Résolution de contraintes géométriques par rigidifications récursive et propagation d'intervalles

Christophe Jermann

THÈSE

présentée à

L'UNIVERSITÉ DE NICE – SOPHIA ANTIPOLIS
UFR SCIENCES

pour obtenir le titre de

DOCTEUR EN SCIENCES

Spécialité

INFORMATIQUE

par

Christophe JERMANN

dirigé par : **Michel RUEHER & Gilles TROMBETTONI**

Résolution de contraintes géométriques par rigidification récursive et propagation d'intervalles

Soutenue publiquement le 20 décembre 2002 devant le jury composé de :

M.	**Frédéric**	**BENHAMOU**	Professeur	Examinateur
M.	**Boi**	**FALTINGS**	Professeur	Rapporteur
M.	**Dominique**	**MICHELUCCI**	Professeur	Rapporteur
M.	**Bertrand**	**NEVEU**	Ingénieur en Chef ENPC	Examinateur
M.	**Michel**	**RUEHER**	Professeur	Directeur
M.	**Gilles**	**TROMBETTONI**	Maître de Conférences	Directeur

Résumé

Les problèmes de satisfaction de contraintes géométriques (GCSP) sont omniprésents dans les applications de CAO, de robotique ou de biologie moléculaire. Ils consistent à chercher les positions, orientations et dimensions d'objets géométriques soumis à des relations géométriques. Le but de la thèse était de proposer une méthode complète et efficace pour la résolution de GCSP.

Dans la première partie, nous comparons des méthodes de résolution et de décomposition, et optons pour la décomposition de Hoffmann *et al.* et la résolution par intervalles. Nous définissons un cadre général pour l'étude de la rigidité, concept central dans les techniques de décomposition géométrique.

Dans la seconde partie, nous analysons la méthode de Hoffmann *et al.*, et les limites inhérentes à toute approche géométrique structurelle. Nous proposons le concept de degré de rigidité pour surmonter certaines de ces limites. Nous introduisons une nouvelle méthode de décomposition, et sa combinaison avec les techniques de réslution par intervalles.

Mots clés : Contraintes, Géométrie, Décomposition, Rigidité, Intervalles

Abstract

Geometric constraint satisfaction problems (GCSPs) are ubiquitous in applications like CAD, robotics or molecular biology. They consist in searching positions, orientations and dimensions of geometric objects bound by geometric constraints. The goal of the thesis was to find an efficient and complete solving method for GCSPs.

In the first part, we compare solving methods and decomposition techniques, and we choose Hoffmann et al's decomposition and interval solving methods . We define a general framework for the study of rigidity in GCSPs, a concept used in all the geometric decomposition methods.

In the second part, we analyse Hoffmann et al's method, and the limits inherent to all the structural geometric approaches. We propose the degree of rigidity concept to overcome some of these limits. We introduce a new decomposition method, and its combination with interval solving techniques.

Keywords : Constraints, Geometry, Decomposition, Rigidity, Intervals

Remerciements

Tout d'abord, merci au professeur Frédéric Benhamou qui a accepté d'assurer la présidence de mon jury de soutenance, ainsi qu'aux professeurs Boi Faltings et Dominique Michelucci dont les élogieux rapports ont été la récompense du labeur de trois années.

Je tiens aussi à remercier Christian Bliek de m'avoir parrainé, avec la société ILOG, pour ma bourse BDI CNRS-Région, et de m'avoir aidé à dégager les voies de recherches intéressantes.

A Michel Rueher, directeur-tacticien de la thèse, vont mes remerciements suivants ; ses conseils avisés et ses traits d'espièglerie m'ont inspiré une nouvelle approche de la science, et une vision différente du monde scientifique et para-scientifique.

Bertrand Neveu et Gilles Trombettoni méritent mes plus sincères remerciements pour la part active qu'ils ont tous deux pris à la thèse. Bertrand pour sa disponibilité et l'attention qu'il sut me prêter à chacune de mes irruptions intempestives dans son bureau ; pour sa grande méfiance face aux intuitions, et son esprit entraîné à repérer les erreurs de raisonnement ; enfin pour le service DGV (déboggage à grande vitesse) qu'il assura avec brio lors de la phase finale d'expérimentation.

Gilles, pour son suivi attentif de la thèse, son soutien et son enthousiasme. Pour nos longues discussions aux détours et contours multiples, parfois productives et souvent enrichissantes. Pour sa capacité à m'amener à approfondir, expliquer et ré-expliquer mes raisonnements tortueux jusqu'à ce qu'ils acquièrent une substance suffisante. Enfin, pour son aide précieuse, et son amitié.

L'équipe Coprin, et en premier lieu son chef, Jean-Pierre Merlet, a droit aussi à mes remerciements, pour le bon temps que j'y ai passé et la mine de connaissances que j'ai pu y trouver.

Mes derniers remerciements vont à ma famille et à mes amis qui m'ont bercé tant d'années dans un cocon douillet. Mes parents, pour tout leur amour ; pour m'avoir fait confiance et m'avoir encouragé à poursuivre mes études toujours plus loin aussi. Stéphanie, sans l'Amour de qui je n'aurais pas eu la force de finir cette thèse, et ma petite Julie, grandie en même temps que la thèse et dont les sourires sont un baume pour le cœur.

Thèse cofinancée par la région Provence-Alpes-Côte d'Azur et le CNRS

À Stéphanie et à Julie,

Table des matières

Introduction

Un système de contraintes géométriques se compose d'un ensemble d'objets géométriques soumis à des contraintes géométriques. Les objets géométriques peuvent être des points, droites et plans par exemple. Les contraintes auxquelles sont soumis les objets géométriques spécifient les positions relatives de ceux-ci : distances, angles, parallélismes et incidences par exemple. Les types d'objets et de contraintes géométriques dépendent du domaine d'application considéré : conception mécanique, dessin, architecture, biologie moléculaire, etc. La taille des systèmes géométriques issus de ces applications peut varier de quelques objets (pour le dessin par exemple) à plusieurs milliers d'objets (en biologie moléculaire par exemple).

La figure 1 présente le système géométrique *Ponts*[1]. Ce système 2D est constitué de 15 points liés par 27 contraintes de distance représentées par des segments sur cette figure.

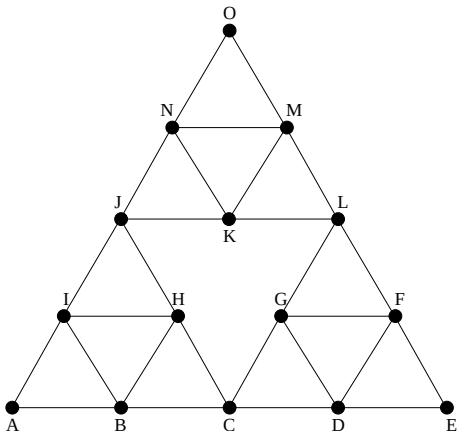

FIG. 1 – Le GCSP *Ponts*

[1]Cette figure est le logo de l'Ecole Nationale des Ponts et Chaussées, exemple didactique utilisé dans la thèse.

Problématique

Le sujet de cette thèse est de proposer une méthode *efficace* pour la *résolution complète* de systèmes de contraintes géométriques.

Résoudre un système de contraintes géométriques consiste à fournir une position, une orientation et des dimensions à chacun de ses objets géométriques de sorte que toutes ses contraintes géométriques soient satisfaites. L'espace géométrique considéré doit donc être muni d'un référentiel global.

Trouver une ou toutes les solutions d'un système géométrique est un problème central pour la plupart des applications où interviennent des relations géométriques : il peut s'agir de mettre à jour un dessin sous contraintes, déterminer les configurations possibles d'un mécanisme ou encore trouver toutes les conformations possibles d'une molécule en 3D.

Résoudre complètement un système de contraintes géométriques consiste à fournir toutes les solutions de ce système. Un système de contraintes géométriques bien-contraint peut en théorie admettre un nombre exponentiel de solutions dans le nombre d'objets géométriques qu'il contient. En pratique, les systèmes de contraintes géométriques comportent suffisamment de contraintes pour admettre un nombre plus restreint de solutions.

Résoudre efficacement un système de contraintes géométriques est nécessaire pour les applications interactives (dessin, CAO) mais aussi pour les applications où les systèmes géométriques sont de grande taille.

Problème de satisfaction de contraintes géométriques

Nous considérons la résolution d'un système géométrique comme un *problème de satisfaction de contraintes géométriques (GCSP)*. Un GCSP est un cas particulier d'un CSP sur domaine continu. Les variables réelles d'un GCSP représentent les positions, orientations et dimensions des objets géométriques ; ses contraintes (équations et inégalités) expriment les contraintes géométriques portant sur les objets géométriques. Le système géométrique *Ponts* est représenté par le GCSP suivant :

- Chaque point P est représenté par le couple de variables réelles (x_P,y_P).
- Chaque contrainte de distance $Distance(P_1, P_2)$ entre un point P_1 et un point P_2 s'exprime par une équation de la forme : $(x_{P_1} - x_{P_2})^2 + (y_{P_1} - y_{P_2})^2 = d^2_{P_1P_2}$

Ce GCSP est constitué de 30 variables réelles provenant des 15 objets géométriques, et de 27 contraintes issues des 27 relations de distances (cf. figure 1).

Résoudre complètement et efficacement un système de contraintes géométriques consiste alors à exploiter une méthode de résolution complète et efficace pour le système d'équations et d'inégalités du GCSP correspondant.

Méthodes de résolution

Pour résoudre le système d'équations et d'inégalités correspondant à un GCSP, on peut utiliser trois types de méthodes : *méthodes formelles, méthodes numériques ou méthodes géométriques.*

Nous avons choisi la méthode numérique de **résolution par intervalles**. Nous fournissons ici une première justification de ce choix :

- **Méthodes formelles** : la résolution symbolique d'un GCSP est souvent très lente. Par exemple, la résolution du système *Ponts* par Maple sur un PII-500MHh-128Mo n'a pas abouti après 24h de calcul.
- **Méthodes numériques** : Bien que très rapide, la résolution par la méthode itérative Newton-Raphson de ce GCSP ne fournit qu'une seule solution ; d'autre part la méthode de résolution par homotopie permet de trouver toutes les solutions mais nécessite une étude particulière du système que l'on cherche à résoudre.
- **Méthodes géométriques** : les méthodes géométriques sont basées sur des règles de construction (à la règle et au compas par exemple) qui ne couvrent pas toutes les instances possibles de GCSP.

La résolution par intervalles permet de fournir un encadrement aussi précis que souhaité de toutes les solutions d'un GCSP. Cette résolution peut cependant prendre un certain temps pour des GCSP composés d'un grand nombre d'équations non-linéaires : environ 1h40 pour produire les 128 solutions du GCSP correspondant à la figure 1 sur un PII-500MHz-128Mo.

Pour améliorer les performances de cette résolution, nous proposons de décomposer le GCSP en sous-problèmes plus faciles à résoudre et dont les solutions peuvent être combinées en solutions du GCSP initial.

Techniques de décomposition

Les techniques de décomposition de CSP sur domaines continus se classent en deux catégories : *techniques équationnelles et techniques géométriques.*

Nous nous sommes intéressés à la technique de **rigidification récursive**, technique géométrique qui décompose un GCSP en sous-systèmes *rigides*, c'est-à-dire indéformables mais déplaçables. Cette technique est dédiée aux systèmes de contraintes géométriques et peut tenir compte de tout type d'objets géométriques et de relations géométriques[2]. Nous donnons une première justification de notre choix ici :

- **Technique équationnelle** : le GCSP *Ponts* se décompose en 8 sous-systèmes de 1, 2 et **14** équations ; rappelons que le temps de résolution n'est pas linéaire dans le nombre d'équations à résoudre et que la taille d'un sous-système à résoudre en une fois constitue donc un critère important quant à l'efficacité de la résolution.
- **Rigidification récursive** : Le même GCSP se décompose en 16 sous-systèmes de 1 et 2 équations.

Nous proposons donc la mise en œuvre de la décomposition par rigidification récursive associée à des méthodes de résolution par intervalles pour la résolution complète et efficace de GCSP.

[2]Dans la limite où les objets sont en eux-mêmes rigides, et les contraintes ne sont pas paramétrées.

Contributions

Les contributions de ce mémoire consistent en la définition d'une nouvelle méthode de rigidification récursive pour la décomposition et la résolution de GCSP en 2D et 3D. Les méthodes de rigidification récursive se découpent généralement en 2 phases :

1. *Décomposition* : cette phase a pour but de produire un ensemble de sous-systèmes plus faciles à résoudre que le GCSP considéré, ainsi qu'un ordre pour la résolution de ces sous-systèmes (plan de construction).
2. *Résolution* : chaque sous-système produit en phase 1 est résolu par intervalles et les solutions de ces sous-systèmes sont combinées en solutions du GCSP entier.

On peut séparer les apports de la thèse en 3 catégories : l'analyse des méthodes de décompositions existantes et du concept de rigidité ; une nouvelle méthode de décomposition, basée sur celle proposée par Hoffmann *et al.* [Hoffmann *et al.*, 1997] ; et la mise en œuvre de la résolution par intervalles d'un GCSP décomposé.

Analyse des méthodes de décomposition géométrique structurelle

Nous proposons une analyse approfondie des limites des méthodes de décomposition géométrique structurelle. Ces méthodes reposent sur l'utilisation de la notion de rigidité qui n'admet pas de caractérisation complète et polynomiale pour les GCSP quelconques.

La plupart des méthodes géométriques, dont celle de Hoffmann *et al.*, utilisent donc une caractérisation approchée et purement structurelle de la rigidité. L'approche géométrique structurelle peut être trompée par la présence de contraintes singulières, redondantes ou inconsistantes.

Après une étude approfondie du concept de rigidité, nous élaborons une explication des limites à la caractérisation de la rigidité. Nous apportons alors des solutions partielles pour surmonter ces limites, tout en expliquant la difficulté inhérente des problèmes posés par toute caractérisation structurelle, qui explique l'absence de solutions complètes et globales.

En particulier, nous présentons quelques mécanismes de détection automatique de redondances et d'inconsistances. Nous proposons de prendre en compte les singularités des GCSP en introduisant le concept de *degré de rigidité* (DDR). Le calcul du degré de rigidité, généralement de l'ordre de la démonstration automatique de théorème géométrique, peut s'avérer plus simple pour certaines classes de GCSP.

Ceci nous permet de proposer une nouvelle caractérisation plus tout à fait structurelle pour caractériser la rigidité : la *rigidité structurelle étendue*. Cette caractérisation est plus proche de la rigidité que ne l'est la rigidité structurelle.

Enfin, nous proposons aussi un nouveau test pour déterminer si l'assemblage de deux sous-GCSP rigides est réalisable, basé lui aussi sur le concept de degré de rigidité.

Nouvelle méthode de décomposition

Parmi les différentes méthodes de rigidification récursive que nous connaissons, celle proposée par Hoffmann, Lomonosov et Sitharam [Hoffmann *et al.*, 1997; 1998; 2000] nous a paru la plus générale :

- elle permet de traiter des GCSP en dimension quelconque ;
- basée sur une approche structurelle, elle n'est pas limitée par un répertoire de règles ;

Méthode de Hoffmann, Lomonosov et Sitharam

La phase de décomposition, définie par Hoffmann *et al.*, est un algorithme de point fixe qui itère sur 3 opérations :

- L'opération de *fusion* a pour but d'identifier un sous-système rigide de petite taille dans le GCSP.
- L'opération d'*extensions* a pour but d'inclure des objets un à un dans le sous-système courant tant que celui-ci demeure rigide.
- L'opération de *condensation* a pour but de remplacer le sous-système identifié par un objet rigide unique qui le représentera dans le GCSP pour les prochaines itérations.

Nous proposons une analyse détaillée de cette méthode ; certains aspects, comme la façon de résoudre le GCSP décomposé ou encore de mettre en collaboration les différentes opérations, n'ont pas été détaillés par Hoffmann *et al.* et nous proposons notre propre solution aux problèmes posés par cette méthode. Nous proposons aussi une amélioration directe de cette méthode consistant à combiner plusieurs des opérations de condensation proposées par Hoffmann *et al.*.

Cette description détaillée de la méthode permet d'en mieux apprécier les limites et les avantages. Cette compréhension profonde nous a permis de proposer une nouvelle méthode de rigidification récursive tout aussi générale mais tirant parti des leçons de notre analyse de la méthode de Hoffmann *et al.*.

Nouvel opérateur de fusion

L'opérateur de fusion proposé par Hoffmann *et al.* est basé sur une analyse des degrés de liberté dans un graphe représentant le GCSP. Cette analyse, effectuée par un algorithme de flot, a pour but de retourner un sous-système rigide de taille minimale. Cet algorithme est incorrect dans le cas général, d'une part car l'approximation de la rigidité qu'il utilise peut se tromper, d'autre part car il ne tient pas compte de la géométrie du système.

Nous définissons un nouvel opérateur de fusion, toujours basé sur un algorithme de flot ; d'une part, il utilise notre caractérisation de la rigidité (strictement meilleure que celle utilisée par Hoffmann *et al.*) et d'autre part il tient compte de la géométrie afin de distribuer le flot de façon correcte.

Par ailleurs, nous proposons une étude approfondie de la complexité du problème de la recherche de sous-systèmes rigides de taille *minimum*, problème prouvé généralement NP-difficile par Hoffmann *et al.* [Hoffmann *et al.*, 1998].

Nouvel opérateur d'extension

Nous présentons un nouvel opérateur d'extension qui permet le partage d'objets géométriques entre sous-systèmes, l'extension sur un sous-système entier en une seule fois et peut éviter certaines étapes de fusion. Cet opérateur améliore ainsi la *finesse* de la décomposition ainsi que le nombre d'itérations nécessaires pour parvenir à un point fixe pour la phase de décomposition. Il consiste à utiliser une règle spéciale lors de l'extension sur des objets appartenant déjà à d'autres sous-systèmes : l'extension globale.

Tout ceci est rendu possible grâce à un nouvel opérateur de condensation.

Nouvel opérateur de condensation

Nous proposons un nouvel opérateur de condensation qui est basé sur un mécanisme de *rigidification automatique*. Cet opérateur consiste à laisser certains objets des sous-GCSP déjà résolus dans le GCSP et à introduire des contraintes entre ces objets afin de les rendre rigides relativement les uns aux autres ; cette rigidité relative représente le fait que ces objets appartiennent déjà à un sous-système rigide résolu.

Nous montrons que ce nouvel opérateur est plus intéressant que ceux proposés par Hoffmann *et al.* car il permet d'augmenter encore le nombre d'extensions possibles, ce qui améliore la finesse de la décomposition et réduit le nombre d'itérations nécessaires.

Résolution par intervalles d'un GCSP décomposé

Nous proposons d'utiliser les méthodes de résolution numérique par intervalles et un processus de retour-arrière afin d'assurer la complétude de la résolution. Un schéma de résolution a été proposé par Bliek *et al.* [Bliek *et al.*, 1998] pour une autre forme de décomposition. La contribution pour cette phase est d'avoir ramené le problème de la résolution d'un GCSP décomposé à ce schéma de résolution d'un *graphe orienté acyclique (DAG) de blocs*, et de proposer plusieurs stratégies de résolution.

Adaptation de la décomposition à la résolution

Le plan produit en phase de décomposition est utilisé pour générer un DAG de blocs. Chaque bloc contient un *sous-système d'équations* du GCSP qui correspond au résultat d'une opération de fusion, ou d'une extension simple. Les arcs du DAG représentent l'ordre partiel imposé pour la résolution de ces blocs.

Nous présentons la **génération du DAG de blocs** correspondant à la décomposition d'un GCSP.

Résolution d'un DAG de blocs par intervalles

La méthode proposée par Bliek *et al.* consiste à résoudre chaque bloc par intervalles et à utiliser un mécanisme de retour-arrière inter-blocs afin d'assurer la complétude de la résolution.

Lors de la résolution d'un bloc, plusieurs stratégies de propagation sont possibles. Nous proposons trois nouvelles stratégies basées chacune sur un compromis différent entre la quantité d'information utilisée et la rapidité du traitement. Des heuristiques permettent d'améliorer les performances de certaines de ces stratégies.

Le mécanisme de retour-arrière peut tirer parti de l'information fournie par la structure du DAG, qui représente un ordre partiel pour la résolution des blocs.

Validation

Nous validons notre méthode sur quelques exemples issus du domaine de la CAO, domaine où les relations entre objets géométriques sont diverses (distance, angle, incidence, parallélisme, ...). Ces expérimentations préliminaires laissent présager d'un gain important par rapport à une méthode de résolution globale, et permettent d'illustrer les avantages des différentes décompositions et des différentes stratégies de résolution.

Plan du mémoire

L'exposé est organisé en 2 parties. La première partie expose un état de l'art concernant la résolution de GCSP. Ce tour d'horizon commence par une définition plus formelle des concepts de GCSP et de résolution de GCSP (chapitre 1). Le chapitre 2 présente l'existant relatif à la notion de rigidité ; cette notion peut être utilisée pour réaliser une étude qualitative du système qui permet de résoudre les GCSP de façon plus renseignée. Les différentes méthodes de résolution que nous avons évoquées dans cette introduction sont présentées au chapitre 3. Nous proposons alors un aperçu de quelques méthodes de décomposition au chapitre 4, qui nous permet de justifier notre intérêt pour la méthode de rigidification récursive proposée par Hoffmann *et al.*.

La seconde partie regroupe essentiellement les contributions de cette thèse que nous avons listées à la section* précédente. Plus précisément, le chapitre 5 expose en détail la méthode proposée par Hoffmann *et al.* et présente quelques réponses à des questions qu'elle laissait en suspens, ainsi que des améliorations directes de cette méthodes. Le chapitre 6 présente une analyse approfondie des limites des méthodes de décomposition géométrique structurelle. Nous y expliquons la difficulté qu'il y a à résoudre ces problèmes, et proposons des solutions partielles pouvant s'avérer très utiles en pratique. Ceci nous permet de définir notre nouvelle caractérisation de la rigidité ainsi qu'un nouveau test déterminant si l'assemblage de deux sous-GCSP rigides est possible. le chapitre 7 présente une nouvelle méthode de rigidification récursive, basée sur le schéma de la méthode proposée par Hoffmann *et al.*. Notre méthode prend en compte les remarques faites et les solutions proposées au chapitre 6 afin de définir de nouveaux opérateurs de fusion, d'extension et de condensation plus corrects et puissants. Nous précisons les limites de la méthode. Le chapitre 8 présente la mise en œuvre de la résolution par intervalles basée sur un plan de construction produit par rigidification récursive.

En conclusion de ce mémoire, nous récapitulons brièvement les contributions de la thèse

à la résolution de GCSP, nous tirons le bilan du travail de thèse et proposons quelques perspectives.

Quatre annexes fournissent des données techniques complémentaires. La modélisation des objets et contraintes géométriques que nous avons utilisée dans le prototype développé au cours de la thèse est présentée à l'annexe A. L'annexe B présente l'étude de la complexité d'un problème en marge de la thèse, le problème MINIMUM-DENSE ; ce problème a été prouvé NP-difficile dans le cas général par Hoffmann *et al.*, mais nous montrons qu'il existe des classes pratiques de GCSP pour lesquelles ce problème peut-être résolu en temps polynomial, alors que les instances utilisées par la démonstration de Hoffmann *et al.* n'existent pas en pratique. L'annexe C propose des tables pouvant être utilisées pour le calcul du degré de rigidité, ainsi que des règles pour la rigidification automatique d'ensembles d'objets géométriques. Enfin, l'annexe D regroupe quelques résultats expérimentaux qui permettent de valider la nouvelle approche présentée aux chapitres 7 et 8 sur des GCSP en 2D et 3D.

Un glossaire, la liste des définitions et théorèmes énoncés dans la thèse, et un index bibliographique terminent le mémoire.

Première partie

Problématique et état de l'art

Chapitre 1

Problématique

Sommaire

Ce chapitre présente de façon plus formelle la problématique de la thèse : *la résolution de systèmes de contraintes géométriques.*

Pour aborder les systèmes géométriques, nous proposons d'utiliser le formalisme des contraintes dans lequel tout problème se décrit par un CSP (*constraint satisfaction problem*). Nous verrons que l'on peut considérer deux formes de CSP équivalentes pour décrire un système géométrique. Chacune des ces formulations a ses avantages et les deux sont utilisées dans les méthodes de résolution et décomposition que nous présentons aux chapitres 3 et 4.

La première forme de GCSP (pour *geometric constraint satisfaction problem*) consiste à considérer les objets géométriques comme les variables du problème et les contraintes géométriques comme les contraintes du GCSP. Il s'agit alors de trouver les positions, orientations et dimensions des objets géométriques de façon à satisfaire les contraintes géométriques.

La seconde forme de GCSP consiste à représenter chaque objet géométrique par un ensemble de paramètres à valeurs dans $\mathbb{R}$, et chaque contrainte géométrique par un ensemble d'équations dont les inconnues sont les paramètres des objets. Les variables du GCSP sont alors les paramètres des objets géométriques, et les contraintes sont les équations représentant les contraintes géométriques. Le problème consiste alors à trouver une valeur réelle pour chaque paramètre de sorte que toutes les équations soient vérifiées.

Notons que ces deux GCSP sont en quelque sorte imbriqués : le second correspond à une expansion des variables et des contraintes du premier. C'est pourquoi ils sont équivalents, et, dans la suite de la thèse, nous parlerons de GCSP pour représenter indifféremment l'une ou l'autre de ces représentations.

Dans les sections qui suivent, nous allons définir formellement la première forme de GCSP qui se situe au niveau géométrique. Nous définirons ensuite la seconde forme de GCSP qui se situe au niveau variables-équations et nous verrons alors que ces deux formulations sont équivalentes. Nous définirons alors ce que nous entendons par résolution de GCSP, et l'approche que nous proposons pour la résolution.

1.1 GCSP : le niveau géométrique

Commençons par rappeler brièvement la définition générale d'un CSP dans le formalisme des contraintes :

Définition 1 CSP
Un CSP est un triplet (X, D, C) où :
- *X est l'ensemble des variables du CSP,*
- *D est l'ensemble des domaines ; un domaine représente un ensemble de valeurs admissibles pour une variable de X,*
- *C est l'ensemble des contraintes portant sur les variables de X.*

Lorsque l'on considère un GCSP au niveau géométrique, l'ensemble X est constitué des objets géométriques, l'ensemble D est l'ensemble des configurations possibles pour tous les objets géométriques et l'ensemble C contient les contraintes géométriques. Plus formellement :

Définition 2 GCSP au niveau géométrique
Un GCSP au niveau géométrique se définit par un quadruplet (E_g, X_g, D_g, C_g) où :
- *E_g est l'espace géométrique considéré ; cet espace est muni d'un repère global,*
- *X_g est l'ensemble des objets géométriques du GCSP,*
- *D_g est l'ensemble des configurations possibles pour tous les objets géométriques de X_g,*
- *C_g est l'ensemble des contraintes géométriques liant les objets géométriques de X_g.*

Dans la suite du mémoire, on utilisera souvent la notation abrégée $S = (O, C)$ pour représenter un GCSP, O étant l'ensemble des objets géométriques, C étant l'ensemble des

contraintes géométriques. L'espace géométrique et les domaines sont omis afin de rendre la notation plus concise.

Considérons à nouveau le système *Ponts* présenté à la figure 1. Sa représentation en tant que GCSP au niveau géométrique est la suivante :

- $X_g = \{A, B, C, D, E, F, G, H, I, J, K, L, M, N, O\}$,
- $D_g = \{Point, Point, Point, ..., Point\}$,
- $C_g = \{Distance(A, B), Distance(A, I), Distance(B, I), ..., Distance(N, O)\}$.

Afin de simplifier les notations dans la suite du mémoire, nous définissons les opérations ensemblistes sur les GCSP :

Définition 3 Opérations ensemblistes sur GCSP
Soient $S = (O, C)$, $S1 = (O1, C1)$ *et* $S2 = (O2, C2)$ *trois GCSP en espace géométrique semblable.*
On dit que $S1$ *est un* sous-GCSP *de* S, *noté* $S1 \subset S$, *si et seulement si* $O1 \subset O$ *et* $C1 = \{c \in C | c$ *ne porte que sur des objets de* $O1\}$. *Un sous-ensemble* $O1 \subset O$ *d'objets géométriques induit donc un unique sous-GCSP* $S1$ *de* S.
Si $S1 \subset S$, *on note* $S \setminus S1$ *le sous-GCSP de* S *induit par* $O \setminus O1$.
Si $S1 \subset S$ *et* $S2 \subset S$, *on note* $S1 \cap S2$ *le sous-GCSP de* S *induit par* $O1 \cap O2$ *; on note* $S1 \cup S2$ *le sous-GCSP de* S *induit par* $O1 \cup O2$.
On dit que S *est le* GCSP vide, *noté* $S = (\emptyset, \emptyset)$, *si* $O = \emptyset$, *ce qui implique que* $C = \emptyset$ *aussi.*

Dans la suite du mémoire, nous dénoterons les sous-GCSP par la séquence des noms des objets qui l'induisent : par exemple, ABI sera le sous-système de *Ponts* constitué des points A, B et I et donc des contraintes de distance $Distance(A, B)$, $Distance(A, I)$ et $Distance(B, I)$.

Afin de compléter la définition de GCSP au niveau géométrique, nous allons maintenant définir les concepts d'objet géométrique, de configuration d'objet géométrique et de contrainte géométrique.

1.1.1 Objets géométriques

Un *objet géométrique* en espace E_g de dimension d se définit par l'ensemble des points de cet espace qui lui appartiennent. Les objets géométriques usuels sont points, droites, cercles, plans, etc ... Une *configuration d'un objet géométrique* définit exactement l'ensemble des points de cet objet ; autrement dit, une configuration définit la position, l'orientation et les dimensions d'un objet géométrique.

Définition 4 Configuration d'un objet géométrique
On appelle **configuration d'un objet géométrique** *la donnée d'une position, d'une orientation et de dimensions pour cet objet géométrique.*
Un objet géométrique muni d'une configuration est dit **déterminé**.

Par exemple, la configuration d'un point en 2D consiste simplement à définir sa position vis à vis du repère global considéré ; la configuration d'un cercle en 3D consiste à donner

la position de son centre, l'orientation du plan qui le contient et la dimension de son rayon. Ainsi, le domaine d'un objet géométrique se définit par l'ensemble des configurations admises par cet objet. On définit l'espace de toutes les configurations possibles pour un type d'objet géométrique donné : Point sera l'ensemble de toutes les configurations possibles pour un point, Droite l'ensemble de toutes les configurations possibles pour une droite, etc ... Dans le cas du système *Ponts*, nous avons défini le domaine de chacun des 15 points comme étant l'espace de toutes les configurations de points. Notons que l'espace des configurations admises par un objet géométrique dépend de l'espace géométrique E_g considéré : en 2D, la configuration d'un cercle correspond seulement à sa position et sa dimension, alors qu'en 3D elle est constituée en plus d'une orientation.

1.1.2 Contraintes géométriques

Une *contrainte géométrique* est une relation entre des objets géométriques qui restreint l'ensemble des configurations possibles pour ces objets. Les contraintes géométriques usuelles sont distances, angles, incidences, parallélismes, etc ... Nous distinguons 2 catégories de contraintes géométriques : les *contraintes valuées* et les *contraintes booléennes*. Une contrainte valuée doit être associée à une valeur pour être définie, alors qu'une contrainte booléenne ne nécessite pas de valeur pour être définie. Par exemple, une contrainte de distance ou d'angle n'est définie que si la valeur de la distance ou de l'angle est donnée ; en revanche, les contraintes d'incidence ou de parallélisme peuvent être posées indépendamment de toute valeur numérique : elles ne portent que sur les objets géométriques.

On peut considérer les contraintes booléennes comme des contraintes valuées particulières : une incidence est une distance valant 0, un parallélisme est un angle valant 0. Nous verrons au chapitre 4 qu'il est important de distinguer contraintes valuées et contraintes booléennes ; en effet, représenter les contraintes booléennes par des contraintes valuées introduit des singularités dans le GCSP qui peuvent faire échouer certaines méthodes de résolution.

Résoudre un GCSP au niveau géométrique revient donc à déterminer tous les objets géométriques, c-a-d. à identifier une (ou toutes les) configuration(s) pour chaque objet géométrique, de telle sorte que toutes les contraintes géométriques soient satisfaites.

1.2 GCSP : le niveau variables-équations

Un GCSP au niveau variables-équations représente les objets géométriques par des ensembles de paramètres à valeurs dans $\mathbb{R}$ et les contraintes géométriques par des ensembles d'équations et inégalités sur ces paramètres. Une fois la modélisation des objets et contraintes géométriques choisie, la définition d'un GCSP au niveau variables-équations est la même que celle d'un CSP quelconque (cf. définition 1, page 12). Nous allons donc nous attarder dans les sections suivantes sur la question de la modélisation des objets et des contraintes géométriques. Nous ne proposerons pas une analyse des différentes modélisations existantes, mais nous proposons un cadre pour la définition de modélisations.

1.2.1 Objets géométriques

Nous avons vu qu'un objet géométrique est défini par l'ensemble des points de l'espace géométrique qui lui appartiennent. Cet ensemble est généralement décrit *en compréhension*, c-a-d. à l'aide d'une relation mathématique, appelée *relation externe*, qui est satisfaite pour tous les points appartenant à l'objet. Par exemple, on considère généralement la formulation analytique $ax + by + c = 0$ pour modéliser une droite en 2D. Cependant, on peut aussi modéliser une droite par l'équation $y = dx+e$ ou encore par l'équation $x = fy+g$. Il existe donc plusieurs représentations possibles de l'ensemble des points appartenant à un objet géométrique. A chacune de ces représentations, on peut associer un ensemble de *paramètres* qui sont les coefficients apparaissant dans la relation décrivant l'objet. Par exemple, pour la droite en 2D, on a la paramétrisation $\{a, b, c\}$ pour $ax+by+c = 0$, $\{d, e\}$ pour $y = dx+e$ ou $\{f, g\}$ pour $x = fy + g$. C'est en choisissant des valeurs (dans $\mathbb{R}$) pour ces paramètres que l'on détermine la position, l'orientation et les dimensions de l'objet géométrique. Par exemple, fixer le paramètres d d'une droite en 2D revient à déterminer sa pente, c-a-d. son orientation, et fixer son paramètre e détermine son ordonnée à l'origine, c-a-d. sa position. Ces paramètres doivent être fixés vis à vis d'un repère, c'est pourquoi l'espace géométrique considéré doit être muni d'un repère global.

On peut donc associer un objet géométrique à un ensemble de paramètres dont les valeurs permettent de représenter la position et l'orientation de cet objet vis-à-vis d'un repère donné. Cependant, certaines paramétrisations ne permettent pas de décrire toutes les configurations possibles d'un objet géométrique. Par exemple, la représentation d'une droite en 2D par les paramètres $\{d, e\}$ pour $y = dx + e$ ne permet pas de représenter les droites verticales du type $x = constante$. On dit alors que la paramétrisation est *incomplète*, par opposition aux paramétrisations *complètes* qui permettent de décrire toutes les configurations d'un objet. Dans la suite de la thèse, nous considèrerons que la paramétrisation des objets géométriques est complète afin d'éviter l'apparition de GCSP insolubles à cause d'une paramétrisation incomplète.

D'autre part, certaines paramétrisations d'un objet ne permettent pas de décrire une configuration d'un objet de façon unique. Par exemple, la paramétrisation de la droite par $\{a, b, c\}$ pour $ax + by + c = 0$ n'est pas unique : cette droite admet toutes les paramétrisation de la forme $\{k * a, k * b, k * c\}$ pour tout $k \in \mathbb{R} - 0$; le choix d'une valeur de k représente le choix d'un vecteur directeur pour la droite, et il existe une infinité de vecteurs directeurs d'une même droite. Afin que la représentation d'une droite associe une unique valuation des paramètres à une configuration donnée, on peut alors introduire une relation d'unicité. Par exemple, dans le cas de la droite représentée par $\{a, b, c\}$, on peut choisir d'introduire la relation $\{a^2 + b^2 = 1, (b \leq 0 \vee (b = 0 \wedge a > 0))\}$ qui représente le fait que les vecteurs directeurs $(-ba)$ des droites ainsi modélisées seront normés $(a^2 + b^2 = 1)$et orientés vers la droite $(b \leq 0)$, ou vers le haut $(b = 0 \wedge a > 0)$. Dans la suite de la thèse, nous considèrerons que la paramétrisation des objets géométriques est unique, si nécessaire par introduction d'une relation d'unicité, appelée *relation interne* à l'objet géométrique. Lorsque la paramétrisation est complète et unique sans adjonction d'une relation interne, on dira qu'elle est *minimale*.

Voici la définition que nous proposons pour la modélisation d'un objet géométrique :

Définition 5 Modélisation d'un objet géométrique
On représente un objet géométrique o par un triplet (P_o, I_o, E_o) *où :*
- P_o *est l'ensemble des* **paramètres** *de l'objet o,*
- I_o *est la* **relation interne** *de l'objet o, nécessaire si la paramétrisation n'est pas unique ; si* $I_o = \emptyset$*, la modélisation est dite* **minimale** *;*
- E_o *est la* **relation externe** *de l'objet o, paramétrée par* P_o*, qui définit l'ensemble des points de l'espace géométrique qui appartiennent à o.*

Les représentations que nous utilisons pour les objets géométriques usuels en 2D et 3D sont regroupées dans l'annexe A.

Puisque nous considérons uniquement les modélisations complètes et uniques des objets géométriques, il existe une bijection entre les valeurs des paramètres et les configurations d'un objet géométrique. En effet, à toute configuration correspond une et une seule valuation des paramètres. Aussi, nous appellerons par la suite *configuration* une valuation des paramètres d'un objet géométrique qui respecte la relation interne de cet objet.

1.2.2 Contraintes géométriques

Une contrainte géométrique est une relation entre des objets géométriques qui restreint l'ensemble des configurations qu'ils peuvent prendre. Comme nous venons de le voir, une paramétrisation d'un objet géométrique permet de représenter l'espace des configurations qu'il peut prendre. Il est donc naturel de vouloir représenter une contrainte géométrique par une relation sur les paramètres des objets géométriques, appelées *relation externe* de la contrainte. Par exemple on peut représenter une contrainte de distance entre 2 points en 2D par une relation du type $(a-c)^2+(b-d)^2 = e^2$, où $\{a, b\}$ et $\{c, d\}$ sont les coordonnées des 2 points et e est la valeur de la distance. On constate que certaines contraintes nécessitent des paramètres : ce sont les contraintes que nous avons appelées valuées. Ainsi, la contrainte de distance nécessite la valeur de la distance, représentée par le paramètre e. On peut aussi constater que la modélisation d'une contrainte dépend de la paramétrisation choisie pour les objets géométriques. Par exemple, si un point était représenté comme l'intersection de 2 droites, c-a-d. avec 6 paramètres, la relation de distance entre 2 points en 2D serait toute autre.

Une contrainte géométrique peut aussi nécessiter l'introduction d'une *relation interne* portant sur ses paramètres. Cette relation interne a pour but de rendre unique et *non dégénérée* la modélisation de la contrainte géométrique. Par exemple, la relation de distance est représentée avec un paramètre e dont la valeur représente la distance ; on peut alors ajouter la relation interne $e > 0$ à la modélisation de cette contrainte. En effet, la contrainte de distance paramétrée par $e = k$ ou par $e = -k$ est équivalente et on choisit donc de supprimer l'une des deux possibilités pour rendre la modélisation unique. De plus, la valeur 0 n'a pas de sens pour le paramètre e : une relation de distance valant 0 entre 2 points n'est pas une relation de distance mais une relation de coïncidence. C'est pourquoi nous introduisons la relation interne $e > 0$ pour la contrainte de distance.

La modélisation d'une contrainte géométrique est donc symétrique de la modélisation d'un objet géométrique :

Définition 6 Modélisation d'une contrainte géométrique
Une contrainte géométrique c se définit par un quadruplet (O_c, P_c, I_c, E_c) où :
- *O_c est l'ensemble des objets géométriques soumis à cette contrainte géométrique.*
- *P_c est l'ensemble des* paramètres *de la contrainte c ; si c est une* **contrainte booléenne**, *$P_c = \emptyset$; sinon, c est une* **contrainte valuée**.
- *I_c est la* **relation interne** *de c qui définit les valeurs possibles pour ses paramètres de contrainte.*
- *E_c est la* **relation externe** *de c, paramétrée par P_c, qui définit les configurations des objets géométriques de O_c qui vérifient la contrainte géométrique c.*

Les représentations que nous utilisons pour les contraintes géométriques usuelles en 2D et 3D sont regroupées dans l'annexe A.

1.2.3 GCSP variables-équations

Un GCSP au niveau variables-équations se définit à partir des modélisations des objets et des contraintes géométriques comme suit :

Définition 7 GCSP correspondant à un système géométrique
Un GCSP (E_g, X_g, D_g, C_g) se définit au niveau variables-équations par un triplet (X, D, C) où :
- *$X = \bigcup_{o \in X_g} P_o$, où P_o est l'ensemble des paramètres géométriques de l'objet géométrique o,*
- *D contient un domaine de valeurs (inclus dans $\mathbb{R}$) pour chaque paramètre géométrique de chaque objet géométrique de X_g,*
- *$C = \bigcup_{o \in X_g} I_o \cup \bigcup_{c \in C_g} E_c$, où I_o est la relation interne à chaque objet géométrique $o \in X_g$ et E_c est la relation externe de chaque contrainte géométrique $c \in C_g$.*

Rappelons que les configurations des objets géométriques de O sont en bijection avec les valeurs réelles possibles pour les paramètres géométriques de cet objet. Pour représenter les domaines des paramètres d'un objet géométrique de façon concise, nous utiliserons les plus petits intervalles contenant toutes les valeurs correspondant à des configurations autorisées pour cet objet géométrique de O. Par exemple, les paramètres $\{a, b, c\}$ d'une droite en 2D peuvent prendre leurs valeurs dans $\mathbb{R}$. En revanche, le paramètre r représentant le rayon d'un cercle en 2D ne peut prendre ses valeurs que dans $]0, +\infty[$. De façon plus générale, les paramètres de position prennent leurs valeurs dans $\mathbb{R}$, les paramètres d'orientation ou de dimension angulaire prennent leurs valeurs dans $[0, 2\pi[$ et les paramètres de dimension représentant une longueur prennent leurs valeurs dans $]0, +\infty[$.

Par définition, les seules variables sont les paramètres des objets géométriques du GCSP. Ainsi, les paramètres P_c d'une contrainte valuée ne sont pas des variables du GCSP : pour qu'un GCSP soit défini, ces paramètres doivent tous être valués (de façon consistante avec

la relation interne I_c) lors de la définition de la contrainte géométrique. Ceci n'est pas vraiment une limitation au cadre des contraintes géométriques car, si la valeur des paramètres d'une contrainte valuée peut être calculée uniquement à partir des autres contraintes géométriques, c'est que cette contrainte est *redondante* vis-à-vis du reste des contraintes. Dans ce mémoire, nous ne nous intéressons pas aux contraintes redondantes valuées dont les paramètres ne sont pas déterminés car elles ne changent ni les propriétés géométriques du GCSP, ni son espace de solutions[1].

Reprenons l'exemple de système *Ponts*. Sa représentation sous forme de GCSP au niveau variables-équations est la suivante :

- $X = \{x_A, y_A, x_B, y_B, x_C, y_C, ..., y_O\}$
- $D = \{D_{x_A} = [-\infty, +\infty], D_{y_A} = [-\infty, +\infty], ..., D_{y_O} = [-\infty, +\infty]\}$
- $C = \{(x_A - x_B)^2 + (y_A - y_B)^2 = d_{AB}^2, (x_A - x_I)^2 + (y_A - y_I)^2 = d_{AI}^2, ..., (x_N - x_O)^2 + (y_N - y_O)^2 = d_{NO}^2\}$

Résoudre un GCSP au niveau variables-équations, c'est trouver une (ou toutes les) valuation(s) des variables dans X tel que toutes les équations dans C soient satisfaites. Nous avons vu que l'ensemble des valuations des paramètres des objets géométrique est en bijection avec l'espace des configurations de ces objets. Il en résulte que la résolution d'un GCSP au niveau variables-équations est équivalente à la résolution d'un GCSP au niveau géométrique.

Dans les chapitres suivants, nous emploierons le terme GCSP pour désigner aussi bien le niveau géométrique que le niveau variables-équations. Ainsi, on pourra parler des objets géométriques d'un GCSP, et de ses contraintes géométriques ou, de façon équivalente, des variables du GCSP qui sont les paramètres géométriques de ses objets géométriques, et des équations du GCSP qui sont les relations représentant ses contraintes géométriques.

1.3 Résolution d'un GCSP

Le formalisme des contraintes laisse une liberté complète quant à la méthode utilisée pour la résolution, puisqu'il est représenté sous la forme d'un ensemble de relations mathématiques, équations et inégalités sur des variables à valeurs dans des intervalles de $\mathbb{R}$. Cependant, les GCSP sont constitués d'équations non linéaires généralement algébriques mais pouvant parfois être non algébriques, comme le GCSP exposé à la figure 1.1. Ce GCSP en 3D est un mécanisme à came constitué de 2 parties mobiles ; la partie verticale est constituée d'une barre pouvant coulisser et dont l'extrémité I doit rester incidente à la courbe sinusoïdale S qui peut coulisser horizontalement. La définition de l'ordonnée du point I est une fonction sinusoïdale dépendant de la distance AB qui définit la position horizontale de la courbe S.

Le chapitre 3 expose un ensemble de méthodes qui peuvent être utilisées pour résoudre un GCSP.

[1]Notons cependant que de telles contraintes peuvent devenir indispensables à la résolution de GCSP contenant des incertitudes ou des imprécisions, car alors les valeurs associées aux paramètres des contraintes valuées deviennent des intervalles représentant l'erreur possible.

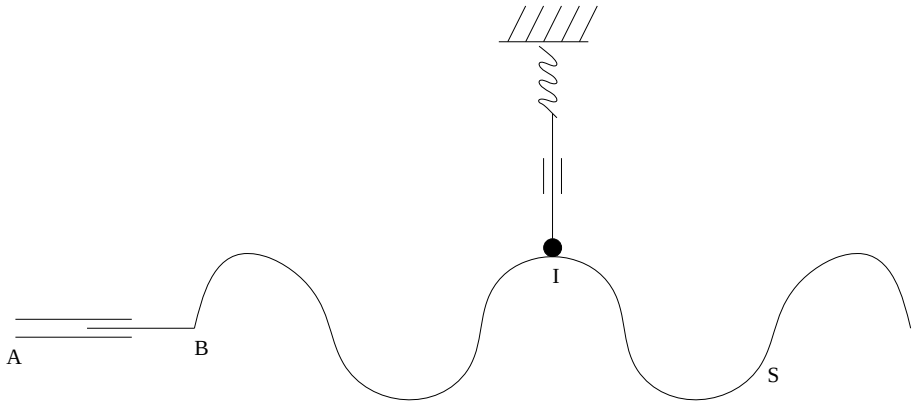

FIG. 1.1 – Exemple de GCSP induisant des relations non-algébriques

Afin de résoudre efficacement des GCSP, nous proposons d'utiliser certaines de leurs propriétés. Plus précisément, nous allons effectuer une analyse du GCSP à résoudre qui permettra :

- d'identifier les sous-systèmes inconsistants ou possédant un ensemble continu de solutions,
- de produire une séquence de sous-problèmes dont les solutions forment des solutions du GCSP.

Au chapitre 4, nous présentons quelques méthodes de décomposition qui peuvent être utilisées pour l'analyse de GCSP. L'utilisation d'une méthode de décomposition implique celle d'une méthode de résolution complète, comme nous l'expliquerons alors. Dans le chapitre suivant, nous présentons le concept de rigidité qui est à la base des méthodes de décomposition géométrique. Cette notion peut être rapprochée, comme nous le verrons au chapitre 2, de la notion de CSP bien-contraint qui est plus connue dans la communauté de programmation par contraintes, et a déjà été utilisée pour proposer des méthodes d'analyse de CSP continus. Nous présentons ce concept à la section suivante.

1.4 CSP bien-, sur- et sous-contraints

On définit 3 catégories de CSP sur domaines continus : CSP bien-contraints, CSP sur-contraints et CSP sous-contraints. Un CSP se classe dans l'une de ces 3 catégories en fonction du nombre de solutions qu'il admet :

Définition 8 Bien-, sur- et sous-contraint
Un CSP est dit **bien-contraint** *s'il admet un ensemble dénombrable de solutions ; il est dit* **sur-contraint** *s'il n'admet aucune solution ; enfin, il est dit* **sous-contraint** *s'il admet un ensemble indénombrable de solutions.*

Considérons les CSP : $P1 = (\{x\}, \{D_x\}, \{sin(x) = 1\})$, $P2 = (\{x, y\}, \{D_x, D_y\}, \{x - y = 0\})$ et $P3 = (\{x, y\}, \{D_x, D_y\}, \{x^2 + y^2 = 0, x + y = 2\})$. $P1$ est bien-contraint, $P2$

est sous-contraint et $P3$ est sur-contraint. En effet, toutes les solutions de $P1$ sont de la forme $x = k\pi$, avec $k \in \mathbb{Z}$. Si $D_x = \mathbb{R}$, l'ensemble de ces solutions est dénombrable, mais si $D_x = [a, b]$, l'ensemble de ses solutions est alors fini. Dans tous les cas, ce CSP est bien-contraint. Dans $P2$, toute valeur v dans $D_x \cap D_y$ correspond à une solution $x = y = v$; donc si $D_x \cap D_y \neq \emptyset$ et $D_x \cap D_y \neq [a, a]$, l'ensemble des solutions de ce CSP est continu, c-a-d. indénombrable. Enfin, $P3$ ne possède aucune solution ; en effet, l'équation $x^2+y^2 = 0$ implique que $x = 0$ et $y = 0$, mais l'équation $x + y = 2$ implique que $x \neq 0$ ou $y \neq 0$. Autrement dit, ce CSP contient une *inconsistance* : il est sur-contraint.

1.4.1 Caractérisation algébrique

Nous avons vu qu'un GCSP se ramène à un système d'équations et d'inégalités. Si le système d'équations est algébrique, on peut caractériser l'espace des solutions complexes d'un GCSP : le calcul d'une base de Gröbner permet de définir la dimension de la plus grande composante de la variété définie par le GCSP, appelée dimension de Hilbert ; cette dimension représente la dimension maximale de l'espace des solutions. Si elle est supérieure à 0, le GCSP admet un ensemble continu de solutions. L'espace des solutions réelles étant contenu dans l'espace des solutions complexes, on peut déduire certaines informations quant au nombre de solutions réelles à partir des informations sur le nombre de solutions complexes :

- si un GCSP est sur-contraint sur $\mathbb{C}$, alors il l'est également sur $\mathbb{R}$;
- s'il est bien-contraint sur $\mathbb{C}$, sur $\mathbb{R}$ il est soit sur-contraint, soit bien-contraint.

De plus, certains algorithmes ont été développés qui permettent, dans le cas où la dimension de Hilbert est 0, de décider si un CSP admet un ensemble vide ou fini de solutions réelles [Rouillier, 1999], ou d'identifier une solution réelle sur chaque composante de la variété d'un système si la dimension est positive [Rouillier *et al.*, 2000]. Malheureusement, les méthodes appliquées dans le cas où la dimension de Hilbert est positive ne permettent pas de déterminer si le CSP est bien- ou sous-contraint dans les réels. De plus, ces méthodes sont inapplicables pour des GCSP non-algébriques, et elles sont trop coûteuses pour être appliquées dans la résolution efficace de GCSP de taille industrielle.

1.4.2 Caractérisation structurelle

On peut avoir recours à une approximation structurelle de la caractérisation du nombre de solutions réelles d'un GCSP. Celle-ci est basée sur l'intuition suivante : un CSP possédant autant d'équations que de variables est *généralement* bien-contraint ; un CSP possédant plus d'équations que de variables est *généralement* sur-contraint ; et un CSP possédant moins d'équations que de variables est *généralement* sous-contraint.

L'intérêt de cette approximation est que tout CSP peut être classé en temps polynomial dans l'une des 3 catégories qu'elle définit. En effet, la décomposition Dulmage-Mendelsohn [Dulmage and Mendelsohn, 1958] permet d'identifier les composantes structurellement bien-, sur- et sous- contraintes de tout CSP à l'aide d'un couplage maximum d'un graphe représentant la structure du CSP (cf. section 4.1.1, page 88).

Définition 9 Structurellement bien-, sur- et sous-contraint
Un CSP (X, D, C) *est* **structurellement bien-contraint**, *noté bien-s_ contraint si* $|X| = |C|$ *et pour tout sous-problème* (X', D', C') [2] $|X'| \geq |C'|$;

ce CSP est **structurellement sur-contraint**, *noté sur-s_ contraint, s'il existe un sous-problème* (X', D', C') *tel que* $|X'| < |C'|$;

il est **structurellement sous-contraint**, *noté sous-s_ contraint si* $|X| > |C|$ *et pour tout sous-problème* (X', D', C'), $|X'| \geq |C'|$.

Limites de la caractérisation structurelle

La caractérisation structurelle ne fournit qu'une approximation du nombre de solutions réelles. La table 1.1 explique les différences entre la caractérisation structurelle du nombre de solution d'un CSP et son véritable nombre de solutions. Un exemple de CSP est associé à chaque case de cette table où la caractérisation structurelle est fausse. Il y a trois causes principales d'échec de la caractérisation structurelle : présence de contraintes inconsistantes, algébriquement redondantes ou singulières.

Définition 10 Inconsistance
Un ensemble de contraintes est **inconsistant** *si et seulement si il n'admet aucune solution. Il est* **minimal** *si le retrait d'une seule contrainte produit un problème consistant. Toute contrainte appartenant à un ensemble de contraintes inconsistant minimal est dite* **inconsistante**.

Notons qu'un CSP ou GCSP contenant une contrainte inconsistante est forcément sur-contraint, quel que soit le nombre d'équations et variables qui le constitue.

Définition 11 Redondances
Dans un CSP, une contrainte est dite **redondante** *si son retrait produit un CSP équivalent, c-a-d. possédant le même ensemble de solutions.*

Dans un CSP, une contrainte est dite algébriquement redondante, notée **A-redondante** *si son retrait produit un CSP appartenant à la même catégorie : bien-, sur- ou sous-contraint.*

Dans un GCSP, une contrainte est dite redondante vis-à-vis de la rigidité, notée **R-redondante**, *si son retrait ne change pas la catégorie du GCSP : bien-, sur- ou sous-rigide.*

On distinguera donc dans ce mémoire 3 types de redondances : la première définition est la définition la plus commune de redondance : une contrainte redondante est une contrainte qui n'apporte aucune information supplémentaire. On pourrait dire qu'une contrainte redondante dans ce sens est *superflue*. Une contrainte algébriquement redondante peut être superflue, mais peut aussi ne pas l'être. Par exemple, le système $x^2 = 4 \wedge x = 2$

[2] un sous-problème (X', D', C') d'un CSP (X, D, C) est induit par les variables de $X' \subset X$, c-a-d. $D' = \{D'_x | x' \in X'\}$ et $C' = \{c \in C | c$ porte uniquement sur $X'\}$.

est redondant car il contient deux équations permettant de déterminer une seule variable ; l'équation $x^2 = 4$ est A-redondante et superflue puisqu'elle ne change pas l'ensemble des solutions ; l'équation $x = 2$, elle, est A-redondante puisque sa présence change l'ensemble des solutions du système. Toute redondance est donc une A-redondance. Enfin, la R-redondance est une notion de redondance similaire à la redondance algébrique, mais relative à la propriété de rigidité plutôt qu'à la dimension de l'espace des solutions. Nous verrons à la section 2.3.3 (page 59) que la notion de rigidité est étroitement liée à la dimension de l'espace des solutions, et qu'il existe donc un lien entre A-redondance et R-redondance.

Définition 12 Singularité
Un ensemble de m équations est **singulier** *s'il permet de déterminer $m + n$ variables, et $n > 0$. On dira d'une équation qu'elle est singulière si elle permet, à elle seule, de déterminer plus d'une inconnue.*

Nous verrons aussi que la notion de singularité revêt un sens géométrique lorsqu'elle est appliquée aux configurations d'un GCSP (cf. section 2.1.4, page 35).

		catégorie du CSP		
		sur	bien	sous
carac.	sur	-	A-redondance (ex.3)	A-redondance (ex.5)
struc.	bien	inconsistance (ex.1)	-	A-redondance (ex.6)
	sous	inconsistance (ex.2)	singularité (ex.4)	-

TAB. 1.1 – Problèmes de la caractérisation structurelle de l'espace des solutions

Ci-dessous, les exemples qui illustrent les différents types d'échec de la caractérisation structurelle de la catégorie d'un CSP.

Exemple 1 Sur-contraint, structurellement bien-contraint
Le CSP $(X = \{x\}, D = \{D_x\}, C = \{x^2 = -1\})$ est sur-contraint ; en effet, il possède un ensemble vide de solutions. En revanche il est structurellement bien-contraint, car il est constitué de 1 équation et de 1 variable. La raison de cette différence est que cette équation est inconsistante : le carré d'un variable réelle ne peut être négatif.

Exemple 2 Sur-contraint, structurellement sous-contraint
Le CSP $(X = \{x, y\}, D = \{D_x, D_y\}, C = \{x^2 + y^2 = -1\})$ est sur-contraint ; en effet, il possède un ensemble vide de solutions. En revanche il est structurellement sous-contraint, car il est constitué de 1 équation et de 2 variables. La raison de cette différence est que cette équation est inconsistante : la somme des carrés de variables réelles ne peut être négative.

Exemple 3 Bien-contraint, structurellement sur-contraint
Le CSP $(X = \{x\}, D = \{D_x\}, C = \{x = 1, 2x = 2\})$ est bien-contraint ; en effet, il possède un ensemble dénombrable de solutions. En revanche il est structurellement sur-contraint,

car il est constitué de 2 équations et de 1 variable. La raison de cette différence est que les 2 équations sont A-redondantes, et même redondantes : il est ici évident que la seconde équation est égale à 2 fois la première.

Exemple 4 Bien-contraint, structurellement sous-contraint
Le CSP $(X = \{x, y\}, D = \{D_x, D_y\}, C = \{x^2 + y^2 = 0\})$ *est bien-contraint; en effet, il possède un ensemble dénombrable de solutions. En revanche il est structurellement sous-contraint, car il est constitué de 1 équation et de 2 variables. La raison de cette différence est que cette équation est singulière : si la somme des carrés des variables est nulle, toutes les variables sont égales à 0.*

Exemple 5 Sous-contraint, structurellement sur-contraint
Le CSP $(X = \{x, y\}, D = \{D_x\}, C = \{x - y = 1, 2x - 2y = 2, 3x - 3y = 3\})$ *est sous-contraint; en effet, il possède un ensemble indénombrable de solutions. En revanche il est structurellement sur-contraint, car il est constitué de 3 équations et de 2 variables. La raison de cette différence est que 2 de ses équations sont A-redondantes (et redondantes) par rapport à la première : l'équation 2 (resp. 3) est égale à 2 (resp. 3) fois la première.*

Exemple 6 Sous-contraint, structurellement bien-contraint
Le CSP $(X = \{x, y\}, D = \{D_x\}, C = \{x - y = 1, 2x - 2y = 2\})$ *est sous-contraint; en effet, il possède un ensemble indénombrable de solutions. En revanche il est structurellement bien-contraint, car il est constitué de 2 équations et 2 variables. La raison de cette différence est que la seconde équation est A-redondante (et redondante) : elle est égale à 2 fois la première.*

1.4.3 Analyse de GCSP

Dans cette thèse, nous nous intéressons aux cas particuliers de CSP continus que sont les GCSP. Un GCSP est souvent constitué de contraintes géométriques indépendantes du repère global de l'espace géométrique considéré[3]. Lorsque tel est le cas, toute solution du GCSP peut être déplacée dans l'espace géométrique, c-a-d. que l'application d'un déplacement (rotation+translation) à une solution d'un GCSP est toujours une solution du GCSP. Un tel GCSP peut être résolu complètement s'il est bien-contraint lorsqu'il est fixé par rapport au repère global. On dit aussi qu'il est bien-contraint dans un repère local, pouvant être déplacé par rapport au repère global. Cette propriété a pour nom *rigidité* et nous la présentons en détail au chapitre 2. Nous verrons en fin de ce chapitre que la seule caractérisation générale et polynomiale de la rigidité est une caractérisation structurelle approchée qui comporte les mêmes failles que la détermination structurelle du nombre de solutions d'un CSP.

[3]Tout GCSP peut d'ailleurs se ramener à ce cas : il suffit d'introduire, comme le proposent Lamure et Michelucci [Lamure and Michelucci, 1998], un objet repère sur lequel on reporte toutes les contraintes dépendantes du repère global ; on obtient ainsi un GCSP dont les contraintes sont indépendantes du repère global.

Chapitre 2

Rigidité

Sommaire

Ce chapitre introduit le concept de *rigidité*. La notion physique de rigidité est assez intuitive : un assemblage d'objets est rigide s'il est *indéformable*, c-a-d. que ses objets n'admettent aucun mouvement relativement les uns aux autres. On oublie souvent une autre propriété des systèmes rigides : un système rigide doit aussi être *déplaçable*, c-a-d. qu'on doit pouvoir appliquer tout mouvement à l'ensemble de ses objets. Cette seconde condition est généralement vérifiée tant que les contraintes géométriques mises en jeu sont indépendantes du référentiel de l'espace géométrique considéré. Vérifier la première condition est par contre plus difficile : c'est sur cette tâche que se sont concentrés l'ensemble des travaux menés sur la notion de rigidité.

La notion de rigidité a été principalement étudiée par la communauté de *topologie structurale* (un tour d'horizon des travaux sur la rigidité dans cette communauté est proposé dans [Graver, 2002]). L'intérêt pour cette notion remonte à la conjecture d'Euler en 1766 : "Une figure fermée de l'espace n'admet pas de déformation tant qu'elle n'est pas disjointe". Cette notion a cependant des incidences particulières en *théorie des mécanismes* où la distinction entre une structure (rigide) et un mécanisme (mobile) repose sur la détermination du degré de mobilité d'un mécanisme. Enfin, en *conception assistée par ordinateur* (CAO) [Owen, 1991; Kramer, 1992; Bouma *et al.*, 1995; Hoffmann *et al.*, 1997], ces notions ont été utilisées pour proposer des méthodes efficaces pour la résolution de GCSP qui sont présentées au chapitre 4.

Nous présentons d'abord les résultats obtenus par la communauté de topologie structurale, qui permettent d'introduire les différentes formes de rigidité. Nous proposons ensuite un aperçu des problèmes traités en théorie des mécanismes et l'intérêt de la notion de rigidité dans cette communauté. Enfin, nous terminons par la notion de *rigidité structurelle* utilisée en CAO, qui est une extension de la *rigidité générique* en topologie structurale et de la *formule de Grübler* en théorie des mécanismes.

2.1 Topologie structurale

La théorie de la rigidité, principal sujet d'étude de la communauté de topologie structurale [Whiteley, 1998; Graver, 2002], est d'origine mathématique (géométrie, algèbre) et exploite certains résultats de la théorie des graphes. Le but de cette théorie est de fournir des outils pour l'étude des propriétés des structures 2D et 3D apparaissant dans des constructions humaines. Plus particulièrement, la question à laquelle cherche à répondre cette théorie est : "Une construction donnée sera-t-elle suffisamment résistante pour supporter la charge qui lui sera appliquée ?". Dans sa recherche d'une réponse à cette question, la théorie de la rigidité simplifie le problème de plusieurs façons :

- les propriétés physiques des matériaux utilisés dans la construction ne sont pas prises en compte,
- les constructions sont réduites à leur squelette, à savoir des barres et des jointures.

Les théorèmes et propriétés que nous présentons dans cette section sont tous issus de la théorie de la rigidité, et sont, pour la plupart, regroupés, démontrés et expliqués dans l'ouvrage [Graver, 2002].

2.1.1 Systèmes à barres

Les GCSP considérés en topologie structurale sont principalement des *systèmes à barres*. Un système à barres est constitué d'un ensemble de barres (segments) de longueurs données jointes par leurs extrémités. Un tel système correspond à un GCSP constitué uniquement de points et de contraintes de distance : les barres de longueurs données deviennent des contraintes de distances entre leurs points extrémités. Plus formellement :

Définition 13 Système à barres
Un **système à barres** *en dimension d se définit par un triplet* (V, B, p) *où :*
- (V, B) *est un graphe de sommets* V *et d'arêtes* B *; chaque sommet représente un point, et chaque arête, appelée barre, représente une contrainte de distance entre les 2 points correspondant aux sommets qu'elle relie.*
- p *est une fonction des sommets vers l'espace euclidien,* $p : V \rightarrow \mathbb{R}^d$.

La fonction p d'un système à barres correspond à la notion de *configuration* d'un objet géométrique (cf. définition 4, page 13) : p associe à chaque sommet/point une configuration dans l'espace E.

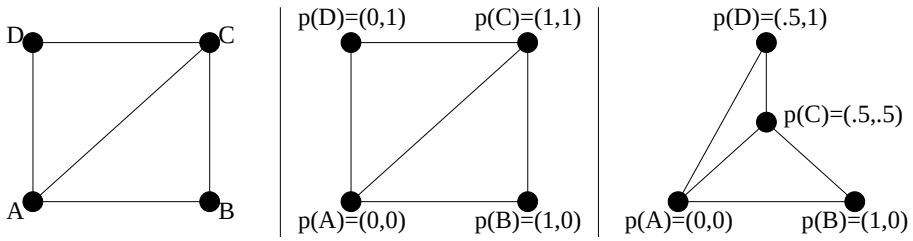

FIG. 2.1 – Deux systèmes à barres partageant le même squelette

La figure 2.1 représente un système à barres en 2D constitué de 4 points (A,B,C et D) et 5 barres. La partie gauche présente le graphe (V,B) de ce système. La partie centrale présente une configuration de ce système où les points A, B, C et D prennent respectivement les positions (0,0), (1,0), (1,1) et (0,1). La partie droite représente une configuration de ce système où les points A, B, C et D prennent respectivement les positions (0,0), (1,0), (0.5,0.5) et (0.5,1).

La définition des systèmes à barres donne un aperçu de leur double nature :
- L'aspect *structurel* d'un système à barres est représenté par son graphe (V, B) ; en effet, des systèmes à barres vérifiant des distances différentes entre les mêmes points partagent le même graphe (V, B). On dit qu'ils ont la même structure, ou même squelette.
- L'aspect *géométrique* d'un système à barres est représenté par la fonction p qui définit les positions des points ($p(A)$ est le point de l'espace E qui correspond au sommet A du graphe (V, B)) ainsi que les distances entre ces points (la longueur de la barre $(A, C) \in B$ est[1] $|p(A)p(C)|$).

[1] On utilise la notation $|AC|$ pour représenter la norme du vecteur $\overrightarrow{AC}$.

Nous verrons à la section 2.3, page 50, que cette double nature n'est pas spécifique aux systèmes à barres mais peut aussi être définie pour des GCSP quelconques.

2.1.2 Rigidité

La rigidité d'un système à barres se définit en fonction des mouvements qu'il admet. On distingue deux types de mouvements possibles :

- Une *déformation* est un mouvement qui préserve les distances entre les sommets liés par une arête mais modifie les distances entre les points qui ne sont pas liés par une arête.
- Un *déplacement* est un mouvement qui préserve les distances entre tous les points d'un système à barres.

Plus formellement :

Définition 14 Mouvements, déformations et déplacements
Un **mouvement** *d'un système à barres (V, B, p) en dimension d est un ensemble M contenant une fonction* continue et différenciable $m_i : [0,1] \to \mathbb{R}^d$ *pour chaque sommet $v_i \in V$ qui vérifie $m_i(0) = p(v_i)$ et $|m_i(t)m_j(t)| = |p(v_i)p(v_j)|$ pour tout $t \in [0,1]$ et tout $(v_i, v_j) \in B$.*

Un mouvement M est une **déformation** *s'il existe $t \in [0,1]$, $v_i \in V$ et $v_j \in V$ tel que $|m_i(t)m_j(t)| \neq |p(v_i)p(v_j)|$.*

Un mouvement M est un **déplacement** *si, pour tout $t \in [0,1]$, tout $v_i \in V$ et tout $v_j \in V$, $|m_i(t)m_j(t)| = |p(v_i)p(v_j)|$.*

Par définition, tout mouvement doit vérifier continuellement les contraintes de distance du système à barres : un mouvement M associe toujours la même longueur aux barres du système. La différence entre une déformation et un déplacement se trouve donc dans la distance entre les sommets/points qui ne sont pas liés par une arête : une déformation modifie ces valeurs, un déplacement ne les modifie pas.

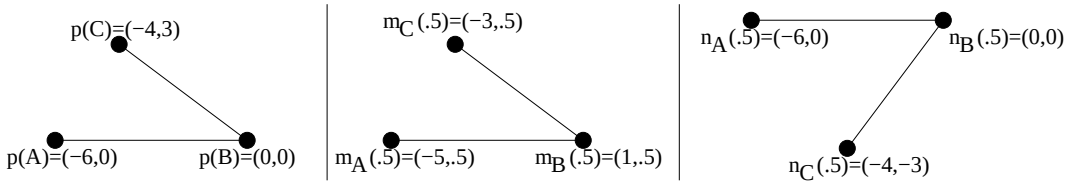

FIG. 2.2 – Différents types de mouvements admis par un système à barres

La figure 2.2 présente un système à barres en 2D constitué de 3 points A, B et C et 2 segments AB et BC. La configuration p présentée sur la figure de gauche associe chaque sommet aux points $p(A) = (-6, 0)$, $p(B) = (0, 0)$ et $p(C) = (-4, 3)$. La figure centrale représente un mouvement $M = \{m_A, m_B, m_C\}$ de la figure de gauche où :

$m_A(t) = (-6+2t, t)$
$m_B(t) = (2t, t)$
$m_C(t) = (-4+2t, 3+t)$

On peut vérifier que M est bien un mouvement car $m_A(0) = p(A)$, $m_B(0) = p(B)$, $m_C(0) = p(C)$ et, pour tout $t \in [0,1]$:

$|m_A(t)m_B(t)| = \sqrt{((-6+2t)-2t)^2+(t-t)^2} = 6$
$|m_B(t)m_C(t)| = \sqrt{(2t-(-4+2t))^2+(t-(3+t))^2} = 5$

Enfin, on peut vérifier que le mouvement M est un déplacement car, pour tout $t \in [0,1]$:
$|m_A(t)m_C(t)| = \sqrt{((-6+2t)-(-4+2t))^2+(t-(3+t))^2} = \sqrt{13}$

La partie droite de la figure 2.2 présente un autre mouvement $N = \{n_A, n_B, n_C\}$ de la figure de gauche où :

$n_A(t) = (-6, 0)$
$n_B(t) = (0, 0)$
$n_C(t) = (-4\cos t\pi - 3\sin t\pi, -4\sin t\pi + 3\cos t\pi)$

On peut vérifier que N est bien un mouvement pour la configuration p car $n_A(0) = p(A)$, $n_B(0) = p(B)$, $n_C(0) = p(B)$ et, pour tout $t \in [0,1]$:

$|n_A(t)n_B(t)| = 6$
$|n_B(t)n_C(t)| = \sqrt{(-4\cos(t\pi) - 3\sin(t\pi))^2 + (-4\sin(t\pi) + 3\cos(t\pi))^2} = 5$

Cependant, N est une déformation de la configuration p car :

$|n_A(0)n_C(0)| = \sqrt{(-6+4)^2+(0-3)^2} = \sqrt{13}$
$|n_A(1)n_C(1)| = \sqrt{(-6-4)^2+(0+3)^2} = \sqrt{109}$

C'est à dire qu'il existe des valeurs de t telles que $|p(A)p(C)| \neq |n_A(t)n_C(t)|$.

La notion de rigidité est liée à la notion de mouvement. En effet, la définition intuitive de rigidité proposée dans l'introduction de ce chapitre indiquait qu'un système rigide est indéformable et déplaçable. Les systèmes à barres se limitent à des contraintes de distance entre points et sont donc toujours déplaçables. Cela signifie donc que le système sera rigide s'il n'admet aucune déformation. Plus formellement :

Définition 15 Rigidité
Un système à barres (V, B, p) est **rigide** *si l'ensemble des mouvements qu'il admet ne contient aucune déformation.*

Déterminer si un système à barres est rigide revient, selon cette définition, à caractériser l'ensemble des mouvements qu'admet ce système.

Caractérisation algébrique de la rigidité

Un système à barres se ramène à un système d'équations quadratiques, une contrainte de distance étant représentée par une somme de différences de coordonnées au carré (par exemple, le système *Ponts*, page 2). Déterminer si le système est rigide consiste à vérifier que la distance entre toute paire de points est une conséquence du système d'équations. L'approche algébrique pour répondre à cette question consiste à utiliser les méthodes de preuve de théorème.

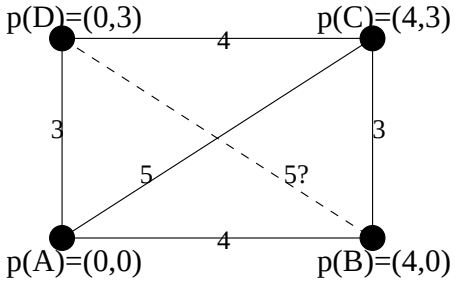

FIG. 2.3 – Rigidité algébrique d'un système à barres

Illustrons cette méthode sur le système à barres en 2D présenté à la figure 2.3. Ce système est constitué de 4 points A, B, C et D et de 5 contraintes de distances. On lui associe la configuration p : $\{A = (0,0), B = (4,0), C = (4,3), D = (0,3)\}$. On cherche à vérifier que, pour chaque paire de points, la distance entre ces points est une déduction du système des 5 contraintes de distance. Ceci est trivialement vérifié pour les paires AB, AC, AD, BC et CD puisqu'elles correspondent aux 5 contraintes de distance. La paire BD a une distance mesurée de 5 pour la configuration p considérée. On cherche donc à montrer que l'équation :

$(x_B - x_D)^2 + (y_B - y_D)^2 = 25$

est une conséquence du système d'équations correspondant à ce système à barres :

$$\begin{aligned}
(x_A - x_B)^2 + (y_A - y_B)^2 &= 16 \\
(x_A - x_C)^2 + (y_A - y_C)^2 &= 25 \\
(x_A - x_D)^2 + (y_A - y_D)^2 &= 9 \\
(x_B - x_C)^2 + (y_B - y_C)^2 &= 9 \\
(x_C - x_D)^2 + (y_C - y_D)^2 &= 16
\end{aligned}$$

La résolution symbolique de ce système nous permet d'identifier les deux valeurs possibles pour la distance entre les points B et D : 5 et $\frac{35}{25}$. La distance correspondant à la configuration p est 5 et il n'existe pas de mouvement permettant de faire passer cette distance de 5 à $\frac{35}{25}$: en effet, il n'existe pas de fonction continue passant uniquement par un ensemble fini de valeurs ; Le système à barres est donc rigide.

De façon générale, si l'ensemble des solutions pour une distance donnée est fini, le système est rigide. Cette méthode est cependant coûteuse, puisqu'il faut vérifier que toutes les distances sont des conséquences du système à barres en configuration p. La difficulté de cette méthode a amené au développement d'une forme plus simple de rigidité : la rigidité de premier ordre.

2.1.3 Rigidité de premier ordre

Un mouvement d'un système à barres étant un ensemble de fonctions continues et *différenciables*, on peut dériver la fonction du mouvement associée à chaque point du système selon le paramètre t de ce mouvement ; le mouvement dérivé est un mouvement au

premier ordre, ou *mouvement infinitésimal*. A partir de cette constatation, on définit les mouvements infinitésimaux d'un système à barres :

Définition 16 Mouvement infinitésimal
Soit (V, B, p) un système à barres en dimension d. Un **mouvement infinitésimal** *de ce système à barres est une fonction $q : V \to \mathbb{R}^d$ qui associe à chaque sommet v un vecteur $q(v)$ tel que le produit scalaire $(p(a) - p(b)).(q(a) - q(b)) = 0$ pour tout $(a, b) \in B$.*

Un mouvement infinitésimal est une **déformation infinitésimale** *si il existe $a \in V$ et $b \in V$ tels que le produit scalaire $(p(a) - p(b)).(q(a) - q(b)) \neq 0$.*

Un mouvement infinitésimal est un **déplacement infinitésimal** *si pour tout $a \in V$ et tout $b \in V$, le produit scalaire $(p(a) - p(b)).(q(a) - q(b)) = 0$.*

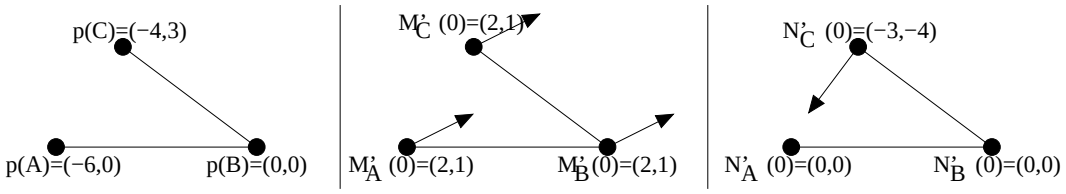

FIG. 2.4 – Illustrations de mouvements infinitésimaux

La figure 2.4 présente le système à barres constitué de 3 points A, B et C et 2 segments AB et BC déjà présenté à la figure 2.2, page 28. Nous reprenons la configuration p déjà proposée pour ce système en partie gauche de la figure. Les parties centrales et droites de la figure représentent les mouvements infinitésimaux M' et N' correspondants aux mouvements M et N déjà étudiés pour ce système à la figure 2.2, page 28.

Les mouvements infinitésimaux M' et N' de la configuration p se définissent par dérivation respectivement du mouvement M et du mouvement N en $t = 0$ (car $M(0) = N(O) = p$)) :

$$\begin{array}{lcl}
M'_A(0) & = & (2,1)|_{t=0} \quad = \quad (2,1) \\
M'_B(0) & = & (2,1)|_{t=0} \quad = \quad (2,1) \\
M'_C(0) & = & (2,1)|_{t=0} \quad = \quad (2,1) \\
N'_A(0) & = & (0,0)|_{t=0} \quad = \quad (0,0) \\
N'_B(0) & = & (0,0)|_{t=0} \quad = \quad (0,0) \\
N'_C(0) & = & (4\pi \sin t\pi - 3\pi \cos t\pi, -4\pi \cos t\pi - 3\pi \sin t\pi)|_{t=0} \quad = \quad (-3,-4)
\end{array}$$

Ces mouvements infinitésimaux sont représentés sous forme de vecteurs sur la figure 2.4. On peut vérifier que M' est un déplacement infinitésimal, alors que N' est une déformation infinitésimale :

$$\begin{array}{lcl}
(p(A) - p(C)).(M'_A - M'_C) & = & (-2,-3).(0,0) \quad = \quad 0 \\
(p(A) - p(C)).(N'_A - N'_C) & = & (-2,-3).(3,4) \quad = \quad -18
\end{array}$$

Le mouvement M (resp. N) correspond donc bien à un mouvement infinitésimal M' (resp. N') qui vérifie les mêmes propriétés : c'est un déplacement (resp. une déformation).

La rigidité de premier ordre se définit relativement à la notion de mouvement infinitésimal de la même façon que la rigidité se définit relativement aux mouvements :

Définition 17 Rigidité de premier ordre
Un système à barres (V, B, p) *est* **rigide au premier ordre**, *si l'ensemble des mouvements infinitésimaux qu'il admet ne contient aucune déformation infinitésimale.*

Vérifier si un système à barres est rigide au premier ordre revient donc à vérifier qu'aucun des mouvements infinitésimaux qu'il admet n'est une déformation.

Caractérisation algébrique de la rigidité de premier ordre

Vérifier qu'un système à barres est rigide au premier ordre se fait de façon semblable à vérifier qu'il est rigide : il faut vérifier que tout mouvement infinitésimal est un déplacement, c-a-d. que les produits scalaires associés aux paires de points qui ne sont pas liés par des barres sont des conséquences du système des produits scalaires correspondants aux barres du système. Pour ce faire, l'approche algébrique consiste à appliquer un mouvement infinitésimal générique, c-a-d. entièrement paramétré, au système à barres et à utiliser les méthodes de preuves de théorème.

Illustrons cette méthode sur le système à barres en 2D présenté à la figure 2.5. Ce système, déjà utilisé pour illustrer la caractérisation algébrique de la rigidité (cf. section 2.1.2, page 29), est constitué de 4 points A, B, C et D et de 5 barres AB, AC, AD, BC et CD. On lui associe la configuration $p : \{A \to (0,0), B \to (4,0), C \to (4,3), D \to (0,3)\}$ et le mouvement infinitésimal $M = \{A \to (u_A, v_A), B \to (u_B, v_B), C \to (u_C, v_C), D \to (u_D, v_D)\}$. Par définition, ce mouvement infinitésimal vérifie un produit scalaire nul pour chaque barre du système :

$$\begin{aligned}
(p(A)-p(B)).(M(A)-M(B)) &= (-4,0).(u_A-u_B, v_A-v_B)\\
&= -4u_A+4u_B\\
&= 0\\
(p(A)-p(C)).(M(A)-M(C)) &= (-4,-3).(u_A-u_C, v_A-v_C)\\
&= -4u_A+4u_C-3v_A+3v_C\\
&= 0\\
(p(A)-p(D)).(M(A)-M(D)) &= (0,-3).(u_A-u_D, v_A-v_D)\\
&= -3v_A+3v_D\\
&= 0\\
(p(B)-p(C)).(M(B)-M(C)) &= (0,-3).(u_B-u_C, v_B-v_C)\\
&= -3v_B+3v_C\\
&= 0\\
(p(C)-p(D)).(M(C)-M(D)) &= (4,0).(u_C-u_D, v_C-v_D)\\
&= 4u_C-4u_D\\
&= 0
\end{aligned}$$

On cherche à vérifier que, pour chaque paire de points qui n'est pas liée par une barre, le mouvement infinitésimal M n'induit pas de déformation, c-a-d. que l'équation suivante est une conséquence du système précédent :

$$\begin{aligned}
(p(B)-p(D)).(M(B)-M(D)) &= (4,-3).(u_B-u_D, v_B-v_D)\\
&= 4u_B-4u_D-3v_D+3v_D\\
&= 0
\end{aligned}$$

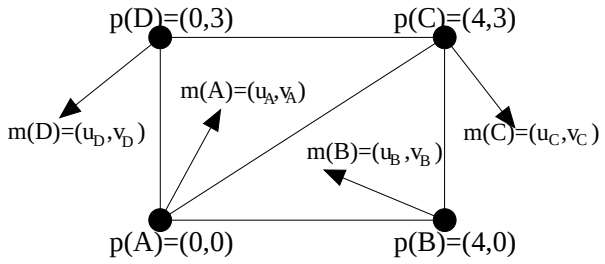

FIG. 2.5 – Vérification algébrique de la rigidité infinitésimale d'un système à barres

Ce système étant linéaire, une méthode de pivot de Gauss permet de vérifier que cette équation est bien une conséquence des 5 premières en temps polynomial. La méthode employée est donc semblable à celle utilisée pour la caractérisation algébrique de la rigidité, à la différence que le système d'équations considéré est linéaire dans le cas de la rigidité de premier ordre, alors qu'il est quadratique dans le cas de la rigidité.

La rigidité de premier ordre semble donc régler le problème de la complexité de la vérification de la rigidité d'un système à barres. Nous allons voir que cela n'est pas tout à fait le cas en nous intéressant au lien entre rigidité et rigidité de premier ordre : ces deux propriétés ne sont équivalentes que sous certaines restrictions.

2.1.4 Lien entre rigidité et rigidité de premier ordre

L'intérêt de la rigidité de premier ordre réside dans la proposition suivante :

Proposition 1 Rigidité de premier ordre et rigidité
Si un système à barres est rigide au premier ordre, alors il est rigide.

Il est aisé de se convaincre de la contraposée de cette proposition : si un système à barres n'est pas rigide, alors il dispose d'une déformation ; cette déformation peut être dérivée et produit alors une déformation infinitésimal ; le système à barres n'est donc pas rigide au premier ordre. Pourtant, cette proposition est fausse : la rigidité de premier ordre n'implique la rigidité *que pour des systèmes à barres en configurations génériques*, comme nous allons le voir.

La proposition inverse est toute aussi fausse : un système peut être rigide sans être rigide au premier ordre. Considérons l'exemple de système à barres donné à la figure 2.6.

Le système à barres représenté dans cette figure est constitué de 5 points A, B, C, D et E et de 7 segments AB, AC, AD, BC, BD, CE et DE. Ce système est rigide : aucun point n'admet de mouvement indépendant. Cependant, ce système n'est pas rigide au premier ordre : considérons le mouvement infinitésimal qui associe à chaque point A, B, C et D un vecteur nul, et au point E un vecteur v_E non nul et perpendiculaire aux segments CE et DE. Un tel vecteur est représenté sur la figure 2.6. On peut montrer que ce mouvement infinitésimal est une déformation infinitésimale en calculant le produit scalaire $\overrightarrow{AE}.\overrightarrow{v_E}$ qui est non nul.

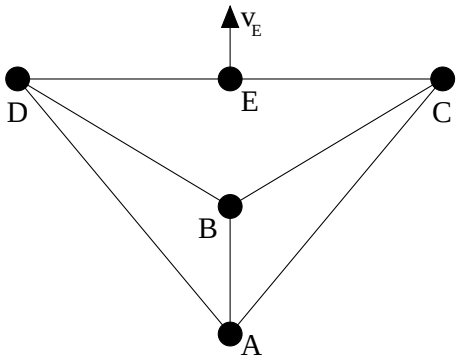

FIG. 2.6 – Différence entre rigidité et rigidité de premier ordre

L'interprétation physique de ce phénomène est que, malgré sa rigidité, ce système à barres comporte une *faiblesse* au niveau du point E : le système est rigide, mais une pression appliquée au point E peut déformer, voire casser la structure physique correspondante. Or, du point de vue géométrique, ce système est rigide et c'est la propriété qui nous importe (cf. section 2.3.1, page 51). De plus, la rigidité de premier ordre ne peut malheureusement être caractérisée par l'approche algébrique que nous avons présentée pour des systèmes à barres en *configurations singulières*, c-a-d. des configurations présentant une dépendance algébrique entre les coordonnées des différents points. Une configuration qui n'est pas singulière est dite *générique*. Avant de donner une définition formelle de généricité, nous proposons d'illustrer le problème de la rigidité de premier ordre sur un exemple.

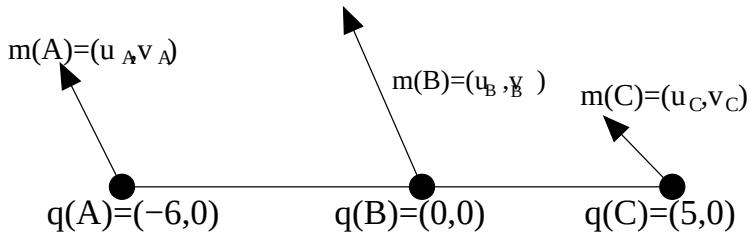

FIG. 2.7 – Rigidité de premier ordre et configurations singulières

Considérons le système à barres représenté à la figure 2.7. Ce système composé de 3 points A, B et C et de 2 barres AB et BC possède la même structure que le système à barres présenté à la figure 2.2, page 28, mais dans une configuration différente : $q = \{A \to (-6,0), B \to (0,0), C \to (5,0)\}$. Rappelons que ce système admet des déformations et n'est donc pas rigide (cf. section 2.1.2, page 28). Considérons à présent le mouvement infinitésimal $M = \{A \to (u_A, v_A), B \to (u_B, v_B), C \to (u_C, v_C)\}$. Ce mouvement induit le système de produits scalaires suivants :

$$\begin{aligned}(q(A)-q(B)).(M(A)-M(B)) &= (-6,0).(u_A-u_B,v_A-v_B)\\ &= -6u_A+6u_B\\ &= 0\\ (q(B)-q(C)).(M(B)-M(C)) &= (-5,0).(u_B-u_C,v_B-v_C)\\ &= -5u_B+5u_C\\ &= 0\end{aligned}$$

et on cherche à savoir si l'équation

$(q(A)-q(C)).(M(A)-M(C)) = (-1,0).(u_A-u_C,v_A-v_C) = -u_A+u_C = 0$

est une conséquence des 2 premières équations. C'est bien le cas puisque les 2 équations précédentes impliquent $u_A = u_B$ et $u_B = u_C$. On peut donc conclure que ce système est rigide au premier ordre, et par le théorème 1 (page 33), que le système est donc rigide, ce qui est faux ! Ce résultat aberrant provient de la configuration singulière q dans laquelle les trois points A, B et C sont alignés.

Configurations génériques et générales

En fait, la rigidité de premier ordre et la rigidité correspondent lorsque la configuration d'un système à barres est *générique*, c-a-d. que le système ne contient aucune singularité. Intuitivement, une configuration est générique si les coordonnées des points dans cette configuration sont algébriquement indépendantes. Avant de fournir une définition formelle de la généricité, nous introduisons la notion de *configuration générale*, une simplification de la notion de configuration générique.

Définition 18 Configuration générale
Un système à barres (V,B,p) est en configuration p générale si et seulement si chaque sous-ensemble de ses points est contenu dans un espace géométrique de dimension maximale.

Un ensemble de n points définit au plus un espace à $n-1$ dimensions. ils sont en configuration générale si et seulement si ils définissent exactement un espace à $n-1$ dimension. Si tel est le cas, les coordonnées de ces points sont linéairement indépendantes, c-a-d. que les $n-1$ vecteurs constructibles en prenant pour origine l'un des n points, et pour arrivée chacun des $n-1$ points restants sont indépendants et constituent une base de l'espace vectoriel de dimension $n-1$. Plus intuitivement :

- en 2D et 3D, 2 points sont généralement contenus dans un espace de dimension 1 (une droite). Ils sont en configuration singulière lorsqu'ils sont confondus (dimension 0).
- en 2D et 3D, 3 points sont généralement contenus dans un espace de dimension 2 (un plan). Ils sont en configuration singulière s'ils sont confondus (dimension 0) ou alignés (dimension 1).
- en 3D, 4 points sont généralement contenus dans une espace de dimension 3. Ils sont en configuration singulière s'ils sont confondus (dimension 0), alignés (dimension 1) ou coplanaires (dimension 2).
- ...

Notons que le système à barres considéré étant plongé dans un espace géométrique de dimension d, les points qui le composent définissent au plus un espace de dimension d. Ainsi, 4 points coplanaires, non-alignés, non-confondus, en 2D sont en configuration générale puisqu'ils définissent un espace de dimension maximale vis à vis de l'espace géométrique considéré. Remarquons aussi qu'une configuration pour un ensemble de points est générale indépendamment d'un quelconque système à barres : une configuration d'un ensemble de points est générale ou pas en fonction des coordonnées des points et non des barres entre ces points.

La notion de configuration générique est plus difficile à définir ; intuitivement, une configuration est générique si les coordonnées des points dans cette configuration sont algébriquement indépendantes ; c'est en cela que les configurations générales sont une simplification des configurations génériques : général = indépendance linéaire, générique = indépendance algébrique.

Une définition plus formelle de configuration générique repose sur la notion de *matrice de rigidité* qui est liée à la rigidité de premier ordre :

Définition 19 Matrice de rigidité
La **matrice de rigidité** *d'un système à barres (V, B, p) affecté par un mouvement infinitésimal q est la jacobienne du système de produits scalaires $(p(a) - p(b)).(q(a) - q(b)) = 0$ pour tout $(a, b) \in B$ où les $q(v)$ sont les variables.*

Reprenons pour exemple le système à barres S présenté à la figure 2.2, page 28. Les barres de ce système induisent le système de produits scalaires déjà présenté à la section 2.1.4, page 33 :

$$\begin{aligned}
eq1 : (p(A) - p(B)).(M(A) - M(B)) &= (-6, 0).(u_A - u_B, v_A - v_B) \\
&= -6u_A + 6u_B \\
&= 0 \\
eq2 : (p(B) - p(C)).(M(B) - M(C)) &= (-5, 0).(u_B - u_C, v_B - v_C) \\
&= -5u_B + 5u_C \\
&= 0
\end{aligned}$$

Ce qui produit la matrice de rigidité M(S) suivante :

$$\begin{array}{c} \\ eq1 \\ eq2 \end{array} \begin{array}{c} \begin{array}{cccccc} u_A & v_A & u_B & v_B & u_C & v_C \end{array} \\ \begin{pmatrix} -6 & 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & -5 & 0 & 5 & 0 \end{pmatrix} \end{array}$$

Cette matrice permet de caractériser tous les mouvements infinitésimaux admissibles par un système à barres. En effet, le système des produits scalaires utilisé pour caractériser algébriquement un mouvement infinitésimal q est un système d'équations linéaires de la forme $A.x = 0$ dont la matrice A est la matrice de rigidité et le vecteur $x \in \mathbb{R}^{d|V|}$ contient les coordonnées de tous les vecteurs composants le mouvement infinitésimal q (c-a-d. $x = (q(v_1), q(v_2), ..., q(v_{|V|}))$).

La matrice de rigidité nous permet de définir la notion de configuration générique comme suit :

Définition 20 Configuration générique
Soit $\mathrm{M}(S)$ *la matrice de rigidité du système à barres* $S = (V, K, p)$ *en dimension* $d = 2$ *ou* 3 *où* (V, K) *est le graphe complet sur les sommets de* V *et* $|V| > d$. *Soit* $\mathbb{S}$ *l'union des courbes de* $\mathbb{R}^{d|V|}$ *solutions du système d'équations polynomiales obtenues en mettant à 0 le déterminant de chaque mineur (non identiquement nul) de* $\mathrm{M}(S)$. *Alors* p *est une* **configuration générique** *si, représenté sous forme de vecteur de* $\mathbb{R}^{d|V|}$ *(*$p = (p(v_1), p(v_2), ..., p(v_{|V|}))$*), il n'appartient pas à* $\mathbb{S}$.

Rappelons que l'intuition de configuration générique est la suivante : une configuration est générique si les coordonnées qu'elle donne aux points sont *algébriquement indépendantes*[2]. Le déterminant d'un mineur de la matrice de rigidité définit une équation algébrique $e(x) = 0$ s'il n'est pas identiquement nul. Si une configuration p est solution de e, alors la configuration n'est pas algébriquement indépendante, et donc pas générique.

A nouveau, une configuration générique est définie indépendamment d'un système à barres particulier : une configuration générique p pour un système à barres (V, B, p) sera aussi générique pour un système à barres (V, B', p) puisque ces deux systèmes à barres sont définis sur le même ensemble de sommets V, c'est à dire qu'il ont le même graphe complété (V, K). Autrement dit, une configuration est générique si les coordonnées qu'elle associe aux points sont algébriquement indépendantes, indépendamment des barres entre ces points.

Le théorème suivant, démontré dans [Graver, 2002], explique l'intérêt des configurations génériques :

Théorème 1 Rigidité et rigidité de premier ordre
En 2D et 3D, rigidité et rigidité de premier ordre sont équivalentes pour tout système à barres en configuration générique.

Les configurations générales sont aussi intéressantes comme nous le verrons dans la suite. D'une part elles sont plus facilement caractérisables que les configurations génériques (une dépendance linéaire est plus simple à identifier qu'une dépendance algébrique) ; d'autre part, elles peuvent être généralisées facilement à l'étude de la rigidité de GCSP quelconques. Enfin, elle donnent un critère simple pour la détection de certaines singularités, car une configuration qui n'est pas générale n'est pas générique. En effet, la notion de configuration générique est une notion plus forte que celle de configuration générale :

Théorème 2 Configuration générale et générique
Toute configuration générique est une configuration générale.

[2]En réalité, l'indépendance algébrique est une condition *suffisante* pour qu'une configuration soit générique, mais non nécessaire : une configuration générique ne doit simplement pas annuler le déterminant de la matrice de rigidité, ce qui représente seulement un nombre fini de dépendances algébriques interdites, et non pas l'interdiction de toute dépendance algébrique.

La figure 2.8 présente un exemple de système à barres en 2D en configuration générale mais non générique [Connelly, 2002]. La configuration de système à barres place les 6 points sur une même ellipse dont l'expression analytique représente une dépendance quadratique entre les coordonnées de ces points. Ce système est bien rigide, mais pas rigide au premier ordre : le produit scalaire correspondant à la paire de points (A, C) n'est pas une conséquence du système des produits scalaires des barres de ce système à barres.

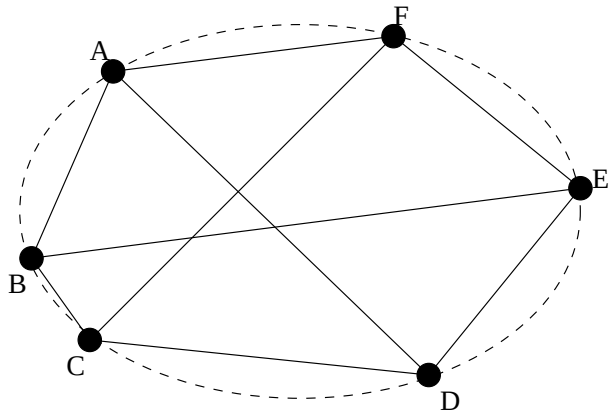

FIG. 2.8 – Configuration générale non-générique d'un système à barres en 2D

2.1.5 Degrés de liberté d'un système à barres

Le nombre de mouvements indépendants admis par un système à barres S est appelé *nombre de degrés de liberté* de ce système. Il est égal à la dimension de l'espace $\mathbb{M}_S$ des mouvements admis par le système S. On distingue deux types de degrés de liberté : ceux basés sur les mouvements et ceux basés sur les mouvements infinitésimaux. Comme nous allons considérer les configurations génériques des systèmes à barres, il n'est pas nécessaire de distinguer ces 2 types de degrés de liberté (cf. théorème 1, page 37).

Le nombre de degrés de liberté d'un système à barres S peut se calculer à partir du nombre de points et de contraintes qu'il comporte :

- Tout point en dimension d confère d degrés de liberté supplémentaires au système ; en effet, en espace de dimension d, un point libre peut être déplacé selon chaque dimension de l'espace, c-a-d. qu'il admet les d translations indépendantes offertes par un espace de dimension d.
- Toute barre *indépendante* retire 1 degré de liberté au système ; en effet, un segment dispose de $2d - 1$ degrés de liberté en dimension d : il est complètement déterminé lorsque ses 2 extrémités sont positionnées ; la première peut être positionnée indifféremment (d translations), la seconde doit respecter la distance, c-a-d. qu'elle appartient à la sphère centrée sur la première extrémité et de rayon égal à la longueur du segment ($d - 1$ rotations).

Le nombre de degrés de liberté $DDL(S)$ d'un système à barres $S = (V, B, p)$ se calcule alors en sommant les degrés de liberté offerts par ses points et en retranchant à cette somme le nombre de barres *indépendantes* $B_i \subseteq B$ de S : $DDL(S) = d * |V| - |B_i|$.

Une barre $(a, b) \in B$ d'un système à barres $S = (V, B, p)$ est *indépendante* si elle n'est pas *R-redondante*. Cette barre est R-redondante si le produit scalaire qui lui est associé dans le calcul des mouvements infinitésimaux est une conséquence algébrique du système d'équations linéaires produit par les autres barres $B \setminus (a, b)$ du système S.

Notons que le terme de R-redondance ne correspond pas exactement à la définition de R-redondance donnée en section 1.4.2 (page 21) : généralement, une contrainte est dite R-redondante si son retrait ne change pas la rigidité du GCSP ; ici, la notion de R-redondance est relative à la rigidté de premier ordre. Cependant, il s'agit de la même propriété, raison pour laquelle nous employons le même terme.

Le nombre de degrés de liberté d'un système à barres donne une indication sur sa rigidité au premier ordre, comme indiqué par le théorème suivant proposé dans [Graver, 2002] :

Théorème 3 Degré de liberté et rigidité
Un système à barres (V, B, p) en configuration générale p en espace de dimension d et vérifiant $|V| > d$ est rigide au premier ordre (et rigide par le théorème 1, page 37) si et seulement si $DDL(S) = \frac{d(d+1)}{2}$.

En effet, un système est rigide au premier ordre s'il n'admet comme mouvements que les déplacements de l'espace qui le contient (cf. définition 17, page 32). Le lemme suivant établit le nombre de déplacements qu'admet un espace géométrique de dimension donnée :

Lemme 1 Nombre de déplacements indépendants
Un espace géométrique E de dimension d admet d translations indépendantes et $\frac{d(d-1)}{2}$ rotations indépendantes.

En effet, une translation dans un espace géométrique de dimension d se définit par un vecteur de $\mathbb{R}^d$. Toute base de l'espace vectoriel $\mathbb{R}^d$ est constituée de d vecteurs indépendants, il existe donc d translations indépendantes. Une rotation dans un espace géométrique de dimension d se définit par une paire de vecteurs indépendante de $\mathbb{R}^d \times \mathbb{R}^d$. Toute base de l'espace vectoriel $\mathbb{R}^d \times \mathbb{R}^d$ contient $\frac{d(d-1)}{2}$ paires de vecteurs générateurs.

Puisque l'espace euclidien de dimension d admet d translations et $\frac{d(d-1)}{2}$ rotations indépendantes, c-a-d. $d + \frac{d(d-1)}{2} = \frac{d(d+1)}{2}$ déplacements indépendants, et qu'un système rigide n'admet aucune déformation, tout système rigide admet au plus $\frac{d(d+1)}{2}$ degrés de liberté. De plus, un système à barres constitué d'au moins d points en espace de dimension d et en configuration générale p emplit tout l'espace géométrique considéré. Il admet alors forcément tous les déplacements indépendants de cet espace, c'est à dire qu'il possède au moins $\frac{d(d+1)}{2}$ degrés de liberté. Ceci prouve le théorème 3.

On en déduit le corollaire suivant :

Corollaire 1 *Un système à barres en dimension d qui admet au moins $\frac{d(d+1)}{2}+1$ degrés de liberté n'est pas rigide.*

En effet, étant donné le lemme 1, si un système à barres en dimension d admet plus de $\frac{d(d+1)}{2}$ degrés de liberté, au moins un de ces degrés de liberté correspond à un mouvement indépendant qui est une déformation.

Compter le nombre de degrés de liberté d'un système à barres en espace géométrique de dimension d donne donc une indication sur sa rigidité :

- si le système possède plus de $\frac{d(d+1)}{2}$, alors il n'est pas rigide (cf. corollaire 1),
- si le système possède exactement $\frac{d(d+1)}{2}$ degrés de liberté, alors il est rigide si sa configuration p est générale (cf. théorème 3).

Ainsi, si l'on s'intéresse à la rigidité d'un système à barres dans toutes ses configurations génériques, on peut se satisfaire d'un simple compte des degrés de liberté du système. Malheureusement, compter les degrés de liberté d'un système implique de déterminer quelles barres de ce système sont indépendantes, ce qui ne peut être caractérisé que par des méthodes algébriques.

La communauté de topologie structurale propose une forme de rigidité qui s'applique à toute configuration générique d'un système à barres qui se base sur la structure de ces systèmes ; c'est la *rigidité générique*.

2.1.6 Rigidité générique

La notion de rigidité générique est une dérivation de la notion de rigidité des systèmes à barres qui ne se base que sur la structure de tels systèmes, c-a-d. le graphe (V, B) des systèmes à barres (V, B, p). Cette notion est donc générique en ce sens qu'elle donne une indication de la rigidité d'un système indépendamment de sa configuration p. Elle trouve son intérêt dans le théorème suivant :

Théorème 4 [Gluck, 1975]
S'il existe une configuration générique p pour une structure (V, B) telle que le système à barres (V, B, p) est (resp. n'est pas) rigide au premier ordre, alors toute configuration générique appliquée à la structure (V, B) produit un système à barres rigide (resp. non rigide) au premier ordre.

On définit les concepts jumeaux de rigidité de graphe et de rigidité générique comme suit :

Définition 21 Rigidité de graphe et rigidité générique
Un **graphe** (V, B) *est* **rigide** *en dimension d, noté d-rigide, s'il existe une configuration générique p telle que le système à barres (V, B, p) en dimension d est rigide.*

Un système à barres (V, B, p) en espace E de dimension d est dit **génériquement rigide** *si et seulement si sa structure (V, B) est un graphe d-rigide.*

La définition de rigidité de graphe repose sur les théorèmes 4 et 1 (page 37). En effet, rigidité et rigidité de premier ordre correspondent pour toute configuration générique. De plus, tous les systèmes basés sur une même structure sont soit rigides pour toute configuration générique, soit rigides pour aucune configuration générique. Le théorème suivant, énoncé dans [Graver, 2002], lie rigidité générique et configuration générale :

Théorème 5 Rigidité générique et configuration générale
Si un système à barres (V, B, p) en configuration générale p est rigide au premier ordre, alors il est génériquement rigide.

Les caractérisations de rigidité que nous avons présentées jusqu'ici pourraient être utilisées pour caractériser les graphes rigides en dimension donnée, et ainsi les systèmes à barres génériquement rigides de la façon suivante :

- associer le graphe (V, B) du système à barres S étudié à une configuration générale p en dimension d.
- vérifier si le système à barres (V, B, p) résultant est rigide au premier ordre.
- si tel est le cas, le théorème 5 assure alors que le système à barres S initial est génériquement rigide en dimension d.

Cette caractérisation est malheureusement semi-déterministe : dans le cas où le système à barres (V, B, p) n'est pas rigide au premier ordre, on ne peut déterminer si le système à barres S est génériquement rigide ou pas. On pourrait envisager d'utiliser le même processus avec une configuration p générique au lieu de générale ; dans ce cas, c'est le théorème 4 qui permet de décider de façon complètement déterministe si S est génériquement rigide ou pas. Cependant, produire une configuration générique est plus difficile que produire une configuration générale : il faut générer des coordonnées de points algébriquement indépendantes.

La rigidité générique peut aussi être caractérisée par des propriétés structurelles, c-a-d. des propriétés du graphe décrivant sa structure, de façon complète en 2D.

2.1.7 Caractérisation structurelle de la rigidité générique

L'idée sur laquelle se base cette caractérisation structurelle est la suivante : si un graphe possède un sous-graphe d-rigide, alors il est lui-même d-rigide. Cette idée est formalisée dans le théorème suivant, proposé dans [Graver, 2002] :

Théorème 6 Rigidité et graphe d-isostatique
*En dimension $d = 1, 2$ ou 3, un graphe $G = (V, E)$ est d-rigide si et seulement si il possède un sous-graphe$G' = (V', E')$ d-*isostatique *tel que $V' = V$.*

Un graphe est d-isostatique s'il est d-rigide et minimal, c-a-d. qu'on ne peut lui retirer aucune arête sans qu'il ne soit alors plus d-rigide. Plus formellement :

Définition 22 Graphe d-isostatique
Un graphe (V, B) est d-isostatique si et seulement si (V, B) est un graphe d-rigide, et tout retrait d'une seule arête $b \in B$ produit un graphe $(V, B \setminus b)$ qui n'est pas d-rigide.

Un graphe d-isostatique est donc un graphe d-rigide minimal et ne comporte donc aucune arête *R-redondante* (cf. section 2.1.5, page 38). Les graphes isostatiques sont des graphes connexes ; en effet, si le graphe d'un système à barres n'est pas connexe, celui-ci ne peut être rigide : les sous-systèmes induits par les composantes connexes de ce graphe peuvent être déplacés indépendamment les uns des autres ; ces mouvements constituent des déformations du système global.

Les graphes 2-isostatiques ont été caractérisés au début du 20^e siècle ; c'est Henneberg [Henneberg, 1911] qui a le premier proposé une caractérisation *constructive* de ces graphes :

Théorème 7 de Henneberg
Un graphe (V, B) est 2-isostatique si et seulement si il peut être obtenu à partir d'une seule arête par une série de 2-extensions et 2-arête-séparation.

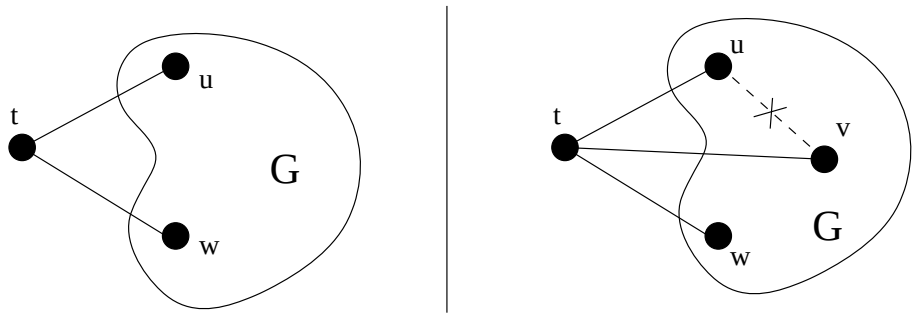

FIG. 2.9 – Exemple de 2-extension et de 2-arête-séparation

La figure 2.9 présente une 2-extension en partie gauche, et une 2-arête-séparation en partie droite. Une 2-extension d'un graphe $G = (V, B)$ consiste à ajouter un sommet t dans V ainsi que 2 arêtes (t, u) et (t, w) dans B, partant de 2 sommets distincts u et w. Une 2-arête-séparation d'un graphe $G = (V, B)$ consiste à supprimer une arête (u, v) du graphe G, et à insérer un sommet t ainsi que 3 arêtes (t, u), (t, v) et (t, w).

Cette caractérisation n'est malheureusement pas exploitable en pratique : le problème de l'identification d'une séquence de Henneberg pour la construction d'un graphe donné est NP-Complet.

Il a fallu attendre Laman en 1970 [Laman, 1970] pour qu'une caractérisation polynomiale des graphes 2-isostatiques soit trouvée :

Théorème 8 de Laman
Un graphe (V, B) est 2-isostatique si et seulement si :
- $|C| \leq 2|U| - 3$ *pour tout* $U \subseteq V$ *vérifiant* $|U| \geq 2$, $C \subseteq B$ *étant l'ensemble des arêtes du graphe (V, B) induit par l'ensemble de sommets U, et*
- $|B| = 2|V| - 3$.

Cette caractérisation est à rapprocher du concept de degré de liberté introduit à la section 2.1.5, page 38. En effet, en 2D, chaque point dispose de 2 degrés de liberté ; par ailleurs, chaque barre indépendante retire 1 degré de liberté. De plus, l'espace de dimension $d = 2$ dispose de $\frac{d(d+1)}{2} = 3$ déplacements indépendants (cf. lemme 1, page 39). Il résulte du théorème 3 (page 39) qu'un système à barres $S = (V, B)$ vérifiant $|B| = 2|V| - 3$ et dont les barres sont placées de façon non-R-redondante est génériquement rigide. Le compte global du théorème de Laman vise à vérifier que le système à barres possède le nombre d'arêtes minimal nécessaire afin d'être rigide, et le compte des barres dans chaque sous-système vise à vérifier que les barres du système sont placées de façon non-R-redondante.

Cette caractérisation a permis d'obtenir des algorithmes polynomiaux ($O(|V|^2)$ pour vérifier si un graphe est 2-isostatique [Sugihara, 1986; Lamure and Michelucci, 1998] [Imai, 1986; Hendrickson, 1992].

Crapo [Crapo, 1996] a proposé une autre caractérisation des graphes 2-isostatiques qui lui a permis d'obtenir un algorithme plus performant, mais toujours quadratique :

Théorème 9 de Crapo
Un graphe qui est l'union de 3 arbres tels que chaque sommet du graphe appartient à exactement 2 de ces arbres et chaque arête du graphe n'appartient qu'à un seul de ces arbres est un graphe 2-isostatique.

Enfin, Lovasz et Yemini [Lovasz and Yemini, 1982] ont proposé une condition suffisante pour caractériser des graphes 2-rigides qui ne sont pas forcément 2-isostatiques :

Théorème 10 de Lovasz et Yemini
Tout système à barres dont le graphe est 6-connexe est génériquement rigide en dimension 2.

Lovasz et Yemini ont de plus prouvé l'existence de graphes 5-connexes non 2-rigides, montrant par là que leur condition de 6-connexité est une borne inférieure pour ce type de caractérisation des graphes 2-rigides.

Caractérisation structurelle en 3D

Les caractérisations que nous avons présentées précédemment sont valides en 2D, c-a-d. qu'elles permettent d'identifier des sous-graphes 2-isostatiques. Malheureusement, les extensions de ces caractérisations en 3D, c-a-d. pour l'identification de graphes 3-isostatiques ne sont pas correctes. En fait, aucune caractérisation structurelle, correcte et complète des graphes 3-isostatiques n'a jusqu'ici été proposée. Nous allons expliquer ici pourquoi l'extension en 3D des caractérisations connues en 2D ne sont pas correctes.

La caractérisation de Henneberg étendue en 3D est la suivante :

Propriété 1 Henneberg en 3D
Un graphe (V, B) est constructible par une séquence de Henneberg en 3D si et seulement si il peut être obtenu à partir d'un triangle par une série de 3-extensions et 3-arête-séparations.

L'extension des conditions de Laman en 3D est la suivante :

Propriété 2 Laman en 3D
Un graphe vérifie la caractérisation de Laman en 3D si et seulement si :
- $|C| \leq 3|U| - 6$ *pour tout* $U \subseteq V$ *vérifiant* $|U| \geq 3$, $C \subseteq B$ *étant l'ensemble des arêtes du graphe* (V, B) *induit par l'ensemble de sommets* U, *et*
- $|B| = 3|V| - 6$.

La figure 2.10 présente en partie gauche un contre-exemple à la caractérisation de Laman en 3D, et en partie droite un contre-exemple à celle de Henneberg en 3D.

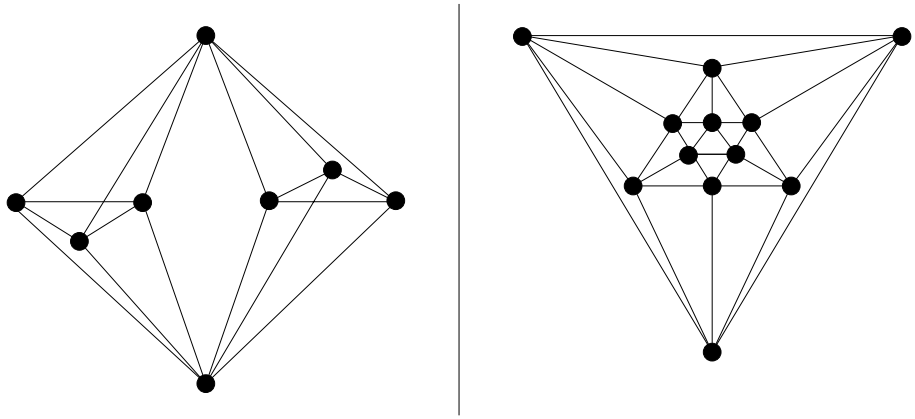

FIG. 2.10 – Contre-exemples 3D aux caractérisations de Laman et Henneberg

La partie gauche de cette figure est un système à barres 3D appelé *double banane*. Ce système n'est évidemment pas rigide : chaque banane peut tourner autour de l'axe défini par leurs extrémités liées. Si ce système n'est pas rigide, c'est qu'il ne contient pas de graphe 3-isostatique couvrant. Cependant, les conditions de Laman étendues en 3D sont vérifiées sur ce système à barres. L'extension de la caractérisation de Laman à la 3D n'est donc pas correcte. Par contre, un graphe qui ne vérifie pas les conditions de Laman étendues en 3D n'est pas rigide. Autrement dit, l'extension de la caractérisation de Laman en 3D est une condition nécessaire à la rigidité d'un graphe :

Condition 1 de Laman
Tout graphe 3*-isostatique vérifie nécessairement la caractérisation 3D de Laman.*

La partie droite de la figure représente le plus petit graphe 3-isostatique (en terme de nombre de sommets) non constructible par la méthode de Henneberg étendue en 3D. Cette méthode, basée sur les opérations de 3-extension et 3-arête-séparation, permet de construire uniquement des graphes 3-isostatiques, mais elle ne permet pas de tous les construire comme le prouve ce contre-exemple. La raison pour laquelle aucune construction de Henneberg 3D ne peut produire ce contre-exemple repose sur les propriétés suivantes :

- Tout graphe 3-isostatique contenant un sommet de degré[3] 3 peut être produit par une 3-extension à partir d'un graphe 3-isostatique plus petit.
- Tout graphe 3-isostatique contenant un sommet de degré 4 peut être produit par une 3-arête-séparation à partir d'un graphe 3-isostatique plus petit.
- Le théorème de Laman permet de montrer que tous les sommets d'un graphe 3-isostatique ont un degré supérieur ou égal à 3, et qu'au moins un de ses sommets a un degré inférieur à 6.

Le contre-exemple présenté en partie gauche de la figure 2.10 est le plus petit graphe 3-isostatique qui ne contient que des sommets de degré supérieur ou égal à 5, ce qui explique qu'il n'est pas constructible par l'extension de la méthode de Henneberg en 3D.

D'autres extensions de la méthode de Henneberg en 3D ont été proposées (incluant, par exemple, la définition de nouveaux opérateurs, comme le 3-double-arête-séparation), mais aucune n'a jusqu'ici permis une caractérisation combinatoire complète de tous les graphes 3-isostatiques. Cependant, les travaux menés sur cette caractérisation on donné lieu a des conditions intéressantes :

Condition 2 de Henneberg

Un graphe construit à partir d'un triangle par 3-extensions et 3-arête-séparations est nécessairement un 3-isostatique.

S'il n'existe aucune séquence de 3-extensions, 3-arête-séparations et 3-double-arête-séparations partant d'un triangle et permettant de produire un graphe donné, ce graphe n'est pas 3-isostatique. Cependant, il existe des séquences de ces opérations qui produisent des graphes qui ne sont pas 3-isostatiques.

Enfin, un calcul probabiliste du déterminant de la matrice de rigidité [Hendrickson, 1992; Lamure and Michelucci, 1998] permet de "décider" si un système à barres en dimension quelconque est génériquement rigide. Cette méthode consiste à évaluer le déterminant, qui comporte symboliquement un nombre exponentiel de termes, uniquement sur des configurations p tirées aléatoirement. La probabilité que des tirages aléatoires produisent des configurations singulières étant nulle en théorie [Lovasz and Plummer, 1986], il suffit que l'une des évaluations de ce déterminant soit non nulle pour que le système soit génériquement rigide. Nous revenons sur cette approche probabiliste au chapitre 6.

2.1.8 Conclusion sur la théorie de la rigidité

La communauté de topologie structurale a proposé des caractérisations algébriques pour la rigidité et la rigidité de premier ordre pour les systèmes à barres, systèmes constitués de points et distances, dans une configuration donnée. Le problème posé par ces caractérisations est la complexité de leur mise en œuvre et les restrictions qui portent sur les configurations traitables : les configurations pour lesquelles ces caractérisations fonctionnent sont dites génériques.

[3]Le degré d'un sommet est égal au nombre d'arêtes qui lui sont incidentes.

Afin de fournir un outil plus puissant et plus souple pour l'étude de la rigidité de ces systèmes, une nouvelle forme de rigidité a été développée, la rigidité générique : un système à barres est génériquement rigide si toute configuration générique appliquée sur sa structure est rigide.

Cette forme de rigidité repose uniquement sur la structure des systèmes à barres, à savoir un graphe dont les sommets sont les points du système et les arêtes représentent les contraintes de distance. Cette forme de rigidité dépend de conditions qui peuvent être vérifiées de façon combinatoire en dimensions 1 et 2. En dimension 3, toutes les extensions de ces méthodes qui ont été proposées jusqu'à présent sont soit incomplètes, soit incorrectes. Elles ont néanmoins permis de mettre en place des conditions soit nécessaires, soit suffisantes pour la rigidité générique d'un système. D'autres caractérisations structurelles de rigidité générique ont été proposées : pour les structures à base de polyèdres, les dessins parallèles, certaines classes de mécanismes, etc [Whiteley, 2002].

Ces approches structurelles sont à l'origine de la rigidité structurelle, forme de rigidité générique appliquée à des GCSP quelconques, définie par la communauté CAO (cf. section 2.3).

2.2 Théorie des mécanismes

La communauté de théorie des mécanismes s'intéresse principalement à l'étude des mouvements dans des mécanismes, GCSP constitués de corps solides liés par des articulations spécifiques. Cette étude porte, entre autres, sur la détermination du *degré de mobilité* d'un mécanisme donné, indice dont la valeur représente le nombre de mouvements indépendants admis par ce mécanisme. Dans cette section, nous allons dans un premier temps définir les mécanismes, puis donner une idée de l'approche proposée par cette communauté pour l'étude des mouvements dans les mécanismes et les résultats relatifs à la rigidité d'un mécanisme.

2.2.1 Mécanismes

Un mécanisme est un GCSP en dimension 2 ou 3 constitué de *pièces mécaniques* assemblées par des contacts mécaniques appelés *liaisons* ou articulations. Plus formellement :

Définition 23 Mécanisme
Un **mécanisme** *en dimension d se définit par une paire (P, L) où :*
- *P est un ensemble de* **pièces mécaniques**, *corps solides (rigides) en dimension d,*
- *L est un ensemble de* **liaisons mécaniques**, *chaque liaison liant une paire de pièces de P.*

Pour l'étude de la mobilité d'un mécanisme, on se limite aux *pièces mécaniques idéales* et aux *liaisons idéales*. Une pièce mécanique idéale est un corps solide de géométrie parfaitement connue. Une pièce mécanique idéale est considérée indéformable, c-a-d. rigide. Une liaison idéale est un contact mécanique parfait entre des pièces mécaniques idéales. Elle ne

présente ni irrégularités, ni jeu, ni frottements dans les mouvements qu'elle autorise entre les pièces impliquées.

On considère généralement les mécanismes dans une *configuration* donnée. Une configuration d'un mécanisme consiste à prescrire la géométrie (dimensions, forme, ...) de chacune de ses pièces mécaniques et à déterminer les paramètres (angles, longueurs, ...) de ses liaisons.

L'étude des mouvements d'un mécanisme en configuration connue peut se faire suivant différentes approches : l'*approche géométrique* considère les relations géométriques (positions et mouvements) du mécanisme, l'*approche statique* considère les forces exercées dans le mécanisme, l'*approche cinématique* considère les vitesses relatives des pièces mécaniques les unes par rapport aux autres et l'*approche dynamique* considère le travail et les puissances développées dans le mécanisme. Ces approches sont duales en un certain sens[4] et permettent de déterminer le *degré de mobilité* d'un mécanisme. Nous présentons dans la section suivante le principe de l'approche cinématique pour la détermination du degré de mobilité d'un mécanisme.

2.2.2 Approche cinématique

Le degré de mobilité d'un mécanisme caractérise le nombre de mouvements indépendants que le mécanisme admet dans la configuration qui lui est associée. Pour le déterminer en utilisant l'approche cinématique, on utilise le principe de la *fermeture cinématique* sur les *boucles fermées* du mécanisme.

Considérons un mécanisme comme un graphe dont les sommets sont les pièces mécaniques et où chaque arête représente une liaison entre 2 pièces mécaniques. Une *boucle fermée* d'un mécanisme est un cycle dans ce graphe. Un ensemble de *boucles fermées* est indépendant si aucune des boucles fermées qu'il contient ne peut être reproduite à partir des arêtes des autres boucles fermées qu'il contient. Le *nombre cyclomatique* NC, qui caractérise le nombre de boucles fermées indépendantes, vaut $NC = |L| - |P| + 1$ dans un mécanisme constitué de $|P|$ pièces dont l'une est fixée, et de $|L|$ liaisons.

Chaque liaison l du mécanisme induit un torseur cinématique qui comporte c_l inconnues cinématiques indépendantes (c_l varie selon les types des liaisons considérées). La fermeture cinématique indique que la somme des vitesses de chaque pièce mécanique dans une boucle fermée est un vecteur nul. L'application de cette fermeture à une boucle fermée indépendante permet d'obtenir un système de 6 équations[5] sur les inconnues cinématiques des liaisons de cette boucle. Pour l'ensemble des boucles fermées indépendantes du mécanisme, on obtient ainsi $E_c = 6 * NC$ équations linéaires sur les $I_c = \sum_{l \in L} c_l$ inconnues cinématiques de l'ensemble L des liaisons du mécanisme.

L'approche cinématique consiste donc à étudier un système de E_c équations à I_c inconnues. Le rang r_c de ce système donne le degré de mobilité m du mécanisme : $m = I_c - r_c$,

[4] Selon le mécanisme et sa configuration, il peut être plus simple d'établir les équations liées à l'une ou l'autre de ces approches.

[5] le principe de fermeture cinématique appliqué à une boucle fermée produit 2 équations sur des vecteurs de $\mathbb{R}^3$, c-a-d. 6 équations au total, et ce quelle que soit la taille de la boucle.

c-a-d. le nombre d'inconnues cinématiques qui ne peuvent être déterminées par ce système.

Lorsque $m = 0$, le mécanisme est immobile : il s'agit d'une structure rigide *isostatique* ; lorsque $m > 0$, le mécanisme possède m mouvements indépendants et n'est donc pas rigide ; enfin, lorsque $m < 0$, le mécanisme est aussi considéré comme étant rigide, mais *hyperstatique* (ou R-redondant). Par rapprochement avec les notions définies en théorie de la rigidité, on pourrait dire que le degré de mobilité représente le nombre de mouvements d'un mécanisme qui sont des déformations (rappelons qu'un système est rigide s'il n'admet aucune déformation).

Cette approche permet de déterminer si un mécanisme dans une configuration connue est une structure rigide ou pas. Cependant, la plupart des mécanismes disposent d'une infinité de configurations possibles : les dimensions des pièces mécaniques, leurs positions et orientations ainsi que les paramètres des liaisons mécaniques sont autant de paramètres de l'ensemble des configurations d'un mécanisme donné. Le calcul du degré de mobilité d'un mécanisme se compare alors à la caractérisation de la rigidité d'un système à barres en configuration donnée : c'est une information plus précise que la rigidité générique, mais moins générale puisqu'elle ne donne aucune information sur la rigidité des autres configurations du même système. Or, un mécanisme est généralement piloté et doit permettre une variation de certains de ces paramètres. Cependant, il devrait être rigide en toute configuration de ces paramètres de pilotage.

2.2.3 Formule de Grübler

Grübler [Grübler, 1917; Angeles, 1982] a proposé une formule *combinatoire* pour le calcul du degré de mobilité d'un mécanisme indépendante de sa configuration :

Proposition 2 Formule de Grübler
Un mécanisme (P, L) en dimension d a pour degré de mobilité $m = \frac{d(d+1)}{2} * (|P| - |L| - 1) + \sum_{l \in L} DDL(l)$, où $DDL(l)$ est le nombre de degrés de liberté de la liaison l.

Le nombre de *degrés de liberté* $DDL(l)$ d'une liaison l correspond au nombre d'inconnues cinématiques du torseur cinématique de l. Intuitivement, ce nombre correspond au nombre de mouvements indépendants et relatifs que la liaison autorise entre les pièces qu'elle lie. Par exemple, une liaison rotoïde autorise uniquement la rotation autour d'un axe particulier : elle dispose donc d'un unique degré de liberté.

Le principe de la formule de Grübler est que le degré de mobilité d'un mécanisme est égal à la somme des degrés de liberté offerts par ses pièces mécaniques hormis le bâti ($\frac{d(d+1)}{2} * (|P| - 1)$) soustrait du nombre de degré de liberté retirés par les liaisons ($\frac{d(d+1)}{2} * |L| - \sum_{l \in L} DDL(l)$).

La formule de Grübler se rapproche donc par certains aspects de la caractérisation de la rigidité générique par Laman en théorie de la rigidité : elle consiste en un compte des degrés de liberté du mécanisme. La caractérisation de Laman est cependant plus précise puisqu'elle vérifie aussi une condition sur tous les sous-systèmes, alors que la formule de Grübler ne propose qu'un compte global.

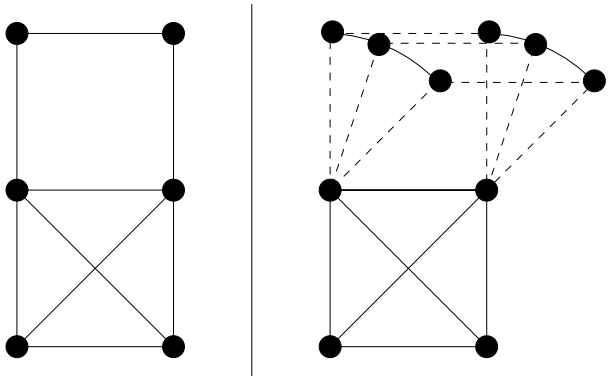

FIG. 2.11 – Mécanisme 2D trompant la formule de Grübler

Cela peut être source d'erreur dans l'estimation du degré de mobilité que la formule propose : en effet, il est possible qu'un mécanisme pour lequel la formule estime un degré de mobilité $m = 0$ contienne en fait une sous-partie hyperstatique et une sous-partie mobile. C'est le cas du mécanisme 2D présenté en partie gauche de la figure 2.11. Ce mécanisme est constitué de 9 barres liées par leurs extrémités. Selon la formule de Grübler, il possède un degré de mobilité $m = 0$, or ce mécanisme possède un mouvement : la partie droite de la figure présente une autre position du mécanisme pour les mêmes paramètres : ce mécanisme est équivalent à un système à barres qui n'est pas rigide, même en configuration générique. A noter que l'approche cinématique pour la détermination du degré de mobilité de ce mécanisme permettrait de déterminer que celui-ci n'est pas rigide.

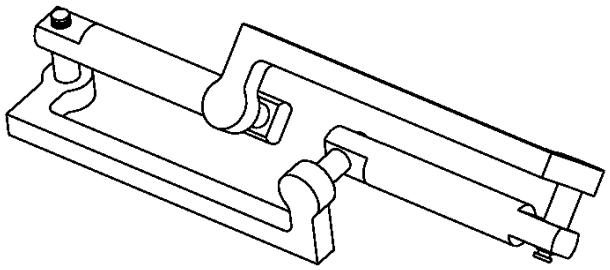

FIG. 2.12 – Mécanisme de Bennett à 1 degré de liberté en configuration singulière

Nous avons vu que la caractérisation de Laman identifie des systèmes à barres rigides en configurations génériques. La formule de Grübler caractérise de façon similaire le degré de mobilité d'un mécanisme en toutes ses configurations *génériques*. La figure 2.12 présente le mécanisme 3D de Bennett constitué de 4 pièces (des barres) et de 4 liaisons rotoïdes. Le degré de mobilité de ce mécanisme selon la formule de Grübler est $m = 0$, indiquant que le mécanisme devrait être une structure rigide. Cependant, la configuration du mécanisme présentée sur cette figure, où les 4 barres sont de même longueur et où les axes des liaisons

rotoïdes forment des angles particuliers, confère à ce mécanisme un degré de liberté. Cet exemple correspond à une *configuration singulière* du mécanisme.

La formule de Grübler permet donc de décider du degré de mobilité d'un mécanisme en général mais peut être fausse pour des mécanismes spécifiques (redondants ou en configurations singulières).

2.2.4 Conclusion sur la théorie des mécanismes

La théorie des mécanismes s'intéresse à l'étude de GCSP 2D et 3D constitués de corps solides reliés par des liaisons mécaniques. Une étude cinématique (ou statique, ou dynamique, ...) d'un mécanisme permet de déterminer son degré de mobilité à partir du rang du système d'équations cinématiques associé au mécanisme. Cette approche fournit une information précise mais est basée sur une configuration particulière du mécanisme, certains mécanismes pouvant admettre un grand nombre de configurations. L'introduction des paramètres du mécanisme (dimensions des pièces et géométrie des articulations) conduit à une formulation paramétrée du système cinématique dont le rang doit alors être étudié formellement, ce qui rend cette approche difficile en pratique.

Grübler a proposé une formule qui permet de calculer le degré de mobilité d'un mécanisme simplement à partir du nombre de pièces et des degrés de liberté des liaisons de celui-ci. Cette formule fournit une information générale et facile à calculer, mais qui peut malheureusement être fausse pour des mécanismes redondants ou en configurations singulières. En cela, cette formule peut être comparée aux caractérisations structurelles proposées par la communauté de topologie structurale pour la rigidité générique de systèmes à barres.

2.3 Rigidité en CAO

En CAO, les GCSP attaqués sont quelconques, c-a-d. constitués d'objets géométriques quelconques liés par des contraintes géométriques quelconques. La notion de rigidité n'a pas été définie de façon générale pour des GCSP quelconques. Cependant, certaines formes de rigidité ont été définies, ou du moins utilisées par des méthodes de résolution proposées par la communauté de CAO (cf. section 4.3 page 99 et section 4.4 page 107).

Cette section constitue donc une contribution de la thèse, puisque nous définissons un cadre général pour l'étude de la rigidité de GCSP quelconques.

Pour ce faire, nous nous basons sur les notions de rigidité développées dans les communautés de topologie structurale et de théorie des mécanismes. Cela nous permet de définir la *rigidité d'une solution d'un GCSP*, puis la rigidité de toutes les solutions d'un GCSP, appelée *rigidité du GCSP*, qui est la propriété intéressante en CAO. Nous montrons que déterminer si un GCSP est bien-, sur- ou sous-rigide est équivalent à déterminer si un CSP est bien-, sur- ou sous-contraint, caractérisation dont la difficulté a été expliquée à la section 1.4, page 19.

Nous faisons alors une synthèse des concepts de rigidité utilisés en CAO, que nous regroupons sous l'appellation de rigidité structurelle. Nous discutons des limites de cette caractérisation qui peut être comparée à la caractérisation structurelle de la rigidité générique en topologie structurale et à la formule de Grübler en théorie des mécanismes. Nous effectuons aussi un rapprochement avec la caractérisation structurelle de bien-, sur- ou sous-contraint pour un CSP, présentée à la section 1.4.2, page 20. Cela permet de mieux comprendre l'intérêt de la communauté de CAO pour la rigidité : un GCSP rigide est l'équivalent géométrique d'un CSP bien-contraint.

2.3.1 Rigidité d'un GCSP

Nous avons vu que la rigidité d'un système à barres ou d'un mécanisme s'étudie sur une configuration du système considéré. Il en va de même pour les GCSP : on peut définir la rigidité d'une configuration-solution p d'un GCSP S. Rappelons qu'une solution p de S est un ensemble qui contient une valeur $p(v_i^o)$ pour chacun des paramètres géométriques v_i^o de chacun des objets géométriques o du GCSP S. On utilise la même notion intuitive de rigidité que celle introduite en début de chapitre : une configuration-solution d'un GCSP est rigide si elle est indéformable et déplaçable dans tout l'espace. La notion de rigidité est donc toujours basée sur la notion de mouvement.

Mouvement d'une solution d'un GCSP

Nous étendons la définition de mouvement aux GCSP quelconques de la façon suivante :

Définition 24 Mouvement d'une solution d'un GCSP
Un **mouvement** *d'une solution p d'un GCSP S est un ensemble M contenant une fonction continue et différenciable $m_i^o : [0,1] \rightarrow \mathbb{R}$ pour chaque paramètre géométrique v_i^o de chaque objet géométrique o de S qui vérifie $m_i^o(0) = p(v_i^o)$ et pour tout $t \in [0,1]$, $m(t) = \{m_i^o(t)\}$ est aussi une solution du GCSP S.*

Contrairement à la notion de mouvement pour les systèmes à barres, il est difficile de distinguer les mouvements d'une solution d'un GCSP qui sont des déplacements de ceux qui sont des déformations en considérant l'ensemble des propriétés du GCSP. En effet, en topologie structurale, il suffit de vérifier que toutes les distances point à point d'un système à barres restent identiques au cours d'un mouvement pour décider qu'il s'agit d'un déplacement. Dans un GCSP quelconque, l'ensemble des relations à vérifier ne peut être défini qu'une fois donnés les types des objets du GCSP. Par exemple, si le GCSP ne contient que des points et des droites en 2D, il faudrait vérifier que toutes les distances point à point et point à droite, mais aussi tous les angles droite à droite restent identiques au cours d'un mouvement pour décider que ce mouvement est un déplacement.

A cette caractérisation dépendante des types des objets du GCSP, nous préférons une définition des déplacements basée sur la constatation géométrique suivante : tout mouvement d'un GCSP qui est un déplacement correspond à un déplacement (rotation et

translation) de l'espace géométrique contenant le GCSP. La définition formelle est la suivante :

Définition 25 Déplacement d'une solution d'un GCSP
Un **déplacement** *d'une solution p d'un GCSP S est un mouvement M tel que pour tout $t \in [0,1]$ il existe un déplacement (rotation+translation) de l'espace géométrique considéré qui, appliqué à chaque objet du GCSP, transforme la solution p en une solution q de S tel que $M(t) = q$.*

Cette caractérisation des déplacements d'une solution d'un GCSP nous permet de définir par complémentarité la notion de déformation d'une solution d'un GCSP :

Définition 26 Déformation d'une solution d'un GCSP
Un mouvement d'une solution p d'un GCSP S qui n'est pas un déplacement est une **déformation**.

Rigidité d'une solution d'un GCSP

Nous pouvons alors définir la notion de rigidité d'une solution d'un GCSP :

Définition 27 Rigidité d'une solution d'un GCSP
Une solution d'un GCSP est **bien-rigide** *si l'ensemble des mouvements qu'elle admet ne contient aucune déformation, mais contient tous les déplacements de l'espace géométrique considéré.*

Elle est **sur-rigide** *si elle n'admet pas tous les déplacements de l'espace géométrique considéré.*

Elle est **sous-rigide** *si elle admet tous les déplacements de l'espace géométrique, mais aussi des déformations.*

Notons que, à l'inverse de la définition de rigidité pour les systèmes à barres, la rigidité d'une solution d'un GCSP implique de vérifier que cette solution est indéformable *et déplaçable*. En effet, cette seconde condition est superflue pour les systèmes à barres car, étant uniquement constitués de contraintes de distances, ils sont toujours déplaçables. Cela n'est pas toujours le cas pour un GCSP quelconque. Par exemple, si l'on autorise la contrainte qui consiste à fixer un point dans le repère global de l'espace géométrique considéré, toute solution d'un GCSP contenant cette contrainte n'est pas déplaçable : on ne peut lui appliquer les translations de l'espace géométrique puisque celles-ci conduiraient à violer la contrainte de fixation.

En fait, une solution d'un GCSP est toujours déplaçable si chaque contrainte géométrique du GCSP est indépendante du repère global de l'espace géométrique considéré. Une contrainte est indépendante du repère global si elle ne fixe les positions et orientations d'objets géométriques que les uns relativement aux autres et jamais relativement au repère global. Les contraintes valuées (distance, angle) et booléennes (incidence, parallélisme) usuelles sont indépendantes du repère global. En revanche, la contrainte qui fixe (même

partiellement) un objet est dépendante du repère global puisqu'elle consiste à donner une position et une orientation à cet objet dans le repère global.

Nous avons défini les notions supplémentaires de *sur-rigidité* et *sous-rigidité* qui nous permettent de classer chaque solution d'un GCSP dans l'une des 3 catégories : bien-rigide, sur-rigide ou sous-rigide.

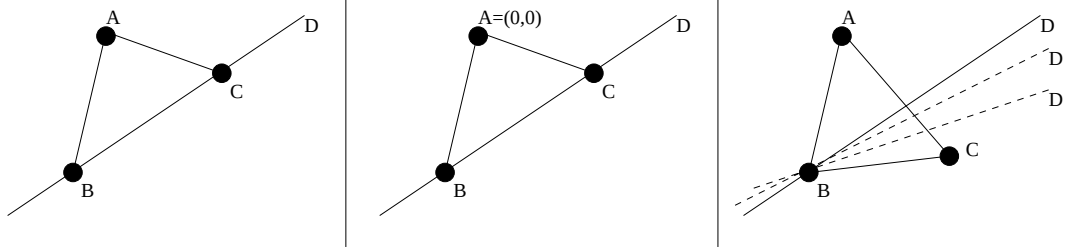

FIG. 2.13 – Exemples de GCSP rigide, sur-rigide et sous-rigide

La figure 2.13 présente 3 solutions de 3 GCSP différents en 2D. Ces 3 GCSP sont constitués du même quadruplet d'objets géométriques : 3 points A, B et C et une droite D. Le GCSP de gauche contient les contraintes suivantes : Distance(A, B), Distance(A, C), Distance(B, C), Incidence(B, D), Incidence(C, D). La solution présentée de ce GCSP est bien-rigide : la droite et les 3 points peuvent être déplacés ensembles dans le plan euclidien, mais n'admettent aucun mouvement qui respecte les contraintes sans être un déplacement. La partie centrale de la figure 2.13 présente une solution du GCSP contenant les contraintes Incidence$(A, (0, 0))$, Distance(A, B), Distance(A, C), Distance(B, C), Incidence(B, D), Incidence(C, D). La solution présentée de ce GCSP est sur-rigide car le point A est fixé à l'origine du repère global du plan euclidien ; elle ne peut donc pas admettre toutes les translations du plan euclidien. La partie droite de la figure présente une solution du GCSP contenant les contraintes Distance(A, B), Distance(A, C), Distance(B, C), Incidence(B, D). Cette solution est sous-rigide car le point C et la droite D peuvent être déplacés relativement l'un à l'autre, ce qui constitue une déformation du système.

Caractérisation algébrique de la rigidité d'une solution

Les méthodes formelles ne permettent pas de caractériser la rigidité d'une solution d'un GCSP. En effet, la méthode décrite à la section 2.1.2, page 29, ne peut être appliquée car :

- les contraintes peuvent être non algébriques (cf. GCSP présenté à la figure 1.1, page 19) ;
- l'ensemble des propriétés métriques qu'un déplacement doit vérifier continûment ne peut être défini qu'en fonction des types d'objets géométriques présents dans le GCSP (cf. remarque suivant la définition 24, page 51).

Nous ne proposons pas d'autres façons de caractériser la rigidité d'une solution d'un GCSP, car cette notion ne présente pas d'intérêt particulier dans le cadre de cette thèse.

En effet, elle n'est qu'une étape intermédiaire vers la notion de rigidité globale d'un GCSP, qui représente la rigidité *a priori* de l'ensemble des solutions d'un GCSP.

Rigidité globale d'un GCSP

En CAO, ce n'est pas tant la rigidité d'une solution d'un GCSP qui est intéressante, mais plutôt la *rigidité de toutes ses solutions*. En effet, la propriété de rigidité est utilisée en CAO pour simplifier la résolution de GCSP en le décomposant en sous-GCSP solubles dans un repère local, c-a-d. dont toutes les solutions sont rigides. Lors d'une telle décomposition (cf. section 4.3 page 99 et section 4.4 page 107), le GCSP n'est pas encore résolu et on ne peut juger de la rigidité de chacune de ces solutions individuellement. Nous allons voir de quelle manière cette rigidité peut être caractérisée. Tout d'abord, nous définissons le concept de rigidité de GCSP :

Définition 28 Rigidité d'un GCSP
Un GCSP est **bien-rigide** *si et seulement si toutes ses solutions sont bien-rigides ;*
il est **sur-rigide** *si et seulement si il ne possède aucune solution ou qu'au moins une de ses solutions est sur-rigide ;*
il est **sous-rigide** *si et seulement si il n'est pas sur-rigide et possède au moins une solution sous-rigide.*

Le concept de rigidité de GCSP est proche de celui de CSP bien-contraint (cf. définition 8, page 19). Un CSP est bien-contraint s'il possède un ensemble dénombrable de solutions. Un GCSP rigide possède un ensemble indénombrable de solutions, puisque toute solution rigide doit être déplaçable. Cependant, si l'on considère un GCSP rigide fixé dans un repère local, il n'admet plus qu'un ensemble dénombrable de solutions. En effet, si tel n'est pas le cas, c'est qu'il existe un mouvement continu permettant de passer de l'une de ses solutions à une autre solution ; ce mouvement ne peut être un déplacement, car le GCSP est fixe dans son repère local, c-a-d. qu'on le résout modulo les déplacements de l'espace géométrique considéré ; il s'agit donc d'une déformation, et le GCSP n'est alors pas rigide puisqu'il contient une solution qui ne l'est pas. Un GCSP est donc rigide s'il est bien contraint lorsqu'il est considéré dans un repère local, c-a-d. modulo les déplacements de l'espace géométrique considéré.

C'est cette propriété qui intéresse la communauté de CAO qui cherche à résoudre des GCSP en tirant parti de la rigidité. En effet, le but des méthodes de décomposition géométrique est d'identifier des sous-GCSP qui puissent être résolus complètement dans des repères locaux, c-a-d. des sous-systèmes bien-contraints modulo les déplacements. De tels sous-GCSP sont rigides au sens de la définition 28.

Aucune caractérisation complète et correcte n'a jusqu'ici été proposée pour la rigidité d'un GCSP. La communauté de CAO a proposé une caractérisation structurelle de la rigidité d'un GCSP, semblable d'une part à la caractérisation structurelle de la rigidité générique d'un système à barres (cf. caractérisation de Laman, théorème 8, page 42), et d'autre part à la caractérisation structurelle des CSP bien-contraints (cf. définition 8,

page 19). C'est la *rigidité structurelle* qui permet de décider (approximativement) si un GCSP est bien-rigide, sur-rigide ou sous-rigide. Cette heuristique de rigidité est basée sur un compte des degrés de liberté dans un GCSP, tout comme la caractérisation de Laman et la formule de Grübler.

Nous allons tout d'abord étendre le concept de degré de liberté et de compte des degrés de liberté aux GCSP avant de présenter la rigidité structurelle.

2.3.2 Degrés de liberté d'un GCSP

En topologie structurale, une façon de compter le nombre de mouvements indépendants admis par un système à barres est de compter le nombre de degrés de liberté de ce système. On peut étendre la définition des degrés de liberté aux GCSP et proposer un calcul similaire pour compter le nombre de mouvements indépendants admis dans un GCSP.

Pour pouvoir définir le nombre de degrés de liberté d'un GCSP, il nous faut définir le nombre de degrés de liberté offerts par ses objets et le nombre de degrés de liberté retirés par ses contraintes.

Degrés de liberté d'un objet géométrique

Le nombre de degrés de liberté d'un objet se définit intuitivement comme le nombre de paramètres géométriques qui permettent de définir cet objet. Plus formellement :

Définition 29 Degrés de liberté d'un objet géométrique
Un objet géométrique o dont la configuration peut être déterminée en fixant k paramètres possède k degrés de liberté ; on note $DDL(o) = k$.

Le nombre de paramètres permettant de fixer la configuration d'un objet dépend généralement de sa modélisation ; la proposition suivante établit le lien entre la modélisation d'un objet et son nombre de degrés de liberté :

Proposition 3 Compte des degrés de liberté d'un objet
Si o est modélisé par un triplet (P_o, I_o, E_o) dont la relation interne I_o est un système d'équation générique et non-A-redondant, alors $DDL(o) = |P_o| - |I_o|$.

En effet, la relation interne I_o d'un objet o permet de déterminer un certain nombre des paramètres géométriques P_o de cet objet à partir des autres paramètres. Le nombre de paramètres déterminés par un système d'équations est généralement égal au nombre d'équations constituant ce système. Ceci est faux si le système est algébriquement singulier ou algébriquement redondant (cf. contre-exemples présentés à la section 1.4.2, page 21).

Pour déterminer le nombre de DDL d'un objet géométrique, il faut donc disposer d'une modélisation générique et non-A-redondante de cet objet. Le plus simple est d'utiliser la *modélisation minimale* d'un objet (cf. définition 5, page 16) : cette modélisation ne contient pas de relation interne, et alors $DDL(o) = |P_o|$.

Voici 2 exemples d'objets géométriques et leur nombre de degrés de liberté :

- Un point A en 2D peut être représenté par le triplet $(\{x_A, y_A\}, \emptyset, (x = x_A) \wedge (y = y_A))$; le nombre de paramètres libres du point A est donc 2, puisque le point ne possède pas de relation interne. Ceci implique qu'un point possède 2 degrés de liberté en 2D.
- Un plan P en 3D peut être représenté par le triplet $(\{a, b, c, d\}, ((a^2+b^2+c^2 = 1) \wedge (d \geq 0) \wedge (a > 0 \vee (a = 0 \wedge b > 0) \vee (a = b = 0 \wedge c > 0))), (a * x + b * y + c * z + d = 0))$. La relation interne au plan P permet de déterminer 1 paramètre du plan parmi a, b et c, une inégalité ne permettant pas de déterminer une inconnue dans le cas général. Un plan en 3D possède donc 3 paramètres géométriques libres, c-a-d. 3 degrés de liberté.

Du point de vue géométrique, les paramètres d'un objet définissent sa position, son orientation et ses dimensions ; chaque degré de liberté d'un objet représente alors une possibilité de variation indépendante de sa position, de son orientation ou de ses dimensions. Par exemple, un point en 2D possède 2 degrés de liberté, et admet effectivement 2 déplacements indépendants : les 2 translations du plan euclidien. Un plan en 3D possède 3 degrés de liberté et dispose effectivement d'une seule translation et de 2 rotations de l'espace euclidien à 3 dimensions. Le tableau 2.1 regroupe le nombre de degré de liberté de quelques objets 2D et 3D[6]. Par extension, on notera $DDL(O)$ la somme des degrés de liberté d'un ensemble d'objets géométriques O ($DDL(O) = \sum_{o \in O} DDL(o)$).

Objet	2D	3D
Point	2	3
Droite	2	4
Plan	-	3
Cercle à rayon variable	3	6
Cercle à rayon fixe	2	5
Sphère à rayon variable	-	4
Sphère à rayon fixe	-	3

TAB. 2.1 – Degrés de liberté des objets usuels en 2D et 3D

Degrés de liberté d'une contrainte géométrique

Les contraintes géométriques empêchent certains mouvements relatifs des objets géométriques. Ce faisant, elles retirent donc des degrés de liberté aux objets sur lesquels elles portent.

Définition 30 Degrés de liberté d'une contrainte géométrique
Une contrainte géométrique c retire un nombre de degré de liberté, noté $DDL(c)$, égal au nombre de paramètres que la contrainte permet de fixer parmi les objets géométriques sur lesquels elle porte.

[6]La modélisation des objets que nous avons utilisée dans notre prototype est présentée à l'annexe A.

Ce nombre est aussi parfois appelé *degrés de restriction* d'une contrainte. Pour déterminer le nombre de paramètres qu'une contrainte permet de déterminer, on peut s'appuyer sur la modélisation de cette contrainte :

Proposition 4 Calcul du nombre de degrés de liberté d'une contrainte
Si c est modélisée par un quadruplet (O_c, P_c, I_c, E_c) dont la relation externe E_c est un système d'équations générique et non-A-redondant, alors $DDL(c) = |E_c|$.

En effet, la relation externe E_c d'une contrainte géométrique c permet généralement de déterminer un nombre de paramètres géométriques des objets O_c égal au nombre d'équations qui la constitue. Ceci est faux si la relation externe de la contrainte est un système d'équations algébriquement singulier ou algébriquement redondant. Considérons par exemple la modélisation d'une contrainte d'incidence entre 2 points A et B en 2D par l'équation $(x_A - x_B)^2 + (y_A - y_B)^2 = 0$. Cette équation correspond à la contrainte Distance(A, B)=0 qui est singulière : elle n'est constituée que d'une seule équation mais permet de déterminer 2 variables.

Remarque : Représenter les contraintes booléennes par des contraintes valuées constitue donc une erreur ; les contraintes valuées représentant des contraintes booléennes sont généralement singulières et ne retirent donc pas le même nombre degré de liberté. Cette représentation des contraintes booléennes par des contraintes valuées fausse donc le compte des degrés de liberté.

Voici quelques exemples de contraintes géométriques et leurs degrés de liberté[7] :

- Une contrainte de distance $distance(A, B)$ entre 2 points $A = (\{x_A, y_A\}, \emptyset, ((x = x_A) \wedge (y = y_A)))$ et $B = (\{x_B, y_B\}, \emptyset, ((x = x_B) \wedge (y = y_B)))$ du plan euclidien, peut se représenter par le quadruplet $(\{A, B\}, \{d\}, (d > 0), ((x_A - x_B)^2 + (y_A - y_B)^2 = d^2))$. La relation externe n'est constituée que d'une seule équation, qui est bien générique à cause de la relation interne à la contrainte. Une contrainte de distance retire donc bien 1 degré de liberté.
- Une contrainte de parallélisme $parallelisme(P, Q)$ entre plans $P = (\{a_P, b_P, c_P, d_P\}, (a_P^2 + b_P^2 + c_P^2 = 1 \wedge (d_P \geq 0) \wedge (a_P > 0 \vee (a_P = 0 \wedge b_P > 0) \vee (a_P = b_P = 0 \wedge c_P > 0))), (a_P * x + b_P * y + c_P * z + d_P = 0))$ et $Q = (\{a_Q, b_Q, c_Q, d_Q\}, (a_Q^2 + b_Q^2 + c_Q^2 = 1 \wedge (d_Q \geq 0) \wedge (a_Q > 0 \vee (a_Q = 0 \wedge b_Q > 0) \vee (a_Q = b_Q = 0 \wedge c_Q > 0))), (a_Q * x + b_Q * y + c_Q * z + d_Q = 0))$ en 3D, peut être représentée par le quadruplet $(\{P, Q\}, \emptyset, \emptyset, (a_P * b_Q - a_Q * b_P = 0, a_P * c_Q + a_Q * c_P = 0)$. La relation externe de cette contrainte est composée de 2 équations indépendantes. Une contrainte de parallélisme entre 2 plans en 3D permet donc de fixer 2 degrés de liberté.

L'interprétation géométrique des degrés de liberté retirés par une contrainte est la suivante : une contrainte géométrique empêche un nombre de mouvements relatifs entre les objets qu'elle contraint égal à son nombre de degrés de liberté. Par exemple, la contrainte de distance entre points empêche les points contraints de se déplacer relativement l'un à l'autre sur la droite passant par ces 2 points : elle empêche le mouvement de translation

[7] La modélisation des contraintes que nous avons utilisée dans notre prototype est présentée et discutée à l'annexe A.

le long de cette droite. La contrainte de parallélisme entre plans empêche ces plans d'avoir une rotation relativement l'un à l'autre, à part la rotation d'axe égal au vecteur normal de ces plans (rotation qui est l'identité pour un plan) : elle empêche donc 2 rotations indépendantes. Le tableau 2.2 regroupe le nombre de degrés de liberté retirés par quelques contraintes en 2D et 3D. Par extension, on notera $DDL(C)$ la somme des degrés de liberté retirés par un ensemble de contraintes géométriques C ($DDL(C) = \sum_{c \in C} DDL(c)$).

Contrainte	2D	3D
Distance(Point,Point)	1	1
Distance(Point,Droite)	1	1
Distance(Point,Plan)	-	1
Distance(Droite,Droite)	1	1
Distance(Droite,Plan)	-	1
Distance(Plan,Plan)	-	1
Angle(Point,Point,Point,Point)	1	1
Angle(Point,Point,Droite)	1	1
Angle(Point,Point,Plan)	-	1
Angle(Droite,Droite)	1	1
Angle(Droite,Plan)	-	1
Angle(Plan,Plan)	-	1
Incidence(Point,Droite)	1	2
Incidence(Point,Plan)	-	1
Incidence(Droite,Plan)	-	2
Parallélisme(Droite,Droite)	1	2
Parallélisme(Droite,Plan)	-	1
Parallélisme(Plan,Plan)	-	2

TAB. 2.2 – Degrés de liberté des contraintes usuelles en 2D et 3D

Compte des degrés de liberté d'un GCSP

Le nombre de degrés de liberté d'un GCSP se définit de la façon suivante :

Définition 31 Degrés de liberté d'un GCSP
Le nombre de degrés de liberté d'un GCSP $S = (O, C)$ représente le nombre de mouvements indépendants admis par ce GCSP.

Pour calculer le nombre de degrés de liberté d'un GCSP, on procède alors de la même façon que pour calculer celui d'un système à barres (cf. section 2.1.5, page 38) : en sommant les degrés de liberté offerts par ses objets géométriques et en retranchant les degrés de liberté retirés par ses contraintes géométriques.

Proposition 5 Calcul du nombre de degrés de liberté d'un GCSP
Le nombre de degrés de liberté d'un GCSP $S = (O, C)$ *vaut* $DDL(S) = DDL(O) - DDL(C)$.

Cette formule ressemble fortement à la formule de Grübler pour le calcul du degré de mobilité d'un mécanisme. La principale différence se trouve dans le fait que la formule de Grübler considère que l'un des solides du mécanisme est fixe et retire ainsi par avance les $\frac{d(d+1)}{2}$ degrés de liberté correspondants aux déplacements de ce solide : le mécanisme est ainsi déterminé dans le repère local de ce solide et les degrés de liberté restants représentent des déformations du mécanisme.

Nous avons vu que la formule de Grübler peut établir un degré de mobilité incorrect dans 2 cas de figure : si le mécanisme est dans une configuration singulière ou si le mécanisme contient des liaisons redondantes. Il en va de même pour la formule de calcul du nombre de degrés de liberté d'un GCSP : le *vrai* nombre de degrés de liberté d'un GCSP devrait tenir compte des singularités et des R-redondances du GCSP (pas seulement des A-redondances dans les modélisations des objets et des contraintes).

Une contrainte R-redondante ne change pas la rigidité du GCSP considéré, c-a-d que les propriétés géométriques (incidence, non-incidence, parallélisme, alignement, ...) qu'elle impose sont des conséquences des autres contraintes $C \backslash c$ du GCSP. Attention : comme nous l'avons déjà signalé au sujet des systèmes à barres (cf. section 2.1.5, page 38) une R-redondance n'est pas une redondance au sens habituel, c-a-d. que l'ensemble des solutions avec ou sans cette contrainte peut différer ; une contrainte R-redondante n'est donc superflue que du point de vue de la rigidité du GCSP.

Etablir qu'une contrainte est R-redondante est un problème difficile dans le cas général. Nous discutons ce problème et quelques solutions possibles à la section 6.1, page 163.

Un ensemble de contraintes géométriques est *singulier* s'il fixe globalement plus (ou moins) de degrés de liberté que la simple somme des degrés de liberté fixés par chacune des contraintes qu'il contient. Nous approfondirons cette notion dans le chapitre 6.

Il en résulte que le calcul du nombre de degrés de liberté par la formule donnée à la proposition 5 n'est valide que pour des GCSP ne contenant pas de contraintes géométriques R-redondantes ou singulières. En revanche, cette formule, à l'instar de la formule de Grübler, donne une information générale sur le GCSP : son nombre de degrés de liberté indépendamment de la solution du GCSP. C'est sur cette formule qu'est basée la notion de rigidité structurelle.

2.3.3 Rigidité structurelle

La rigidité structurelle d'un GCSP est une propriété basée sur un compte des degrés de liberté dans le GCSP. C'est une généralisation aux GCSP des conditions dérivées en fin de section 2.1.5, page 38, sur le nombre de degrés de liberté d'un système à barres : un GCSP possédant plus de mouvements que le nombre de déplacements indépendants de l'espace géométrique considéré est sous-rigide (définition de sous-s_rigide) ; un GCSP qui n'admet pas tous les déplacements de l'espace considéré, ou n'admet aucune solution, n'est pas

déplaçable, c-a-d. sur-rigide (définition de sur-s_rigide) ; enfin, un GCSP qui admet uniquement les déplacements et tous les déplacements est rigide (définition de bien-s_rigide). Rappelons que le nombre de déplacements indépendants dans un espace géométrique de dimension d est $\frac{d(d+1)}{2}$ (cf. lemme 1, page 39).

Définition 32 Rigidité structurelle
Un GCSP $S = (O, C)$ en dimension d est **structurellement bien-rigide**, *noté* **bien-s_rigide**, *si et seulement si $DDL(S) = \frac{d(d+1)}{2}$ et $\forall S' \subseteq S$, $DDL(S') \geq \frac{d(d+1)}{2}$.*

Il est **structurellement sur-rigide**, *noté* **sur-s_rigide**, *si et seulement si $\exists S' \subseteq S$ tel que $DDL(S') < \frac{d(d+1)}{2}$.*

Il est **structurellement sous-rigide**, *noté* **sous-s_rigide**, *si et seulement si $DDL(S) > \frac{d(d+1)}{2}$ et $\forall S' \subset S$, $DDL(S') \geq \frac{d(d+1)}{2}$.*

Cette définition est à rapprocher de la caractérisation de Laman pour les systèmes à barres rigides en 2D. Cette dernière indique qu'un système à barres est rigide en 2D s'il possède suffisamment de barres et que ses barres sont disposées de façon non-R-redondante ; de même, un GCSP est rigide si les contraintes qu'il contient retirent suffisamment de degrés de liberté à ses objets (condition vérifiée par le compte global des degrés de liberté) et que ces contraintes sont *placées* de façon non-R-redondante (condition assurée par le compte des degrés de liberté dans les sous-systèmes du GCSP).

Limites de la caractérisation structurelle

Nous avons vu que la notion de GCSP bien-rigide peut être mise en correspondance avec la notion de CSP bien-contraint (cf. fin de la section 2.3.1). Il en va de même pour les caractérisations structurelles de ces 2 notions ; la table 2.3 présente la classification d'un GCSP en bien-, sur- ou sous-contraint et bien- sur- ou sous-rigide en fonction du nombre de degrés de liberté qu'il possède :

DDL	< 0	0	$]0, \frac{d(d+1)}{2}[$	$\frac{d(d+1)}{2}$	$> \frac{d(d+1)}{2}$
contraint	sur	bien	sous		
rigide	sur			bien	sous

TAB. 2.3 – Lien entre CSP bien-, sur- et sous-contraint et GCSP bien-, sur- et sous-rigide

Les deux caractérisations structurelles sont donc équivalentes à $\frac{d(d+1)}{2}$ degrés de liberté près, c-a-d. à un repère local près, comme nous l'avons expliqué pour les notions de rigidité et bien-contraint. Nous avons montré et expliqué à la section 1.4.2, page 21, que la classification structurelle de CSP en bien-, sur- ou sous-contraint peut être fausse pour 3 raisons : inconsistance, redondance algébrique et singularité. Il en va de même pour la caractérisation structurelle de la rigidité d'un GCSP, comme le montre la table 2.4 qui est le pendant de la table 1.1 (page 22). Chaque type d'échec de la caractérisation structurelle est illustré sur un exemple de GCSP.

		rigidité		
		sur	bien	sous
rigidité structurelle	sur	-	R-redondance (ex.9)	R-redondance (ex.11)
	bien	inconsistance (ex.7)	-	R-redondance (ex.12)
	sous	inconsistance (ex.8)	singularité (ex.10)	-

TAB. 2.4 – Différences entre rigidité structurelle et rigidité d'un GCSP

Exemple 7 Sur-rigide, bien-s_rigide
Le GCSP $S = (O = \{P_1, P_2, P_3\}, C = \{$ *Distance*(P_1, P_2)*=3, Distance*(P_1, P_3)*=1, Distance*(P_2, P_3)*=1*$\})$ *en 2D est sur-rigide ; en effet, Le triangle formé par les points* P_1, P_2 *et* P_3 *doit vérifier* $Distance(P_1, P_2) \leq Distance(P_1, P_3) + Distance(P_2, P_3)$, *et l'ensemble de contraintes de* S *est donc inconsistant. Il est cependant bien-s_rigide :* $DDL(S) = 3$, *et pour tout* $S' \subset S$, $DDL(S') \geq 3$.

Exemple 8 Sur-rigide, sous-s_rigide
Le GCSP $S = (O = \{P_1, P_2, P_3, P_4\}, C = \{Distance(P_1, P_2) = 3, Distance(P_1, P_3) = 1, Distance(P_2, P_3) = 1\})$ *en 2D est sur-rigide ; en effet, Le triangle formé par les points* P_1, P_2 *et* P_3 *doit vérifier* $Distance(P_1, P_2) \leq Distance(P_1, P_3) + Distance(P_2, P_3)$, *et l'ensemble de contraintes de* S *est donc inconsistant. Il est cependant sous-s_rigide :* $DDL(S) = 5 > 3$, *et pour tout* $S' \subset S$, $DDL(S') \geq 3$.

Exemple 9 Bien-rigide, sur-s_rigide
Le GCSP $S = (O = \{P_1, P_2\}, C = \{Distance(P_1, P_2) = 1, Distance(P_1, P_2) = 1\})$ *en 2D est bien-rigide ; en effet, la distance entre les points* P_1 *et* P_2 *est spécifiée 2 fois mais de façon R-redondante. Cependant,* S *est sur-s_rigide :* $DDL(S) = 2 < 3$.

Exemple 10 Bien-rigide, sous-s_rigide
Le GCSP $S = (O = \{P_1, P_2, P_3\}, C = \{Distance(P_1, P_2) = 1, Distance(P_2, P_3) = 0\})$ *en 2D est bien-rigide ; en effet, le point* P_3 *est confondu avec le point* P_2 *et le segment* P_1P_2 *est un GCSP bien-rigide. Cependant,* S *est sous-s_rigide :* $DDL(S) = 4 > 3$, *et pour tout* $S' \subset S$, $DDL(S') \geq 3$. *Ceci est dû à la singularité de la contrainte* $Distance(P_2, P_3) = 0$ *: une telle contrainte n'est pas une contrainte de distance, mais plutôt une contrainte de coïncidence entre les points* P_2 *et* P_3.

Exemple 11 Sous-rigide, sur-s_rigide
Le GCSP $S = (O = \{P_1, P_2, P_3\}, C = \{Distance(P_1, P_2) = 1, Distance(P_1, P_2) = 1\})$ *en 2D est sous-rigide ; en effet, la distance entre les points* P_1 *et* P_2 *est spécifiée 2 fois de façon R-redondante mais le points* P_3 *peut être déplacé indépendamment des 2 autres points, ce qui constitue une déformation. Cependant,* S *est sur-s_rigide :* $DDL(S)' = 2 < 3$ *avec* $S' = (O = \{P_1, P_2\}, C = \{Distance(P_1, P_2) = 1, Distance(P_1, P_2) = 1\})$. *Ceci est dû à la R-redondance des 2 contraintes du GCSP.*

Exemple 12 Sous-rigide, bien-s_rigide
$S = (O = \{D_1, D_2, D_3\}, C = \{Angle(D_1, D_2) = \frac{\pi}{3}, Angle(D_1, D_3) = \frac{\pi}{3}, Angle(D_2, D_3) = \frac{\pi}{3}\})$ est un GCSP sous-rigide en 2D ; en effet, les 3 angles spécifiés entre les droites D_1, D_2 et D_3 sont R-redondants puisque la somme des angles d'un triangle en 2D vaut π, ce qui permet à chacune des 3 droites d'être translatée indépendamment n'importe où dans le plan 2D. Cependant, S est bien-s_rigide : $DDL(S) = 3$, et pour tout $S' \subset S$, $DDL(S') \geq 3$.

Une autre limite

Les limites que nous venons de présenter sont connues de la communauté CAO qui utilise la rigidité structurelle comme caractérisation pour identifier des sous-GCSP rigides. Cependant, il existe une autre limite à cette caractérisation qui apparaît même dans le cas où le GCSP est générique et non R-redondant.

Par exemple, si l'on considère le segment constitué de 2 points et d'une contrainte de distance les liant en 3D, ce GCSP est bien rigide mais ne dispose que de 5 DDL. Un simple segment est donc considéré comme un GCSP sur-s_rigide, et par définition de la s_rigidité, tout GCSP contenant un tel segment est alors aussi sur-s_rigide.

Cette limite a déjà été constatée dans la communauté de CAO qui a introduit le concept de *sous-GCSP trivial* afin de pallier ce nouveau problème.

Définition 33 Sous-GCSP trivial
Soit S un GCSP en dimension d et $S' = (O', C')$ un sous-GCSP de S. S' est trivial *si $|O'| \leq d$.*

Proposition 6 s_rigidité et trivialité
La s_rigidité des sous-GCSP triviaux *n'a pas de signification.*

En considérant la proposition 6, la s_rigidité devient alors :

Définition 34 Rigidité structurelle (bis)
Un GCSP $S = (O, C)$ **non-trivial** *en dimension d est bien-s_rigide, si et seulement si $DDL(S) = \frac{d(d+1)}{2}$ et $\forall S'$ sous-GCSP* **non-trivial** *de S, $DDL(S') \geq \frac{d(d+1)}{2}$.*

Il est sur-s_rigide, si et seulement si $\exists S'$ sous-GCSP **non-trivial** *de S tel que $DDL(S') < \frac{d(d+1)}{2}$.*

Il est sous-s_rigide, si et seulement si $DDL(S) > \frac{d(d+1)}{2}$ et $\forall S'$ sous-GCSP **non-trivial** *de S, $DDL(S') \geq \frac{d(d+1)}{2}$.*

Pour montrer que cette modification de la rigidité structurelle ne règle en rien le problème que nous avons mentionné, considérons un autre exemple : soit S le GCSP en 3D constitué de 4 droites A, B, C et D liées par 8 contraintes : Parallélisme(A, B), Parallélisme(B, C), Parallélisme(C, D), Distance(A, B), Distance(A, C), Distance(B, C), Distance(A, D), Distance(C, D). Ce GCSP est non-trivial et bien-rigide en 3D : il est constitué de 4 droites parallèles à distances prescrites et donc ne pouvant s'éloigner ou

tourner indépendamment les unes autour des autres. Cependant, ce GCSP ne dispose que de 5 DDL : 4 pour chacune des droites, -2 pour chaque parallélisme et -1 pour chaque distance.

Qui plus est, la s_rigidité (bis) perd la capacité à détecter certaines sur-rigidités que la s_rigidité peut pourtant détecter. Considérons par exemple le GCSP en 3D constitué d'un point A et d'une droite B liés par 2 contraintes de distance. Ce GCSP est R-redondant, puisque la distance de A à B est prescrite 2 fois. Dans le cas générique où ces 2 distances sont en plus différentes, ce GCSP est sur-contraint et donc sur-rigide. Par ailleurs, il dispose de 5 DDL et est donc sur-s_rigide. Or il s'agit d'un GCSP trivial et il n'est donc pas caractérisé par la s_rigidité (bis). Ainsi, tout GCSP incluant un sous-GCSP semblable ne sera pas détecté comme sur-s_rigide (bis), bien qu'il soit sur-s_rigide et sur-rigide.

La définition 34 ne devrait donc pas être utilisée car elle ne résout pas le problème de la s_rigidité et y introduit un nouveau problème.

En fait, la taille du GCSP (en nombre d'objets) n'a pas d'influence sur cette limite de la s_rigidité ; cette erreur est plutôt liée à l'utilisation de la valeur unique $\frac{d(d+1)}{2}$ dans la définition de s_rigidité. En effet, cette valeur représente le nombre de déplacements indépendants *maximum* dans un espace géométrique (cf. lemme 1, page 39), or il existe des GCSP ne disposant pas de tous ces mouvements. Ces GCSP sont généralement considérés singuliers, car l'absence de certains déplacements indique que leur ensemble d'objet ne remplit pas un sous-espace géométrique de dimension maximale. Cependant, nous avons vu que le segment n'est pas singulier mais pose pourtant problème, alors que les 4 droites parallèles sont obligatoirement en configurations singulières à cause des contraintes de parallélisme. Nous approfondirons cette analyse et proposerons une solution au chapitre 6.

Intérêt de la caractérisation structurelle

La caractérisation structurelle peut être vérifiée en temps polynomial et fournit, à l'instar de la formule de Grübler, une information générale sur le GCSP : toutes les solutions génériques du GCSP sont rigides. Ceci permet d'étudier la rigidité d'un GCSP uniquement à partir de sa structure, contrairement aux approches algébriques qui consistent à étudier les mouvements d'une solution particulière d'un GCSP.

Les approches à base de règles constituent la seule alternative pour une étude globale de la rigidité. Elles sont cependant nécessairement incomplètes lorsque l'on considère les GCSP quelconques ; nous avons vu (section 2.1.7, page 43) qu'aucun ensemble de règle ne permet d'identifier tous (et uniquement) les systèmes à barres rigides en 3D.

Le concept de rigidité structurelle est globalement équivalente à la notion de densité utilisée dans [Hoffmann *et al.*, 1997], à la propriété P énoncée dans [Lamure and Michelucci, 1998] et à d'autres caractérisations utilisées par la communauté CAO [Bouma *et al.*, 1995; Owen, 1991; Verroust *et al.*, 1992] pour identifier des sous-systèmes pouvant être résolus dans un repère local.

L'algorithme de flot proposé par Hoffmann *et al.*pour détecter des sous-systèmes denses est l'algorithme le plus général qui permet d'identifier des GCSP s_rigides en dimension quelconque. Nous présenterons cet algorithme en détail au chapitre 5.

2.4 En résumé ...

La théorie de la rigidité a été développée principalement dans la communauté de topologie structurale. Cette communauté a utilisé une approche mathématique du problème de la caractérisation de système à barres rigides. Une caractérisation algébrique en dimension quelconque est toujours possible, mais limitée à l'étude d'une configuration particulière d'un système à barres et basée sur des méthodes formelles de preuve de théorème dont la complexité est exponentielle. L'étude de la rigidité de premier ordre apporte une solution à ce problème en ramenant le système à barres à une forme linéaire. Cependant, cette étude est toujours limitée à une seule configuration d'un système à barres. Une autre approche consiste à étudier la rigidité d'un système à barres en toute configuration générique à partir d'une caractérisation structurelle de ces systèmes. Cette caractérisation, basée sur des propriétés de graphe, est complète en 2D. Les systèmes à barres génériquement rigides en 3D n'ont pour l'instant pas de caractérisation structurelle polynomiale.

La communauté de théorie des mécanismes est intéressée par l'étude des mouvements dans des mécanismes en 2D et 3D. Cette étude propose elle aussi une approche spécifique qui permet d'étudier exactement les mouvements admis par un mécanisme en une configuration particulière de celui-ci, et une approche plus générale basée sur un compte des degrés de liberté dans le mécanisme, la formule de Grübler. Cette dernière caractérisation est polynomiale et s'applique à toute configuration générique de tout mécanisme non-redondant.

L'étude de la rigidité de GCSP constitués d'objets et de contraintes quelconques est plus difficile. Nous avons proposé une définition des mouvements d'un GCSP quelconque, et une caractérisation des mouvements qui sont des déplacements. Cependant, la caractérisation formelle n'est pas possible en toute généralité, c-a-d. sans connaître *a priori* les types d'objets et de contraintes contenus dans le GCSP. Une approche structurelle proche de celle proposée en topologie structurale et en théorie des mécanismes est alors proposée. Elle permet de caractériser de façon heuristique mais polynomiale des GCSP globalement rigides, c-a-d. des GCSP dont toutes les solutions sont rigides. Cette caractérisation n'est ni complète ni correcte mais est considérée comme une bonne approximation pour identifier de GCSP globalement rigides. Nous discuterons à nouveau des limites de l'identification structurelle de GCSP rigides au chapitre 6. Nous proposerons alors une nouvelle caractérisation, la *rigidité structurelle étendue*, plus conforme à la rigidité car renseignée sur les propriétés géométriques du GCSP considéré.

Dans le chapitre suivant, nous introduisons plusieurs méthodes de résolutions pouvant être utilisées pour la résolution de GCSP décomposés. Nous discuterons alors des critères nécessaires pour une telle résolution, et justifierons notre choix de la méthode de résolution par intervalles.

Chapitre 3

Méthodes de résolution

Sommaire

Ce chapitre présente un ensemble de méthodes pouvant être utilisées pour la résolution de GCSP. Nous les classons en deux catégories principales : les *méthodes formelles* qui procèdent par manipulation symbolique du système d'équations à résoudre, et les *méthodes numériques* qui procèdent par approximations numériques successives des solutions.

Avant de décrire ces méthodes, nous rappelons brièvement les caractéristiques des systèmes d'équations et d'inégalités correspondant à des GCSP : les équations sont non-linéaires, généralement algébriques mais parfois non-algébriques (cf. section 1.3, page 18). De plus, afin de pouvoir résoudre efficacement des GCSP de grande taille, nous allons utiliser des méthodes de décomposition. Ces méthodes impliquent deux critères supplémentaires quant à la méthode de résolution que nous pouvons utiliser : elle doit être complète, c-a-d. capable de fournir toutes les solutions d'un sous-système, et correcte, c-a-d. capable de fournir des solutions exactes pour chaque sous-système. Nous revenons sur ces critères en fin de chapitre avant de discuter des méthodes de résolution pouvant être utilisées pour résoudre un GCSP décomposé.

Dans ce chapitre, nous présentons quelques méthodes de résolution répondant à ces critères, et évoquons d'autres méthodes de résolution parfois utilisées pour résoudre des GCSP. Nous découpons cet exposé en 3 sections : une section pour les méthodes formelles, une pour les méthodes numériques et enfin une dernière qui offre une comparaison de ces méthodes et motive notre choix de la méthode numérique de résolution par intervalles.

3.1 Méthodes formelles

Les méthodes formelles ont pour objectif la résolution exacte et formelle d'un système d'équations. Les variables sont généralement considérées comme des inconnues complexes, mais parfois aussi réelles ou rationnelles. Les principaux avantages de ces approches sont la correction et la complétude : elles fournissent l'ensemble des solutions sous forme symbolique, sans erreur d'arrondi numérique. Leurs principales limites sont, de façon générale, liées aux types de systèmes gérés (systèmes algébriques uniquement) et à la complexité des algorithmes qu'elles mettent en jeu (souvent exponentielle).

Parmi ces méthodes, on distingue les méthodes générales de résolution de systèmes algébriques et les méthodes dédiées plus particulièrement aux systèmes algébriques issus de GCSP. Dans les sections qui suivent, nous allons donner un aperçu de quelques méthodes dans ces deux catégories et évoquer la façon dont elles ont été mises en œuvre pour la résolution de GCSP le cas échéant.

3.1.1 Bases de Gröbner

La méthode des bases de Gröbner est l'une des plus classiques approches formelles. Cette méthode est issue de la théorie des idéaux polynomiaux. Un idéal est un espace de polynômes généré par une famille de polynômes par opération d'addition interne[1] et de multiplication externe[2], c-a-d. qu'un polynôme p appartient à l'idéal engendré par un ensemble de polynômes $(p_1, p_2, ..., p_n)$ si et seulement si $p{=}q_1 * p_1 + q_2 * p_2 + ... + q_n * p_n$ où $(q_1, q_2, ..., q_n)$ est un ensemble de polynômes quelconques.

Le principe de la méthode, initialement proposée par Buchberger [Buchberger, 1985], est de trouver une famille génératrice indépendante, c-a-d. une base de l'idéal engendré par un système polynomial $S = (p_1, p_2, ..., p_n)$. Cette famille, appelée base de Gröbner, a la particularité de posséder les mêmes solutions que S et d'être plus facilement soluble. Le calcul d'une base de Gröbner peut être effectué en temps exponentiel dans le degré du système S, c-a-d. le produit des degrés des équations du système S, par l'algorithme de Buchberger [Buchberger, 1985]. La base obtenue dépend du choix d'un ordre sur les termes des polynômes considérés ; on peut par exemple considérer l'ordre lexicographique induit par un ordre total sur les noms des variables. Cet ordre a un intérêt particulier pour la résolution d'un système S car il permet l'obtention d'une base de Gröbner *quasiment*

[1] L'addition interne ne permet d'additionner que des polynômes membres de l'idéal.

[2] La multiplication externe permet de multiplier un polynôme membre de l'idéal par un polynôme quelconque.

triangulée. Le système correspondant à cette base peut alors être résolu par simple substitution, à la façon d'un système linéaire. Ainsi, le calcul d'une base de Gröbner permet de calculer formellement toutes les solutions d'un CSP algébrique S en calculant les solutions d'un système plus aisément soluble.

La méthode des bases de Gröbner est particulièrement utilisée en preuve de théorème géométrique ; elle a été intégrée au logiciel GEOTHER qui effectue de la preuve géométrique en 2D [Chou, 1988; Wang, 2002]. La méthode des bases de Gröbner a aussi été utilisée pour la résolution de GCSP [Kondo, 1992; Ericson and Yap, 1988].

L'inconvénient de l'approche est principalement sa complexité exponentielle ; certaines heuristiques permettent d'améliorer les performances en pratique, mais ce ne sont que des heuristiques et la complexité reste exponentielle en pire cas. Ceci rend la méthode inadaptée pour attaquer des GCSP de grande taille. Le GCSP *Ponts*, par exemple, est constitué de 27 contraintes de distance et constitue donc un GCSP de degré total 2^{27}.

3.1.2 Méthode Ritt-Wu

La méthode de Ritt-Wu, aussi appelée méthode des ensembles caractéristiques, fut introduites par Ritt [Ritt, 1950] avant d'être redécouverte par Wu [Wu, 1986; Chou, 1988] dans le contexte de la preuve automatique de théorèmes géométriques. Elle est aussi destinée exclusivement aux systèmes polynomiaux.

Elle est basée sur le calcul de l'ensemble caractéristique associé à un système polynomial S. L'ensemble caractéristique est un ensemble de polynômes qui sont des combinaisons algébriques des polynômes de S. La propriété intéressante de l'ensemble caractéristique est que l'ensemble des solutions du système S est égal[3] à l'ensemble des solutions de l'ensemble caractéristique de S ; aussi, l'ensemble caractéristique représente un système plus simple à résoudre que S.

Le calcul de l'ensemble caractéristique d'un système polynomial S est basé sur l'algorithme de pseudo-division de polynômes. Cet algorithme est de complexité exponentielle, ce qui confère à la méthode les mêmes limites que celles de la méthode des bases de Gröbner.

Cette méthode est principalement utilisée pour la preuve de théorème en géométrie [Wang, 2002], tâche pour laquelle elle fut initialement proposée. Wang affirme d'ailleurs que cette méthode permet de résoudre beaucoup de problèmes de ce type trouvés dans la littérature. Cependant, la résolution de grands GCSP par cette approche est aussi très coûteuse.

3.1.3 Méthode du résultant

Parmi les autres approches de calcul formel, on peut citer la *méthode des résultants* [Gelfand, 1994] qui est basée sur la théorie des déterminants. Elle s'applique uniquement aux systèmes algébriques dont la dimension de l'espace des solutions (complexes) est nulle, c-a-d. les systèmes bien-contraints.

[3] sauf introduction de singularités dues aux combinaisons algébriques qui peuvent faire apparaître des *racines parasites*.

Le résultant est la généralisation aux systèmes polynomiaux de la notion de déterminant pour les systèmes linéaires : une application linéaire peut être représentée par une matrice ; le déterminant de cette matrice indique si le système admet un ensemble fini ou pas de solutions : si ce déterminant est nul, le système linéaire admet un ensemble indénombrable de solutions.

On peut calculer le résultant d'un système polynomial à partir d'une matrice, appelée matrice des résultants ; de telles matrices ont été proposées par Macaulay [Macaulay, 1902] ou encore la matrice Bézoutienne [Elkadi and Mourrain, 1998]. C'est le déterminant de cette matrice qui est appelé résultant du système polynomial. Si le résultant est non-nul, le système est de dimension 0. Dans ce cas, il est possible de générer les *opérateurs de multiplication*, des matrices qui permettent ensuite de calculer toutes les solutions du système S [Mourrain, 1999].

Cette méthode a été utilisée par Bondyfalat [Bondyfalat, 2000] pour la résolution du problème d'étalonnage de caméra en vision artificielle.

3.1.4 Méthodes géométriques

Les méthodes géométriques sont spécialisées dans la résolution de sous-GCSP particuliers. Les méthodes de construction formelle à la règle et au compas en sont un exemple. Une méthode géométrique prend en entrée un certain nombre d'objets déjà déterminés et produit les positions, orientations et dimensions d'autres objets géométriques. Par exemple, la méthode d'intersection droite-cercle prend en entrée les paramètres de la droite et du cercle et produit en sortie les coordonnées de l'un des points d'intersection. Ces méthodes sont câblées, c-a-d. que le système d'équations qu'elles résolvent est diagonalisé afin de déterminer chacune des variables de sortie de la méthode. Chaque méthode est donc un algorithme de calcul qui produit l'ensemble des solutions d'un petit GCSP, soit de façon formelle, soit de façon numérique. Par exemple, la méthode d'intersection droite-cercle peut produire les 2 solutions (cf. algorithme 1, page 69, où le choix de la solution est déterminé par la valeur du paramètre $SolNb$).

Une méthode géométrique permet de résoudre un sous-GCSP particulier. Ne pouvant résoudre le GCSP entièrement, elles sont principalement utilisées en conjonction avec des méthodes de décomposition. Elles ont l'avantage de permettre une résolution formelle du GCSP, et d'être capable de gérer les singularités du GCSP. Par exemple, la méthode d'intersection droite-cercle peut traiter le cas où la droite et le cercle n'ont pas d'intersection ou une intersection unique. Ces méthodes sont donc très bien adaptées pour la résolution de GCSP. Cependant, elles ne peuvent être appliquées qu'aux GCSP auxquels elles sont dédiés et souffrent donc d'un manque de généralité. Malgré tout, elles sont utilisées dans la plupart des méthodes de décomposition géométrique à base de règles (cf. section 4.4, page 107). Elles peuvent aussi être utilisées avec l'algorithme GPDOF dans son adaptation aux GCSP (cf. section 4.2.1, page 98).

Algorithme 1 IntersectionCD$_{2D}$ (C : un cercle en 2D ; D : une droite en 2D ; $SolNb$: un entier) **retourne** P : un point en 2D

Requiert: $SolNb$, qui indique le numéro de la solution souhaitée, vaut 1 ou 2

Modélisation des objets : $C : \{x_o, y_o, r\}$, $D : \{a, b, c\}$, $P : \{x_p, y_p\}$

Si $(-(b^2+1)x_p + aby_p - ac)^2 + (-bc + abx_p - (a^2-1)y_p)^2 \leq r^2$ $\{D1 \cap C1 \neq \emptyset\}$ **Alors**

Si $a = 0$ $\{\Rightarrow b = 1\}$ **Alors**

$y_p \leftarrow -c$

Si $SolNb = 1$ **Alors**

$x_p \leftarrow x_o + \sqrt{(y_o + c)^2 + r^2}$

Sinon

$x_p \leftarrow x_o - \sqrt{(y_o + c)^2 + r^2}$

Fin Si

Sinon

Si $SolNb = 1$ **Alors**

$y_p \leftarrow -abx_o + a^2 y_o - bc + a\sqrt{-2abx_o y_o - 2bcy_o - b^2 y_o^2 + b^2 r^2 - 2acxo - a^2 x_o^2 - c^2 + a^2 r^2}$

Sinon

$y_p \leftarrow -abx_o + a^2 y_o - bc - a\sqrt{-2abx_o y_o - 2bcy_o - b^2 y_o^2 + b^2 r^2 - 2acxo - a^2 x_o^2 - c^2 + a^2 r^2}$

Fin Si

$x_p \leftarrow \frac{-by_p - c}{a}$

Fin Si

Sinon

$x_p \leftarrow y_p \leftarrow -\infty$ $\{C$ *et* D *n'ont pas d'intersection*$\}$

Fin Si

Return P

3.2 Méthodes numériques

Les méthodes numériques peuvent en général résoudre tout système d'équations et d'inégalités, mais font face au problème de l'inexactitude du calcul numérique. En effet, s'il est possible de résoudre exactement de façon numérique les systèmes linéaires (sous réserve de savoir calculer de façon exacte dans le corps des coefficients), cela est impossible de façon générale sur les systèmes d'équations quelconques, mêmes s'ils sont limités aux équations algébriques. Ainsi, la grande majorité des méthodes numériques produisent des approximations des solutions, pouvant parfois s'avérer totalement fausses. Les méthodes de résolution par intervalles lèvent ce problème en permettant d'effectuer des calculs numériques de façon robuste. Elles produisent des approximations exactes des solutions, représentées par des intervalles dont la finesse peut-être réglée à souhait.

Dans cette section, nous ne décrivons que brièvement les méthodes numériques itératives classiques, comme la méthode de Newton-Raphson, car celles-ci ne répondent pas aux critères attendus pour pouvoir résoudre un GCSP à l'aide d'une méthode de décomposi-

tion : elles ne fournissent qu'au mieux une solution du système à résoudre et ne sont donc pas complètes ; elles peuvent produire des solutions fausses si des erreurs d'arrondi sont faite durant les calculs, et ne sont donc pas correctes.

3.2.1 Méthodes itératives *classiques*

Les méthodes numériques itératives, aussi appelées méthodes numériques locales, procèdent toutes du même principe : elles calculent les termes d'une suite définie de façon inductive, dont la base est généralement fournie et qui doit converger vers une solution (approchée) en un nombre fini d'itérations. Parmi les méthodes de ce type, la plus connue est sans doute la méthode de Newton-Raphson, qui utilise la jacobienne J du système $F(X)$ à résoudre dans la définition récursive de la suite : $X_{n+1} = X_n - J^{-1}(X_n)F(X_n)$.

Cette méthode à l'avantage de converger très rapidement vers une solution si la valeur initiale X_0 fournie est *adéquate*. Elle implique cependant certaines conditions ($\forall n \geq 0, det(J(X_n)) \neq 0$) qui, si elles ne sont pas vérifiées à l'une des étapes de la méthode, peuvent faire échouer la résolution. Enfin, les bassins d'attraction à partir de la valeur initiale ont des formes fractales [Peitgen and Richter, 1986] : il en résulte que même si la valeur initiale est topologiquement proche d'une solution, la méthode peut converger vers une solution plus lointaine, ou ne pas converger. Par exemple, le CSP présenté à la figure 3.1 possède 1 solution $x = 0$, et 2 solutions en $x \to \pm\infty$. Or, si la valeur initiale est prise hors de l'intervalle $]-a, +a[$, la méthode ne converge pas (c-a-d. converge vers l'une des 2 *solutions* à l'infini).

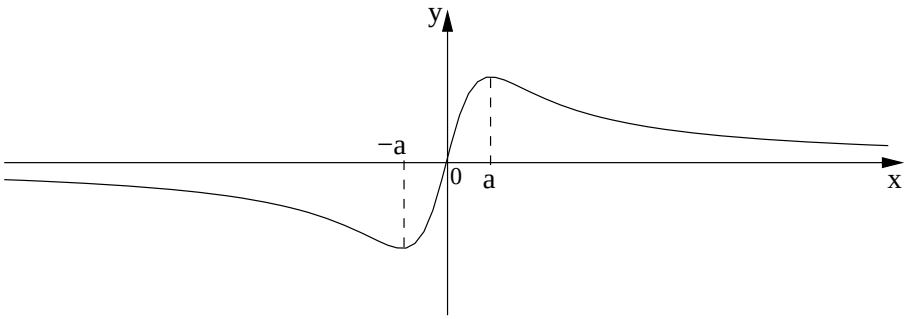

FIG. 3.1 – Représentation graphique de la fonction $y = \frac{x}{x^2+a^2}$

D'autres méthodes sont issues de l'optimisation locale [Dennis and Schnabel, 1983]. Afin d'être adaptées aux problèmes de satisfaction, on peut définir la fonction objectif comme la somme des contraintes violées, ce qui permet d'atteindre la satisfaction lorsque la fonction objectif atteint 0. D'autres méthodes encore utilisent une fonction objectif ayant un sens pour un domaine d'application particulier : distance à une esquisse en terme de somme de moindres carrés en dessin, minimisation d'une fonction énergétique en conformation moléculaire, etc. Ces méthodes peuvent cependant rester bloquées dans un minimum local de la fonction objectif, ce qui ne permet pas de garantir la pertinence de la solution trouvée.

Les méthodes numériques ne peuvent pas être utilisées dans notre contexte pour la résolution de GCSP car elles ne remplissent pas les conditions nécessaires à l'utilisation d'une décomposition (complétude et correction). Elles ont cependant été très utilisées pour la résolution de GCSP non décomposés dans la communauté de CAO ou DAO[4] en raison de leur efficacité [Sutherland, 1963; Borning, 1979; Heydon and Nelson, 1994; Light *et al.*, 1981].

3.2.2 Méthode par homotopie

La méthode d'homotopie, aussi appelée méthode par continuation homotopique, remonte au début du 20^e siècle [Lahaye, 1934]. Elle a depuis été utilisée pour la résolution de GCSP issus de la mécanique, robotique ou encore CAO [Morgan, 1983; Wampler *et al.*, 1990; Michelucci and Lamure, 1994; Durand, 1998]. Cette méthode consiste à déformer les solutions d'un système simple en solutions du système à résoudre en suivant les chemins de ces solutions, généralement dans l'espace complexe.

Plus formellement, on définit la fonction homotopique d'un système F comme suit :

Définition 35 Homotopie
Soit F un système d'équations sur l'ensemble de variables X, $X \in \mathbb{X}$, produisant des valeurs dans $\mathbb{S}$. La fonction homotopique $H : \mathbb{X} \times [0,1] \rightarrow \mathbb{S}$ de F est définie par $H(X,t) = (1-t)G(X) + tF(X)$; $G : \mathbb{X} \rightarrow \mathbb{S}$ est un système d'équations possédant autant de solutions que F.

La méthode homotopique consiste à faire varier itérativement la valeur de t de 0 à 1 pour faire évoluer, étape par étape, la fonction homotopique de G vers F ; plus exactement, chaque solution X^* du système G est associée à une courbe $H(t, X^*) : [0,1] \rightarrow \mathbb{R}$ ou $\mathbb{C}$; la méthode homotopique consiste à *suivre* itérativement ces courbes en partant de $t = 0$ afin d'obtenir les solutions de F lorsque $t = 1$.

Définition du système initial

Le choix du système G est une étape cruciale de la méthode : G doit avoir au moins autant de solutions que F, et toutes ces solutions doivent être connues pour servir de points de départ aux chemins homotopiques. Lorsque le système F à résoudre est polynomial, il existe une définition standard pour le système G : G est constitué d'autant d'équations que F, chacune de ces équations étant de la forme $g_i = x_i^{d_i} - L(X)^{d_i}$ où $x_i \in X$, $L(X)$ est un polynôme linéaire de la forme $a_1x_1 + a_2x_2 + ... + a_nx_n + a_{n+1}$ où tous les coefficients a_i sont non-nuls et choisis aléatoirement dans l'espace réel (ou complexe) et les d_i sont les degrés de chacun des polynômes de F, de sorte que $\prod_{\forall i} d_i = \deg(F)$. Cette définition produit un système G possédant un ensemble de solutions connues en nombre borné (borne de Bézout), ce qui est en général une sur-estimation du nombre de solutions de F ; il existe des estimations plus fines du nombre de solutions données par le théorème de Bernstein par exemple.

[4] DAO = dessin assisté par ordinateur.

Lorsque le système F est non-algébrique, la question du dénombrement de solutions n'a pas de solution générale (cf. section 1.4, page 19) et il n'existe alors pas de définition standard pour le système G. Il est même impossible, de façon générale, de produire un système G ayant autant de solutions que le système F, et la résolution complète n'a pas de sens étant donné que le système F peut avoir un ensemble infini de solutions même s'il est bien contraint (cf. figure 1.1, page 19). La méthode homotopique est donc en théorie applicable à tout système F, mais identifier un système équivalent G dont les solutions sont connues et en nombre au moins égal à celles de F n'est réalisable en général que pour des systèmes F algébriques.

Algorithme 2 Homotopie (F : un système d'équations ; X : son vecteur d'inconnues) **retourne** S : ensemble des zéros de F

$G \leftarrow$ `Système-Equivalent(`F`)`
$S' \leftarrow$ `Ensemble-Des-Solutions(`G`)`
$H(t, X) \leftarrow t * F(X) + (1 - t) * G(X)$
Pour chaque $X_0 \in S'$ **Faire**
  $t \leftarrow 0$
  Tant que $t \neq 1$ **Faire**
    $(t', X_0') \leftarrow$ `Prediction(`t, X_0`)`
    $(t, X_0) \leftarrow$ `Correction(`t', X_0'`)`
  Fin Tant que
  $S \leftarrow S \cup \{X_0\}$
Fin Pour
Return S

Prédiction, correction

Une fois connu le système G et ses solutions, la méthode homotopique consiste à suivre les chemins homotopiques partant de chaque solution de G. Pour ce faire, la méthode procède de façon itérative par prédiction-correction. L'étape de prédiction consiste à proposer une approximation du prochain point sur chaque chemin homotopique, et l'étape de correction à améliorer la qualité de la prédiction par une méthode itérative.

L'algorithme général de la méthode homotopique est décrit à l'algorithme 2. La méthode de prédiction peut utiliser une linéarisation de H afin de produire une approximation du prochain point en suivant un gradient. La taille du pas est un paramètre qui peut varier en fonction du succès de l'étape de correction. Cette dernière est généralement basée sur une méthode numérique itérative qui consiste à réduire l'erreur de l'approximation jusqu'à la précision souhaitée. Cette étape peut ne pas converger, ou converger trop lentement ; il est alors possible de l'interrompre après un nombre d'itérations donné et de reprendre l'étape de prédiction avec un pas de taille inférieure. La taille du pas influe directement sur le nombre d'itérations nécessaires pour atteindre la valeur $t = 1$ et donc sur l'efficacité de la méthode.

De plus, il est possible que certains chemins homotopiques ne passent jamais par la valeur $t = 1$, et il faut donc borner le nombre d'itérations afin d'éviter de boucler dans ce cas. Enfin, il se peut que des chemins homotopiques se croisent ou se confondent, créant aussi des problèmes pour la méthode de suivi de chemin. Ces problèmes peuvent être résolus en suivant les chemins correspondants aux solutions complexes, bien que leur nombre soit supérieur à ceux correspondants aux solutions réelles.

Avantages et limites de l'homotopie

Cette méthode a l'avantage d'être plus robuste que les méthodes numériques itératives de type Newton-Raphson, et d'avoir un comportement plus intuitif car ses bassins d'attraction sont plus réguliers (ce sont des ensembles semi-algébriques)[5].

Elle permet l'obtention de toutes les solutions, au moins dans le cas algébrique bien-contraint[6]. Comme nous l'avons expliqué, dans le cas non-algébrique, la définition du système initial G peut poser quelques difficultés.

La thèse de Durand [Durand, 1998] utilise et compare plusieurs formes d'homotopie pour la résolution de sous-GCSP non décomposables. Ces sous-systèmes apparaissent typiquement dans la décomposition de GCSP 3D constitués de points, droites et plans sur lesquels portent des contraintes de distance et d'angle. Il démontre que l'approche automatique et standard est moins efficace qu'une approche basée sur une étude personnalisée de chaque système, utilisant parfois des simplifications formelles ou géométriques de sous-systèmes afin de permettre une résolution plus efficace.

3.2.3 Résolution par intervalles

La méthode de résolution par intervalles est issue du mariage de l'analyse par intervalles et des techniques de programmation par contraintes [Davis, 1987]. L'analyse par intervalles [Moore, 1966] permet un calcul numérique exacte en représentant les réels par des intervalles dont les bornes appartiennent à un sous-ensemble fini[7] de $\mathbb{R}$. La programmation par contraintes [Waltz, 1972; Montanari, 1974; Mackworth, 1977] utilise des opérateurs de réfutation, appelés opérateurs de *filtrage*, afin de parvenir à la *consistance* du CSP considéré. Il existe plusieurs formes de *consistances partielles*, la *consistance globale* étant atteinte lorsque le domaine de chaque variable est réduit à l'union des valeurs réelles de cette variable apparaissant dans les solutions du CSP.

[5] Cette propriété a d'ailleurs été utilisée dans [Essert-Villard, 2001] pour proposer une méthode de sélection automatique de la solution d'un GCSP utilisant une esquisse comme système initial.

[6] L'homotopie permet aussi de trouver des solutions à des GCSP sous-contraints [Michelucci and Lamure, 1994]. Pour ce faire, on sélectionne les solutions *intéressantes* du système de départ et on suit leurs chemins homotopiques. Lamure et Michelucci affirment que cela permet, à partir d'une esquisse servant de système initial, de faire converger l'homotopie vers une solution intuitivement proche de l'esquisse : un point contraint à être incident à une droite mais ne l'étant pas dans l'esquisse initiale se rapproche de la droite par le plus court chemin.

[7] En pratique, les implantations utilisent souvent les nombres flottants comme ensemble de valeurs pour les bornes.

La résolution par intervalles consiste à isoler les intervalles-solutions contenus dans les domaines des variables du CSP. Ainsi, la solution fournie ne sera pas une valeur numérique, mais un intervalle encadrant *probablement des solutions* du GCSP :

- *probablement*, car les opérateurs de filtrage sont des opérateurs de réfutation, c-a-d. qu'ils assurent qu'aucune solution n'est perdue ; autrement dit, s'il existe des solutions, elles se trouvent forcément dans les intervalles-solutions. Cependant, ces opérateurs ne peuvent décider s'il existe effectivement des solutions dans les intervalles solutions.
- *des solutions*, et non pas *une solution* car la résolution par intervalles utilise le principe dichotomique pour isoler les solutions, mais cette technique ne peut pas être appliquée au-delà d'une certaine précision. En particulier, lorsque les bornes d'un intervalle sont 2 valeurs consécutives dans l'ensemble des valeurs de bornes, par exemple 2 nombres flottants consécutifs, l'intervalle ne peut plus être affiné. Comme expliqué au point précédent, on ne peut de façon générale décider si l'intervalle contient alors une, plusieurs, ou même aucune solution[8].

Cette approche basée sur le calcul d'intervalles est cependant intéressante car elle permet d'effectuer du calcul numérique exact : l'arithmétique d'intervalles définit tous les opérateurs calculatoires sur les intervalles de façon à ce qu'aucune solution ne soit perdue.

La résolution par intervalles combine donc les avantages des méthodes numériques (efficacité, polynomial en espace) et des méthodes formelles (complétude, calcul exact).

Nous donnons dans les sections qui suivent les fondements techniques de la méthode : arithmétique d'intervalles, consistances, filtrage, découpage et résolution par intervalles.

Arithmétique d'intervalles

L'arithmétique d'intervalles permet le calcul sur des intervalles de $\mathbb{R}$. L'ensemble des intervalles de $\mathbb{R}$ est noté $\mathbb{I}$. Les bornes a et b d'un intervalle $[a, b]$ sont prises dans un sous-ensemble fini de $\mathbb{R}$, généralement les nombres flottants. L'ensemble des nombres flottants est noté $\mathbb{F}$. Afin d'assurer l'exactitude des calculs, le résultat de toute opération sur les nombres flottants est arrondi au nombre flottant inférieur (resp. supérieur) pour la borne inférieure (resp. supérieure) de l'intervalle.

On définit alors l'extension des fonctions de $\mathbb{R}^n \rightarrow \mathbb{R}$ aux intervalles [Moore, 1966; Hansen, 1992] :

Définition 36 Extension aux intervalles
Une fonction $F : \mathbb{I}^n \rightarrow \mathbb{I}$ *est une extension de la fonction* $f : \mathbb{R}^n \rightarrow \mathbb{R}$ *si et seulement si* $\forall I_1, ..., I_n \in \mathbb{I}$, $r_1 \in I_1$, ..., $r_n \in I_n \Rightarrow f(r_1, ..., r_n) \in F(I_1, ..., I_n)$.

Pour définir l'extension aux intervalles, il faut disposer d'une fonction permettant de traduire des réels en intervalles. L'opérateur I, appelé opérateur d'encadrement, transforme un ensemble de réels en un intervalle ; il est défini de la façon suivante :

[8] Des théorèmes, comme celui de Kantorovich par exemple, permettent de certifier l'existence et l'unicité de solution dans un intervalle

Définition 37 Opérateur d'encadrement
L'opérateur d'encadrement est une fonction de $\mathbb{R}^n$ *dans* $\mathbb{I}$ *qui transforme un ensemble* $E = \{e \in \mathbb{R}\}$ *en l'intervalle* $I(E) = [pred_{\mathbb{F}}(inf(E)), succ_{\mathbb{F}}(sup(E))[$, *où* $inf(E)$ *(resp.* $sup(E)$*) est le plus grand (resp. petit) réel contenu dans* E, *et* $pred_{\mathbb{F}}(k)$ *(resp* $succ_{\mathbb{F}}(k)$*) retourne le nombre flottant directement inférieur ou égal (resp. strictement supérieur) au réel* k.

Lorsque $E = \{k\}$, *l'ensemble réduit à un réel, on note* $I(k) = I(E)$.

L'*extension naturelle* aux intervalles [Moore, 1966; Neumaier, 1990] F d'une fonction réelle f consiste à substituer dans f toutes les constantes k par l'intervalle $I(k)$, toutes les variables par une variable équivalente sur intervalles, et tous les opérateurs par des opérateurs équivalents sur intervalles. Par exemple, l'extension naturelle de la fonction $f(x, y, z) = x + 3 * x * y + z * z$ est la fonction sur intervalles $F(X, Y, Z) = X + [3] * X * Y + Z * Z$. Les opérations arithmétiques sur intervalles se définissent à partir des opérations arithmétiques sur les réels comme suit :

- $[a, b] + [c, d] = [a + c, b + d]$
- $[a, b] - [c, d] = [a - d, b - c]$
- $[a, b] * [c, d] = [min(a * c, a * d, b * c, b * d), max(a * c, a * d, b * c, b * d)]$
- $[a, b]/[c, d] = [min(a/c, a/d, b/c, b/d), max(a/c, a/d, b/c, b/d)]$, $0 \notin [c, d]$

Les propriétés de l'arithmétique sur les intervalles ne sont pas les mêmes que celles de l'arithmétique sur les réels [Moore, 1966]. En particulier, la distributivité des opérateurs est perdue sur les intervalles. Ainsi $[1, 2] * ([1, 2] - [1, 2]) = [-2, 2]$ alors que $[1, 2] * [1, 2] - [1, 2] * [1, 2] = [-3, 3]$. Aussi, la définition syntaxique d'une fonction a-t-elle une importance capitale dans son extension naturelle aux intervalles.

Consistances

Un CSP est globalement consistant lorsque toutes les valeurs réelles contenues dans les domaines de ses variables appartiennent à au moins une solution du CSP. En particulier, la consistance globale est atteinte lorsque toutes les solutions du CSP sont connues. Etablir la consistance globale d'un CSP est très coûteux, aussi a-t-on généralement recours à des consistances locales (ou partielles) moins coûteuses [Lhomme, 1993; Faltings, 1994; Benhamou *et al.*, 1994] qui permettent la définition *d'opérateurs de filtrage* efficaces pour la résolution de CSP. Un opérateur de filtrage a pour but d'éliminer des parties de l'espace de recherche qui sont inconsistantes ; ce sont des opérateurs de réfutation : les parties éliminées ne contiennent aucune solution.

Parmi les consistances locales les plus utilisées, citons la 2B-consistance, 3B-consistance, Box-consistance et Bound-consistance. Une comparaison entre ces différentes consistances est proposée dans [Delobel, 2000]. Nous présentons ici la 2B-consistance [Lhomme, 1993] qui est une forme de consistance d'arc [Mackworth, 1977] affaiblie (car réduite aux bornes des domaines). Elle s'applique sur chaque contrainte d'un CSP : une contrainte c est 2B-consistante si les bornes du domaine de chaque variable liée à c disposent d'un support dans les domaines des autres variables liées à c. Plus formellement :

Définition 38 2B-consistance
Une contrainte $c \in C$ portant sur les variables $(x_1, ..., x_n)$ d'un CSP (X, D, C) est **2B-consistante** *si et seulement si $\forall i \in [1..n]$, $D_i = I\{v_i \in D_i | \exists v_1 \in D_1$, ..., $v_{i-1} \in D_{i-1}$, $v_{i+1} \in D_{i+1}$, ..., $v_n \in D_n$ tel que $c(v_1, ..., v_n)$ est vérifiée$\}$.*
Un CSP dont toute contrainte est 2B-consistante est dit **2B-consistant**.

On peut aussi exprimer la 2B-consistance directement à partir de l'extension aux intervalles d'une contrainte :

Proposition 7 Contrainte 2B-consistante
Une contrainte $c \in C$ portant sur les variables $(x_1, ..., x_n)$ et ne contenant pas d'occurrences multiples de ces variables est **2B-consistante** *si et seulement si $\forall x_i$ ayant pour domaine $D_{x_i} = [a, b]$:*
- *$c(D_{x_1}, ..., D_{x_{i-1}}, I\{a\}, D_{x_{i+1}}, ..., D_{x_n})$ est vérifiée et,*
- *$c(D_{x_1}, ..., D_{x_{i-1}}, I\{b\}, D_{x_{i+1}}, ..., D_{x_n})$ est vérifiée.*

Un CSP 2B-consistant n'admet pas forcément de solution. En effet, la 2B-consistance considère simplement les contraintes séparément et seulement les bornes des intervalles-domaines ; il se peut que la conjonction des contraintes, elle, ne soit pas consistante et induise l'absence de solution. Par exemple, le CSP $(\{x, y\}, \{D_x = [-2, +2], D_y = [-3, +3]\}, \{x^2 = 4, y^2 = 9, x * y = 1\})$ est 2B-consistant, mais n'admet pas de solution.

La définition de consistances plus fortes repose sur l'utilisation de consistances faibles pour refuter, de façon dichotomique, des parties de l'espace de recherche [Lhomme, 1993; Benhamou *et al.*, 1994].

Filtrage

Une consistance K est généralement associée à un opérateur de filtrage $\texttt{Filtrage}_{\mathrm{K}}$ qui transforme un CSP $P = (X, D, C)$ en $\texttt{Filtrage}_{\mathrm{K}}(P) = (X, D', C)$, un CSP équivalent, c-a-d. ayant les mêmes solutions, qui est K-consistant. Un opérateur de filtrage agit sur les domaines des variables d'un CSP. Il produit un nouveau domaine pour chaque variable qui est plus petit mais contient encore toutes les solutions. Les propriétés attendues d'un opérateur de filtrage sont les suivantes :

1. Contractance : $\forall d_i \in D$, $d'_i \subseteq d_i$
2. Correction : $\forall d_i \in D$ et $c \in C$, $d_i \cap c \subseteq d'_i$
3. Monotonie : $\forall d_i \in D$ et $e_i \subset d_i$, $e'_i \subset d'_i$
4. Idempotence : $\forall d_i \in D$, $d''_i = d'_i$

L'algorithme général des opérateurs de filtrage est décrit à l'algorithme 3. C'est l'opération $\texttt{filtrage}(D, x, c, \mathrm{K})$ qui calcule la réduction du domaine D à partir du calcul de K-consistance pour la variable x dans la contrainte c. Lorsqu'une telle réduction s'opère $(D' \neq D)$, toutes les variables liées à toutes les contraintes faisant intervenir la variable x sont à nouveau examinées. C'est donc un algorithme de point-fixe qui termine lorsqu'aucune réduction ne s'opère plus.

Algorithme 3 Filtrage$_{\mathrm{K}}$ $((X, D, C)$: CSP$)$ **retourne** (X, D', C) : CSP K-consistant

Assure: (X, D', C) est équivalent à (X, D, C)
  $Q \leftarrow \{(x_i, c_j) | c_j \in C$, et x_i intervient dans $c_j\}$
  Tant que $Q \neq \emptyset$ **Faire**
    $(x, c) \leftarrow$ `Element(`Q`)`
    $Q \leftarrow Q - \{(x, c)\}$
    $D' \leftarrow$ `filtrage(`$D, x, c,$ K`)`
    Si $D' \neq D$ **Alors**
      $Q \leftarrow Q \cup \{(x', c') | c' \neq c$ et x et x' interviennent dans $c'\}$
    Fin Si
  Fin Tant que
  Return (X, D', C)

La fonction `filtrage(`$D, x, c, 2B$`)` utilise la *projection de la contrainte* c sur la variable x afin de réduire le domaine D_x de x. La projection consiste à évaluer la contrainte c par l'arithmétique d'intervalles en remplaçant chaque variable $y \neq x$ qu'elle contient par le domaine D_y. La projection n'est généralement pas un intervalle et, pour éviter d'avoir à gérer les unions d'intervalles, on considère alors l'intervalle le plus petit englobant cette projection. C'est l'intersection de cet intervalle avec le domaine actuel D_x de x qui constitue le nouveau domaine D'_x.

Pour rendre possible le calcul des fonctions de projection, le CSP est mis sous forme simplifiée : toute occurrence multiple d'une variable dans une contrainte est éliminée par introduction d'une variable et d'une contrainte supplémentaires (processus de renommage) ; les contraintes sont généralement rendues ternaires. Ces simplifications améliorent les calculs de la 2B-consistance, mais limitent la puissance du filtrage. En effet, la 2B-consistance est une consistance locale qui ne considère qu'une contrainte à la fois. L'éclatement d'une contrainte en plusieurs contraintes du à la simplification induit donc une *perte d'information* pour la 2B-consistance : on ne peut filtrer directement sur les contraintes initiales, mais seulement sur les contraintes simplifiées.

En pratique, le filtrage d'un CSP peut être long et coûteux si l'on désire atteindre une précision au flottant près. On introduit donc un paramètre de précision ω qui définit la taille minimale des réductions prises en compte dans l'algorithme 3. Ainsi, si un appel à `filtrage(`$D, x, c,$ K`)` produit une réduction des bornes inférieure à ω, cette réduction ne sera pas comptabilisée et on considèrera que $D' = D$, ce qui n'entraînera pas l'examen de nouvelles contraintes. Cette modification fait perdre la propriété de confluence du filtrage : le point fixe atteint après un filtrage utilisant un ω dépend de l'ordre de traitement des contraintes.

Des techniques d'amélioration du filtrage ont été proposées [Lhomme *et al.*, 1998] : elles utilisent des techniques issues de l'analyse par intervalles, des méthodes d'extrapolation et la détection des cycles de convergence lente.

Résolution par intervalles

Les techniques de résolution par intervalles sont des méthodes de résolution globale : elles permettent de produire toutes les solutions d'un CSP sur domaines continus. Pour ce faire, elles utilisent 2 types d'opérations :

- $\texttt{Filtrage}_{\mathrm{K}}$, une opération de *filtrage* qui réduit la taille des domaines ; cette étape est généralement connue sous le nom de propagation d'intervalles dans la communauté de programmation par contraintes ;
- `Découpe`, une opération de *découpage*, généralisation du principe dichotomique, qui découpe un domaine en plusieurs sous-domaines, permettant une approche arborescente de l'exploration de l'espace de recherche.

Algorithme 4 RésolutionIntervalles ($P = (X, D, C)$: un CSP) **retourne** S : ensemble des intervalles-solutions de P

$P \leftarrow \texttt{Filtrage}_{\mathrm{K}}(P)$
Si $\forall d \in D$, $|d| < \omega_2$ **Alors**
  Return $S \cup D$
Sinon
  $x \leftarrow$ `ChoixVariable(`P`)`
  Pour chaque $P' \in$ `Découpe(`P, x`)` **Faire**
    $S \leftarrow S \cup$ `RésolutionIntervalles(`P'`)`
  Fin Pour
  Return S
Fin Si

L'algorithme général de résolution d'un CSP par intervalles est présenté à l'algorithme 4. Au premier appel, on considère que l'ensemble S est vide. D'autre part, on considère qu'une solution est trouvée lorsque tous les domaines atteignent la précision ω_2 souhaitée. Si le filtrage $\texttt{Filtrage}_{\mathrm{K}}$ ne permet pas de produire les solutions, on a recours à l'opérateur de découpage `Découpe(`x, P`)` qui découpe le domaine d'une variable x choisie selon une procédure de décision (`ChoixVariable`) et génère un CSP par sous-domaine résultant du découpage. L'ensemble des solutions de ces sous-CSP produit l'ensemble des solutions du CSP initial.

La résolution par intervalles permet donc une résolution numérique sans erreur de calcul et complète. En revanche, on ne sait pas décider en général si les intervalles-solutions produits contiennent on non des solutions.

3.3 Comparaison des méthodes de résolution

Dans cette section, nous proposons une comparaison entre les quelques méthodes de résolution que nous avons présentées dans ce chapitre (cf. table 3.1). Pour ce faire, nous évaluons chaque méthode face aux critères suivants :

- type : le type de CSP que peut traiter la méthode, c-a-d les restrictions que la méthode impose sur le domaine des variables, sur le type d'équations et sur le nombre de solutions attendues.
- comp, corr : complétude et correction de la résolution, c-a-d. aptitude à produire l'ensemble des solutions et à ne produire que des solutions.
- t1, tn, esp : les complexités en temps pour trouver 1 solution (t1) ou toutes les solutions (tn) et en espace (esp), en pire cas.

méthode	type	comp	corr	t1	tn	esp
Résolution formelle						
Gröbner	éq. algébriques	oui	oui	expo	expo	expo
Ritt-Wu	éq. algébriques	oui	oui	expo	expo	expo
Résultant	éq. algébriques CSP 0-dim.	oui	oui	expo	expo	expo
Géométrique	uniquement les règles implantées	oui	oui	poly	poly*	poly
Résolution numérique						
Newton-Raph.	pas d'inégalité	non	non	poly	-	poly
Homotopie	tous en théorie, algébrique en pratique	oui	non	poly	poly@	poly
Prop. Int.	tous	oui	calcul=oui sol.=non	expo	expo	poly

$*$: si le nombre de solutions (@ : et de chemins suivis) est polynomial.

TAB. 3.1 – Comparaison des méthodes de résolution de GCSP

3.3.1 Critères pour la résolution de GCSP décomposé

La décomposition d'un GCSP a pour but de produire un ensemble de sous-problèmes dont la résolution individuelle permettra l'obtention de solutions du GCSP initial. Afin de pouvoir résoudre efficacement un GCSP décomposé, nous proposons d'utiliser une méthode répondant à 3 critères : **complètude** (la méthode doit permettre de produire toutes les solutions des sous-problèmes) ; **correction** (la méthode doit ne produire que des solutions) ; **efficacité** (la méthode doit être la plus efficace possible).

Complétude

La complétude est un critère absolument nécessaire pour qu'une méthode de résolution puisse être employée pour la résolution d'un GCSP décomposé. Nous expliquons pourquoi

à l'aide du GCSP présenté en figure 3.2.

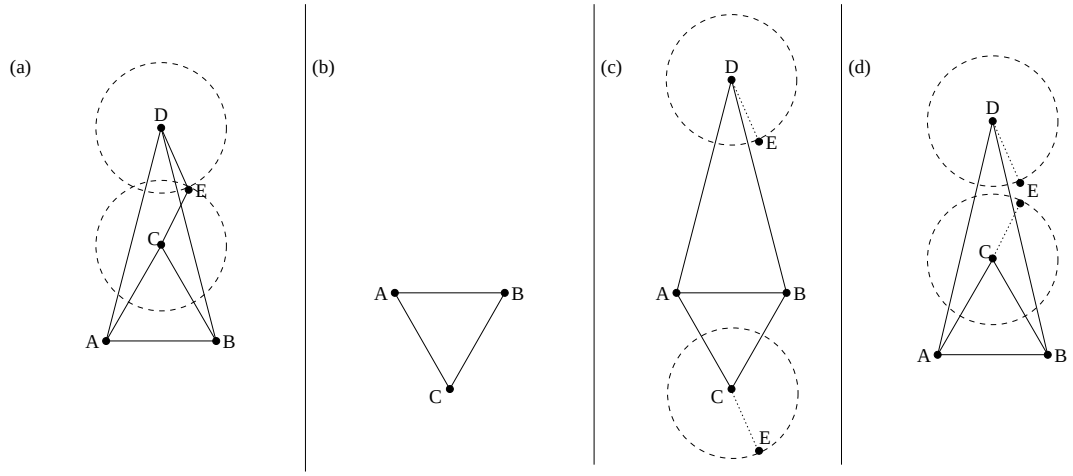

FIG. 3.2 – Importance de la complétude et de la correction de la résolution d'un GCSP décomposé

Ce GCSP en 2D est constitué de 4 points A, B, C, D et E liés par 7 contraintes de distance : Distance(A,B)=1, Distance(A,C)=1, Distance(A,D)=2, Distance(B,C)=1, Distance(B,D)=2, Distance(C,E)=0.6, Distance(D,E)=0.6. Ce GCSP admet une unique solution[9] représentée en figure 3.2-a. Supposons que ce GCSP soit décomposé en 3 sous-problèmes $S1$, $S2$ et $S3$: $S1$ consiste à résoudre le triangle ABC (3 distances), $S2$ à positionner le point D relativement à A et B (2 distances) et $S3$ à positionner le point E par rapport aux points C et D (2 distances). Supposons maintenant que ces 3 sous-problèmes soient résolus par une méthode incomplète, produisant une seule solution par sous-problème. Alors, la résolution de $S1$ peut produire la configuration présentée en figure 3.2-b, puis celle de $S2$ la configuration de D présentée en figure 3.2-c ; si cela se produit, le sous-problème $S3$ n'admet pas de solution car les cercles de rayon 0.6 centrés sur les points C et D produits pour $S1$ et $S2$ n'ont pas d'intersection (cf. figure 3.2-c). La résolution incomplète des sous-problèmes du GCSP décomposé peut donc ne produire aucune solution alors qu'il en existe.

C'est pourquoi, afin d'assurer qu'une solution soit trouvée s'il en existe une , il faut obligatoirement utiliser une méthode de résolution complète. De plus, cela permet d'identifier l'ensemble des solutions du GCSP.

Correction

La correction est aussi absolument nécessaire à la résolution d'un GCSP décomposé. En effet, comme exposé en début de section 3.3.1, la méthode choisie sera utilisée pour

[9]En fait, ce GCSP admet 1 autre solution qui est la symétrie de celle présentée par rapport à la droite passant par A et B.

résoudre des sous-problèmes indépendamment, dont les solutions seront ensuite combinées en solutions du GCSP initial. Si la solution proposée pour un sous-problème est fausse, il peut alors ne pas exister de solution dans un autre sous-problème ou pas de solutions pour le GCSP initial. Pour illustrer ce point, considérons à nouveau l'exemple présenté à la section précédente (cf. figure 3.2). Admettons que nous utilisons une méthode complète mais incorrecte. Alors, il est possible que la résolution du point D dans le sous-problème $S2$ produise une configuration de D fausse, comme illustré à la figure 3.2-d. La configuration fausse engendrée par la résolution incorrecte de $S2$ laisse alors $S3$ sans solution, car le point D est positionné de façon incorrecte trop loin de C, les 2 cercles n'ayant à nouveau pas d'intersection.

Il est donc crucial d'utiliser une méthode de résolution correcte si l'on veut pouvoir produire au moins une solution d'un GCSP décomposé.

Efficacité

L'efficacité de la méthode de résolution est un critère facultatif d'un point de vue théorique, mais absolument crucial d'un point de vue applicatif : la décomposition permet d'améliorer le temps de résolution d'un GCSP, comme nous l'expliquerons au chapitre suivant. Cependant, la résolution de chaque sous-problème engendré par la décomposition d'un GCSP doit être aussi efficace que possible afin de rendre la résolution du GCSP efficace.

3.3.2 Discussion

Nous reprenons la comparaison effectuée entre les différentes méthode de résolution exposées dans ce chapitre afin de discuter des choix possibles pour le couplage avec une méthode de décomposition.

Méthodes formelles

Les méthodes formelles sont complètes et correctes mais de complexité exponentielle en temps et en espace, ce qui les rend inapplicables pour la résolution de grands GCSP. En revanche, elle peuvent être utilisées pour la résolution de GCSP décomposés. Ainsi, on peut envisager d'avoir recours à ces méthodes pour résoudre les sous-GCSP de petite taille produits par une décomposition. Cependant, la décomposition que nous proposons n'assure pas que tous les sous-systèmes produits soient de petite taille, ni qu'ils soient tous algébriques, et on ne peut donc pas assurer que l'utilisation de ces méthodes sera efficace ou possible.

La méthode formelle à base de règles géométriques est un peu à part : chaque méthode étant câblée dans une procédure de calcul séparée, la complexité en espace et en temps dépend seulement de l'évaluation numérique des formules câblées dans l'algorithme de calcul, et est donc polynomiale. Cependant, c'est la taille du code de calcul qui peut être exponentielle, étant donné que le calcul de chaque solution y est implanté et que le nombre

de solutions peut être exponentiel. De plus, nous avons expliqué qu'un ensemble fini de telles méthodes ne permet pas la résolution de tout GCSP, tout comme un répertoire fini de règles géométriques ne permet pas la reconstruction de tout GCSP (cf. section 4.4, page 107). Malgré tout, on peut avoir recours à un répertoire de telles méthodes et les appliquer autant que possible car elles sont la forme de résolution la plus efficace et la plus adaptée pour un sous-système d'un GCSP décomposé. Cette approche coopérative entre des méthodes formelles géométriques et d'autres méthodes de résolution avait déjà été envisagée par Latham et Middleditch [Latham and Middleditch, 1996] (cf. section 4.1.2, page 93).

Méthodes numériques

Les méthodes numériques locales, telles que Newton-Raphson ne peuvent pas être utilisées pour la résolution de GCSP décomposés : elles ne fournissent qu'au mieux une solution par sous-GCSP, et celle-ci ne peut pas toujours être combinée avec la solution trouvée pour un autre sous-GCSP ; lorsque ce cas se produit, la méthode ne permet pas de trouver de solution du GCSP entier. Qui plus est, nous nous intéressons dans ce mémoire à la résolution complète de GCSP.

Les méthodes d'homotopie sont intéressantes et peuvent être utilisées pour la résolution de GCSP décomposés, comme le montre la thèse de Durand [Durand, 1998], à condition que le GCSP ne se compose que de contraintes et objets pouvant être définis de façon algébrique. Une limite à ces approches se trouve dans le nombre de chemins homotopiques à suivre : de façon générale, le nombre de chemins suivis est une sur-évaluation du nombre de solutions, ce qui nuit à l'efficacité de la méthode. Durand a montré que plusieurs ordres de grandeur peuvent être gagnés sur le temps de résolution en étudiant les sous-systèmes à résoudre au cas par cas afin de proposer une homotopie adaptée. De plus, des erreurs de calcul peuvent apparaître, sauf si l'on utilise une méthode de calcul en multi-précision qui nuit encore aux performances.

Enfin, la méthode de résolution par intervalles permet une résolution complète et exacte de tout CSP, mais ne peut assurer de façon générale l'existence de solutions effectives dans les intervalles-solutions qu'elle produit. C'est cependant la méthode que nous choisissons car elle est générale (permet de traiter toutes équations), complète et correcte au sens où les intervalles-solutions qu'elle fournit sont consistants avec les contraintes du problème.

Collaboration de résolveurs

Dans ce mémoire, nous proposons donc la mise en œuvre de la résolution par intervalles avec la méthode de décomposition par rigidification récursive. Cette mise en œuvre est présentée au chapitre 8. On doit considérer ce choix comme une illustration d'un processus de résolution complet et général, car le principe de la décomposition permet d'envisager la collaboration de plusieurs résolveurs : chaque sous-GCSP produit par la décomposition peut être résolu par une méthode de résolution différente, du moment qu'elle répond aux 3 critères énoncés en début de section 3.3.1 : complétude, correction, efficacité. Comme ce

dernier critère n'est pas universel[10], l'utilisation de plusieurs méthodes de résolution peut être intéressante.

On peut, par exemple, envisager de faire collaborer les méthodes formelles, les règles de résolution géométriques, l'homotopie et la résolution par intervalles dans un même outil de CAO :

- Les méthodes à base de règles géométriques seraient utilisées pour tous les sous-problèmes qu'elles peuvent résoudre, étant donné leur adéquation aux GCSP ;
- Les méthodes formelles pourraient traiter les sous-GCSP algébriques de petite taille afin de produire des sous-systèmes triangulés correspondants dont l'évaluation numérique (éventuellement par intervalles) serait ensuite très rapide ;
- Les méthodes d'homotopie pourraient être utilisées sur les sous-GCSP étudiés par Durand qui correspondent à des cas basiques de sous-GCSP apparaissant dans des décompositions ;
- La résolution par intervalles servirait alors de méthode générale, prenant le pas lorsque les autres méthodes ne sont pas adaptées ou échouent à résoudre un système en un temps borné.

Le chapitre suivant présente quelques méthodes de décomposition pouvant être utilisées pour simplifier la résolution de GCSP.

[10]La meilleure méthode de résolution n'existe pas : les performances de chaque méthode varient selon le problème à résoudre.

Chapitre 4

Méthodes de décomposition

Sommaire

La complexité de la résolution d'un CSP se mesure généralement à la taille de celui-ci, c-a-d. au nombre de variables, à la taille de leurs domaines et au nombre d'équations (et à leur *difficulté*) : pour des contraintes de même difficulté, plus grand est le CSP, plus longue est la résolution. De plus, la complexité de la résolution d'un CSP est bien souvent exponentielle en la taille de celui-ci, comme nous l'avons vu au chapitre 3. On conçoit alors que le temps de résolution d'un CSP sera généralement supérieur au temps

total de résolution de plusieurs CSP de taille inférieure. Les méthodes de décomposition se basent sur cette constatation et appliquent le principe *diviser pour régner* afin d'améliorer la résolution de CSP. Pour ce faire, elles cherchent à identifier une séquence de sous-problèmes pouvant être résolus séparément et dont les solutions respectives peuvent être combinées en des solutions du CSP initial.

Rappelons que la méthode de résolution employée pour résoudre un GCSP décomposé doit obligatoirement être complète et correcte (cf. section 3.3.1, page 79). Aussi, toutes les méthodes que nous allons présenter dans les sections suivantes et qui n'utilisent pas une méthode de résolution répondant à ces critères seront incomplètes et surtout inaptes à trouver même une unique solution.

On peut classer les méthodes de décomposition applicables aux GCSP en 2 catégories : les *méthodes de décomposition équationnelle* et les *méthodes de décomposition géométrique*. Les premières sont basées sur la décomposition d'un système d'équations *quelconques*, les secondes sont adaptées aux GCSP. Les méthodes de ces deux catégories peuvent être subdivisées en 2 schémas généraux : les *méthodes à base de règles* reposent sur l'utilisation d'un répertoire de modèles de sous-GCSP solubles qu'elles essayent de retrouver dans le GCSP à décomposer ; les *méthodes structurelles* se basent sur la structure du GCSP à décomposer pour identifier des sous-systèmes *structurellement* solubles. Les méthodes à base de règles sont généralement incomplètes mais peuvent être correctes (c-a-d. qu'elles ne permettent pas de décomposer tous les GCSP, mais les décompositions qu'elles produisent sont généralement solubles), alors que les méthodes structurelles sont généralement incorrectes mais peuvent être complètes (c-a-d. qu'elles permettent de décomposer la plupart des GCSP, mais les décompositions qu'elles produisent peuvent être insolubles).

Nous étudierons les différences générales entre ces quatre types de décomposition à la section 4.5, page 114.

Ce chapitre est découpé en 5 sections ; les 4 premières donnent un aperçu des travaux existants dans chacune des 4 classes de méthodes que nous avons définies : méthodes équationnelles structurelles, méthodes équationnelles à base de règles, méthodes géométriques structurelles et méthodes géométriques à base de règles. La dernière section propose une analyse critique des avantages et des limites de ces différentes méthodes et nous permet de motiver le choix de la méthode géométrique structurelle due à Hoffmann, Lomonosov et Sitharam [Hoffmann *et al.*, 1997] comme base pour notre propre méthode de résolution de GCSP.

4.1 Décompositions équationnelles structurelles

Les méthodes de décomposition équationnelle structurelle représentent la catégorie la plus générale des méthodes de décomposition : elles peuvent s'appliquer à tout CSP et utilisent uniquement la structure afin d'identifier des sous-problèmes solubles. La structure d'un CSP est généralement représentée par un *graphe variables-équations*, graphe biparti non-orienté $G = (X, C, E)$ dont les sommets $X \cup C$ sont les variables X et les contraintes C du CSP. Il existe une arête (x, c) entre une variable x et une contrainte c si x intervient dans

c. La figure 4.1 présente le graphe variables-équations correspondant au GCSP *Ponts*. Sur ce graphe, les sommets-variables sont représentés par des cercles et les sommets-contraintes par des carrés.

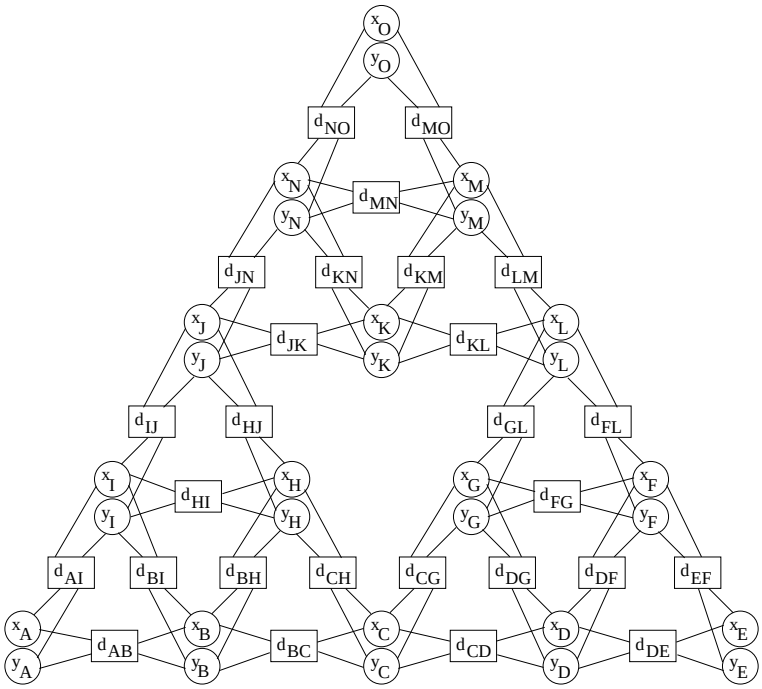

FIG. 4.1 – Graphe variables-équations représentant la structure du GCSP *Ponts*

Les méthodes de décomposition équationnelle structurelle sont toutes basées sur le principe du *couplage maximum* du graphe variables-équations. Nous rappelons ici les quelques définitions et propriétés liées au problème du couplage maximum.

Un *couplage* d'un graphe G est un sous-ensemble M d'arêtes du graphe G tel que, tout sommet de G a au plus une arête de M qui lui est incidente. Un sommet v de G est *couvert* par un couplage M si et seulement si il existe une arête de M incidente à v. Un couplage M est un *couplage maximum* de G si et seulement si $|M| \geq |M'|$ pour tout couplage M' de G, c-a-d. que le nombre de sommets couplés est maximum. Un couplage M est un *couplage parfait* si tout sommet de G est couvert par M. Le problème de la recherche d'un couplage maximum dans un graphe biparti est résolu en temps polynomial par exemple à l'aide d'un algorithme de flot [Cormen *et al.*, 1990].

4.1.1 Décomposition DM-CFC

La décomposition DM (Dulmage et Mendelsohn [Dulmage and Mendelsohn, 1958]) utilise un couplage maximum du graphe variables-équations d'un CSP pour le séparer en 3 composantes : structurellement bien-contrainte, structurellement sur-contrainte et structurellement sous-contrainte. Cette décomposition est issue de la théorie de l'élimination de

variables dans les matrices creuses pour la simplification de système d'équations linéaires. La décomposition CFC (pour composantes fortement connexes) produit des sous-systèmes structurellement bien-contraints de taille minimale en cherchant les composantes fortement connexes dans le graphe variables-équations d'un CSP associé à un couplage maximum. Ces 2 décompositions ont été associées pour la décomposition de CSP par Jégou, Michelucci et Ait Aoudia [Ait-Aoudia *et al.*, 1993] puis par Bliek, Neveu et Trombettoni [Bliek *et al.*, 1998]. Ces deux méthodes étant semblables dans leurs grandes lignes, nous proposons une description générale adaptée aux deux.

Décomposition DM

La décomposition DM est basée sur le théorème suivant dû à Dulmage et Mendelsohn :

Théorème 11 [Dulmage and Mendelsohn, 1958]
L'ensemble $X \cup C$ des sommets d'un graphe biparti $G = (X, C, E)$ peut être séparé en 3 sous-ensembles distincts, chacun pouvant être éventuellement vide :

- *P est l'ensemble des sommets pour lesquels il existe un couplage maximum ne les couvrant pas,*
- *Q est l'ensemble des sommets adjacents aux sommets de P,*
- *R est l'ensemble des sommets restants $(X \cup C) \setminus (P \cup Q)$.*

Cette décomposition est unique et induit 3 sous-graphes :

- *$G1 = (R1, R2, E1)$ où $R1 = R \cap C$, $R2 = R \cap X$ et $E1 = \{r_1 r_2 \in E | r_1 \in R1, r_2 \in R2\}$;*
- *$G2 = (P1, Q2, E2)$ où $P1 = P \cap C$, $Q2 = Q \cap X$ et $E2 = \{p_1 q_2 \in E | p_1 \in P1, q_2 \in Q2\}$;*
- *$G3 = (Q1, P2, E3)$ où $Q1 = Q \cap C$, $P2 = P \cap X$ et $E1 = \{q_1 p_2 \in E | q_1 \in Q1, p_2 \in P2\}$.*

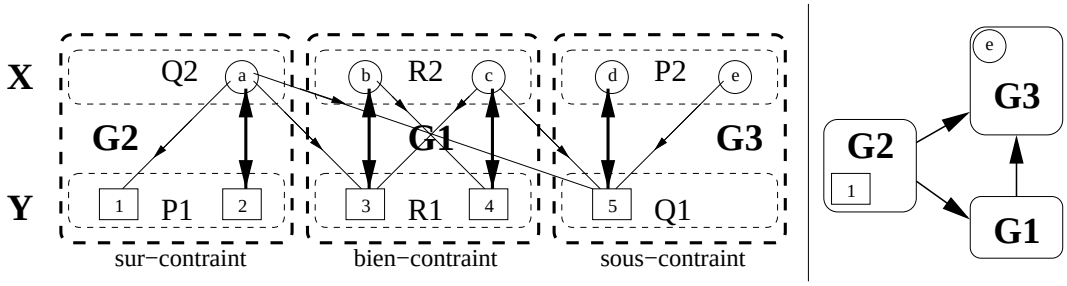

FIG. 4.2 – Décomposition DM d'un graphe biparti et DAG résultant

La figure 4.2 présente la décomposition DM d'un graphe biparti. On peut y voir les arcs d'un couplage maximum en gras, ainsi que les ensembles $P1$, $P2$, $Q1$, $Q2$, $R1$ et $R2$ et les graphes $G1$, $G2$ et $G3$.

La décomposition DM d'un graphe biparti $G = (X, C, E)$ peut être calculée à partir d'un couplage maximum M de G. Pour ce faire, il suffit de calculer un couplage M quelconque puis de suivre des *chemins alternés* dans le graphe G. Un chemin alterné dans un graphe G associé à un couplage M est un chemin de M dont une arête sur 2 appartient à M.

La composante $G2$ est le sous-graphe de G induit par l'ensemble des sommets atteignables par un chemin alterné partant de l'un des sommets de C non-couvert par le couplage M ; réciproquement, la composante $G3$ est le sous-graphe de G induit par l'ensemble des sommets atteignables par un chemin alterné partant de l'un des sommets de X non-couvert par le couplage M ; enfin, la composante $G1$ est induite par les sommets de G qui n'appartiennent ni à $G2$, ni à $G3$.

Le calcul du couplage maximum M peut être réalisé en temps polynomial, et le calcul des sommets atteignables à partir d'un sommet de G peut se faire à l'aide d'un simple parcours en temps linéaire dans la taille du graphe ; il en résulte que la décomposition DM peut être réalisée en temps polynomial.

La décomposition DM d'un graphe biparti satisfait les propriétés suivantes :

Propriété 3 Dulmage et Mendelsohn

1. *il n'y a pas d'arête entre $P1$ et $R2$, entre $P2$ et $R1$ et entre $P1$ et $P2$;*
2. *tout couplage maximum M de G se compose d'un couplage parfait de $G1$, d'un couplage maximum de $G2$ et d'un couplage maximum de $G3$, c-a-d. $|M| = |R1| + |Q1| + |Q2|$, $|R1| = |R2|$, $|P2| > |Q1|$, $|P1| > |Q2|$;*
3. *les arêtes entre $R1$ et $Q2$, entre $R2$ et $Q1$ et entre $Q1$ et $Q2$ n'appartiennent à aucun couplage maximum de G.*

Appliquée au graphe variables-équations $G_S = (X, Y, E)$ d'un CSP $S = (X, D, Y)$, la décomposition DM permet d'identifier les 3 composantes de S :

- $G1$ représente la composante *structurellement* bien-contrainte (autant de variables que d'équations car $|R1| = |R2|$),
- $G2$ représente la composante *structurellement* sur-contrainte (moins de variables que d'équations car $|P1| > |Q2|$),
- $G3$ représente la composante *structurellement* sous-contrainte (plus de variables que d'équations, car $|P2| > |Q1|$).

Décomposition CFC

La décomposition CFC d'un CSP consiste à identifier les composantes fortement connexes dans chacune des 3 composantes d'un CSP obtenues par décomposition DM ; une composante fortement connexe représente un circuit de taille maximale dans un graphe orienté. Une composante fortement connexe du graphe variables-équations admet un couplage parfait et contient autant de variables que d'équations ; elle correspond donc à un

sous-système structurellement bien-contraint du CSP. Les composantes fortement connexes produites par la décomposition CFC sont appelées *blocs* par la suite.

Pour identifier les blocs, on calcule tout d'abord un couplage maximum M du graphe variables-équations G d'un CSP ; notons qu'on peut réutiliser le couplage déjà calculé pour la décomposition DM. Ce couplage M induit la version orientée G' de G déjà présentée à la section précédente. Dans le graphe orienté G', identifier les composantes fortement connexes peut se faire en temps linéaire $O(|X| + |Y|)$ à l'aide d'un parcours en profondeur de G' et de ${}^tG'$ (graphe G' où tous les arcs ont été inversés) [Cormen *et al.*, 1990].

Le théorème suivant, dû à König, établit une propriété intéressante de la décomposition CFC :

Théorème 12 [König, 1916]
Tout couplage parfait d'un graphe biparti induit une décomposition unique en composantes fortement connexes.

Il résulte de ce théorème que la décomposition CFC de la composante bien-contrainte d'un CSP est unique et ne dépend donc pas du couplage maximum M calculé ; en effet, la propriété 3.2 indique que le couplage M réduit à la composante bien-contrainte est toujours parfait. En revanche, les blocs correspondants aux composantes fortement connexes des sous-graphes $G2$ (composante sur-contrainte) et $G3$ (composante sous-contrainte) ne sont pas toujours irréductibles. La figure 4.3 présente 2 couplages maximum d'un graphe biparti qui produisent des blocs différents.

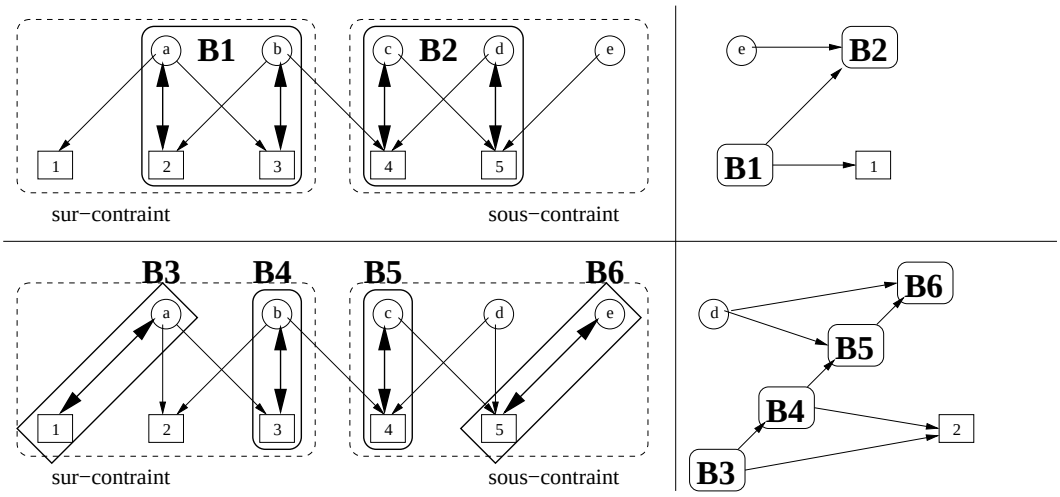

FIG. 4.3 – Décompositions CFC d'un même graphe biparti

La décomposition CFC induit aussi un ordre partiel pour la résolution des blocs. Cet ordre est obtenu de façon similaire à celui sur les composantes sur-, bien- et sous-contraintes : dans le graphe orienté G', on condense chaque bloc en un sommet unique pour obtenir un *graphe orienté sans circuit (DAG) de blocs* (cf. figure 4.3, parties droites).

Un arc d'un bloc $B1$ vers un bloc $B2$ indique que $B2$ utilise, pour sa résolution, la valeur de variables déterminées par la résolution de $B1$; $B2$ doit donc être résolu après $B1$. Un tri topologique [Cormen *et al.*, 1990] du DAG de blocs fournit une séquence de résolution des blocs.

La méthode proposée par Jégou, Michelucci et Ait Aoudia s'arrête là et résout les séquences de blocs correspondant aux composantes bien- et sur-contraintes par une méthode de type Krawczyck-Moore [Moore and Qi, 1982; Kearfott, 1987]. Ils expliquent que les contraintes non-couplées de la partie sur-contrainte peuvent être négligées pour la résolution, mais peuvent ensuite être vérifiées sur l'ensemble des solutions obtenues.

Leur méthode ne permet donc de résoudre que des GCSP bien- ou sur-contraints, sous réserve qu'ils ne contiennent ni A-redondances, ni singularités puisque l'approche est structurelle.

Décomposition fine de la partie sous-contrainte

Bliek, Neveu et Trombettoni vont plus loin en proposant deux algorithmes, OpenPlan dû à Trombettoni [Trombettoni, 1997] et BTF-DIS présenté dans [Bliek *et al.*, 1998], aptes à identifier une décomposition CFC plus intéressante pour la composante sous-contrainte. L'algorithme OpenPlan, initialement issu du schéma PDOF en propagation locale (cf. section 4.2, page 96), construit un DAG de bloc pour la composante sous-contrainte d'un GCSP en sélectionnant itérativement un bloc libre de taille minimum ; un bloc est libre si ses variables ne sont liées qu'aux contraintes du bloc. Cette notion, ainsi que le schéma PDOF, seront décrits plus en détails à la section 4.2.1, page 98. L'algorithme OpenPlan est malheureusement exponentiel, et le problème auquel il s'attaque, à savoir minimiser la taille du plus gros bloc de la décomposition CFC d'un système sous-contraint, n'a jusqu'à présent pas été classé. Une autre version de l'algorithme OpenPlan consiste à ne pas minimiser la taille des blocs libres sélectionnés, mais à utiliser une heuristique qui permet d'identifier des petits blocs. Cette version n'assure pas de minimiser la taille du plus gros bloc, mais elle est polynomiale. Ce dernier algorithme a l'avantage de permettre d'éviter la sélection de blocs de taille arbitraire, problème auquel est confronté l'approche CFC lorsqu'elle est utilisée pour la décomposition de la composante sous-contrainte.

La figure 4.4 présente la décomposition CFC du GCSP *Ponts* par l'algorithme OpenPlan. Ce GCSP est composé uniquement d'une composante sous-contrainte puisqu'il est bien-rigide (cf. lien entre bien-contraint et bien-rigide, section 2.3.1, page 51). A la première itération d'OpenPlan les blocs $B1$, $B2$ et $B3$ sont libres. En effet, les variables x_A et y_A ne sont liées qu'aux équations d_{AI} et d_{AB}, et il en va de même pour les 2 autres blocs. Ces 3 blocs étant de même taille, ils peuvent être choisis indifféremment. La sélection du bloc $B1$ entraîne le retrait du sous-graphe constitué des sommets x_A, y_A, d_{AI} et d_{AB}, qui correspondent aux variables et aux équations du bloc $B1$. Une fois ce sous-graphe retiré, la seconde itération n'offre pas de nouveau bloc, si bien que $B2$ est sélectionné. Son retrait libère un nouveau bloc, le sous-système $B4$ constitué de 14 équations et variables. Sa taille implique que c'est alors $B3$ qui est sélectionné par OpenPlan. Viennent ensuite $B4$ qui est le seul bloc libre à l'itération 4, puis $B5$, $B6$, $B7$ et finalement $B8$ aux 4 itérations

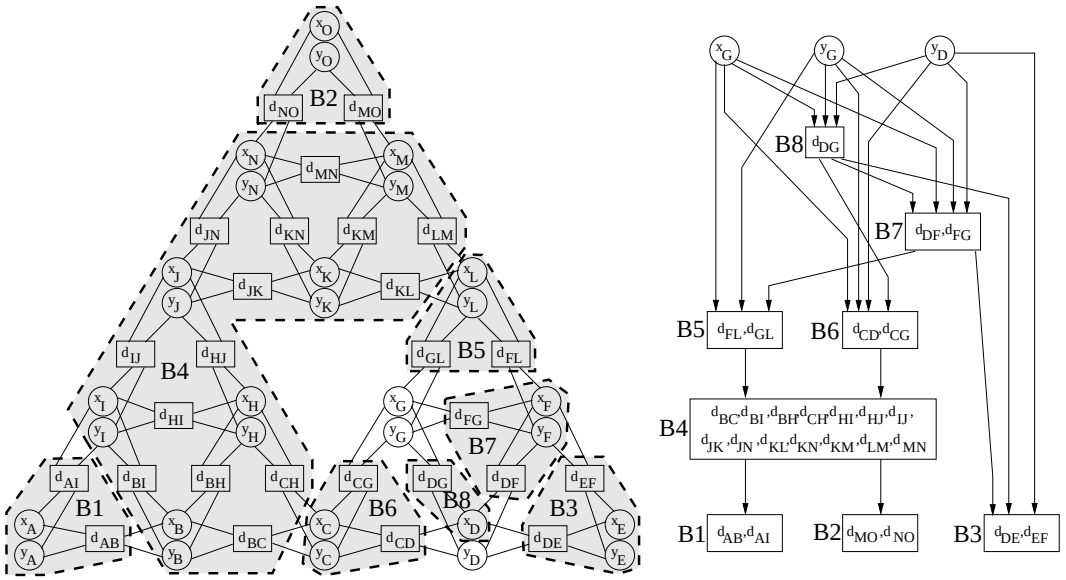

FIG. 4.4 – Décomposition du GCSP *Ponts* par la méthode OpenPlan

suivantes. Le DAG résultant est présenté en partie droite de la figure 4.4. Il s'agit de la décomposition CFC la plus fine du GCSP *Ponts*. Nous verrons à la section 4.3.2 que ce même GCSP peut être décomposé en blocs comportant au plus 2 équations par une méthode de décomposition géométrique.

L'algorithme BTF-DIS (pour *Backtrack-free driving-input selection*), linéaire lui, a été inspiré des applications en robotique et en automatique où certaines variables représentent les paramètres d'un mécanisme ; il permet la sélection interactive de *variables poignées* dans la composante sous-contrainte, c-a-d. des variables dont la valeur sera fournie par un utilisateur, une esquisse, un programme de simulation, ou tout autre processus. La sélection d'une variable poignée parmi les variables de la partie sous-contrainte se traduit par l'ajout d'une contrainte unaire portant sur cette variable. Après l'ajout de toute contrainte unaire, La décomposition DM est recalculée[1] et la partie sous-contrainte est diminuée au minimum d'une variable : la poignée séléctionnée. Les variables de la nouvelle partie sous-contrainte sont à nouveau proposées à la sélection comme variables poignées. L'algorithme termine lorsque la partie sous-contrainte devient vide. La partie bien-contrainte admet alors une décomposition CFC unique (cf. théorème 12, page 90).

Du point de vue de la résolution, Bliek, Neveu et Trombettoni proposent d'utiliser les techniques de résolution par intervalles pour la résolution complète de chaque bloc. Un mécanisme de retour-arrière intelligent, appelé GPB (pour *generalized partial-order backtracking*) et imaginé par Bliek, exploite l'ordre partiel induit par le DAG : lorsqu'un bloc n'admet pas de solution, ce mécanisme permet de remettre en cause la solution choisie

[1]Il suffit généralement d'inverser un chemin dans le couplage maximum M afin de mettre à jour incrémentalement la décomposition DM.

pour le bloc le plus proche qui est probablement la cause de l'absence de solution. Ce mécanisme est basé sur la mémorisation des conflits dans des *no-goods*. L'utilisation d'un mécanisme de retour-arrière permet également de calculer l'ensemble des solutions du CSP initial. La résolution d'un DAG de blocs sera traitée plus en détail au chapitre 3 car nous avons adopté ce schéma pour la résolution de GCSP décomposés par notre méthode de rigidification récursive.

Cette méthode permet donc de résoudre un GCSP bien-, sur- ou sous-contraint, toujours sous réserve que ce GCSP ne contienne ni A-redondance, ni singularité.

4.1.2 Autres approches

Méthode de Latham et Middleditch

Latham et Middleditch [Latham and Middleditch, 1996] ont proposé une méthode pour décomposer un GCSP en sous-systèmes solubles indépendamment. Cette méthode utilise le *graphe objets-contraintes*, graphe biparti dont les sommets sont les objets et les contraintes géométriques ; il existe une arête entre un objet o et une contrainte c si c porte sur o. Le graphe objets-contraintes du GCSP *Ponts* est représenté à la figure 4.5. Remarquons que ce graphe est une version condensée du graphe variables-équations, où toutes les variables/paramètres d'un même objet géométrique sont fusionnées en un seul sommet, et toutes les équations décrivant une contrainte géométrique sont aussi fusionnées en un seul sommet.

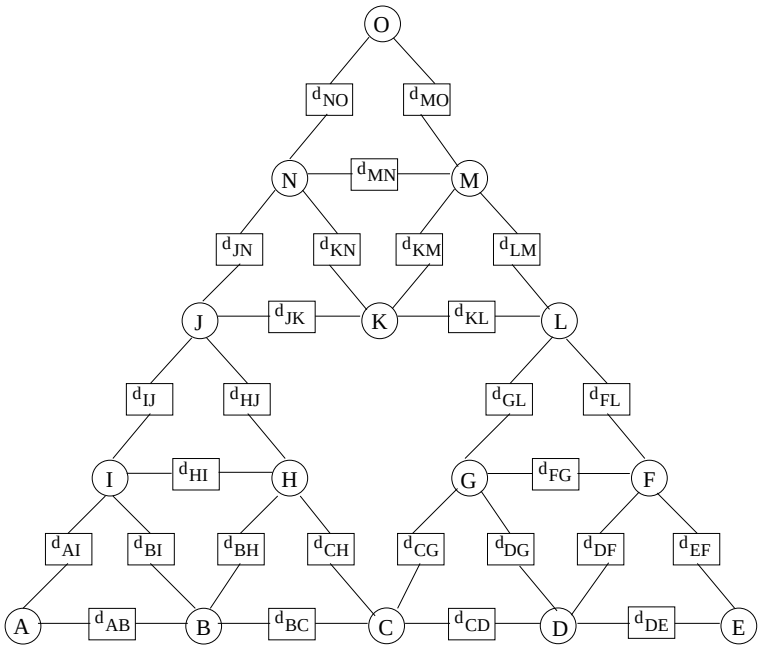

FIG. 4.5 – Graphe objets-contraintes du GCSP *Ponts*

Latham et Middleditch décorent ce graphe avec les degrés de liberté des objets et des contraintes et recherchent une *pondération maximum* dans ce graphe. Une pondération maximum est obtenue en affectant les valeurs maximum aux arêtes de telle sorte que la somme des poids des arêtes incidentes à tout sommet est inférieure au poids de ce sommet ; si elle est égale, le sommet est *saturé*. Un algorithme basé sur le couplage maximum pondéré [Gondran and Minoux, 1984] permet de calculer une pondération maximum. Un objet non-saturé représente une partie sous-contrainte, une contrainte non-saturée représente une partie sur-contrainte.

La méthode consiste alors à obtenir un graphe dont tous les sommets sont saturés en retirant un à un les sommets non-saturés et en adaptant la pondération maximum à chaque étape. Ceci élimine *de façon heuristique* les parties sur-contraintes et sous-contraintes du GCSP. Une fois le GCSP réduit à un système bien contraint, il est décomposé en sous-systèmes équilibrés, correspondant intuitivement aux composantes fortement connexes de la décomposition CFC. La résolution est effectuée par des méthodes de types règle-et-compas et par des méthodes numériques classiques. Ceci n'assure ni la complétude de la résolution, ni même l'obtention d'une solution.

Méthode de Hsu et Brüderlin

Il en va de même pour la méthode proposée dans [Hsu and Brüderlin, 1997]. Cette méthode consiste à représenter un GCSP par un graphe de dépendance dont les sommets sont les objets géométriques et les arêtes (ou hyper-arêtes dans le cas de contraintes n-aires) sont les contraintes géométriques. Ce graphe est orienté par le choix d'un objet racine o_r : tout arête incidente à o_r est un arc entrant dans o_r et sortant pour son (ou ses) autre(s) extrémité(s). Le graphe est aussi décoré par les degrés de liberté des objets et des contraintes, et le but de la méthode est de trouver une distribution des degrés de liberté des contraintes sur les objets.

La méthode procède en 2 étapes répétées itérativement jusqu'à ce qu'un point fixe soit atteint : une affectation gloutonne propose une première distribution des degrés de liberté, puis une passe de migration heuristique rétablit les *équations d'équilibres* liées à chaque objet. Ces équations définissent une règle semblable à celle utilisée par Latham et Middleditch : la somme de degrés de liberté retirés à un objet ne doit pas dépasser son nombre de degrés de liberté. La méthode n'est pas complète car le graphe de dépendance ne peut pas toujours être orienté (présence de cycles) et la passe de migration utilise des règles heuristiques.

Méthode de Lesage

Enfin, la méthode proposée par Lesage, Léon et Serré [Lesage *et al.*, 2000] utilise la décomposition DM-CFC mais sur un système qui n'est pas exactement le système des contraintes géométriques. Cette méthode permet de résoudre des problèmes de CAO contenant des contraintes géométriques, mais pouvant contenir d'autres contraintes, en 2D. Points et vecteurs sont les objets basiques utilisés par cette méthode qui considère des contraintes

de distance et d'angle ; les contraintes d'incidence et de parallélisme sont modélisées par des contraintes valuées ayant des valeurs particulières, ou par introduction d'*objets cachés*, c-a-d. des objets géométriques supplémentaires introduits pour faciliter la modélisation de certaines contraintes.

La méthode consiste à ne pas considérer les coordonnées des objets géométriques mais plutôt leurs distances et angles relativement les uns aux autres. Ceci permet de s'affranchir de la notion de repère qui apparaît dans la plupart des méthodes de décomposition géométrique. Les inconnues deviennent alors les distances et angles entre les objets géométriques du GCSP et les contraintes fournissent des valeurs pour certaines de ces inconnues. Pour déterminer les distances et angles inconnus, la méthode utilise 2 graphes : le premier contenant uniquement les relations de distance et le second uniquement les relations angulaires. Les objets considérés dans les 2 graphes sont des vecteurs : une distance s'exprime entre 2 points qui définissent un vecteur ; un angle s'exprime entre 4 points, c-a-d. 2 vecteurs.

Dans le graphe des distances, un cycle permet d'écrire une relation vectorielle correspondant au principe fondamental de la cinématique : la somme des vecteurs constituant le cycle est le vecteur nul. Puisque le système est en 2D, cette équation vectorielle génère 2 équations par projection sur 2 axes orthogonaux ; les axes sont choisis de façon heuristique pour chaque équation vectorielle, certainement de façon à permettre l'élimination d'un maximum de termes. Un cycle génère donc une paire d'équations sur les distances inconnues du système. Dans le graphe des angles, un cycle permet d'écrire une relation de Châles entre les angles du cycle. Cette relation produit une équation dont les inconnues sont les angles du cycle.

La méthode consiste alors à identifier une couverture en cycles indépendants (si possible de petites tailles) dans chacun des 2 graphes ce qui génère un système d'équations indépendantes. C'est ce système qui est décomposé par la méthode DM-CFC puis résolu par homotopie ou Newton-Raphson. Rappelons que la méthode Newton-Raphson utilisée avec une méthode de décomposition n'assure pas de permettre la production d'une solution (cf. section 3.3.1, page 79). De plus, la complexité de l'identification d'une couverture maximum en cycles indépendants de tailles minimales n'est pas établie.

Méthode de Hendrickson

La méthode de Hendrickson [Hendrickson, 1992] est aussi basée sur une forme de couplage maximum. Comme elle s'applique aux GCSP constitués de points et distances en 2D, elle consiste à tripler n'importe quelle contrainte de distance avant de procéder à une décomposition DM. Tripler une contrainte de distance revient, pour cette classe de GCSP en 2D, à poser un repère local, ce qui fait passer le problème de détecter un sous-système rigide à détecter une composante bien contrainte. C'est de cette approche qu'est issu le travail de Lamure et Michelucci [Lamure and Michelucci, 1998].

Cette approche est très limitée, car elle ne peut être étendue en 3D de façon correcte (cf. problème du repère local, page 116) et ne peut prendre en compte d'autres types d'objets ou de contraintes géométriques.

4.2 Décompositions équationnelles à base de règles

Les méthodes de décomposition équationnelle à base de règles sont principalement issues de la propagation locale. Cette communauté s'intéresse à la résolution de CSP fonctionnels, notés *FCSP*. Un FCSP (X, D, C, M) est le CSP (X, D, C) associé à un ensemble M de méthodes de résolutions, dites *r-méthodes*. Une r-méthode $m_{C'}^{V'} \in M$ d'un FCSP (X, D, C, M) est un processus qui satisfait un ensemble $C' \subset C$ de contraintes en calculant les valeurs d'un ensemble $V' \subset V$ de variables liées à ces contraintes, utilisant pour ce faire les valeurs associées aux autres variables liées aux contraintes de C'. Une r-méthode est dite *simple* si $|C'| = 1$, *générale* sinon. Rappelons qu'une r-méthode doit permettre une résolution exacte et complète du sous-GCSP correspondant si l'on souhaite que la résolution d'un GCSP décomposé soit complète et correcte.

Pour la résolution de GCSP, les r-méthodes que l'on peut utiliser sont par exemple tirées des méthodes de résolution à la règle et au compas. Une règle définit un modèle de sous-graphe objets-contraintes ; par exemple, la règle d'intersection de 2 cercles défini le modèle $(\{Cercle1, Cercle2, Point\}, \{Incidence(Point, Cercle1), Incidence(Point, Cercle2)\})$.

La recherche de tous les sous-graphes du graphe objets-contraintes d'un GCSP correspondant au modèle de sous-graphe associé à une règle, on peut générer l'ensemble des r-méthodes correspondant à cette règle apparaissant dans le GCSP.

Voici deux exemples de telles règles en 2D :

- Détermination d'une coordonnée d'un point $P2$ dont l'une des coordonnées est fixée et qui se trouve à distance d d'un point $P1$ fixé. Le modèle de sous-graphe associé à cette règle est présenté en partie gauche de la figure 4.6. Cette règle génère des r-méthodes simples qui déterminent une seule variable (x_{P2} ou y_{P2}) à l'aide d'une seule équation ($|P1P2| = d$).
- Détermination d'un point $P3$ à distances connues $r1$ et $r2$ de deux points $P1$ et $P2$. Cette règle s'identifie au sous-graphe modèle présenté en partie droite de la figure 4.6. Elle permet de générer des r-méthodes générales, puisqu'elle résout 2 variables (x_{P3}, y_{P3}) à l'aide de 2 équations ($|P1P3| = r1$, $|P2P3| = r2$).

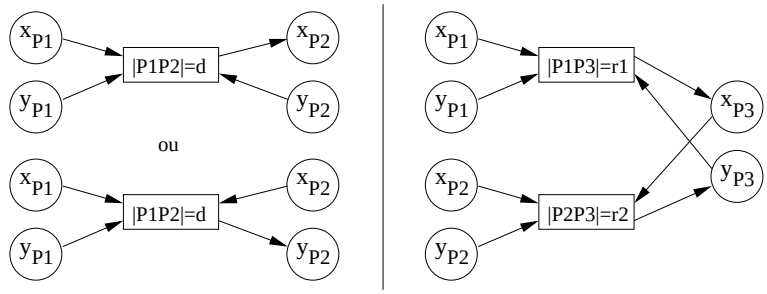

FIG. 4.6 – Deux modèles de r-méthodes géométriques

Le but[2] des algorithmes de propagation locale est de trouver un graphe orienté de r-

[2]La propagation locale a plus exactement pour but de fournir des algorithmes incrémentaux de résolu-

méthodes, appelé *plan*, qui satisfait l'ensemble des contraintes du FCSP. On distingue deux types de plans : les plans sans circuits qui forment un DAG de r-méthodes, et les plans avec circuits. Une fois un plan obtenu, une phase d'évaluation exécute une séquence de r-méthodes de résolution compatible avec le plan.

Les algorithmes de propagation locale se classent en 3 catégories selon le schéma qu'ils suivent :

- Propagation de conflits (PC) : ce schéma consiste à propager les conflits de variables dans [Borning, 1979; Borning and Freeman-Benson, 1995; Sanella, 1994] le graphe jusqu'à l'obtention d'une solution. Un conflit de variables survient lorsque 2 r-méthodes doivent calculer la même variable de sortie. Le principe est alors de modifier l'une des 2 méthodes en conflit jusqu'à atteindre une planification sans conflits. Les algorithmes qui utilisent ce schéma sont capables de produire des plans avec ou sans circuits, et de planifier des r-méthodes générales mais avec une complexité en temps exponentielle dans la taille du FCSP. Cela rend le schéma inintéressant vis-à vis des 2 autres schémas pour lesquels des algorithmes polynomiaux et plus généraux ont été proposés.
- Couplage maximum (MM) : ce schéma est basé sur l'algorithme de couplage maximum et produit des plans avec ou sans circuits. Initialement proposé par Gangnet et Rosenberg [Serrano, 1987; Gangnet and Rosenberg, 1992], ce schéma n'est capable de traiter que les FCSP dont l'ensemble M contient une r-méthode simple m_c^v pour chaque contrainte c et chaque variable v liée à c. Certaines méthodes issues du schéma MM regroupent les méthodes d'une même composante fortement connexe dans un sous-FCSP à résoudre en une seule fois, à la façon de la méthode DM-CFC (cf. section 4.1.1, page 87), .
- Propagation des degrés de liberté (PDOF) : ce schéma cherche à produire un plan sans circuit en partant des feuilles et en remontant vers les racines [Sutherland, 1963; Vander Zanden, 1996; Trombettoni, 1997]. Cette approche, initialement proposée par Sutherland dans son logiciel de dessin *Sketchpad* [Sutherland, 1963], a été généralisée aux FCSP à r-méthodes générales par Trombettoni [Trombettoni, 1998]. Nous présentons l'algorithme GPDOF, adapté au traitement de GCSP, à la section suivante. L'algorithme de décomposition équationnelle structurelle OpenPlan [Trombettoni, 1997; Bliek *et al.*, 1998] présenté à la section 4.1.1 (page 87) est aussi issu de ce schéma.

La phase d'évaluation diffère selon que le plan contient des circuits de r-méthodes ou non. S'il en contient, des méthodes de résolution comme l'évaluation simple [Sanella, 1994], l'évaluation itérative [Sanella, 1994], la relaxation [Sutherland, 1963; Borning, 1979] peuvent être envisagées. La plus générale consiste à considérer le cycle comme un sous-système qui doit être résolu en une fois, et utiliser une méthode de résolution quelconque qui remplace les r-méthodes. Cela dépasse cependant le cadre des méthodes à base de règles. Lorsque le plan ne contient pas de circuit, un tri topologique du graphe de r-méthodes permet de trouver une séquence de résolution pour les r-méthodes, de façon semblable à

tion de contraintes qui permettent le rétablissement de la consistance d'un FCSP *perturbé* par l'introduction ou la suppression d'une contrainte.

celle proposée pour la décomposition DM-CFC, cf. section 4.1.1, page 87.

Dans la section suivante, nous présentons l'algorithme PDOF-général (GPDOF) proposé par Gilles Trombettoni [Trombettoni, 1997]. Cet algorithme, dérive du schéma PDOF et permet de traiter des FCSP généraux. Nous expliquons comment il pourrait être utilisé pour résoudre des GCSP.

4.2.1 PDOF-général

L'algorithme GPDOF est une généralisation de l'algorithme PDOF initialement proposé par Sutherland [Sutherland, 1963]. Le schéma de ces 2 algorithmes est le même : construire de façon ascendante un plan de r-méthodes par sélection itérative de *r-méthodes libres*. Une r-méthode est libre si ses variables de sortie ne sont liées qu'aux contraintes satisfaites par cette r-méthode. GPDOF ne peut donc produire que des plans sans circuit, car aucune des r-méthodes d'un circuit n'est libre.

Algorithme 5 GPDOF (G : graphe variables-équations ; M : ensemble de r-méthodes de G) **retourne** P : un plan de r-méthodes

$P \leftarrow \emptyset$
// *P est vide au début*
$Libres \leftarrow$ `R-Methodes-Libres`$(G, M, \emptyset)$
Tant que $G \neq GrapheVide$ **et** $Libres \neq \emptyset$ **Faire**
  $m \leftarrow$ `Premier-Elément`$(Libres)$
  $M \leftarrow M \setminus \{m\}$
  `Planifier`(m, P)
  `Effacer-Contraintes`(m, G)
  `Effacer-Sorties`(m, G)
  `R-Methodes-Libres`(G, M, m)
Fin Tant que
Return P

L'algorithme GPDOF, présenté à l'algorithme 5, prend en entrée le graphe variables-équations G du FCSP à décomposer et l'ensemble M des r-méthodes associées à ses contraintes. Il produit en sortie un plan P qui est un DAG de r-méthodes. L'algorithme maintient un ensemble *Libres* de r-méthodes libres et termine soit lorsque le FCSP a été entièrement planifié ($G = GrapheVide$), soit lorsqu'il n'existe plus de r-méthodes libres pour planifier le reste du FCSP ($Libres = \emptyset$) ; la complétude de cet algorithme, prouvée dans [Trombettoni, 1997], assure que s'il n'existe plus de r-méthodes libres, il n'existe alors aucun plan sans circuit pour le FCSP considéré.

Le point crucial de cet algorithme, qui fait sa différence avec l'algorithme PDOF, est que la fonction `R-Methodes-Libres` met à jour l'ensemble des r-méthodes M en modifiant certaines r-méthodes de cet ensemble. Sans entrer dans le détail, ceci permet d'obtenir une séquence finale de r-méthodes impliquant des r-méthodes du répertoire se recouvrant partiellement, comme si certaines variables étaient résolues plusieurs fois, et certaines contrain-

tes satisfaites plusieurs fois. C'est ce point précis qui assure la complétude et la complexité de l'algorithme GPDOF.

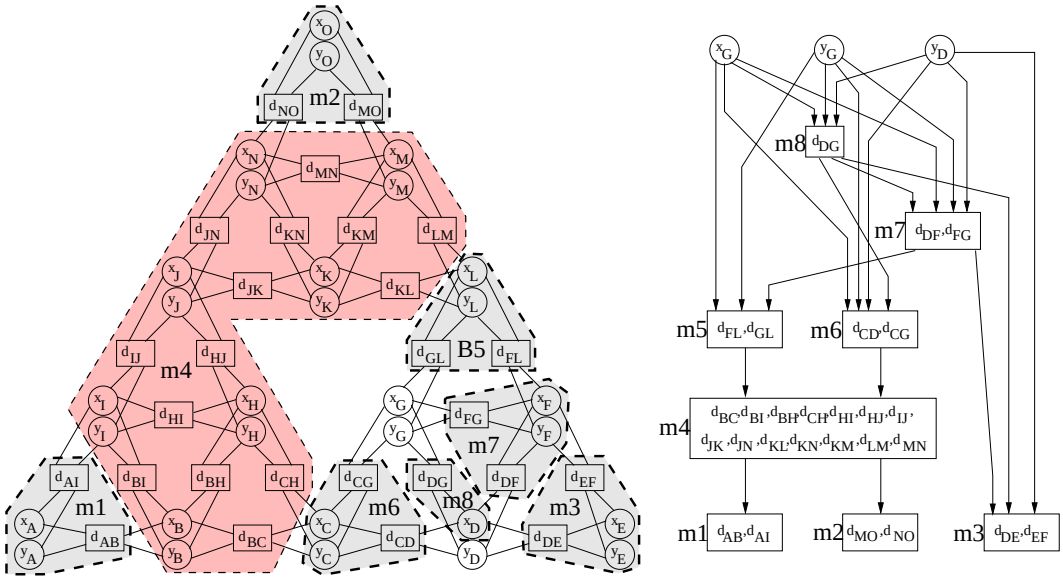

FIG. 4.7 – Décomposition du GCSP *Ponts* par la méthode GPDOF

Considérons le GCSP *Ponts* et sa décomposition par l'algorithme GPDOF. Cette décomposition est illustrée à la figure 4.7. On considère que les r-méthodes associées au GCSP *Ponts* sont les méthodes de résolution à la règle et au compas présentées à la section précédente, à savoir 1 distance entre 2 points et 2 distances entre 3 points. Ces 2 schémas permettent de générer de nombreuses r-méthodes sur le GCSP *Ponts*, dont $m1$, $m2$, $m3$, $m5$, $m6$, $m7$ et $m8$ présentées sur la figure 4.7. Malheureusement, en l'absence de la r-méthode $m4$, ce GCSP ne peut être planifié par l'algorithme GPDOF. Cette méthode ne correspond vraiment à aucune règle classique de résolution, ce qui démontre la limite de GPDOF : même un problème aussi simple que le GCSP *Ponts* ne pourra pas être planifié simplement à l'aide de r-méthodes de type règle et compas. En admettant que la r-méthode $m4$ puisse être fournie avec ce GCSP, le plan obtenu est alors semblable à celui obtenu par l'algorithme OpenPlan, vu à la section 4.1.1, page 91.

4.3 Décompositions géométriques structurelles

Les méthodes de décomposition géométrique utilisent toutes la même propriété géométrique afin de décomposer un GCSP : identifier des sous-systèmes solubles dans un repère local, c-a-d. des sous-GCSP rigides (cf. section 2.3.1, page 51). Les méthodes de décomposition géométrique structurelle cherchent à identifier de tels systèmes de façon structurelle ; nous avons vu au chapitre 2 qu'aucune caractérisation structurelle de la rigidité n'a été pro-

posée pour la rigidité de GCSP dans le cas général. La plupart des méthodes utilisent donc une caractérisation approchée, comme la rigidité structurelle (cf. section 2.3.3, page 59).

On distingue deux types de méthodes de décomposition géométrique structurelle : celles qui proposent une approche descendante (ex. Owen [Owen, 1991]), et celles qui proposent une approche ascendante (ex. Fudos [Fudos and Hoffmann, 1997] et HLS [Hoffmann *et al.*, 1997]). Nous allons présenter, dans les sections suivantes, ces trois méthodes ainsi que certaines améliorations qui leur ont été apportées.

4.3.1 Méthode d'Owen

La méthode d'Owen [Owen, 1991] permet de résoudre des GCSP en 2D constitués de points et droites soumis à des contraintes de distance et d'angle[3]. Par extension du modèle, on peut représenter des cercles et arcs de cercles à l'aide de points et droites, et des contraintes d'incidence, de tangence ou de parallélisme à l'aide de contraintes de distance et angle à valeurs particulières. Le système d'équations correspondant à un tel GCSP est constitué d'équations quadratiques uniquement.

Owen utilise la version non-bipartie du graphe objets-contraintes, aussi appelée graphe de contraintes géométriques dans la littérature. Dans cette version, seuls les objets géométriques sont des sommets, les contraintes étant représentées par les arêtes liant ces objets. La version bipartie est mieux adaptée aux GCSP généraux car elle n'est pas limitée aux contraintes binaires. L'utilisation de la version non-bipartie de ce graphe signifie donc que la méthode d'Owen est limitée aux contraintes binaires, c-a-d. qu'Owen n'autorise pas les contraintes d'angles entre triplets ou quadruplets de points. Une justification possible est que l'on peut toujours ramener une telle contrainte à des contraintes binaires moyennant l'adjonction d'objets au GCSP : on ajoute les droites passant par les points dont on veut contraindre l'angle et on reporte la contrainte d'angle sur celles-ci ; cependant, cette transformation augmente la taille du GCSP à résoudre.

Owen propose une méthode de décomposition descendante basée sur la constatation suivante : 2 sous-systèmes rigides partageant 2 objets géométriques peuvent être assemblés. Cette méthode de décomposition consiste à identifier les *paires d'articulations* dans le graphe objets-contraintes afin de découper celui-ci en sous-graphes pouvant être assemblés. Une *paire d'articulation* est un couple de sommets du graphe qui, si elle est dupliquée, décompose le graphe en 2 composantes connexes de plus de 3 objets partageant une paire d'objets : la paire d'articulation. Une *arête virtuelle* est placée entre les 2 sommets des paires d'articulations à chaque scission. Un processus de simplification vient retirer les arêtes virtuelles qui ne sont pas nécessaires afin de permettre de futures scissions. La scission récursive aux paires d'articulations du graphe produit une décomposition du graphe initial en triangles et composantes 3-connexes plus complexes. Si elle se réduit à des triangles, le système est soluble par la méthode d'Owen.

[3]Cette méthode a été intégrée et développée plus avant dans des logiciels commercialisés par la société D-Cubed ; ces logiciels sont aptes à traiter des GCSP en 3D et d'autres types d'objets et de contraintes, mais la méthode utilisée n'a pas été rendue publique.

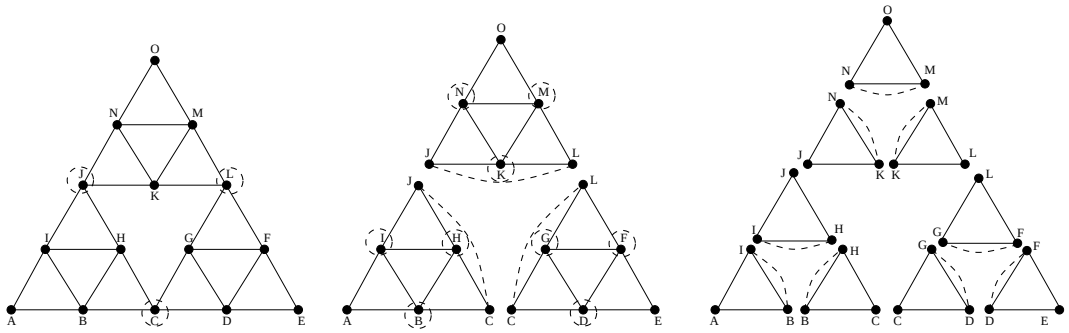

FIG. 4.8 – Décomposition du GCSP *Ponts* par la méthode Owen

La figure 4.8 présente la décomposition du graphe objets-contraintes correspondant au GCSP *Ponts*. Cette décomposition s'effectue en 3 itérations. A chacune, on peut voir les paires d'articulations sélectionnées : les sommets les constituant sont entourés par des tirets. Les arêtes virtuelles sont représentées par des traits en tirets ; aucune n'étant nécessaire dans la décomposition de ce GCSP, elles sont toutes retirées avant l'itération suivante. La décomposition se termine lorsque les sous-graphes obtenus sont tous des triangles.

Le système est résolu dans l'ordre inverse de la décomposition : chaque triangle ne contenant pas d'arête virtuelle est d'abord résolu par une méthode algébrique (manipulation symbolique restreinte aux opérations arithmétiques et à la racine carrée). Les sous-graphes partageant une même paire d'articulation sont alors assemblés par mise en coïncidence des objets dupliqués. Ceci permet de calculer les dimensions de certaines arêtes virtuelles qui permettent à leur tour de résoudre d'autres triangles et de les assembler et ainsi de suite.

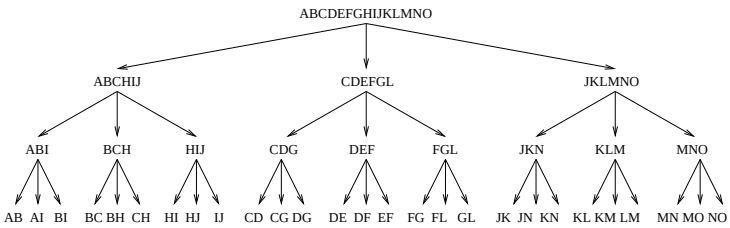

FIG. 4.9 – Décomposition arborescente du GCSP correspondant à celle de la figure 4.8

Joan-Arinyo *et al.* [Joan-Arinyo *et al.*, 2002] ont prouvé que la classe des problèmes traités par cette méthode se compose de tous les GCSP qui admettent une décomposition arborescente en triplets de sous-graphes partageant 2 à 2 un sommet et couvrant le graphe initial. Dit autrement, les problèmes traitables sont ceux pour lesquels on peut assembler récursivement 3 par 3 des sous-graphes partageant 2 à 2 un objet ; l'ensemble des sous-graphes initiaux est constitué d'un sous-graphe par arête du graphe objets-contraintes. On peut voir à la figure 4.9 la décomposition arborescente du GCSP *Ponts*, qui correspond à la décomposition par la méthode d'Owen.

Cette classe de GCSP ne couvre pas l'ensemble des GCSP en 2D constitués de points, droites, distances et angles. Par exemple, le GCSP constitué de 6 points A, B, C, D, E et F lié par 9 contraintes de distance et dont le graphe objets-contraintes est le graphe biparti complet K_{33} (cf. figure 2.8, page 38) ne peut être décomposé de la sorte : le graphe K_{33} est en effet 3-connexe et n'admet aucune paire d'articulation. Il s'agit pourtant d'un GCSP structurellement rigide et rigide si ses contraintes ne sont pas dégénérées.

4.3.2 Méthode de Fudos-Hoffmann

La méthode de Fudos et Hoffmann est une méthode de décomposition ascendante (dite constructive) [Bouma *et al.*, 1995; Fudos and Hoffmann, 1996; 1997] limitée à quelques règles d'assemblage. Elle permet de résoudre le même type de GCSP que la méthode d'Owen, à savoir des GCSP 2D constitués de points et droites soumis à des contraintes de distance et d'angle. Cette méthode utilise aussi la version non bipartie du graphe objets-contraintes et est donc limitée aux contraintes binaires, tout comme la méthode d'Owen.

Le schéma général des méthodes de décomposition ascendantes (ou méthodes constructives) est le suivant :

1. *Phase d'analyse* : cette phase produit une séquence d'opérations pour la résolution du GCSP par réduction (et décomposition) du graphe objets-contraintes.
2. *Phase de construction* : cette phase résout le GCSP en exécutant la séquence d'opérations produite en phase d'analyse.

Cette méthode permet de résoudre des GCSP structurellement sur-, sous- ou bien-rigides. Par souci de clarté, nous présentons d'abord la méthode complète pour des GCSP rigides, puis nous expliquerons la démarche proposée pour les GCSP sur- ou sous-rigides.

La méthode de Fudos et Hoffmann procède par assemblage d'*agrégats*. Un agrégat est un sous-système du GCSP qui possède 3 degrés de liberté, c-a-d. qu'il est structurellement rigide en 2D, et est produit par *assemblage* à partir des *agrégats initiaux*. Les agrégats initiaux sont constitués de tous les sous-systèmes réduits à une seule contrainte du GCSP, c-a-d. une seule arête du graphe objets-contraintes. L'opérateur d'assemblage consiste à fusionner en un unique agrégat, 3 agrégats dont l'intersection 2 à 2 ne contient qu'un seul objet géométrique ; concrètement, cette fusion se traduit par le retrait du graphe objets-contraintes de toutes les arêtes des 3 agrégats assemblés. L'application récursive de l'opérateur d'assemblage produit donc des agrégats de plus en plus grands et réduit le graphe objets-contraintes. Si ce graphe est soluble par la méthode de Fudos et Hoffmann (cf. fin de la section précédente pour la définition de la classe des GCSP solubles par cette approche), il sera finalement réduit à un agrégat unique contenant l'ensemble du GCSP. L'algorithme 6 présente l'algorithme de la méthode Fudos *et al.*.

La figure 4.9, page 101, présente le plan d'assemblage du GCSP *Ponts* : c'est un arbre dont les feuilles sont les contraintes du GCSP et la racine est l'ensemble du GCSP, s'il est structurellement rigide et décomposable sous cette forme. La méthode construit ce plan d'assemblage des feuilles vers la racine par des opérations d'assemblage : initialement,

Algorithme 6 FudosHoffmann ($G = (O, C)$: graphe objets-contraintes) **retourne** P : plan d'assemblage

$Agregats \leftarrow C$
$G1, G2, G3 \leftarrow$ `Identifier-Agrégats-Assemblables(`$Agregats$`)`
Tant que $G1, G2, G3 \neq GrapheVide$ **Faire**
  $K \leftarrow G1 \cup G2 \cup G3$
  `Planifier(`K, P`)`
  $Agregats \leftarrow (Agregats \cup K) \setminus \{G1, G2, G3\}$
Fin Tant que
Return P

chaque arête de ce GCSP constitue un agrégat, représenté par une feuille du plan d'assemblage ; à chaque itération, 3 agrégats assemblables $G1$, $G2$ et $G3$ sont fusionnés en un agrégat G qui est inséré sous forme de nœuds dans le plan d'assemblage, avec pour fils $G1$, $G2$ et $G3$. Par exemple, les agrégats AB, AI et BI sont assemblables car ils partagent 2 à 2 un objet géométrique. Ils sont fusionnés dans l'agrégat ABI qui devient leur père dans le plan d'assemblage.

Joan-Arinyo *et al.* [Joan-Arinyo *et al.*, 2002] ont prouvé que la classe des problèmes traités par cette méthode est exactement la même que celle des problèmes traités par la méthode d'Owen. Il est donc clair que cette méthode n'est pas complète, même vis-à-vis du type de GCSP attaqués, c-a-d. les GCSP en 2D constitués de points, droites, distances, angles (cf. contre exemple en fin de section précédente). Fudos et Hoffmann ont proposé quelques autres opérateurs d'assemblage d'agrégats, mais la méthode reste cependant incomplète dans le cas général : il existe des GCSP rigides nécessitant l'assemblage d'un nombre arbitraire d'agrégats en une seule étape[4]. Le contre-exemple en fin de section précédente nécessite l'assemblage simultané de ses 9 contraintes. Hoffmann, Lomonosov et Sitharam ont levé cette limitation en proposant la méthode HLS (cf. section 4.3.3) capable d'identifier des assemblages quelconques.

La phase d'assemblage procède à la résolution effective du GCSP en suivant le plan d'assemblage défini par la phase d'analyse : chaque agrégat initial est résolu dans un repère local ; chaque assemblage est résolu par déplacement des repères locaux des 3 agrégats fusionnés afin de mettre en correspondance les 3 objets géométriques partagés. Cette méthode d'assemblage ne fonctionne pas lorsque les 3 objets partagés par 3 agrégats sont des droites car elles ne peuvent être positionnées de façon rigide les unes par rapport aux autres à l'aide de contraintes d'angles ; autrement dit, le partage de 3 droites par 3 agrégats ne permet pas de définir un déplacement unique à appliquer aux 3 agrégats pour résoudre leur assemblage. Enfin, Fudos et Hoffmann proposent une méthode de sélection de solution basée sur le respect des orientations des angles par rapport à une esquisse.

Fudos et Hoffmann proposent de considérer toute sur-rigidité comme une R-redondance (cf. section 2.3.2, page 58) lors de la phase d'analyse. Ils ajoutent alors l'opérateur d'assemblage suivant : lorsque 2 agrégats partagent 2 objets géométriques, ils sont assemblés

[4] Cette limitation a été prouvée par Hoffmann, Lomonosov et Sitharam dans [Hoffmann *et al.*, 1997]

en un seul agrégat. Cet opérateur se traduit à la phase d'assemblage par une vérification : les 2 agrégats ne pourront être assemblés que si les positions des 2 objets qu'ils partagent dans l'un et l'autre de ces agrégats peuvent coïncider par déplacement, c-a-d. s'il existe un déplacement des repères locaux de ses agrégats capable de mettre les 2 objets partagés en correspondance. Si tel n'est pas le cas, le plan d'assemblage ne peut être résolu.

Pour traiter les GCSP structurellement sous-rigides, Fudos *et al.* proposent une méthode de décomposition descendante qui ajoute des contraintes à la volée afin de rendre le GCSP structurellement rigide. Cette méthode est comparable à la méthode proposée par Owen et présentée à la section 4.3.1 : le graphe est décomposé en composantes connexes qui sont décomposées en composantes 2-connexes qui sont décomposées en composantes 3-connexes.

4.3.3 Méthode Hoffmann, Lomonosov et Sitharam

La méthode HLS permet de traiter les GCSP constitués de tout type d'objets géométriques et de tout type de contraintes géométriques. Elle impose cependant 2 restrictions :

- les objets géométriques doivent être rigides, c-a-d. ne disposer d'aucune dimension variable ;
- les contraintes géométriques doivent être indépendantes du repère global considéré, c-a-d. que les contraintes permettant de fixer la position ou l'orientation d'un objet vis-à-vis du repère global de l'espace géométrique considéré sont proscrites.

Ces limitations sont dues à la façon dont opère la phase d'analyse d'une part : elle considère chaque objet comme un sous-GCSP rigide initialement ; et au fait qu'elle utilise la propriété de rigidité d'autre part : un GCSP fixé dans le repère global n'est pas rigide.

La méthode HLS utilise le schéma classique des méthodes de décomposition géométrique structurelle : elle est composée d'une *phase d'analyse* et d'une *phase d'assemblage*.

Phase d'analyse

La phase d'analyse a pour but de produire un plan d'assemblage qui sera exécuté lors de la phase d'assemblage. Elle consiste à regrouper peu à peu les agrégats jusqu'à ce que de tels regroupements ne soient plus possibles. Initialement, chaque objet géométrique constitue un agrégat. Les agrégats sont récursivement assemblés par la répétition d'une séquence de 3 opérations : *fusion*, *extensions* et *condensation*. Cette séquence est répétée jusqu'à ce qu'elle ne permette plus d'agréger aucun sous-GCSP.

L'opération de *fusion* a pour objectif de produire un nouvel agrégat de taille aussi petite que possible sur la base des agrégats existants ; elle est réalisée par un algorithme de calcul de flot maximum sur un réseau représentant le GCSP.

Une *extension* consiste à regrouper exactement 2 agrégats ; l'opération d'*extensions* procède à autant d'extensions que possible à partir de l'agrégat produit par la fusion.

Enfin, l'opération de *condensation* met à jour le GCSP vis à vis du nouvel agrégat identifié. Hoffmann *et al.* proposent 3 opérations de condensation différentes : *CA, FA et MFA*. FA et MFA permettent de faire partager des sous-parties entre plusieurs agrégats,

alors que CA ne le permet pas. Cependant, CA est plus simple et assure la terminaison de la planification sans modification des opérations de fusion et d'extension.

Exemple de planification

La figure 4.10 illustre le déroulement de la planification sur le graphe objets-contraintes du GCSP *Ponts* ; nous avons utilisé ici l'opération de condensation CA. La procédure `Planifier(`K, P`)` ajoute l'agrégat K au plan P : K est introduit sous forme de sommet dans le plan et des arcs sont introduits depuis les agrégats regroupés dans K vers le sommet représentant K dans P. La structure du plan est donc un DAG (cf. figure 4.10-f).

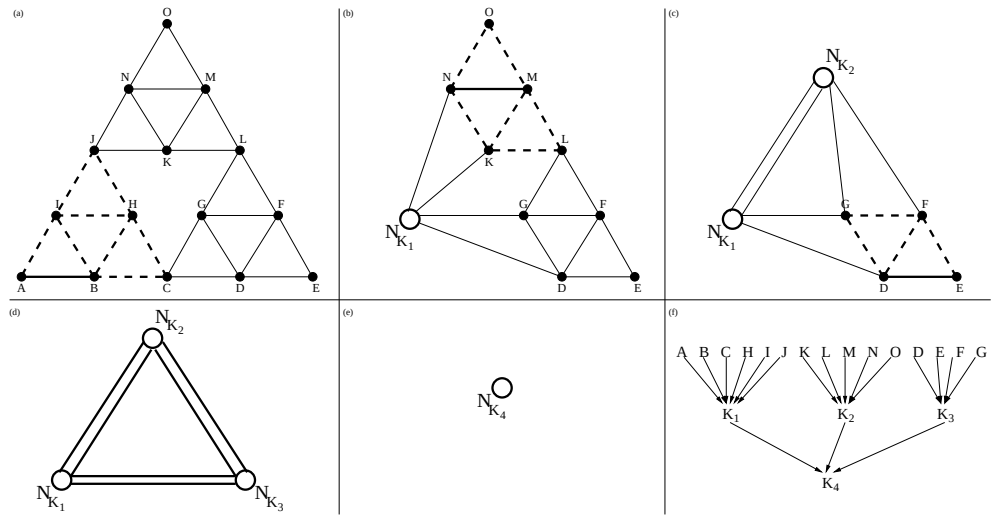

FIG. 4.10 – Décomposition du GCSP *Ponts* par la méthode HLS

La première étape de fusion identifie le sous-graphe $K_1 = AB$ de G (cf. figure 4.10-a). L'opération d'extensions ajoute des agrégats à K_1 tant que sa rigidité est préservée ; les agrégats qui peuvent lui être ajoutés ainsi sont C, H, I et J. L'agrégat K_1 est donc $ABCHIJ$ après l'opération d'extensions. Un nouveau sommet N_{K_1} est ajouté à G et les sommets A, B, I, H, C, J sont retirés du graphe (ainsi que toutes les contraintes les reliant, cf. figure 4.10-b). K_1 est ajouté au plan d'assemblage P ; dans ce plan, il dépend des agrégats initiaux A, B, C, H, I et J et un arc établit cette dépendance.

L'étape de fusion suivante identifie $K_2 = MN$ qui est ensuite étendu en $K_2 = MNOKL$ par l'opération d'extensions. Notons que K_2 aurait pu être étendu sur J si MN avait été détecté en premier à la place de AB. L'utilisation de FA ou MFA à la place de CA rendrait cela possible. Le plan et le graphe sont mis à jour pour prendre en compte le nouvel agrégat K_2 (cf. figure 4.10-c).

La troisième étape de fusion identifie $K_3 = DE$, étendu à $K_3 = DEFG$ par l'opération d'extensions. Là encore, C et L auraient pu être inclus dans $K3$ si cet agrégat avait été

identifié en premier. Là encore, l'utilisation de FA ou MFA à la place de CA rendrait cela possible. Le graphe et le plan sont à nouveau mis à jour avec K_3 (cf. figure 4.10-d).

A la quatrième itération, l'opération de fusion retourne l'agrégat $K_4 = K_1K_2K_3$. L'opération d'extensions n'a aucun agrégat à ajouter à K_4 et le plan et le graphe sont mis à jour pour prendre en compte ce nouvel agrégat (cf. figure 4.10-e).

Le dernier appel à l'opération de fusion ne retourne aucun nouvel agrégat puisque le graphe est réduit à un seul sommet. Le plan est donc terminé (cf. figure 4.10-f). On peut constater que le plan produit par cette exécution a une structure d'arbre renversé. C'est une propriété générale des plans produits avec l'opération de condensation CA. En effet, l'algorithme opère sur les agrégats de l'étape précédent et tente de les assembler en un nouvel agrégat, qui devient alors un descendant commun de tous les agrégats assemblés. L'opérateur de condensation CA retire alors tous les agrégats regroupés, ce qui explique que chaque agrégat ne peut avoir qu'un unique descendant dans le plan d'assemblage.

Phase d'assemblage

La phase d'assemblage n'a jamais été décrite en détail dans les articles de Hoffmann *et al.*. Nous proposons ici quelques déductions sur ce qu'elle pourrait être, partant des principes de résolution utilisés dans les autres méthodes de décomposition géométrique et de notre analyse personnelle de la méthode HLS.

La phase d'assemblage exécute le plan de construction : chaque agrégat est résolu dans un repère local différent. Un agrégat est résolu lui-même en plusieurs étapes : l'opération de fusion qui l'a initialisé correspond à un système d'équations à résoudre, et chaque extension de l'opération d'extensions produit aussi un sous-système indépendant à résoudre.

Hoffmann, Lomonosov et Sitharam ne précisent pas la méthode de résolution utilisée pour résoudre ces sous-systèmes. En revanche, le plan fournit un ordre partiel pour leur résolution, et un guide pour l'assemblage.

La dernière étape consiste à assembler les différents repères locaux des agrégats pour obtenir des solutions du GCSP entier dans un même repère. Cet assemblage final est réalisé en appliquant un déplacement (translation + rotation) à chaque agrégat.

Ainsi, pour le GCSP *Ponts*, le plan produit par la décomposition illustrée à la figure 4.10 induit un ensemble de 13 sous-systèmes (4 étapes de fusion et 9 étapes d'extensions) dont 3 comportent une seule équation (les fusions de K_1, K_2 et K_3), 9 comportent 2 équations (les 9 extensions) et le dernier (fusion pour K_4) comporte 6 équations.

4.3.4 Autres approches géométriques structurelles

Méthode de Bondyfalat

Bondyfalat [Bondyfalat, 2000] propose une méthode pour trouver un ordre constructif d'un GCSP représentant une scène à reconstruire en 3D. Cette méthode se base sur une analyse des degrés de liberté et sélectionne itérativement un objet à reconstruire à chaque étape selon des heuristiques : nombre de degrés de liberté minimal, nombre de con-

traintes liées maximal, ... L'ordre constructif produit peut ne pas permettre de résoudre la scène entière, et il propose alors un mécanisme de retour-arrière qui remet en cause l'ordre constructif. Cela rend la planification exponentielle en temps, ce qui est une mauvaise solution puisque le problème de la planification admet des solutions polynomiales. Cependant, cette méthode permet de traiter les A-redondances, courantes dans le domaine d'application visé : la vision artificielle.

Méthode de Hoffmann et Joan-Arinyo

Hoffmann et Joan-Arinyo [Hoffmann and Joan-Arinyo, 1997] ont proposé un solveur hybride capable de traiter les contraintes géométriques et d'autres contraintes portant sur les dimensions des contraintes géométriques, c-a-d. les valeurs des angles et distances. La partie géométrique est résolue par la méthode structurelle déjà proposée par Fudos et Hoffmann (cf. section 4.3.2). Un résolveur équationnel basé sur la recherche de sous-système soluble par couplage dans un graphe biparti (méthode proche de celle présentée à la section 4.1.1, page 87) permet de déterminer certaines dimensions lorsqu'aucune règle géométrique ne peut plus être appliquée. Les 2 résolveurs opèrent en parallèle et échangent leurs résultats, à savoir les valeurs des variables qu'ils ont pu résoudre. Ainsi, le résolveur géométrique fournit au résolveur équationnel les valeurs des paramètres géométriques qu'il a pu déterminer, ce qui permet d'identifier de nouveaux sous-systèmes équationnels solubles ; le résolveur équationnel fournit pour sa part les valeurs des dimensions qu'il a pu déterminer, ce qui peut permettre l'application de nouvelles règles géométriques dans le résolveur géométrique.

4.4 Décompositions géométriques à base de règles

Les méthodes de décomposition géométrique à base de règles utilisent aussi la propriété de rigidité pour résoudre des GCSP. Elles sont cependant basées sur des répertoires de règles qui sont adaptés à certains types de GCSP, ou aux GCSP issus d'applications particulières (cf. méthode de Kramer adaptée aux mécanismes). Nous présentons ici 3 méthodes représentatives de l'approche : la méthode de Sunde (reprise et étendue par Verroust *et al.*), la méthode de Kramer et la méthode de Mathis.

4.4.1 Méthode de Kramer

La méthode de Kramer [Kramer, 1992] vise à résoudre des GCSP en 3D par une *analyse des degrés de liberté* dans le GCSP. Cette analyse permet d'identifier une séquence d'opérations élémentaires, appelée plan, qui produira un assemblage des solides satisfaisant les contraintes. Kramer développe essentiellement sa méthode pour la résolution et la simulation cinématique de mécanismes en 3D. Un mécanisme est constitué de pièces mécaniques disposant chacune de 6 degrés de liberté et liées par des articulations mécaniques (cf. section 2.2, page 46, pour plus de détails sur les mécanismes).

Chaque pièce mécanique est modélisée par un ensemble de repères locaux nommés *marqueurs*. Les marqueurs associés à une pièce mécanique servent à définir sa position, son orientation et ses dimensions. Par exemple, une brique peut être représentée par 3 marqueurs $m1$, $m2$ et $m3$: $m1$ représente la position et l'orientation de la brique, alors que la position de l'origine de $m2$ et $m3$ permettent de définir sa largeur, sa longueur et sa hauteur (cf. figure 4.11).

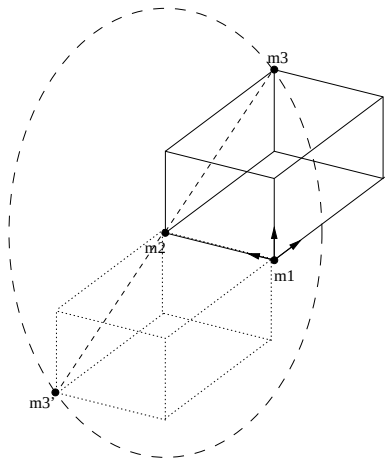

FIG. 4.11 – Représentation d'une barre en 3D par 3 marqueurs

Les articulations entre pièces mécaniques sont représentées par des contraintes sur les marqueurs des pièces mécaniques : coïncidence, alignement, etc. Pour résoudre une contrainte entre deux marqueurs, Kramer propose de n'utiliser que des déplacements des pièces mécaniques (rotations ou translations). Tout déplacement d'une pièce doit respecter les contraintes déjà satisfaites. Concrètement, cela se traduit par le fait que chaque action retire une certain nombre de degrés de liberté à la pièce mécanique qu'elle déplace. Il faut donc maintenir le compte du nombre de degrés de liberté disponible de chaque pièce mécanique. Kramer propose de distinguer les différents types de degrés de liberté : *translationnels, rotationnels et dimensionnels*. Les pièces mécaniques en 3D disposent toutes de 3 degrés de liberté rotationnels et 3 degrés de libertés translationnels, mais aucun degré de liberté dimensionnel : toute pièce mécanique est rigide en théorie des mécanismes.

Analyse des degrés de liberté

L'analyse des degrés de liberté a pour but de produire un plan d'assemblage du GCSP. La planification se base sur la version non-bipartie du graphe objets-contraintes du GCSP. L'algorithme 7 présente la phase d'analyse des degrés de liberté proposée par Kramer.

L'identification de sous-systèmes rigides se fait par application de la formule de Grübler aux boucles d'un mécanisme (cf. section 2.2, page 46). Cette formule donne une *estimation* du degré de mobilité, c-a-d. du nombre de degrés de liberté hors déplacements, dans une

Algorithme 7 Kramer (G : graphe objets-contraintes) *retourne* P : plan d'assemblage

Répéter
$G' \leftarrow$ `SousSystèmeRigide(`G`)`
$K_{G'} \leftarrow$ Analyse-Action-Et-Lieux(G', P).
$G \leftarrow G \cup K_{G'} \setminus G'$
Jusqu'à $G' = GrapheVide$ **ou** $K'_G = NoeudVide$

boucle fermée indépendante. Si cette estimation est négative ou nulle, la boucle est considérée comme rigide et constitue alors le sous-système G' à planifier par *analyse d'actions* et *analyse des lieux*. Kramer propose aussi un certain nombre de méthodes pour identifier les chaînes ouvertes rigides et affiner l'estimation de la formule de Grübler.

Analyse d'actions

L'analyse d'actions a pour but de produire les déplacements nécessaires à la résolution de toutes les contraintes de G'. Ces actions sont sélectionnées dans la *table des fragments de plan* qui associe à chaque contrainte une action dépendant des degrés de liberté disponibles dans la pièce mécanique à déplacer ; cette table définit en outre la mise à jour des degrés de liberté de la pièce déplacée une fois l'action effectuée. La sélection d'un fragment de plan se fait donc en fonction de la contrainte à résoudre, des degrés de liberté disponibles dans la pièce à déplacer, mais aussi en fonction des marqueurs déjà déterminés ; par exemple la rotation de la brique de la figure 4.11 qui amène le marqueur $m3$ en position $m3'$ nécessite que la position du marqueur $m3'$ soit connue afin d'être sélectionnable. La table des fragments de plan proposée par Kramer [Kramer, 1992] contient environ une centaine de règles et permet de traiter les articulations mécaniques classiques. Kramer prouve que l'analyse d'actions est complète et canonique, c-a-d. que l'ordre de sélection des fragments de plan n'importe pas dans la planification.

Analyse de lieux

L'analyse des lieux des marqueurs intervient lorsqu'aucune règle de la table de fragments de plan ne peut être utilisée. Le *lieu* d'un marqueur d'une pièce mécanique est déterminé par les marqueurs fixés de cette pièce et le nombre de degrés de liberté de chaque type restant à cette pièce. Ainsi, le lieu du marqueur $m3$ de la brique de la figure 4.11 est un cercle lorsque les marqueurs $m1$ et $m2$ sont positionnés. Ce cercle est représenté en tirets à la figure 4.11. L'analyse des lieux produit des marqueurs virtuels aux intersections[5] des lieux des marqueurs non déterminés par l'analyse d'actions. C'est la *table d'intersection des lieux* qui définit le type et la façon de calculer les intersections de lieux des marqueurs libres. Des contraintes de coïncidence sont placées entre les marqueurs libres et les marqueurs virtuels

[5]Un seul marqueur virtuel est créé à l'intersection de 2 lieux donnés. Il peut cependant exister plusieurs intersections de 2 lieux donnés : 2 cercles co-planaires peuvent se couper en 2 points. C'est alors à l'utilisateur de choisir le point d'intersection où il souhaite créer un marqueur virtuel.

générés aux intersections de lieux, ce qui permet d'utiliser à nouveau des règles de la table des fragments de plan.

Limites de la méthode

Kramer prouve que cette double analyse, basée sur les 2 tables qu'il a définies, peut résoudre tout mécanisme contenant des boucles fermées rigides d'au plus 3 pièces mécaniques[6] et dont l'expression mathématique peut être résolue de façon analytique. L'utilisation de la formule de Grübler, et la manière structurelle de sélectionner une règle dans la table des fragments de plan, confère cependant à cette méthode les inconvénients des méthodes structurelles (cf. section 2.2.3, page 48, et section 4.5), page 114.

4.4.2 Méthode de Sunde

La méthode proposée par Sunde [Sunde, 1986], puis reprise et étendue par Verroust *et al.* [Verroust *et al.*, 1992], permet de résoudre des GCSP en 2D constitués de points et de segments, représentés par des paires de points, liés par des contraintes de distance et d'angle.

Cette méthode propose de regrouper des sous-parties du GCSP sous la forme de *constrained angle sets* (CA) et *constrained distance sets* (CD). Un CA est un ensemble de segments contraints par des angles. Un CD est un ensemble de points contraints par des distances. Dans la méthode de Verroust *et al.*, un repère local est attaché à chaque CD, et le GCSP est résolu lorsque tous les points qu'il contient appartiennent au même CD. Intuitivement, un CD correspond à un sous-système rigide : tous les points d'un CD disposent d'une position fixe dans le repère local associé au CD.

Initialement, un CA est créé pour chaque paire de segments sur laquelle porte une contrainte d'angle, et un CD *et un CA* sont créés pour chaque paire de points sur laquelle porte une contrainte de distance : un CD contenant la paire des points contraints, et un CA contenant le segment correspondant à cette paire de points. Le repère local associé à chaque nouveau CD se définit en plaçant l'un des points en $(0, 0)$ et l'autre en $(d, 0)$ si leur distance est contrainte à valoir d.

Ces CA et CD initiaux sont alors regroupés en CA et CD d'ordre supérieur par application de règles d'assemblage. Les règles initialement proposées par Sunde permettent simplement d'agréger 2 CA partageant un même segment en un seul CA, et 2 CD partageant un même point et formant un angle fixe en un seul CD. Verroust *et al.* adjoignent 6 règles supplémentaires qui permettent de reconstruire des triangles, des parallélogrammes et des quadrilatères à partir d'un ensemble de CD et CA vérifiant certaines conditions.

Chaque règle fait disparaître un ensemble de CD et CA pour les remplacer par un ensemble réduit de CD. Chaque règle assemble donc un certain nombre de points déjà résolus dans des repères locaux séparés. Cet assemblage se fait par des changements de repères calculés à partir des objets en commun entre les CD (rappelons que seuls les CD sont associés à des repères locaux).

[6] Il est démontré qu'une boucle fermée rigide contient au plus 6 pièces mécaniques en 3D.

Chaque règle est aussi associée à une procédure de vérification qui détecte l'impossibilité de la résolution associée à la règle : elles sont ainsi aptes à prendre en compte les singularités du GCSP. De telles singularités sont dues à des valeurs particulières des distances et des angles ; par exemple, la construction d'un point C à partir de 2 points A et B à distance d_{AB} à l'aide de 2 contraintes de distance d_{AC} et d_{BC} n'est possible que si $A \neq B$ et $d_{AB} < d_{AC} + d_{BC}$.

Cette méthode peut aussi détecter les R-redondances car elle procède par insertion incrémentale des contraintes dans le GCSP : à chaque insertion d'une contrainte, toutes les règles sont déclenchées et les CA et CD sont agrégés jusqu'à ce qu'un point fixe soit atteint ; une contrainte est alors R-redondante si c'est une contrainte d'angle qui porte sur 2 segments déjà dans un même CA, ou si c'est une contrainte de distance qui porte sur une paire de points déjà dans un même CD.

Verroust *et al.* démontrent l'existence de GCSP en 2D, constitués de points, segments, distance et angles, qui ne peuvent pas être reconstruits à partir de leurs règles, ainsi que des classes de GCSP vérifiant des propriétés de connexité qui peuvent être résolus entièrement par leur méthode.

4.4.3 Méthode de Mathis

Mathis [Mathis, 1997; Dufourd *et al.*, 1998] propose un schéma général de décomposition géométrique ayant pour but de fédérer les autres méthodes d'assemblage de GCSP. Ce schéma est décrit pour des GCSP en 2D constitués de points, droites, cercles, distances, angles, incidences, parallélismes. La méthode de Mathis procède aussi en 2 étapes : *planification* puis *évaluation numérique* du système planifié.

Planification

La planification produit un ensemble de sous-figures pouvant être résolues dans des repères locaux et assemblées par unification de ces repères. L'architecture générale de la phase de planification est un système multi-agents avec tableau noir et superviseur :

- Le tableau noir contient les informations partagées par les agents, c-a-d. les données du problème et le plan en cours de production.
- Chaque agent correspond à un opérateur de construction qui transforme les données du problème en fragments du plan de construction. Concrètement, les agents augmentent ou produisent des sous-figures résolues à partir de méthodes de construction qui leurs sont propres.
- Le superviseur déclenche les agents par ordre de priorité ; lorsqu'un agent échoue, le superviseur lance l'agent de priorité directement inférieur, alors que si l'agent produit un nouveau fragment de plan, le superviseur relance l'agent de priorité la plus haute. Le superviseur est aussi chargé de l'assemblage des sous-figures ayant des repères en commun.

Les agents de résolution locaux proposés par Mathis sont au nombre de 3 : 2 agents à base de règles et un agent structurel. Les 2 agents à base de règles sont basés sur une

axiomatisation de l'univers géométrique : les règles de la géométrie euclidienne, l'égalité et les contraintes constituent les axiomes du système. Les opérateurs de construction sont des règles géométriques ayant pour prémisses des propositions de la base de faits (le tableau noir) et pour conclusion un fragment de plan pour le GCSP, c-a-d. une méthode de résolution pour une sous-figure. Par exemple, $onl(A, D)$ est un prédicat qui signifie que le point A appartient à la droite D. Ce prédicat, qui correspond soit à une contrainte initiale du GCSP, soit à une déduction issue d'une règle précédente, peut être associé à la méthode de résolution $D := lpla(A, E, a)$ lors de la planification ; cette méthode construit la droite D passant par le point A et formant un angle a avec la droite E. $fixe(A)$, $fixe(E)$ et $fixe(a)$ sont des prémisses supplémentaires nécessaires pour que cette règle puisse être appliquée ; cela signifie que le point A, la droite E et l'angle a doivent avoir été déterminés auparavant dans la construction.

L'un de ces 2 agents, dit *agent risqué*, dispose de règles introduisant des objets géométriques intermédiaires qui n'existent pas initialement dans le GCSP. La production de tels objets en présence d'un cycle de règles peut conduire à une séquence infinie de construction d'objets intermédiaires, ce qui provoque une boucle infinie dans la planification du système. Cet agent risqué n'est appliqué que lorsque l'autre agent, qui ne dispose de telles règles, ne parvient à produire aucune construction.

L'agent structurel permet de reconstruire des parties du GCSP qui ne peuvent pas être traitées par les 2 agents à base de règles. Il n'est déclenché que si les 2 agents à base de règles ne permettent pas de construire de nouvelles sous-figures ou d'augmenter des sous-figures existantes. Cet agent est basé sur un couplage maximum dans un graphe biparti variables-équations qui permet d'identifier un sous-système carré à résoudre (cf. section 4.1.1, page 87). L'agent structurel ne peut pas échouer si le GCSP initial est structurellement bien-contraint. La planification reprend alors avec l'agent de priorité la plus élevée : l'agent à base de règles non risquées.

A chaque fois qu'une sous-figure est produite ou augmentée par un agent, son *bord* est mis à jour par le superviseur. Le bord d'une sous-figure résolue correspond aux objets géométriques de cette sous-figure sur lesquels portent encore des contraintes du GCSP, et qui peuvent donc être partagés par d'autres sous-figures. Les valeurs à associer aux contraintes valuées pouvant être déduites entre les objets du bord d'une sous-figure sont placées sous forme de contraintes entre ces objets. Ces contraintes sont paramétrées afin de respecter les propriétés de la sous-figure résolue. Par exemple, si le bord d'une sous-figure contient 2 points, une contrainte de distance peut être placée entre ces points. La valeur de cette distance est paramétrée de sorte qu'elle soit toujours égale à la distance entre ces points produite par la résolution de la sous-figure.

Le superviseur est aussi chargé d'assembler des sous-figures. Deux sous-figures sont assemblables si elles partagent des objets (de leurs bords respectifs) définissant au moins un repère. Un repère en 2D se définit par un point et une direction. Leurs repères locaux sont alors unifiés et l'une des sous-figures est déplacée (rotation+translation) pour mettre en correspondance les objets qu'elles partagent. Cette opération se traduit par une séquence d'assemblage dans le plan de construction, c-a-d. le calcul et l'application du déplacement à effectuer pour assembler les sous-figures.

Evaluation numérique du plan

La phase d'évaluation numérique du plan utilise des méthodes précalculées pour chaque fragment du plan généré par l'un des agents à base de règles. Ces méthodes sont capables de fournir toutes les solutions d'une construction correspondant à une règle. Les fragments de plan produits par l'agent structurel sont résolus par la méthode de Newton-Raphson, qui ne peut produire toutes les solutions du fragment considéré. La méthode d'évaluation n'est donc pas complète si l'agent structurel est utilisé. Pire, l'évaluation numérique peut échouer si la solution produite par Newton-Raphson pour une sous-figure n'est compatible avec aucune des solutions des autres sous-figures.

Essert-Villard [Essert-Villard, 2001] propose une extension de l'évaluation de la méthode de Mathis. Elle prend en compte une esquisse initiale et propose plusieurs méthodes, automatiques et semi-automatiques, permettant d'identifier la solution attendue et de guider l'évaluation numérique du plan.

4.4.4 Autres approches géométriques à base de règles

Méthode de Sosnov et Macé

Sosnov et Macé [Sosnov and Macé, 2002] proposent une méthode assez proche de celle de Bondyfalat (cf. section 4.3.4, page 106). En utilisant l'algèbre de Grassman-Cayley, ils proposent de résoudre symboliquement les contraintes affines et projectives, c-a-d. les contraintes d'incidence et de parallélisme. La méthode est basée sur un mécanisme d'inférence qui produit des contraintes supplémentaires à partir d'un jeu réduit de règles d'inférences. Ces règles ne permettent pas de produire toutes les déductions possibles, mais elles permettent de déduire toute incidence ou parallélisme déductible sans introduction d'objet supplémentaire. Cela permet de vérifier la consistance structurelle du GCSP. Une analyse du graphe objets-contraintes permet de déterminer une séquence d'opérations de construction. Cette séquence est ensuite exécutée : l'évaluation numérique peut se faire de façon exacte sur les rationnels car les contraintes sont toutes linéaires ou bilinéaires. La méthode ne permet cependant pas de traiter les contraintes valuées représentant des dimensions. L'application visée est la reconstruction de scène 3D où les distances et angles ne sont pas donnés : la reconstruction se fait à des facteurs d'échelle près selon chaque dimension.

Méthode de Joan-Arinyo et Soto-Riera

Dans la continuité de ses travaux avec Hoffmann, Joan-Arinyo a proposé en collaboration avec Soto-Riera [Joan-Arinyo and Soto-Riera, 1999] un autre solveur hybride capable de traiter les contraintes géométriques et d'autres contraintes portant sur les dimensions, la partie géométrique étant cette fois résolue par un système de réécriture très proche de la méthode de Verroust *et al.* (cf. section 4.4.2, page 110) : c'est un système à base de règles qui permettent de reconstruire des triangles et des quadrilatères à partir de CA, CD et CH ; un CH est un sous-GCSP qui contient un point et un segment contraints par une distance orthogonale. Un résolveur équationnel et le principe de sa collaboration avec le résolveur

géométrique reste le même que celui déjà proposé dans [Hoffmann and Joan-Arinyo, 1997] (cf. section 4.3.4, page 106). Cette collaboration est cependant formalisée comme une règle afin de respecter la description du double résolveur comme un système de réécriture.

4.5 Analyse des différentes décompositions

Dans cette section, nous proposons de comparer les différentes méthodes que nous avons présentées sur les critères suivants (cf. tableau 4.1) :

- **type** : type de GCSP traitable, c-a-d. dimension, types d'objets et de contraintes ;
- **sous, sur** capacité à gérer les GCSP sous- ou sur-rigides, c-a-d. à les détecter au moins, à proposer une façon de les résoudre au mieux ;
- **r&s** : capacité à détecter et traiter les A-redondances/R-redondances et singularités ;
- **comp, corr.** : complétude et correction. Par complétude, on entend que l'algorithme termine et fournit une décomposition s'il en existe une ; par correction, on entend que l'algorithme fournit une décomposition correcte, c-a-d. soluble.
- **temps** : classe de complexité en temps de la planification (hors temps de résolution).

méthode	type	sur	sous	r&s	comp	corr	temps
Décompositions équationnelles structurelles							
DM-CFC	tous	oui	oui	non	oui	non	poly
OpenPlan	tous	oui	oui	non	oui	non	expo
Latham-Mid.	tous	oui	oui	non	non	non	poly
Hsu-Brüder.	tous	non	oui	non	non	?	?
Décompositions équationnelles à base de règles							
GPDOF	règles nécessaires	non	oui	non	non	oui	poly
Décompositions géométriques structurelles							
Owen	2D, point, droite distance, angle	non	oui	non	oui	non	poly
Fudos-Hoff.	2D, point, droite distance, angle	oui	oui	non	oui	non	poly
Bondyfalat	algébriques	oui	oui	oui	oui	oui	expo
Hoff.-Arin.	2D, point, droite distance, angle	oui	oui	non	oui	non	poly
HLS	objets rigides	oui	oui	non	oui	non	poly
Décompositions géométriques à base de règles							
Kramer	mécanismes 3D	non	non	non	non	non	poly
Sunde-Verr.	2D, point, segment distance, angle	oui	non	oui	non	oui	poly
Mathis	2D, tous	oui	non	non	non	non	poly
Arinyo-Rie.	2D, point, segment distance, angle	oui	oui	oui	non	oui	poly

TAB. 4.1 – Comparaisons des méthodes de décomposition de GCSP

4.5.1 Equationnelle VS Géométrique

Les méthodes équationnelles travaillent au niveau des variables et des équations, alors que les méthodes géométriques travaillent au niveau des objets et des contraintes géométriques. Nous avons expliqué au chapitre 1 que le niveau variables-équations correspond à un affinement du niveau objets-contraintes puisque chaque objet géométrique est représenté par plusieurs variables, alors que chaque contrainte géométrique est représentée par un système d'équations.

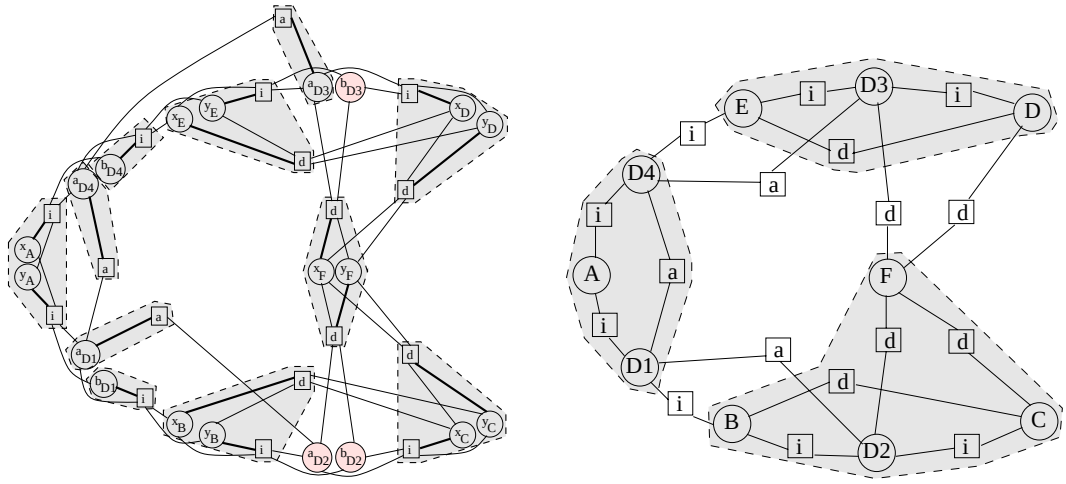

FIG. 4.12 – Exemple de GCSP décomposé plus finement par DM-CFC que par HLS

On pourrait donc penser que les décompositions équationnelles sont généralement plus fines que les décompositions géométriques. En réalité, les 2 sont incomparables, car il existe aussi des GCSP décomposés plus finement par des méthodes géométriques que par des méthodes équationnelles :

- le GCSP *Ponts* est décomposé plus finement par la méthode HLS (cf. figure 4.10, page 105) que par OpenPlan (cf. figure 4.4, page 92) : avec OpenPlan, il reste un sous-système de 14 équations, alors que le plus gros sous-système produit par la méthode HLS contient 6 équations.
- En revanche, le GCSP présenté à la figure 4.12 est décomposé plus finement par OpenPlan que par HLS : la décomposition par OpenPlan, présentée en partie gauche de la figure 4.12, produit des sous-systèmes d'au plus 2 équations, alors que la décomposition HLS, présentée en partie droite de cette même figure, contient un sous-système de 6 équations. Ce GCSP est composé de 6 points A, B, C, D, E et F et de 4 droites $D1$, $D2$, $D3$ et $D4$, liés par les contraintes Angle$(D1, D2)$, Angle$(D3, D4)$, Angle$(D1, D4)$, Distance(B, C), Distance(C, F), Distance(F, D), Distance(D, E), Distance$(F, D2)$, Distance$(F, D3)$, Incidence$(A, D1)$, Incidence$(A, D4)$, Incidence$(B, D1)$, Incidence$(B, D2)$, Incidence$(C, D2)$, Incidence$(D, D3)$, Incidence$(E, D3)$, Incidence$(E, D4)$.

Nous avons vu que la raison pour laquelle une méthode équationnelle peut décomposer plus finement qu'une méthode géométrique est qu'elle travaille au niveau variables-équations ; ceci lui permet ainsi de résoudre des variables d'un même objet géométrique dans différents sous-systèmes, ce qui n'est pas possible pour une méthode géométrique.

La raison pour laquelle une méthode géométrique peut décomposer plus finement qu'une méthode équationnelle est qu'elle tire parti de la géométrie du GCSP. En effet, toutes les méthodes géométriques que nous avons présentées ont pour point commun de résoudre des sous-GCSP indépendamment dans des repères locaux différents puis de procéder à un assemblage des différents repères locaux. Ceci permet à ces méthodes de reprendre plusieurs fois les degrés de liberté correspondant à un repère local, ce qui peut briser des composantes fortement connexes, comme c'est le cas pour le GCSP *Ponts*.

En résumé, ce sont les spécificités suivantes qui rendent les méthodes équationnelles et géométriques incomparables :

- Une méthode équationnelle peut résoudre des variables d'un même objet dans des sous-systèmes différents.
- Une méthode géométrique peut utiliser plusieurs fois les degrés de liberté d'un GCSP rigide.

Nous pensons que l'avantage des méthodes géométriques est plus important lorsque l'on envisage de résoudre des GCSP de grande taille. Par exemple, si l'on colle 3 fois le GCSP *Ponts* pour former une pyramide de taille supérieure, la décomposition OpenPlan devient catastrophique (un bloc contient 50 équations sur les 81 équations du GCSP entier) alors que la décomposition HLS reste inchangée (toujours des sous-systèmes d'au plus 6 équations). Nous avons donc opté pour une méthode de décomposition géométrique.

Cependant, on peut aussi constater que :

- Les méthodes équationnelles peuvent traiter tout type de GCSP, même des GCSP sous-rigides, du moment qu'elles savent traiter la composante sous-contrainte d'un CSP.
- Les méthodes géométriques ne peuvent traiter, au mieux, que les GCSP rigides.

Mais rien n'empêche de compléter une décomposition géométrique par une décomposition équationnelle. Nous ne proposons pas de schéma de collaboration détaillé, cependant, on peut imaginer que chaque sous-GCSP produit par une décomposition géométrique est ensuite décomposé par une méthode équationnelle qui peut éventuellement affiner la décomposition.

Le problème du repère local

Les méthodes géométriques ont donc pour avantage de résoudre plusieurs sous-GCSP dans des repères locaux différents, ce qui permet de ré-utiliser plusieurs fois les degrés de liberté d'un GCSP rigide et peut produire une décomposition plus fine qu'une décomposition équationnelle. Ce principe implique qu'il faut ensuite être capable d'assembler les sous-GCSP résolus dans des repères locaux différents.

Chaque méthode utilise une façon unique de procéder à l'assemblage, mais toutes ont cependant un point commun : elles utilisent la possibilité d'appliquer un même déplacement

à tous les objets d'un sous-système rigide. Généralement l'assemblage consiste donc à découvrir le déplacement T qu'il faut appliquer à un sous-système $S1$ résolu indépendamment afin de le positionner dans le repère local d'un sous-système $S2$.

Il existe une condition pour que ce déplacement puisse être calculé : il faut que $S2$ contienne un sous-ensemble $O1$ d'objets de $S1$ *qui définisse un repère*. En effet, si tel est le cas, la résolution de $O1$ dans $S1$ et dans $S2$ définit 2 repères et il est possible de calculer le changement de repère qui correspondra au déplacement à appliquer à tous les objets de $S1$ afin de l'assembler avec $S2$. En revanche, si $O1$ ne permet pas la définition d'un repère, c'est que $O1$ ne permet pas de fixer les $\frac{d(d+1]}{2}$ DDL qui représentent des déplacements de $S1$, et il est donc impossible d'assembler $S1$ et $S2$: $S1$ admettra une infinité de position dans $S2$.

Considérons par exemple le GCSP présenté comme contre-exemple à la caractérisation de Laman en 3D (cf. exemple 2.10, page 44). Cette *double-banane* en 3D est constituée de 2 sous-GCSP bien-rigides : chaque *banane* est en effet bien rigide en elle même. Cependant, leur assemblage par les pointes engendre un GCSP sous-rigide ; pourtant cet assemblage peut être réalisé par simple déplacement de chacune des bananes. Ce qui explique que ce GCSP est sous-rigide malgré tout, c'est le fait que ces 2 sous-GCSP rigides sont assemblés à l'aide de 2 points seulement. Or, une paire de points ne permet pas de définir un repère local en 3D.

Le problème du repère local consiste à se demander si un sous-ensemble d'objets géométriques dans un GCSP permet de définir un repère local dans lequel les objets restants seront positionnés de façon unique si le GCSP est rigide. Plus formellement :

Problème 1 Repère local
Données : *Soit $S = (O, C)$ un GCSP en dimension d et O' un sous-ensemble d'objets dans ce GCSP.*
Question : *O' permet-il la définition d'un repère local en dimension d ?*

Cette question est, comme nous venons de le voir, cruciale pour les méthodes de décomposition géométrique qui procèdent par assemblage de sous-problèmes.

Rappelons qu'un repère local se définit généralement par une origine et un ensemble de $d-1$ vecteurs orthonormés en dimension d. On peut donc utiliser un ensemble d'objets géométriques pour définir un repère en définissant son origine et ses vecteurs. Par exemple, en 3D, un ensemble de 3 points A, B, et C permet de définir un repère local de la façon suivante :
$(A, \frac{\overrightarrow{AB}}{||\overrightarrow{AB}||}, \frac{\overrightarrow{AB} \wedge \overrightarrow{AC}}{||\overrightarrow{AB} \wedge \overrightarrow{AC}||})$

Cependant, cette définition n'a de sens que si les points A ,B et C ne sont pas alignés, car dans ce cas le produit vectoriel est nul et ne permet pas de définir le second vecteur du repère local. Il existe donc des conditions géométriques pour qu'un ensemble d'objets permette la définition d'un repère local, ce qui explique que ce problème n'admette pas de solution structurelle.

Nous exposerons à la section 6.4 une solution au problème du repère local, basée sur le nouveau concept de *degré de rigidité*.

4.5.2 Règles VS Structurelles

Une méthode à base de règles, équationnelle ou géométrique, n'est complète que relativement à l'ensemble de ses règles, c-a-d. qu'elle ne permet de résoudre que les GCSP peuvant se décomposer en une séquence des règles dont elle dispose.

A l'opposé, les méthodes structurelles, qu'elles soient équationnelles ou géométriques, sont généralement complètes mais peuvent être incorrectes. En effet, ces méthodes utilisent des caractérisations structurelles inaptes à identifier et traiter les A-redondances/R-redondances et singularités, ce qui peut rendre la décomposition fausse : les parties identifiées comme bien-contraintes ou bien-rigides ne le sont que structurellement (cf. sections 1.4 (page 21) et 2.3.3 (page 59)).

Une méthode structurelle nous paraît cependant plus intéressante de par la généralité qu'elle offre : aucun système à base de règles apte à résoudre une majorité d'instances de GCSP en 3D n'a encore été proposé, simplement car il s'agit d'un problème très difficile, et qu'un GCSP bien-rigide de taille minimale (c-a-d ne contenant aucun sous-GCSP bien-rigide) peut être arbitrairement grand, tout comme il peut exister des composantes fortement connexes de taille arbitrairement grande dans un graphe. Un outil de CAO ne peut donc pas être basé uniquement sur des méthodes à base de règles.

4.5.3 Choix d'une méthode de décomposition

Nous avons donc choisi d'utiliser une méthode de décomposition géométrique structurelle. Parmi les 3 méthodes, la méthode HLS est la plus générale :

- elle permet de traiter tout GCSP en dimension quelconque,
- elle est capable d'identifier des sous-GCSP rigides minimaux de taille arbitraire,
- elle peut réaliser des assemblages de taille arbitraire.

Dans la suite de ce mémoire, nous allons détailler cette méthode et expliquer ses limites, dues à sa nature structurelle mais aussi aux algorithmes qu'elle utilise. Nous proposerons des solutions à certains problèmes laissés en suspens ou dont la solution n'a jamais été détaillée par Hoffmann *et al.*

Nous proposons ensuite de lever certaines limites des approches géométriques structurelles en analysant de plus près la rigidité structurelle et ses limites et en proposant le concept de degré de rigidité qui permet de prendre en compte la géométrie du GCSP en plus de sa structure. Nous verrons que ce concept permet par ailleurs de répondre au problème du repère local et donc de déterminer si un assemblage est possible ou pas.

Enfin, nous proposerons une nouvelle méthode de rigidification récursive, basée sur le schéma de la méthode HLS mais utilisant de nouvelles opérations de fusion, d'extension et de condensation ; nous expliciterons les limites de l'approche, et discuterons des solutions qui permettent de vérifier si les conditions d'utilisation de notre méthode sont remplies. Nous décrirons aussi une phase d'assemblage basée sur la résolution d'un DAG de blocs par techniques de résolution par intervalles.

Deuxième partie

Contributions

Chapitre 5

Méthode Hoffmann, Lomonosov, Sitharam (HLS)

Sommaire

Dans ce chapitre, nous détaillons la méthode proposée par Hoffmann, Lomonosov et Sitharam [Hoffmann *et al.*, 1997; 1998; 2000]. Cette méthode de décomposition géométrique structurelle ascendante généralise celle initialement proposée par Fudos et Hoffmann : elle se base sur une analyse des degrés de liberté dans le graphe objets-contraintes afin d'identifier des sous-systèmes rigides, appelés *agrégats*, qui sont assemblés récursivement ; cette méthode appartient à la classe des méthodes dites de *rigidification récursive* de GCSP. Elle a été déclinée en 3 versions, *CA*, *FA* et *MFA* [Hoffmann *et al.*, 2000] qui suivent le même schéma général et ne diffère que par l'*opérateur de condensation* qu'elles utilisent.

Sitharam *et al.* [Oung *et al.*, 2001] ont utilisé la version MFA[1] de cette méthode pour réaliser un outil de CAO, appelé FRONTIER. Une version détaillée de cet article explique que certaines modifications furent nécessaires afin de pouvoir réaliser un logiciel basé sur la méthode. Au cours d'échanges avec Meera Sitharam [Sitharam, 2000], nous avons pu en apprendre plus sur la façon dont la méthode a été adaptée afin de pouvoir être utilisée en pratique.

C'est donc une description détaillée de la méthode HLS et de ses dernières modifications que nous allons présenter ici. Nous proposerons aussi nos propres solutions à certains problèmes laissés en suspens par Hoffmann *et al.*, tels que la collaboration des différentes opérations lors de la phase d'analyse, ou encore l'association de la phase d'analyse avec la phase d'assemblage. Hoffmann *et al.* ont toujours utilisé une description de leur méthode en termes de graphe et de transformations de graphe. Nous proposons d'expliquer cette méthode comme opérant directement sur le GCSP, ce qui clarifie les opérations et fait ressortir certaines difficultés.

5.1 GCSP traitables

La méthode HLS est très générale puisqu'elle permet de traiter des GCSP constitués d'objets géométriques et de contraintes géométriques quelconques en espace géométrique de dimension quelconque. Elle impose cependant deux restrictions :

- les objets géométriques doivent être rigides, c-a-d. ne disposer d'aucune dimension variable ;
- les contraintes géométriques doivent être indépendantes du repère global considéré, c-a-d. que sont proscrites les contraintes permettant de fixer la position ou l'orientation d'un objet vis-à-vis du repère global de l'espace géométrique considéré.

La première limitation est liée à la façon dont procède la phase d'analyse : initialement chaque objet géométrique est considéré comme un sous-GCSP rigide en lui-même ; la méthode consiste ensuite à agréger peu à peu des sous-GCSP rigides. Cette limitation est assez importante : en pratique, on aimerait pouvoir gérer les cercles à rayons variables,

[1] Dans l'article [Oung *et al.*, 2001], c'est FA qui est mentionné ; toutefois, au vu de la description succincte qui en est donnée dans cet article, il semblerait que ce soit en réalité MFA qui est utilisé dans FRONTIER.

et autres objets à dimensions non fixes. Des travaux récents [Hoffmann and Chiang, 2001] visent à lever cette limitation en 2D.

La seconde limitation est plus générale, puisqu'elle est liée à toute méthode géométrique qui utilise le principe d'assemblage : la phase de planification décompose un GCSP ou sous-GCSP rigides qui sont ensuite assemblés par déplacements (rotation+translation) ; de tels déplacements ne sont possibles que si les objets ne sont pas fixés dans le repère global, car alors le GCSP serait par définition indéplaçable, c-a-d. sur-rigide. Lamure et Michelucci [Lamure and Michelucci, 1998] ont expliqué comment il est possible de lever cette limitation de façon générale : l'introduction d'un nouvel objet géométrique repère r dans le GCSP permet de reporter les contraintes dépendantes du repère global sur r. Puisque r est un objet géométrique comme les autres, le GCSP obtenu est alors bien indépendant du repère global. Après la décomposition et la résolution du GCSP décomposé, il est alors possible de satisfaire ces contraintes en choisissant de positionner cet objet repère à l'origine du repère global par déplacement.

5.2 Schéma général de la méthode

Un aperçu général de la méthode a déjà été proposé en section 4.3.3, page 104. Nous revenons ici sur les aspects plus techniques liés aux phases d'analyse (ou de planification) et d'assemblage constituant la méthode.

5.2.1 Phase d'analyse

La phase d'analyse a pour but de produire un *plan d'assemblage* pour un GCSP en appliquant itérativement une séquence de 3 opérations : fusion, extensions et condensation. Elle termine lorsque l'étape de fusion échoue.

L'opération de fusion a pour but de retourner un sous-GCSP bien-rigide, appelé *agrégat*[2], de taille minimale.

L'opération d'extensions[3] produit autant d'extensions que possible ; une extension consiste à inclure un objet *voisin* dans l'agrégat courant ; elle n'est possible que si le sous-GCSP résultant est bien un agrégat, c-a-d. qu'il est bien-rigide.

L'opération de condensation remplace alors l'ensemble des objets regroupés dans l'agrégat courant par un nouvel ensemble d'objets dans le GCSP, et met à jour le plan d'assemblage. Hoffmann *et al.* proposent 3 opérations de condensation différentes : *CA*, *FA* et *MFA*.

La phase de planification est un algorithme de point fixe qui applique opérations de fusion, d'extensions et de condensation en séquence jusqu'à ce que l'opération de fusion échoue à identifier un nouvel agrégat. La complétude de cette opération assure alors qu'il n'existe plus de sous-GCSP rigides dans le GCSP.

[2]Nous avons traduit par le terme agrégat le terme anglais classique *cluster*.

[3]Nous utilisons le terme, peu correct du point de vue grammatical, opération d'extensions pour mettre l'accent sur le fait que cette opération réalise *plusieurs* extensions.

La caractérisation de la rigidité étant difficile dans des GCSP quelconques (cf. section 2.3, page 50), la méthode utilise la *rigidité structurelle* présentée à la section 2.3.3, page 59. Nous avons expliqué que cette caractérisation approximative, basée sur un compte de degrés de liberté, peut être trompée par la présence d'inconsistances, de R-redondances ou de singularités dans le GCSP. Nous verrons que c'est l'utilisation de cette approximation purement structurelle qui confère ses principales limites à la méthode HLS, mais que certains des algorithmes qu'elle emploie posent aussi d'autres problèmes.

Le plan d'assemblage produit par cette phase contient chaque agrégat identifié et toutes les informations relatives à son identification : fusion (flot + sous-GCSP) et extensions (chaque extension dans l'ordre). Il induit une relation d'ordre partielle sur les agrégats pour leur résolution : un agrégat ne peut être résolu avant que tous les agrégats qu'il regroupe ne le soient.

5.2.2 Phase d'assemblage

La phase d'assemblage n'a jamais été décrite en détail dans les articles de Hoffmann *et al.*. Il y est fait référence dans la version détaillée de l'article [Oung *et al.*, 2001].

Elle consiste à résoudre chaque agrégat produit durant la phase de planification dans un repère local différent avant de les assembler. Chaque agrégat est en fait constitué de plusieurs sous-systèmes d'équations : un sous-système d'équations correspondant au sous-GCSP identifié à l'opération de fusion, puis un sous-système d'équations par extension. La première étape consiste à générer tous ces sous-systèmes.

La résolution complète des agrégats doit respecter l'ordre partiel induit par le plan d'assemblage : un agrégat doit être résolu une fois que tous les agrégats qu'il regroupe sont résolus.

Pour résoudre chaque sous-système d'un agrégat, FRONTIER [Oung *et al.*, 2001] utilise un résolveur algébrico-numérique capable de résoudre un agrégat de façon complète et de retourner une formulation algébrique de la variété des solutions dans le cas où le système est sous-contraint (ensemble infini de solutions). La ou les méthodes de résolution employées ne sont pas spécifiées.

Les sous-systèmes d'équations peuvent admettre plusieurs solutions. Sitharam *et al.* utilisent des heuristiques afin de ne sélectionner qu'une seule solution par sous-système. Nous avons expliqué à la section 3.3.1 (page 79) l'importance d'utiliser une méthode de résolution complète afin d'assurer qu'au moins un assemblage sera possible. Sitharam *et al.* ne précisent pas le mécanisme utilisé afin d'assurer qu'au moins une solution assemblée est trouvée.

Une fois les agrégats résolus dans des repères locaux différents, il faut procéder à l'assemblage. Nous avons déjà évoqué l'assemblage dans la section 4.5.1, page 116. L'assemblage consiste à calculer les différents changements de repères qui permettent d'assembler les solutions des agrégats. Nous verrons que selon l'opération de condensation utilisée, l'assemblage est effectué de façon différente (cf. section 5.5).

Nous proposons au chapitre 8 une nouvelle phase d'assemblage détaillée pour notre méthode de rigidification récursive et nous verrons que cette phase d'assemblage peut être

utilisée pour la méthode HLS ; elle entrelace résolution par intervalles et assemblage grâce à un processus de retour-arrière, afin d'éviter la génération de solutions incompatibles pour les agrégats.

5.3 Opération de fusion

L'opération de fusion vise à identifier un sous-GCSP bien-s_rigide dans un GCSP. Pour ce faire, Hoffmann *et al.* introduisent le concept de *densité*. Intuitivement, la densité correspond au compte des degrés de liberté d'un GCSP. Plus formellement :

Définition 39 Densité
*Soit S un GCSP et k un entier. On dit que S est k***dense** *si $DDL(S) \leq k$.*

La notion de densité est définie sur des graphes pondérés dans [Hoffmann *et al.*, 1997], et c'est le graphe objets-contraintes pondéré par les degrés de liberté des objets et des contraintes qui fait le lien entre cette notion de graphe et les GCSP. Nous jugeons inutile d'utiliser l'intermédiaire du graphe, c'est pourquoi nous proposons la définition précédente basée directement sur le GCSP.

Nous précisons cette définition comme suit :

- S est **bien-$k$dense** si et seulement si $DDL(S) = k$
- S est **sur-$k$dense** si et seulement si $DDL(S) < k$
- S est **sous-$k$dense** si et seulement si $DDL(S) > k$

Nous établissons le lien entre la définition de s_rigidité (cf. définition 32, page 60) et la notion de densité par la proposition suivante :

Proposition 8 Densité et s_rigidité
Soit S un GCSP en dimension d. Alors :

- *S est bien-s_rigide si et seulement si S est bien-$\frac{d(d+1)}{2}$dense et $\forall S' \subset S$, S' est bien- ou sous-$\frac{d(d+1)}{2}$dense.*
- *S est sur-s_rigide si et seulement si $\exists S' \subset S$ tel que S' est sur-$\frac{d(d+1)}{2}$dense.*
- *S est sous-s_rigide si et seulement si S est sous-$\frac{d(d+1)}{2}$dense et $\forall S' \subset S$, S' est bien- ou sous-$\frac{d(d+1)}{2}$dense.*

Cette proposition peut être vérifiée simplement à partir des définitions de densité et de s_rigidité.

Hoffmann *et al.* définissent aussi le concept de sous-GCSP *minimal* dense ; intuitivement, un sous-GCSP est minimal dense s'il ne contient aucun sous-GCSP dense. Plus formellement :

Définition 40 Minimal dense
Soit S un GCSP et S' un sous-GCSP de S. S' est dit **minimal** k**dense** *si et seulement si S' est $k$dense et $\forall S'' \subset S'$, S'' est sous-$k$dense.*

L'intérêt des sous-GCSP minimaux denses se trouve dans la proposition suivante que nous proposons :

Proposition 9 Minimal dense et minimal s_rigide
Soit S un GCSP et $S' \subset S$ un sous-GCSP de S. Il y a équivalence entre les 2 propositions suivantes :

1. *S' est minimal $\frac{d(d+1)}{2}$dense*
2. *S' est minimal bien- ou sur-s_rigide*

En effet, un sous-GCSP minimal bien-$k$dense possède au plus k DDL et tout ses sous-GCSP possèdent strictement plus de k DDL. En remplaçant k par $\frac{d(d+1)}{2}$, on peut constater que cela correspond à la définition d'un sous-GCSP bien- ou sur-s_rigide :

- si S' possède moins de $\frac{d(d+1)}{2}$ DDL, il est sur-$\frac{d(d+1)}{2}$dense et sur-s_rigide ; comme il ne contient aucun sous-GCSP $\frac{d(d+1)}{2}$dense, il est alors minimal sur-s_rigide
- s'il possède exactement $\frac{d(d+1)}{2}$ DDL, le fait qu'il ne contienne aucun sous-GCSP bien- ou sur-$\frac{d(d+1)}{2}$dense fait de lui un sous-GCSP minimal bien-s_rigide.

La proposition 9 indique que la recherche de sous-graphes minimaux $\frac{d(d+1)}{2}$dense dans un GCSP permet d'identifier des sous-GCSP bien- ou sur-s_rigides de taille minimale. Identifier de tels sous-systèmes est intéressant pour l'algorithme HLS car :

1. Un sous-GCSP minimal s_rigide constitue un agrégat minimal ; un tel agrégat résultant de l'opération de fusion est intéressant car le GCSP identifié correspondra à un système d'équations à résoudre en une fois lors de la phase d'assemblage. En minimiser la taille améliore donc la finesse de la décomposition.

   On peut alors se poser la question de savoir si un sous-GCSP *minimum s_rigide* ne serait pas *encore plus intéressant*? La recherche d'un sous-GCSP minimum $k$dense est généralement un problème NP-difficile [Hoffmann *et al.*, 1998].

   En étudiant cette démonstration, nous avons remarqué qu'elle repose sur une hypothèse inexistante en pratique : elle suppose que le nombre de DDL des objets et des contraintes n'est pas borné. Nous proposons en annexe B (correspondant à l'article [Jermann and Trombettoni, 2002]) une analyse plus exacte de la classe de complexité de ce problème. Cette analyse nous a permis d'identifier des sous-classes polynomiales qui contiennent des classes pratiques de GCSP, mais pas tous les GCSP.
2. Un sous-GCSP minimal sur-s_rigide constitue une explication minimale pour la sur-rigidité d'un GCSP. En effet, tout GCSP contenant un sous-GCSP sur-rigide est par définition sur-rigide. Présenter un sous-GCSP minimal sur-s_rigide permet donc à un utilisateur de mieux cerner d'où provient la sur-s_rigidité de son système pour y remédier. On peut ainsi, comme nous le verrons à la section 5.6.1 utiliser cet algorithme pour identifier automatiquement certaines R-redondances d'un GCSP : celles qui correspondent à des sur-s_rigidités.

Afin de permettre l'identification de sous-GCSP minimaux $k$denses, Hoffmann *et al.* ont proposé l'algorithme `Minimal-Dense` [Hoffmann *et al.*, 1997]. Cet algorithme, basé sur

le calcul d'un flot maximum dans un réseau représentant le GCSP, permet d'identifier un sous-graphe $k$dense pour une valeur de k positive ou nulle. C'est le premier algorithme purement structurel permettant l'identification de sous-systèmes bien- ou sur-s_rigides qui n'est pas limité par un répertoire de règles d'assemblage particulières. En effet, cet algorithme peut identifier un sous-GCSP minimal s_rigide de taille quelconque ; ce point est important car il peut exister de tels sous-GCSP de taille arbitraire, qui échappent donc à toute technique basée sur un répertoire :

Proposition 10 Taille de minimal dense
Pour tout entier positif t, on peut construire un GCSP minimal bien-$k$dense contenant t sommets.

Nous prouvons cette proposition pour $t > 10$ et $k = 6$; les valeurs inférieurs de t peuvent être facilement exhibées, et la valeur $k = 6$ est pertinente pour les GCSP en 3D (cf. proposition 8, page 125). Nous avons vu qu'un GCSP peut être représenté par son graphe objets-contraintes dont les sommets sont les objets du GCSP, et les arêtes sont ses contraintes ; ce graphe peut être pondéré par les DDL des objets et des contraintes. On peut alors étendre la définition de densité au graphe objets-contraintes pondéré $G = (V, E, w)$ en utilisant la fonction de pondération $w(G) = w(V) - w(E)$, qui représente alors le calcul des DDL d'un sous-GCSP. Pour démontrer la proposition 10, nous proposons la définition inductive suivante d'une famille de graphes tous minimaux (6)denses :

Base : G_0 est le graphe constitué de 10 sommets ayant chacun 6 DDL (des repères en 3D par exemple) liés par 54 arêtes retirant chacune 1 DDL (des distances ou angles en 3D par exemple), tel qu'aucune paire de sommets n'est liée par plus d'une arête ; ce graphe est le graphe complet K_{10} dans lequel on retire 1 arête. Il est minimal bien-(6)dense : $\forall G' \subset G_0$, $w(G') = |V'| * 6 - |E'| \geq |V'| * 6 - \frac{|V'|(|V'|+1)}{2} > 6$ car $|V'| < 10$. La figure 5.1-a présente le graphe G_0.

Induction : G_{n+1} est construit à partir de G_n en introduisant un sommet u de poids 6 lié par 7 nouvelles arêtes (u, v_1), ..., (u, v_7) à 7 sommets v_1, ..., v_7 de G_n tel que (v_1, v_2) est une arête de G_n qui est retirée dans G_{n+1}.

On démontre que si G_n est minimal-(6)dense, alors G_{n+1} l'est aussi :

- d'une part, $w(G_{n+1}) = w(u) - 7 + w(G_n) + 1 = w(G_n)$;
- d'autre part, $\forall G' = (V', E') \subset G_{n+1}$,
  - soit $G' \subset G_n$ alors $w(G') > 6$ car G_n est minimal-(6)dense par hypothèse ;
  - soit $G' = (\{u\} \cup V'', \{(u, v_{i_1}), ...(u, v_{i_j})\} \cup E'')$ avec $G'' = (V'', E'') \subset G_n$; alors $w(G') = w(u) - w((u, v_{i_1})) - ... - w((u, v_{i_k})) + w(G'') = 6 - \delta + w(G'')$;
    si $j = 7$, $w(G') > 7$ car l'arête (v_1, v_2) a été retirée dans G_{n+1} ;
    si $j < 7$, alors $\delta < 7$ et on obtient $w(G') \geq w(G'')$; or $G'' \subset G_n$ donc $w(G'') > 6$, d'où $w(G') > 6$.
    Il en résulte que $\forall G' \subset G_{n+1}$, G' est sous-(6)dense.

La figure 5.1-b présente la construction de G_{n+1} à partir de G_n. L'opération utilisée pourrait être appelée 7-arête-séparation car elle constitue une généralisation des opérations de 2-arête-séparation de la construction de Henneberg (cf. section 2.1.7, page 42).

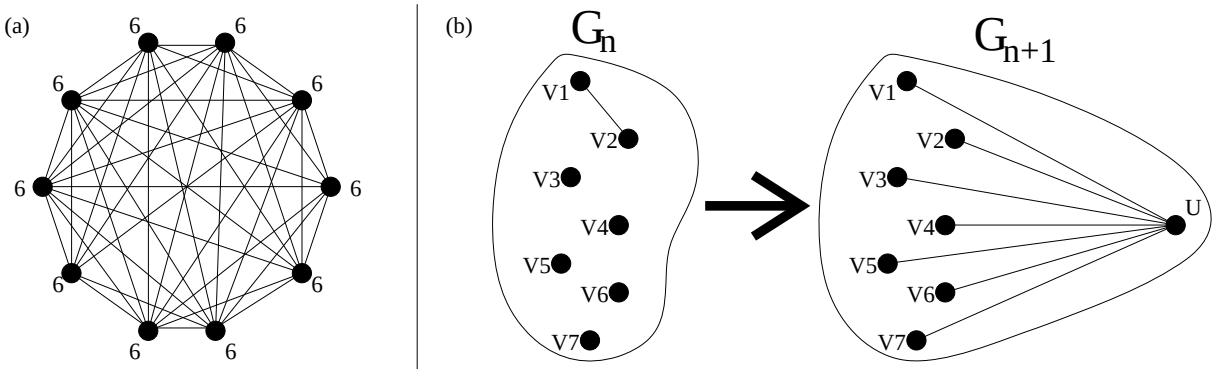

FIG. 5.1 – Illustration de la construction de graphes minimaux bien-(6)denses

Dans les sections suivantes, nous présentons l'algorithme `Dense`, proposé par Hoffmann *et al.* dans [Hoffmann *et al.*, 1997], qui permet d'identifier des sous-graphes $k$denses pour une valeur quelconque (mais positive) de k. Cet algorithme est à la base de l'algorithme `Minimal-Dense` qui est utilisé pour l'étape de fusion de la méthode HLS.

5.3.1 Algorithme `Dense`

L'algorithme `Dense` a été présenté initialement dans l'article [Hoffmann *et al.*, 1997]. Il a depuis été implanté dans FRONTIER [Oung *et al.*, 2001], un outil de CAO. Nous avons eu quelques échanges avec Meera Sitharam [Sitharam, 2000] qui nous ont permis de mieux cerner les spécificités de l'algorithme ; nous proposons de décrire ici une version détaillée de cet algorithme inspirée de celle qui est implantée dans FRONTIER et non pas la version originale publiée dans [Hoffmann *et al.*, 1997].

L'algorithme est basé sur le calcul d'un flot maximum dans le réseau objets-contraintes G_S qui modélise le GCSP S.

Réseau objets-contraintes

Le réseau objets-contraintes d'un GCSP a été défini dans [Hoffmann *et al.*, 1997]. Il est basé sur la version bipartie du graphe objets-contraintes et représente donc la structure du GCSP. Comme le graphe objets-contraintes, il est décoré à l'aide des degrés de liberté des objets et des contraintes géométriques. La source est liée aux contraintes géométriques, alors que les objets géométriques sont liés au puits. L'orientation des arcs permet de distribuer le flot de la source à travers les contraintes puis les objets vers le puits. Ainsi, un flot permet de représenter une distribution des degrés de liberté des contraintes sur les degrés de liberté des objets.

Définition 41 Réseau objets-contraintes
Soit $S = (O, C)$ un GCSP. Le réseau objets-contraintes $G_S = (\{s, t\} \cup V, E, w)$ est construit à partir du GCSP S comme suit :

- **s** *est la source ;*
- **t** *est le puits ;*
- **V** $= O \cup C$ *: l'ensemble des nœuds du réseau contient un nœud par objet et par contrainte géométrique de S ;*
- **E** *est l'ensemble des arcs de G_S ; pour chaque objet géométrique o et chaque contrainte géométrique c, E contient un arc $s \rightarrow c$, un arc $o \rightarrow t$ et, si o est contraint par c, un arc $c \rightarrow o$.*
- **w** *est la fonction de pondération des arcs du réseau qui fournit leurs capacités ; pour chaque objet géométrique o et chaque contrainte géométrique c, $w(s \rightarrow c) = DDL(c)$, $w(o \rightarrow t) = DDL(o)$ et, si $c \rightarrow o$ existe, $w(c \rightarrow o) = +\infty$.*

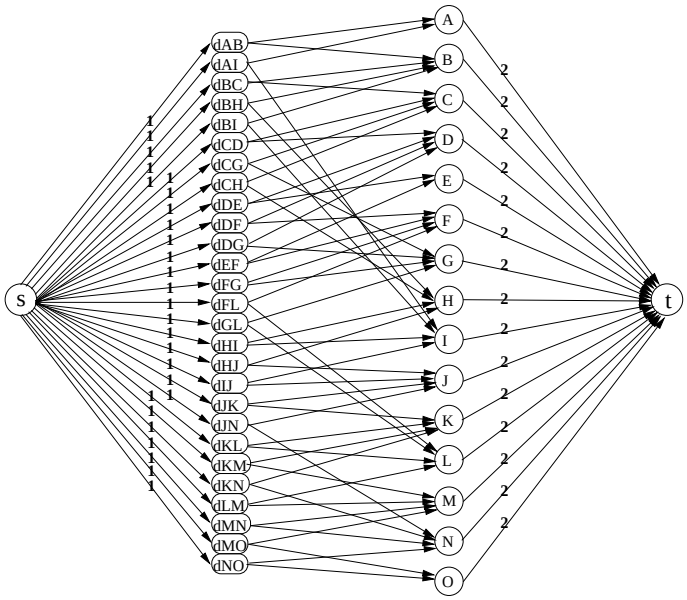

FIG. 5.2 – Réseau objets-contraintes correspondant au GCSP *Ponts*

Le réseau objets-contraintes correspondant au GCSP *Ponts* est présenté à la figure 5.2. Les capacités des arcs provenant de la source et de ceux entrant dans le puits sont représentées : elles correspondent respectivement aux degrés de libertés des contraintes (contraintes de distance à 1 DDL) et des objets géométriques (points en 2D à 2 DDL). Les capacités des autres arcs sont infinies et ne sont pas représentées sur la figure.

Flot dans le réseau objets-contraintes

La distribution d'un flot dans ce réseau représente une répartition des degrés de liberté des contraintes sur les degrés de liberté des objets géométriques. En effet, le flot circule

des contraintes vers les objets, et chaque unité du flot distribuée représente 1 DDL d'une contrainte qu'il est possible de distribuer vers les objets qui lui sont liés. Ainsi, le calcul du flot maximum dans ce réseau représente une distribution maximum des degrés de liberté des contraintes sur les degrés de liberté des objets géométriques.

Sachant que chaque degré de liberté d'une contrainte représente une équation et que chaque degré de liberté d'un objet géométrique représente une variable, on peut faire le rapprochement entre le calcul d'un flot maximum dans le réseau objets-contraintes et le calcul d'un couplage maximum sur le graphe biparti équations-variables (cf. section 4.1.1, page 87). Dans le cas du couplage maximum, on détermine les composantes structurellement bien-, sur- et sous- contraintes. Dans le cas du flot, la distribution maximum permet de déterminer si un sous-GCSP sur-contraint existe. En effet, si un flot maximum ne sature pas toutes les arêtes provenant de la source, c'est qu'il existe un sous-GCSP sur-contraint puisque tous les degrés de liberté des contraintes ne peuvent être absorbés par ceux des objets géométriques.

Pour identifier le GCSP S' sur-contraint qui empêche de saturer un arc a provenant de la source, il faut analyser la distribution du flot maximum. Hoffmann *et al.* énoncent et démontrent la proposition suivante [Hoffmann *et al.*, 1998] :

Proposition 11 Identification d'un sous-GCSP $k$dense par flot maximum
Soit $S = (O, C)$ un GCSP, $G_S = (\{s, t\} \cup V, E, w)$ son réseau objets-contraintes et **a** *un arc issu de la source de G_S qui n'est pas saturé après le calcul d'un flot maximum f dans G_S. Alors S', le sous-GCSP de S induit par $O' = \{o \in V | o$ est traversé durant la recherche d'un chemin augmentant partant de* **a**$\}$*, est sur-s_contraint.*

Autrement dit, le sous-GCSP sur-s_contraint, c-a-d. (-1)dense, est induit par l'ensemble des objets atteignables par un parcours du le graphe résiduel, résultant du calcul d'un flot maximum dans G_S, partant de a.

Identification de sous-GCSP $k$denses

Hoffmann *et al.* généralisent ce principe à la recherche de sous-GCSP $k$denses pour un k quelconque. Afin d'identifier des sous-GCSP $k$denses, la capacité de chaque arc issu de la source est tour à tour augmentée de $k + 1$. Il en résulte que la présence d'un arc non saturé issu de la source après calcul du flot maximum dans ce réseau modifié indique que les objets géométriques ne peuvent pas absorber $k + 1$ degrés de liberté supplémentaires, c-a-d. qu'il existe un sous-GCSP S' qui est $k$dense. Ce sous-GCSP est identifié comme énoncé à la proposition 11. Notons que ce sous-GCSP $k$dense n'est pas forcément *minimal* $k$dense.

L'algorithme `Dense` qui identifie un sous-GCSP $k$dense pour un k quelconque est décrit à l'algorithme 8. Le calcul du flot maximum par la fonction `FlotMaximum` est spécialisé aux contraintes binaires dans l'article [Hoffmann *et al.*, 1997], et réalisé de façon spécifique dans FRONTIER [Oung *et al.*, 2001]. Cependant, il peut très bien être effectué par un algorithme de flot classique [Ford and Fulkerson, 1962]. La complexité de cet algorithme

est alors $O(m * n^2 * (m+n))$ si n est le nombre d'objets géométriques et m le nombre de contraintes géométriques.

Algorithme 8 Dense ($S = (O, C)$: un GCSP ; k : un entier) **retourne** S' : sous-GCSP $k$dense

$G_S \leftarrow$ `RéseauObjetsContraintes(`S`)` *{Rappelons que* $G_S = (\{s,t\} \cup O \cup C, E, w)$*}*
Pour chaque $c \in C$ **Faire**
  $w(s \rightarrow c) \leftarrow w(s \rightarrow c) + k + 1$ *{remplacer par* $w(s \rightarrow c) \leftarrow w(s \rightarrow c) + k$ *pour obtenir l'algorithme* `Sur-Dense`*}*
  $flots \leftarrow$ `FlotMaximum(`G_S`)`
  Si `NonSaturé ?(`$s \rightarrow c, flots$`)` **Alors**
    $O' \leftarrow$ `SommetsCheminAugmentant(`$G_S, s \rightarrow c, flots$`)`
    $S' \leftarrow$ `SousGCSPInduit(`O', S`)`
    Return S' *{évite l'examen de toutes les contraintes si un sous-GCSP* k*dense a été trouvé.}*
  Fin Si
Fin Pour
Return $(\emptyset, \emptyset)$ *{aucun sous-GCSP* k*dense n'a été identifié}*

Améliorations de l'algorithme `Dense`

Cette complexité peut être abaissée à $O(n^2 * (m+n))$ si l'on procède au calcul du flot de façon incrémentale. En effet, la seule différence entre les 2 réseaux utilisés à 2 itérations différentes de l'algorithme est la position de la surcharge : elle est appliquée sur un arc différent issu de la source. Ainsi, si l'on calcule une seule fois la distribution maximum dans le réseau non surchargé (en $O(n^2 * (m+n))$), calculer chacune des m distributions surchargées revient à essayer de distribuer uniquement la surcharge, c-a-d. à calculer au plus $k+1$ chemins augmentants (en $O((k+1) * (m+n)) = O(m+n)$, car k vaut $\frac{d(d+1)}{2}$ en pratique et peut donc être considéré constant par rapport à la taille du GCSP).

Hoffmann *et al.* proposent une version de l'algorithme qui consiste à construire incrémentalement le réseau objets-contraintes en y introduisant les objets géométriques un à un, avec toutes les contraintes géométriques induites par l'ensemble d'objets actuellement présent dans le réseau [Hoffmann *et al.*, 1997]. Les arcs considérés pour la surcharge à chaque étape de construction du réseau sont ceux correspondants aux nouvelles contraintes introduites à cette étape. Cette modification n'améliore pas la complexité en pire cas de l'algorithme mais peut améliorer ces performances en pratique ; elle permet de plus à Hoffmann *et al.* de dériver des propriétés supplémentaires du sous-GCSP identifié (cf. propositions 12 et 13, page 133).

Exemple d'exécution de l'algorithme

La figure 5.3 illustre le déroulement de l'algorithme `Dense` sur le GCSP en 3D composé d'un point A, d'une droite B et d'un plan C liés par les contraintes Distance(A, B),

Angle(B, C) et Incidence(A, C). Le réseau objets-contraintes correspondant à ce GCSP est dépeint à la figure 5.3-a ; les nombres en gras sur les arcs représentent les capacités, c-a-d. les degrés de liberté des objets et des contraintes. Les figures 5.3-b, 5.3-c et 5.3-d représentent les 3 itérations produites par un appel à `Dense`$(S, 5)$. Les nombres en italique sur les arcs représentent les flots alloués après calcul d'un flot maximum. A chaque itération, la surcharge est appliquée sur l'arc représenté en gras sur la figure.

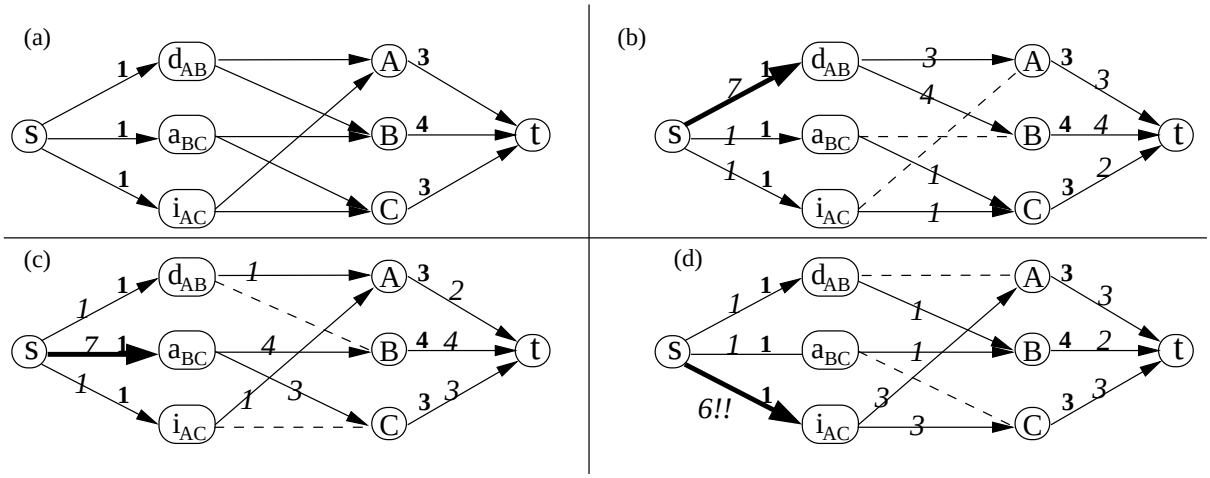

FIG. 5.3 – Illustration du déroulement de l'algorithme `Dense`

A la première itération, la surcharge $k = 5 + 1$ est appliquée sur l'arc liant la source à la contrainte Distance(A, B). Une distribution maximum est calculée et permet de saturer tous les arcs issus de la source (cf. figure 5.3-a). Aucun sous-GCSP (5)dense n'est donc identifié.

La surcharge est ensuite appliquée à l'arc liant la source à la contrainte Angle(B, C) et un calcul de flot maximum permet de saturer tous les arcs issus de la source. Là encore, aucun sous-GCSP (5)dense n'est identifié.

Enfin, la surcharge est appliquée sur la contrainte Incidence(A, C) et le flot maximum ne permet pas de saturer l'arc correspondant issu de la source. Lors de la recherche d'un chemin augmentant dans ce réseau, les nœuds traversés sont A et C : le sous-GCSP identifié est donc AC qui est (5)dense.

Algorithme `Minimal-Dense`

L'algorithme `Minimal-Dense` proposé par Hoffmann, Lomonosov et Sitharam, utilise l'algorithme `Dense` pour construire une série de sous-GCSP $k$denses imbriqués dont le dernier est un sous-GCSP minimal $k$dense. Cet algorithme est présenté à l'algorithme 9.

Le principe classique de minimisation consiste à essayer de retirer les objets un par un dans le sous-GCSP $k$dense identifié par un appel à l'algorithme `Dense`, tout en maintenant la densité du sous-GCSP restant. Plus précisément, dans le sous-GCSP S' $k$dense identifié

par un appel à l'algorithme `Dense`, on retire un objet géométrique o ; un appel à l'algorithme `Dense` sur le sous-GCSP $S' - o$ permet d'identifier un sous-GCSP $S'' \subset S'$ $k$dense s'il en existe un ; s'il n'en existe pas, c'est que l'objet o appartient au sous-GCSP minimal $k$dense. Chaque objet géométrique dans le sous-GCSP S' identifié par l'appel initial à `Dense` est donc examiné au plus une fois, et la complexité en pire cas de l'algorithme `Minimal-Dense` est $O(n * C_{Dense})$ où C_{Dense} est la complexité de l'algorithme `Dense` et n est le nombre d'objets géométriques du sous-GCSP S'.

Algorithme 9 Minimal-Dense ($S = (O, C)$: un GCSP ; d : dimension) **retourne** $S' = (O', C')$: sous-GCSP minimal bien- ou sur-es_rigide

$S' \leftarrow$ `Dense(`$S, \frac{d(d+1)}{2}$`)`
$Necessaires \leftarrow \emptyset$
Pour chaque $o \in$ (`Objets(`S'`)` $\setminus Necessaires$ **Faire**
  $S'' \leftarrow$ `SousGCSPInduit(`$O' \setminus \{o\}, S'$`)` *{S'' est le sous-GCSP de S' ne contenant pas l'objet o}*
  $S'' \leftarrow$ `Dense(`$S'', \frac{d(d+1)}{2}$`)`
  Si $S'' \neq (\emptyset, \emptyset)$ **Alors**
    $S' \leftarrow S''$ *{o n'est pas nécessaire ; le sous-GCSP S' devient le nouveau sous-GCSP S'' identifié}*
  Sinon
    $Necessaires \leftarrow Necessaires \cup \{o\}$ *{o est nécessaire ; il est conservé afin de ne pas être testé à nouveau}*
  Fin Si
Fin Pour
Return S'

La version incrémentale de l'algorithme `Dense` a permis à Hoffmann *et al.* d'énoncer et de démontrer dans [Hoffmann *et al.*, 1998] les 2 propositions suivantes :

Proposition 12
Le dernier objet géométrique o introduit dans le réseau appartient à tout sous-GCSP $S'' \subset S'$ $k$dense du sous-GCSP S' identifié par l'algorithme `Dense`.

Proposition 13
Si S'' est un sous-GCSP $k$dense du sous-GCSP S' identifié par l'algorithme `Dense`, *alors il existe un entier $k' > k$ tel que S'' est k'dense et S' n'est pas k'dense.*

Ces propositions permettent de réduire le nombre d'étapes de minimisation. En effet, la proposition 12 donne un point de départ pour l'algorithme de minimisation puisque le dernier objet appartient nécessairement au sous-GCSP minimal $k$dense résultant de la minimisation. La proposition 13 assure la convergence rapide de l'opération de minimisation car la séquence de sous-GCSP imbriqués produite augmente strictement en densité et que la densité du dernier est au plus k (puisqu'il est minimal $k$dense). Ainsi le nombre d'étapes

de minimisation est borné par la différence de densité entre le premier sous-GCSP $k$dense identifié par un appel à l'algorithme `Dense` et la valeur limite k. De plus, si le sous-GCSP S' retourné par le premier appel à `Dense` dispose de k degrés de liberté, alors il sera déjà minimal et aucune étape de minimisation ne sera nécessaire.

Manque de généralité de la proposition 13

Nous apportons une petite précision quant à ces résultats : la proposition 13 n'est vraie que lorsque toutes les contraintes sont binaires. En effet, si le GCSP contient une contrainte ternaire par exemple, il est possible qu'un sous-GCSP S' identifié par un appel à `Dense` contienne un sous-GCSP $S'' \subset S'$ de densité inférieure. Nous illustrons ce fait sur le GCSP présenté à gauche de la figure 5.4. Ce GCSP en 3D est constitué d'une droite A et de 2 points B et C liés par les contraintes Distance(A, C), Distance(B, C) et SymétrieAxiale(A, B, C). Le réseau correspondant à ce GCSP est dépeint à droite de cette même figure. L'appel à `Dense(`$S, 5$`)` va appliquer une surcharge de $5 + 1$ sur la contrainte SymétrieAxiale(A, B, C) et la distribution maximum ne permettra pas de saturer l'arc correspondant. Le sous-GCSP identifié est alors S tout entier, qui dispose de 6 DDL. Cependant, l'étape de minimisation permettra d'en extraire le sous-GCSP S' réduit aux objets B et C qui ne dispose que de 5 DDL.

Dans le cas général, où les contraintes peuvent être d'arité quelconque, l'étape de minimisation n'est donc jamais superflue.

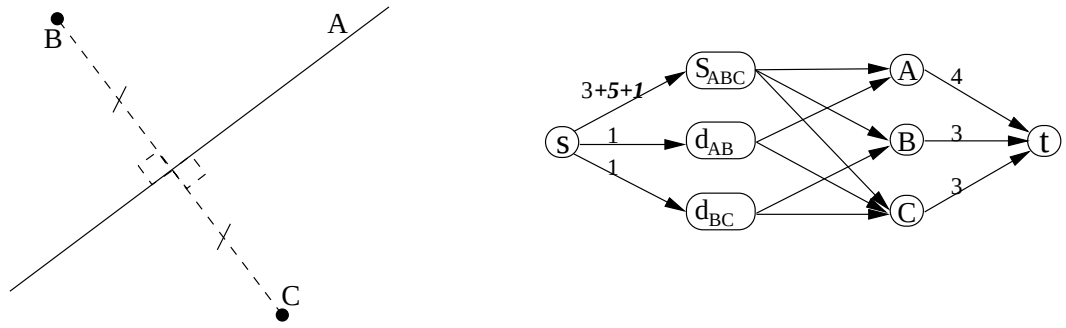

FIG. 5.4 – Un GCSP violant la proposition 13

5.3.2 Utilisation de `Dense` pour l'opération de fusion

La proposition 8, page 125, indique qu'un sous-GCSP minimal $\frac{d(d+1)}{2}$dense est un sous-GCSP minimal bien- ou sur-s_rigide en dimension d. On peut donc utiliser l'algorithme `Minimal-Dense` pour identifier des sous-GCSP bien- ou sur-s_rigides. Il suffit d'utiliser comme valeur pour le paramètre k la constante $\frac{d(d+1)}{2}$ qui ne dépend que de la dimension du GCSP considéré.

L'opérateur de fusion n'est sensé retourner que des sous-systèmes bien-rigides. Dans le cas où le sous-GCSP S' identifié par l'opération est sur-s_rigide, on peut envisager 2 attitudes :

- considérer que S' est sur-rigide, c-a-d que les contraintes en trop introduisent une inconsistance qui doit être éliminée pour que le GCSP puisse être résolu. Alors, S' est retourné à l'utilisateur comme explication minimale de la sur-s_rigidité de son GCSP ;
- considérer que S' est rigide et que les contraintes en trop sont des R-redondances. Alors, on traite S' comme s'il disposait en réalité de $\frac{d(d+1)}{2}$ DDL pour la suite de la planification. C'est cette solution qui est retenue dans le logiciel FRONTIER. S'il s'avère lors de la phase d'assemblage que la résolution de S' est impossible car il était en réalité inconsistant, Sitharam *et al.* proposent alors de retourner S' à l'utilisateur pour qu'il retire les contraintes en trop.

Limites à l'utilisation de `Minimal-Dense` pour la fusion

On distingue 2 catégories de limites à l'utilisation de l'algorithme Minimal-Dense pour réaliser l'opération de fusion : celles liées à l'utilisation de la s_rigidité au lieu de la rigidité, et celles liées à l'algorithme lui-même.

Limites liées à la s_rigidité

L'algorithme `Minimal-Dense` est basé sur la s_rigidité, caractérisation structurelle de la rigidité dont nous avons exposé les limites à la section 2.3.3, page 60. Il est donc possible que le GCSP retourné par l'étape de fusion soit :

- bien-s_rigide mais sous-rigide ; ceci est dû à la présence de R-redondances ;
- bien-s_rigide mais sur-rigide, à cause d'une inconsistance ;
- sur-s_rigide mais bien- ou sous-rigide, de part la présence de R-redondances.

Pour pallier certains problèmes majeurs de la s_rigidité, comme le fait qu'un segment en 3D est sur-s_rigide, Hoffmann *et al.* proposent d'utiliser la s_rigidité *bis* (cf. définition 34, page 62). Nous avons expliqué à la section 2.3.3, page 62, que cette définition ne règle pas le problème et affaiblit la s_rigidité puisqu'elle ne permet plus de détecter certaines sur-s_rigidités.

Hoffmann *et al.* proposent alors d'utiliser une table annexe, la *Look-up Table*. Cette table contient certains modèles de sous-GCSP rigides triviaux, comme le segment en 3D. Elle contient aussi des règles de constructions pour produire des sous-GCSP rigides non triviaux à partir de sous-GCSP rigides triviaux. Par exemple, la table peut fournir la règle produisant un tétraèdre à partir d'un segment en 3D.

Cette table est utilisée avec la s_rigidité, et non pas la s_rigidité bis, de la façon suivante : lorsque l'algorithme `Minimal-Dense` retourne un sous-GCSP trivial, on consulte la *Look-up Table* pour déterminer s'il est rigide. Si tel est le cas, on essaye d'utiliser une règle de la table afin de produire un sous-GCSP plus grand qui soit rigide mais pas trivial. Par exemple, si un segment est retourné par l'algorithme `Minimal-Dense`, après avoir constaté

que ce segment est bien-rigide, la règle qui produit un tétraèdre sera appliquée : s'il existe un tétraèdre contenant le segment identifié dans le GCSP, alors c'est ce tétraèdre qui sera retourné par l'opération de fusion à la place du segment.

La *Look-up Table* est par nature incomplète : elle est basée sur un ensemble fini de règles, alors qu'il est possible que le plus petit sous-GCSP bien-rigide basé sur un sous-GCSP rigide trivial soit de taille arbitraire (cf. proposition 10, page 127). Qui plus est, elle ne règle pas le problème du GCSP bien-rigide en 3D constitué de 4 droites parallèles (cf. section 2.3.3, page 62). Elle permet cependant d'effectuer un traitement géométrique et non plus structurel de certains cas particuliers.

Enfin, Sitharam *et al.* proposent dans le logiciel FRONTIER de gérer les erreurs de décomposition dues à la s_rigidité directement durant l'assemblage : si un sous-GCSP s_rigide s'avère sur- ou sous-rigide à la résolution (pas de solutions, ou une variété continue de solution), le sous-GCSP incriminé est renvoyé à l'utilisateur qui doit retirer les inconsistances (si le sous-GCSP est sur-rigide), ou fixer les paramètres de la variété afin de choisir une solution ou encore ajouter des contraintes (si le sous-GCSP est sous-rigide). Rappelons que toute modification des contraintes du GCSP entraîne normalement une re-planification complète du GCSP, puisqu'elle modifie la s_rigidité dans des sous-GCSP. Sitharam *et al.* proposent d'user d'heuristiques afin de prendre en compte les contraintes introduites ou retirées en cours de résolution.

Limites liées à l'algorithme

En plus des problèmes causés par l'utilisation de la s_rigidité (ou de la s_rigidité bis), l'utilisation de `Minimal-Dense` pour réaliser l'opération de fusion pose un autre problème : la façon de distribuer la surcharge dans le réseau objets-contraintes n'est pas conforme à la nature géométrique du problème.

En effet, du point de vue géométrique, la surcharge valant $\frac{d(d+1)}{2} + 1$ représente une tentative de poser un repère ($\frac{d(d+1)}{2}$), et de fixer un DDL supplémentaire ($+1$), sur un ensemble d'objets du GCSP afin de déterminer s'il contient un sous-GCSP rigide. Appliquer cette surcharge sur une contrainte c revient alors à placer un repère local sur l'ensemble O_c d'objets liés à la contrainte c. Or, rien n'assure que l'ensemble d'objets géométriques O_c permette de définir un repère local. Par exemple, Lamure et Michelucci ont remarqué [Lamure and Michelucci, 1998] qu'il faut considérer 3 points simultanément afin de pouvoir placer un repère local en 3D, alors que 2 points sont insuffisants. Or, si une contrainte de distance est liée à 2 points en 3D, la surcharge distribuée via cette contrainte de distance représentera une tentative, dans l'algorithme `Minimal-Dense`, de poser un repère local sur 2 points seulement.

L'algorithme `Minimal-Dense` peut donc distribuer la surcharge de façon incorrecte vis-à-vis de la géométrie, ce qui constitue un autre inconvénient de l'utilisation de cet algorithme pour réaliser l'opération de fusion.

Ainsi, rien n'assure que le sous-GCSP retourné par l'algorithme `Minimal-Dense` permette la définition d'un repère local ; s'il ne le permet pas, ce sous-GCSP ne pourra pas être assemblé de façon rigide avec le reste du GCSP. On retrouve le problème du repère

local, que nous avons introduit à la section 4.5, page 116. Nous y avons expliqué qu'aucune solution structurelle exacte ne peut être donnée à ce problème. Nous donnerons notre solution, basée sur le concept de *degré de rigidité*, à la section 6.4, page 191. Nous proposerons alors une nouvelle opération de fusion (cf. section 7.3, page 200).

5.4 Opérateur d'extensions

L'opération d'extensions permet d'accroître la taille de l'agrégat produit par l'opération de fusion. Pour ce faire, l'agrégat courant K est étendu vers des *objets voisins* ; le voisinnage se définit de la façon suivante :

Définition 42 voisinnage d'un agrégat
Soit $S = (O, C)$ un GCSP et $K = (O_K, C_K)$ un agrégat dans ce GCSP. Le voisinnage de K est le sous-ensemble O' des objets géométriques de S qui sont liés par des contraintes uniquement à des objets de K : $O' = \{o \in O | \exists c \in C$ qui porte sur les objets$O_c = \{o\} \cup O'' \wedge O'' \subset O_K\}$.

Les contraintes liant un objet voisin v à un agrégat K sont toutes utilisées pour fixer v vis-à-vis de K. On ne considère donc que les contraintes portant sur un objet extérieur à K à la fois pour l'extension, l'inclusion de plusieurs objets à la fois pouvant introduire une combinatoire nocive à la complexité de l'opération d'extension. Du point de vue de l'assemblage, l'extension consistera à placer l'objet voisin v dans le repère local de l'agrégat courant K.

L'opération d'extension n'a jamais été décrite précisément dans les articles de Hoffmann *et al.*, cependant nous avons pu nous faire une idée de la façon dont elle est réalisée lors de nos échanges avec Meera Sitharam [Sitharam, 2000]. Nous proposons à l'algorithme 10 (page 138) un algorithme décrivant l'opération d'extension.

Cet algorithme maintient l'ensemble $Voisins$ qui contient tous les objets dans le voisinnage de l'agrégat courant. Cet agrégat est nommé K' au cours de l'extension, pour être distingué de l'agrégat K pris en entrée et qui correspond à celui retourné par l'étape de fusion. L'opération d'extensions est terminée lorsqu'aucune extension n'est plus possible, c-a-d. lorsque l'ensemble $Voisins$ devient vide.

A chaque itération, l'algorithme considère un agrégat v dans le voisinnage de K'. Si l'extension vers ce voisin est *possible*, elle est réalisée en incluant v et toutes les arêtes qui le lient uniquement à des objets de K' dans l'agrégat K' courant. Le point crucial de l'opération réside donc dans le test qui détermine si une extension est possible ou non.

5.4.1 Extension possible

Une extension de l'agrégat K' sur un objet v voisin de K' n'est possible que si le sous-GCSP $K' \cup v$ résultant de cette extension est bien-s_rigide. Ce test est réalisé par la fonction `Extension-Possible` dans l'algorithme 10. Nous avons expliqué que l'extension

consiste à ajouter dans K' l'objet v et toutes les contraintes le liant uniquement à des objets de K'. Cet ensemble de contraintes est appelé *Liens* dans l'algorithme 10.

Algorithme 10 Extensions ($K = (O_K, C_K)$: un agrégat de S ; $S = (O, C)$: le graphe objets-contraintes pondéré d'un GCSP) **retourne** $K' = (O_{K'}, C_{K'})$: un agrégat contenant K

$O_{K'} \leftarrow O_K$
$C_{K'} \leftarrow C_K$
$Voisins \leftarrow \{o \in O | \exists c \in C \wedge c$ porte sur les objets$O_c = \{o\} \cup O' \wedge O' \subset O_K\}$
Tant que $Voisins \neq \emptyset$ **Faire**
  $v \leftarrow$ `UnObjet(`$Voisins$`)`
  $Voisins \leftarrow Voisins \setminus \{v\}$
  $Liens \leftarrow \{c \in C | c$ porte sur les objets$O_c = \{v\} \cup O' \wedge O' \subset O_K\}$
  Si `Extension-Possible(`K', v`)` **Alors**
    $O_{K'} \leftarrow O_{K'} \cup \{v\}$
    $C_{K'} \leftarrow C_{K'} \cup Liens$
    $Voisins \leftarrow \{o \in O | \exists c \in C \wedge c$ porte sur les objets$O_c = \{o, v\} \cup O' \wedge O' \subset O_K\}$ *{Le voisinnage n'est mis à jour que lorsque K' est étendu}*
  Fin Si
Fin Tant que
Return $K' = (U', F')$

Vérifier si un GCSP $K' \cup v$ est bien-s_rigide revient, par définition, à vérifier que $DDL(K' \cup v) = \frac{d(d+1)}{2}$ et pour tout $S' \subset K' \cup v$, $DDL(S') \geq \frac{d(d+1)}{2}$. Hoffmann *et al.* proposent un test composé des deux vérifications suivantes :

1. $DDL(v) = DDL(Liens)$; c-a-d. que le nombre de DDL fixés par les contraintes liant K' à v est égal au nombre de DDL de v.
2. $|\{o \in O_{K'} | \exists c \in Liens \wedge c$ porte sur $o\}| > d$ en dimension d ; c-a-d. que l'ensemble des objets de K' auxquels est relié v par des contraintes contient au moins $d + 1$ objets.

La première condition assure effectivement que le compte global des DDL de $K' \cup v$ est vérifié, car $DDL(K') = \frac{d(d+1)}{2}$ et $K' \cup v = (O'_K \cup v, C'_K \cup Liens)$. La seconde condition n'a pas été expliquée par Hoffmann *et al.*. Nous avons constaté qu'elle assure le compte local des DDL, mais uniquement en 2D et 3D. Nous démontrons cette propriété en calculant le nombre de DDL de tout sous-GCSP $S' \subset K' \cup v$. Pour ce faire, on distingue 3 cas :

- $S' \subset K'$, c-a-d. que S' ne contient pas l'objet v. En ce cas, S' est un sous-GCSP d'un agrégat (K') et vérifie donc bien $DDL(S') \geq \frac{d(d+1)}{2}$.
- $S' = K'' \cup v$ et K'' comporte au moins deux objets géométriques. Notons $Liens' \subset Liens$ les contraintes liant v à K''. Alors $DDL(S') = DDL(K'') + DDL(v) - DDL(Liens')$. Or $DDL(v) = DDL(Liens)$ par hypothèse (condition 1 de Hoffmann *et al.*) et donc $DDL(v) \geq DDL(Liens')$. De plus, $DDL(K'') \geq \frac{d(d+1)}{2}$ puisque K'' est un sous-GCSP de l'agrégat K' non réduit à un objet. Il en résulte que $DDL(S') \geq \frac{d(d+1)}{2}$.

– $S' = K'' \cup v$ et $K'' = (\{u\}, \emptyset)$, c-a-d que K'' n'est pas un sous-GCSP de K' car il ne contient qu'un seul objet, u. Notons alors $Liens' \subset Liens$ les contraintes liant u à v. Alors $DDL(S') = DDL(u)+DDL(v)-DDL(Liens')$. Or $DDL(v) = DDL(Liens)$ et $Liens$ contient des contraintes portant au moins sur d objets en dehors de u, chacune retirant au moins 1 DDL (condition 1 et 2 de Hoffmann *et al.*) ; il en résulte que $DDL(v) - d \geq DDL(Liens')$. Donc $DDL(S') \geq DDL(u) + DDL(V) - DDL(v) + d$, c-a-d. $DDL(S') \geq DDL(u) + d$. Or, en dimension $d = 2$ ou $d = 3$, tout objet rigide admet au moins d DDL, et donc $DDL(S') \geq 2 * d$. Toujours en dimension $d = 2$ ou $d = 3$, $2 * d \geq \frac{d(d+1)}{2}$, et donc $DDL(S') \geq \frac{d(d+1)}{2}$.

□

On constate donc que les deux conditions proposées par Hoffmann *et al.* permettent bien d'assurer le compte global et le compte local des DDL et assure ainsi qu'une extension est possible.

Cependant, la condition 2 de ce test peut exclure des extensions qui sont en réalité possibles. En particulier, elle invalide toute extension vers des objets géométriques possédant seulement d DDL en dimension d, par exemple les points en 2D et 3D, et les plans en 3D, car ils ne peuvent pas être liés de façon rigide à plus de d objets de l'agrégat K' : si un objet v possède seulement d DDL est lié par au moins $d + 1$ contraintes retirant chacune au moins 1 DDL à un agrégat, la condition 1, c-a-d. le compte global pour la s_rigidité, sera violée.

Amélioration du test d'extension possible

Afin d'assurer la s_rigidité de l'agrégat résultant d'une extension, nous proposons d'utiliser l'algorithme `Dense`. En effet, nous avons vu que, lorsqu'il est utilisé avec une valeur de surcharge valant $\frac{d(d+1)}{2}$ au lieu de $\frac{d(d+1)}{2} + 1$, cet algorithme permet d'identifier un sous-GCSP sur-s_rigide en temps polynomial s'il en existe un. La fonction `Extension-Possible` consiste alors simplement à retourner le résultat du test suivant :

$(DDL(v) = DDL(Liens)) \wedge$ (`Dense`$((O_{K'} \cup \{v\}, C_{K'} \cup Liens), \frac{d(d+1)}{2} - 1) = \emptyset$)

En effet, l'absence de sur-s_rigidité assure que le compte local des DDL est vérifié. Cependant, ce test est plus coûteux que celui proposé par Hoffmann *et al.*, puisqu'il implique le calcul d'un flot maximum avec distribution de surcharge pour chaque contrainte liée à v.

5.4.2 Discussion

Problèmes de l'extension

L'opération d'extension étant basée elle aussi sur la s_rigidité, elle peut opérer des extensions incorrectes du point de vue géométrique dans les cas où la s_rigidité ne correspond pas à la rigidité : inconsistances, R-redondances et singularités. Par exemple, la méthode HLS permet, quelle que soit l'opération de condensation considérée, de planifier le GCSP présenté à la figure 2.10, page 44, qui est, rappelons-le, un GCSP bien-s_rigide

mais sur-rigide dans le cas générique (il contient une contrainte R-redondante qui est inconsistante avec le reste du GCSP dans le cas générique) ou sous-rigide dans le cas singulier (lorsque les contraintes permettent la coïncidences des 2 points formant l'axe de rotation de ce système).

Liens avec d'autres opérations

Le principe d'extension peut être rapproché de celui de 2-extension défini par Henneberg pour les systèmes à barres (cf. section 2.1.7, page 41) : la 2-extension intègre de façon rigide un point en 2D dans un système à barres rigide ; une extension intègre de façon rigide un agrégat en dimension d dans un autre agrégat rigide.

L'opération d'extensions est une heuristique qui permet de minimiser le nombre d'étapes de fusion : tout agrégat généré par extension pourrait être généré par fusion. Cependant, l'opération d'extension est en pratique moins coûteuse que la fusion : même si elle est basée sur l'algorithme `Dense` (si l'on utilise le test que nous proposons pour décider si une extension est possible), le réseau utilisé lors de la fusion est généralement plus grand que celui utilisé lors du test pour déterminer si une extension est possible.

La différence fondamentale entre fusion et extension réside dans la taille des sous-systèmes correspondants à résoudre lors de l'assemblage : une opération de fusion est capable d'identifier des sous-GCSP minimaux s_rigides de taille quelconque qui correspondent à des sous-systèmes d'équations de taille proportionnelle pour la résolution ; à l'inverse, une extension n'est capable de produire que des agrégats correspondant à l'assemblage d'un agrégat et d'un objet géométrique, mais le système d'équations correspondant est alors de taille minimale[4].

5.4.3 Exemple d'extensions

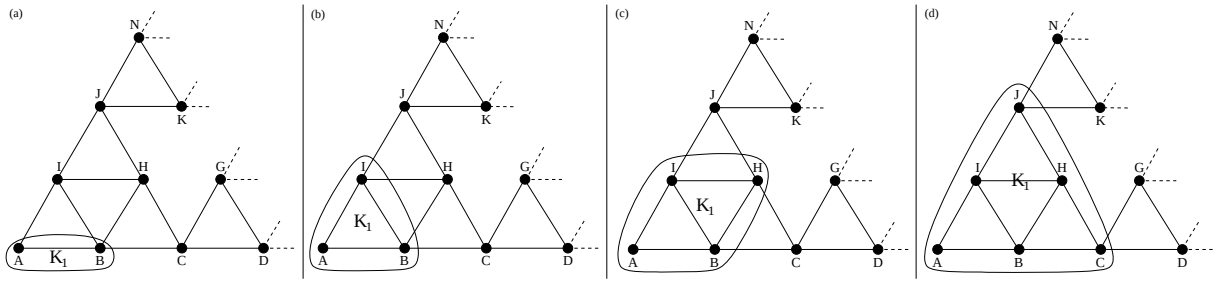

FIG. 5.5 – Illustration du déroulement de l'algorithme `Extensions`

Nous illustrons l'opération d'extensions sur le GCSP *Ponts*, à l'étape présentée en figure 4.10-a, page 105. L'agrégat produit par l'opération de fusion à cette étape est $K_1 =$

[4]Les objets géométriques étant l'entité atomique de la méthode, le plus petit sous-système correspond à la résolution d'un seul objet.

AB (cf. figure 5.5-a). L'opération d'extensions commence par produire la liste des agrégats voisins qui sont candidats à l'extension : $Voisins \leftarrow \{C, H, I\}$. L'extension de K_1 sur C ou sur H est impossible car le compte global n'est pas assuré. En revanche, l'extension de K_1 sur I est possible (cf. figure 5.5-b). La liste des voisins est donc remise à jour : $Voisins \leftarrow \{C, H, J\}$. L'extension de K_1 sur C ou J est impossible, mais elle est possible sur H (cf. figure 5.5-c). La liste des voisins est à nouveau mise à jour : $Voisins \leftarrow \{C, J\}$. Cette fois, tous les voisins peuvent être intégrés par extension (cf. figure 5.5-d). Les nouveaux voisins sont $Voisins = \{D, G, K, N\}$ et aucun ne peut être intégré à K_1 par extension. L'opération se termine donc avec l'agrégat $K_1 = ABCHIJ$.

5.5 Opérateurs de condensation

L'opération de condensation a la charge de mettre à jour les structures de données utilisées par la phase de planification : le GCSP et le plan d'assemblage. Hoffmann *et al.* proposent trois opérations de condensations différentes : *CA*, *FA* et *MFA*. Ces trois opérations ont été introduites comme des mises à jour du graphe objets-contraintes associé au GCSP seulement.

Pour chaque opération, nous proposons sa description telle que donnée par Hoffmann *et al.*, avant de proposer notre interprétation de son effet au niveau du GCSP et de la façon dont doit s'opérer l'assemblage pour cette condensation. Cette présentation souligne les problèmes que peuvent poser chacune de ces opérations lorsque l'on s'intéresse à ce qu'elle signifie pour le GCSP et nous proposerons le cas échéant des solutions pour rendre leur utilisation possible.

Nous discutons alors les limites de chaque approche, et en particulier les conditions sous lesquelles ces condensations sont correctes et nous comparons les trois approches proposées par Hoffmann *et al.*.

5.5.1 Mise à jour du plan

Avant de détailler chacune des trois opérations de condensation proposées par Hoffmann *et al.*, nous décrivons la mise à jour du plan d'assemblage dont le schéma est commun à ces trois opérations. Nous verrons les différences entre les plans produits par CA, FA et MFA à la section 5.5.5.

Une opération de condensation effectue la mise à jour du plan d'assemblage en y introduisant le nouvel agrégat identifié. Le plan d'assemblage est généralement représenté comme un graphe orienté. Un agrégat K y est inséré sous forme d'un nouveau nœud N_K. Un arc lie tout nœud n représentant un objet ou un agrégat agrégé dans K au nouveau nœud N_K.

Le plan servira d'intermédiaire entre la phase de planification et la phase d'assemblage ; aussi stocke-t-il non seulement chaque nouvel agrégat K, mais aussi toutes les informations concernant son identification : étape de fusion avec la distribution de flot et le sous-GCSP minimal s_rigide identifié et chaque extension ayant ensuite intégré un objet ou agrégat

supplémentaire dans K. Toutes ces informations seront utiles à la génération des sous-systèmes d'équations à résoudre lors de la phase d'assemblage [Oung *et al.*, 2001] : chaque opération de fusion, et chaque extension sur un objet correspondra à un sous-système d'équations à résoudre séparément. Comme Sitharam *et al.* travaillent au niveau du graphe objets-contraintes et non du GCSP, il proposent un mécanisme de génération des équations à partir des arêtes du graphe qu'ils utilisent.

Nous verrons au chapitre 8 comment notre phase d'assemblage utilise le plan produit en phase d'analyse.

Correction des opérateurs de condensation

La méthode HLS étant basée sur la s_rigidité, elle peut être trompée par la présence de singularités, R-redondances ou inconsistances. On ne peut donc pas démontrer la correction des opérations de condensation qu'elle utilise vis à vis de la rigidité. Cependant, on peut vérifier que ces opérations sont correctes vis à vis de la s_rigidité. Intuitivement, une opération de condensation est correcte si et seulement si elle préserve la s_rigidité du GCSP. Afin de démontrer qu'une opération de condensation est correcte il faut et il suffit de démontrer la proposition suivante :

Proposition 14 Correction de la condensation
Soit K un agrégat identifié par fusion et extensions dans un GCSP S. Soit S' le GCSP résultant de la condensation de K dans S. L'opération de condensation est correcte si et seulement si les 4 propriétés suivantes sont vérifiées :

1. $DDL(S) = DDL(S')$,
2. *si S contient un sous-GCSP S_0 sur-s_rigide, alors S' contient un sous-GCSP S_1 sur-s_rigide,*
3. *si S' contient un sous-GCSP S_1 sur-s_rigide, alors S contient un sous-GCSP S_0 sur-s_rigide,*
4. *l'ensemble des solutions de $K \cup S'$ est le même que l'ensemble des solutions de S.*

En effet, les propositions 14-1, 14-2 et 14-3 assurent que la s_rigidité du GCSP est préservée :

- si S est sur-s_rigide, la proposition 14-2, assure que S' restera sur-s_rigide ;
- si S est bien- ou sous-s_rigide, alors il ne contient aucun sous-GCSP sur-s_rigide ; la contraposée de la proposition 14-3 assure alors que S' n'en contient pas non plus. De plus, la proposition 14-1 assure que S et S' possèdent le même nombre de DDL globalement, et donc qu'ils ont même s_rigidité : si S est sous-s_rigide, alors S' l'est aussi, alors que si S est bien-s_rigide, alors S' l'est aussi.

La proposition 14-4 assure quant à elle que l'ensemble des solutions est préservé *par assemblage*. L'assemblage consiste à reporter les solutions de l'agrégat K dans le GCSP S' mis à jour par condensation. Démontrer que l'ensemble des solutions reste identique consiste alors à démontrer que la résolution de K puis du sous-GCSP S' n'engendre ni ne fait disparaître de solutions par rapport à S.

5.5.2 Condensation par CA

L'opération de condensation de CA fut la première présentée [Hoffmann *et al.*, 1997] et est la plus intuitive. En terme de graphe, elle consiste à condenser l'agrégat K produit par les opérations de fusion et d'extensions en un sommet unique N_K dans le graphe objets-contraintes associé au GCSP S. Les arêtes qui étaient incidentes à des agrégats à l'intérieur de K sont reportées sur N_K.

L'exemple de déroulement de la méthode HLS présenté à la figure 4.10, page 105 utilisait l'opération de condensation CA ; elle illustre donc son application.

CA au niveau GCSP

Au niveau du GCSP, CA consiste à retirer du GCSP les objets regroupés dans K alors qu'un nouvel objet géométrique N_K y est introduit. Nous proposons que cet objet soit un *objet repère*. Ceci est possible car K est bien-s_rigide et dispose donc de $\frac{d(d+1)}{2}$ DDL, qui sont tous des déplacements ; un objet repère dispose aussi de $\frac{d(d+1)}{2}$ DDL et est aussi rigide.

Un objet repère N_K en dimension d se définit par un point origine p_K et un ensemble de $d-1$ vecteurs formant une base orthonormée $(v1_K, v2_K, ..., v(d-1)_K)$. Utiliser une telle représentation assure l'unicité de représentation d'une même configuration d'un objet repère. La modélisation que nous proposons pour les objets repères en 2D et 3D est présentée dans l'annexe A.

Les contraintes qui impliquaient des objets de K doivent alors être reportées sur l'objet repère N_K. Pour ce faire, il faut considérer que cet objet repère représentera le repère local de l'agrégat K ; il est alors possible de fixer chaque objet de K dans N_K, en exprimant tous les paramètres des objets de l'agrégat relativement à ceux de N_K.

Par exemple un point P en 2D paramétré par ses coordonnées x et y peut être exprimé relativement au repère N_K en considérant $P_{N_K} = p_K + x * v1_K + y * v1_K^{\perp}$, où $v1_K^{\perp}$ est le vecteur perpendiculaire à $v1_K$ qui forme avec lui un repère orthonormé. Si l'on pose $p_K = (x_r, y_r)$ et $v1_K = (a, b)$, on peut alors exprimer les coordonnées (x, y) du point P en fonction des paramètres du repère r et de ces coordonnées (x_K, y_K) dans r de la façon suivante :

$$\begin{aligned} x &= x_r + a * x_K - b * y_K \\ y &= y_r + b * x_K + a * y_K \end{aligned}$$

Nous présentons à l'annexe A la façon de reformuler chaque objet géométrique que nous définissons vis-à-vis d'un objet repère. Ces formules correspondent aux changements de repères classiques pour les objets de type points, droites et plans en 2D et 3D.

Reporter sur N_K toutes les contraintes C_P qui portent sur P consiste alors à substituer toute occurrence de x et y dans les équations modélisant les contraintes C_P par leur expression en fonction des paramètres de N_K. C'est la fonction `Traduire-Contrainte` qui réalise cette substitution, pour une contrainte c et un objet o de K donnés, dans l'algorithme présenté à l'algorithme 11.

La complexité en temps de cette opération est en $O(n * m)$ où n est le nombre d'objets géométriques dans K et m est le nombre de contraintes de S' incidentes à des objets de

Algorithme 11 CA ($K = (O_K, C_K)$: agrégat, $S = (O, C)$: le GCSP) **retourne** $S' = (O', C')$ un GCSP équivalent

$O' \leftarrow O \setminus O_K$ *{retrait des objets agrégés}*
$C' \leftarrow C \setminus C_K$ *{retrait des contraintes agrégées}*
$O' \leftarrow O' \cup \{N_K\}$ *{introduction de l'objet repère remplaçant K}*
Pour chaque $o \in O_K$ **Faire**
  Pour chaque $c \in C$ **tel que** c porte sur o **Faire**
    $c' \leftarrow$`Traduire-Contrainte`(c, o, N_K) *{report sur N_K de chaque contrainte c portant sur un objet o de K}*
    $C' \leftarrow C' - \{c\} \cup \{c'\}$
  Fin Pour
Fin Pour
Return $S' = (O', C')$

K. On considère que la traduction d'une contrainte s'effectue en temps constant : dans la grande majorité des contraintes géométriques, le nombre d'occurrence d'une même variable dans une même équation dépasse rarement trois.

Incorrection de CA

CA est une opération de condensation incorrecte dans le cas général : elle peut faire disparaître les sur-s_rigidités d'un GCSP, ce qui viole la proposition 14-2, proposition nécessaire à la correction de toute opération de condensation.

Considérons par exemple le GCSP présenté à la figure 5.6. Ce GCSP est constitué de 6 points A, B, C, D, E et F liés par 9 contraintes de distances (représentées par des segments sur la figure). Ce GCSP est sur-s_rigide : le sous-GCSP $ABCD$ ne dispose que de 2 degrés de liberté, ce qui est inférieur à $\frac{d(d+1)}{2} = 3$ en 2D. Or, après une étape de la méthode HLS, cette sur-s_rigidité disparaît lorsque l'agrégat $K = DEF$ identifié est condensé en un objet unique N_K doté de 3 DDL.

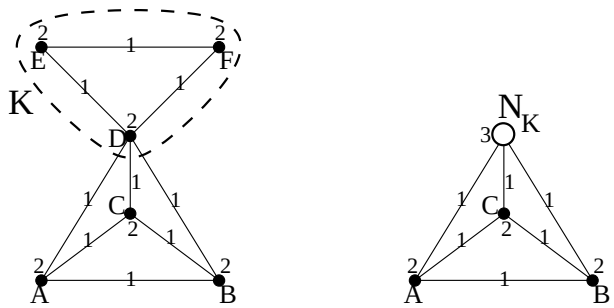

FIG. 5.6 – Exemple de GCSP pour lequel CA est incorrect

Correction conditionnelle de CA

Cependant, nous démontrons que CA est correct sous l'hypothèse d'absence de sur-s_rigidité dans le GCSP initial. Pour ce faire, nous allons démontrer que la proposition 14, page 142, est vérifiée si le GCSP S est initialement exempt de sur-s_rigidité. Notons S le GCSP contenant un agrégat K, et S' le GCSP dans lequel K a été condensé par CA.

Démonstration de la proposition 14-1 Remarquons tout d'abord que $S' = (S \setminus K) \cup N_K$. Calculons alors le nombre de DDL de S' : $DDL(S') = DDL(S) - DDL(K) + DDL(N_K)$. Or un objet repère dispose de $\frac{d(d+1)}{2}$ DDL en dimension d, comme tout sous-GCSP bien-s_rigide, en particulier K. Donc $DDL(S') = DDL(S)$. □

Démonstration de la proposition 14-2 Par hypothèse, le GCSP S ne contient initialement pas de sur-s_rigidité. Or, la proposition 14-3 assure que CA ne peut pas faire apparaître de sur-s_rigidité, et il n'en existera donc jamais au cours des itérations de la méthode HLS. S'il n'en existe jamais, CA ne peut pas en faire disparaître. □

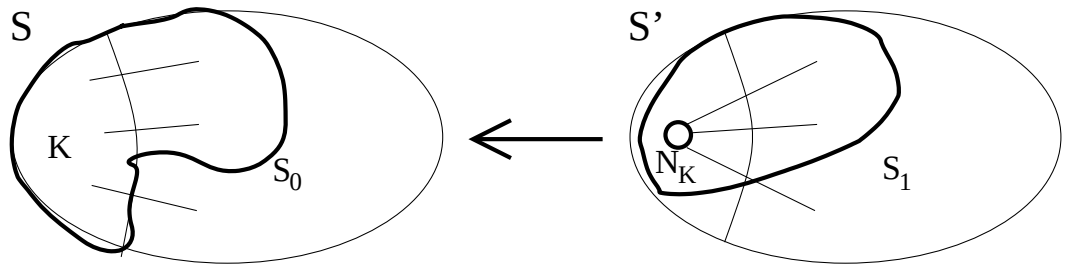

FIG. 5.7 – Sous-GCSP utilisés dans la démonstration de la proposition 14-3 pour CA

Démonstration de la proposition 14-3 Considérons S_1, un sous-GCSP sur-s_rigide de S'. Deux cas peuvent se présenter :

- S_1 ne contient pas l'objet repère N_K qui représente K dans S' ; alors, S_1 est aussi un sous-GCSP de S, qui contient donc aussi un sous-GCSP sur-s_rigide.
- S_1 contient l'objet repère N_K qui représente K dans S' ; alors, considérons $S_0 = (S_1 \setminus \{N_K\}) \cup K$, c-a-d. que S_0 est le sous-GCSP de S induit par tous les objets de S_1 sauf N_K, et tous les objets de l'agrégat K à la place. La figure 5.7 illustre les sous-GCSP S_1 et S_0 considérés. Calculons alors le nombre de DDL de S_0 : $DDL(S_0) = DDL(S_1) + DDL(K) - DDL(N_K) + DDL(C') - DDL(C'')$; C' est l'ensemble des contraintes liant N_K à un objet de S_1 dans S', alors que C'' est l'ensemble des contraintes liant un objet de K à un objet de $S_1 - N_K$ dans S. Par définition de l'opération de condensation CA, C' représente le contraintes de C'' traduites pour porter sur l'objet repère N_K ; donc $DDL(C') = DDL(C'')$. De plus, $DDL(K) = DDL(N_K)$, car K est bien-s_rigide et N_K est un objet repère : ils disposent tous

deux de $\frac{d(d+1)}{2}$ DDL. Donc $DDL(S_0) = DDL(S_1)$, et S_0 est un sous-GCSP sur-s_rigide de S.

□

Démonstration de la proposition 14-4 CA remplace un agrégat K par un objet repère N_K sur lequel les contraintes incidentes à des objets de K sont reportées. Aucune contrainte n'est insérée, et CA est donc correct pour peu que l'agrégat puisse être remplacé par un repère local. Nous verrons comment on peut répondre à ce problème dans le chapitre suivant. Lorsque la s_rigidité correspond à la rigidité, un agrégat s_rigide peut être remplacé par un repère local, ce qui justifie CA.

L'assemblage consiste simplement à déplacer l'agrégat K pour le positionner à la place où se trouve le repère N_K qui le représente dans S'. Le déplacement est correct et n'engendre pas de solutions parasites si les objets de K permettent la définition d'un repère local. Ceci correspond au problème du repère local (cf. section 4.5.1, page 116) et n'a pas été soulevé par Hoffmann *et al.*. Sous l'hypothèse où la s_rigidité correspond à la rigidité, on peut cependant constater que les $\frac{d(d+1)}{2}$ DDL d'un agrégat s_rigide correspondent bien à tous les déplacements possibles de l'espace géométrique de dimension d : l'assemblage sera possible et correct (nous discuterons de ce problème du repère local et de l'assemblage plus en détail au chapitre suivant).

5.5.3 Condensation par FA

C'est sans doute l'incorrection de CA qui a conduit Hoffmann, Lomonosov et Sitharam à proposer l'opérateur de condensation FA (*Frontier Algorithm*) [Hoffmann *et al.*, 2000]. Cette opération repose sur la notion de *frontière*. Intuitivement, la frontière d'un agrégat est composée de tous les objets de l'agrégat sur lesquels portent des contraintes extérieures à l'agrégat. Plus formellement :

Définition 43 Frontière d'un agrégat
Soit $S = (O, C)$ un GCSP, et $K = (O_K, C_K)$ un agrégat dans S. Un objet $o \in O_K$ est dit **frontière** *si et seulement si $\exists c \in C \setminus C_K$ telle que c porte sur o. On note F_K l'ensemble des objets frontières de K, et K_F le sous-GCSP de K induit par F_K.*

Par opposition, les objets de K n'appartenant pas à F_K sont dits internes *; l'ensemble des objets internes est noté I_K, et le sous-GCSP de K induit par I_K est noté K_I.*

En terme de mise à jour du graphe objets-contraintes, l'opération FA est décrite de la façon suivante par Hoffmann *et al.* [Hoffmann *et al.*, 2000] : elle consiste à remplacer les sommets internes de l'agrégat K par un unique sommet N_K de poids $\frac{d(d+1)}{2}$, à retirer toutes les arêtes de K et à relier N_K à chaque sommet $v \in F_K$ par une arête de poids $DDL(v)$. Notons que le sommet N_K n'est introduit que si l'ensemble des sommets internes de K est non vide ; en cas contraire, l'agrégat K n'est pas condensé du tout par FA.

La figure 5.8 présente deux exemples d'application de FA pour la condensation d'un agrégat K dans un GCSP S. On peut constater que lorsque l'agrégat K ne contient aucun objet interne, la condensation par FA est l'identité, c-a-d. qu'elle ne fait rien.

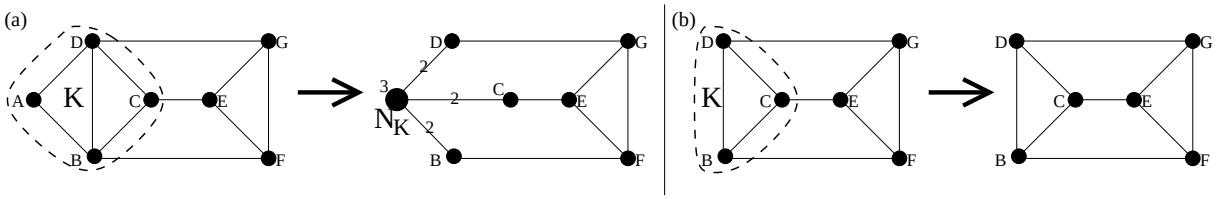

FIG. 5.8 – Exemples d'application de l'opération de condensation FA

FA au niveau GCSP

Au niveau du GCSP, il est possible d'expliquer FA de la façon suivante : le sommet unique introduit par FA représente un objet repère N_K, et l'arête C_o introduite entre chaque objet frontière $o \in F_K$ et l'objet repère N_K est une contrainte fixant o dans N_K. Il est nécessaire de fixer chaque objet frontière dans N_K afin de préserver dans S' le fait que les objets frontières de K appartiennent déjà à un agrégat rigide.

Nous avons déjà expliqué à la section 5.5.2 comment introduire un objet repère N_K et comment fixer un objet o vis-à-vis de cet objet repère : en introduisant les relations qui définissent les paramètres de o en fonction des paramètres de N_K.

Cependant, au lieu de substituer les paramètres des objets frontières par ces expressions dans toutes les contraintes du GCSP, FA propose de dupliquer les objets frontières afin de permettre qu'ils puissent être résolus indépendamment dans plusieurs agrégats. Les instances dupliquées remplacent les instances précédentes dans toutes les contraintes restantes du GCSP, et ce sont donc ces copies qui sont fixées relativement à N_K.

L'algorithme réalisant cette opération de condensation est présenté à l'algorithme 12. Rappelons que l'agrégat n'est condensé que s'il contient au moins un objet interne.

La complexité de l'opération FA est donc en $O(n*m)$ tout comme celle de CA, n étant le nombre d'objets frontières de K et m le nombre de contrainte incidentes à ses objets frontières. On peut considérer que la duplication d'un objet frontière o, l'introduction d'une contrainte le fixant par rapport au repère N_K et la substitution de o par sa copie dans une contrainte c se font en temps constant.

La correction de FA a été démontrée par Hoffmann *et al.* dans [Hoffmann *et al.*, 2000].

Problèmes posés par FA

L'opération FA permet de réaliser certaines extensions qui n'étaient pas réalisables avec CA. Nous allons voir qu'elle pose cependant quelques problèmes, d'une part car elle peut parfois ne pas condenser l'agrégat identifié par fusion et extensions, d'autre part car elle n'assure pas la terminaison de la méthode HLS, et enfin car elle ne permet plus certaines extensions.

FA sans effet Comme nous l'avons fait remarqué (cf. figure 5.8-b, page 147), la condensation par FA consiste parfois à ne rien changer au GCSP ; c'est le cas lorsque l'agrégat

Algorithme 12 FA ($K = (O_K, C_K)$: agrégat, $S = (O, C)$: le GCSP) **retourne** $S' = (O', C')$ un GCSP équivalent

$F_K \leftarrow \{o \in O_K | \exists c \in C \wedge \exists o' \in O \setminus O_K \wedge c$ porte sur o et $o'\}$
$I_K \leftarrow O_K \setminus F_K$
$O' \leftarrow \emptyset$
$C' \leftarrow \emptyset$
Si $|I_K| > 0$ **Alors**
  $O' \leftarrow O \setminus I_K$ *{retrait des objets internes de K}*
  $C' \leftarrow C \setminus C_K$ *{retrait des contraintes utilisées pour ces agrégations}*
  $O' \leftarrow O' \cup \{N_K\}$ *{introduction de l'objet repère N_K}*
  Pour chaque $o \in F_K$ **Faire**
    $o' \leftarrow$ `Nouvelle-Instance(`o`)` *{duplication de o}*
    $O' \leftarrow (O' \setminus \{o\}) \cup \{o'\}$ *{substitution de o par sa nouvelle instance o'}*
    $C' \leftarrow C' \cup \{$`Fixer(`o', N_K`)`$\}$ *{fixation de o' vis-à-vis du repère N_K}*
    Pour chaque $c \in C'$ **tel que** c porte sur o **Faire**
      $c' \leftarrow$`Substituer-Objet(`c, o, o'`)` *{substitution de o par o' dans la contrainte c}*
      $C' \leftarrow C' - \{c\} \cup \{c'\}$
    Fin Pour
  Fin Pour
Fin Si
Return $S' = (O', C')$

identifié par fusion puis extension ne contient aucun objet interne. Dans ce cas, cet agrégat est inintéressant car il ne permet pas de simplifier la résolution de S : en effet, $S' = S$, et donc la résolution de $S' \cup K$ n'offre aucun intérêt, puisque K est déjà résolu entièrement dans S'. Nous proposons d'ignorer les agrégats qui ne peuvent pas être condensés par FA : ils ne seront pas introduits dans le plan d'assemblage.

D'autre part, le fait que FA soit parfois l'opération identité pour un GCSP peut poser un problème de terminaison de la planification ; en effet, rien n'empêche que l'opération de fusion re-sélectionne *ad vitam aeternam* le même agrégat. Nous discutons de ce problème dans le paragraphe suivant, car il est général à la condensation par FA, et pas seulement lié au fait que, parfois, FA ne fait rien.

Problème de terminaison La condensation par FA met en péril la terminaison de la phase de planification. Considérons par exemple qu'un agrégat K est identifié dans S et condensé en K' dans S' ; à l'itération suivante, il est possible que l'agrégat identifié soit à nouveau K', car le sous-GCSP K' induit par $\{N_K\} \cup F_K$ est aussi bien-s_rigide (par définition de FA) ; de plus, K était maximal pour l'extension, et K' ne pourra donc être étendu plus. La mise à jour de K' par FA est l'identité, et le même phénomène peut se reproduire à l'itération suivante, et ainsi de suite...

Il faut donc assurer que l'agrégat K' ne peut pas être identifié aux itérations suivantes de la méthode.

Pour ce faire, Sitharam *et al.* proposent d'utiliser une fonction de distribution de flot supplémentaire dans le logiciel FRONTIER. Il est difficile de juger de ce que fait exactement cette fonction car l'algorithme de distribution de flot utilisé par FA est déjà constitué d'un ensemble de fonctions de distribution à la récursivité croisée difficile à appréhender.

Nous avons présenté une opération de distribution de flot utilisant une seule fonction de distribution basée sur un algorithme de calcul de flot maximum classique. Nous proposons une solution pour pouvoir continuer à utiliser l'algorithme `Minimal-Dense` tel que décrit à la section 5.3.1.

Notre solution consiste à générer un réseau objets-contraintes de façon quelque peu différente de celle décrite à la définition 41, page 129 : le nouveau réseau objets contraintes contiendra un nœud unique pour représenter chacun des anciens agrégats identifiés, c-a-d que les objets frontières ne seront pas représentés dans ce réseau. On peut comparer ce réseau à celui qui est généré lorsque l'opération de condensation utilisée est CA.

Générer ce réseau à chaque fois nécessite de reprendre en compte tous les agrégats déjà identifiés et peut s'avérer fastidieux, car un même objet peut appartenir à plusieurs agrégats. Nous proposons donc que le réseau soit conservé et mis à jour durant la phase de condensation par un algorithme dédié, présenté à l'algorithme 13.

Algorithme 13 MaintenanceRéseau ($G_S = (\{s,t\} \cup V, E, w)$: réseau ; $K = (O_K, C_K)$: agrégat) **retourne** $G_{S'} = (\{s',t'\} \cup V', E', w')$: réseau

$s' \leftarrow s$
$t' \leftarrow t$
$V' \leftarrow V$
$E' \leftarrow E$
$w' \leftarrow w$
$V' \leftarrow V' \cup \{N'_K\}$ *{introduction du nœud N'_K représentant K}*
$E' \leftarrow E' \cup \{N'_K \rightarrow t'\}$ *{lier le nœud introduit au puits, comme un objet}*
$w'(N'_K \rightarrow t') \leftarrow \frac{d(d+1)}{2}$ *{la capacité de cet arc est fixé à $\frac{d(d+1)}{2}$, le nombre de DDL de K}*
Pour chaque $v \in V'$ **tel que** v représente un agrégat dans K **Faire**
  $E' \leftarrow E' \setminus \{v \rightarrow t\}$ *{chaque nœud représentant un agrégat de K est retiré de $G_{S'}$}*
  Pour chaque $u \in V'$ **tel que** $(u \rightarrow v) \in E'$ **Faire**
    $E' \leftarrow E' \setminus \{u \rightarrow v\} \cup \{u \rightarrow N'_K\}$ *{chaque arc liant un nœud-contrainte à un nœud-objet de K est reporté sur N_K}*
    $w'(u \rightarrow N'_K) \leftarrow +\infty$
  Fin Pour
Fin Pour
Pour chaque $u \in V'$ **tel que** $\{v \in V' | (u \rightarrow v) \in E'\} = \{N_K\}$ **Faire** *{{si la contrainte u ne porte que sur N_K, c'est que c'est une contrainte interne à l'agrégat K}}*
  $V' \leftarrow V' \setminus \{u\}$ *{elle est retirée du réseau}*
  $E' \leftarrow E' \setminus \{u \rightarrow N_K, s \rightarrow u\}$ *{ainsi que tous les arcs dépendant d'elle}*
Fin Pour
Return $G_{S'} = (\{s',t'\} \cup V', E', w')$

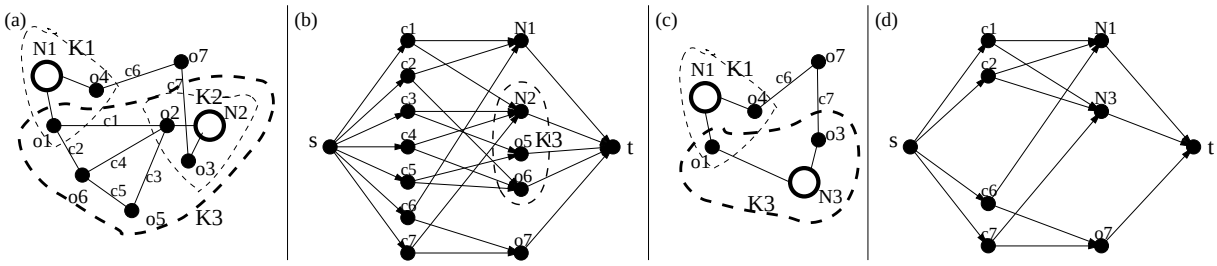

FIG. 5.9 – Illustration du déroulement de l'algorithme `Maintenance-Réseau`

La figure 5.9, illustre la mise à jour du réseau objets-contraintes lors de la condensation d'un agrégat $K3$. A la figure 5.9-a, on aperçoit le graphe objets-contraintes du GCSP S où les agrégats $K1$, $K2$ et $K3$ sont représentés. On peut constater que l'agrégat $K1$ partage un de ses objets frontières, $o1$, avec le nouvel agrégat $K3$. D'autre part, tous les objets $\{o2, o3, N_2\}$ de $K2$ sont intégrés dans $K3$.

La figure 5.9-b présente le réseau objets-contraintes correspondant à ce GCSP S. Les objets de $K1$ et de $K2$ n'y apparaissent plus puisque les agrégats y sont représentés par un nœud unique, respectivement $N1$ et $N2$. En revanche, les contraintes $c1$ et $c2$ qui lient les objets $o6$ et $o3$ à $o1$ et ont permis son inclusion dans $K3$ apparaissent : $c1$ est placée entre $N1$ et $N2$ car $o1$ appartient à $K1$, et $o3$ appartient à $K2$; $c2$ est placée entre $o6$ et $N1$. De même, les contraintes $c3$ et $c4$ qui lient les objets $o5$ et $o6$ de $K3$ à l'objet frontière $o2$ de $K2$ apparaissent et sont liées directement à $N2$.

Les figures 5.9-c et 5.9-d représentent le graphe objets-contraintes après condensation de $K3$ par FA et le réseau objets-contraintes après la mise à jour par l'algorithme `MaintenanceRéseau`. Le graphe objets-contraintes contient maintenant un sommet N_3 qui représente tous les objets internes de $K3$; on peut noter que N_2 a disparu puisque $K2$ était entièrement inclus dans $K3$. Les objets frontières de $K3$ sont $o1$ et $o3$, qui sont dupliqués en $o1'$ et $o3'$. On peut noter que $o1$ était auparavant un objet frontière de $K1$, alors que $o3$ était un objet frontière de $K2$. $o1'$ et $o3'$ sont liés à N_3 par une arête qui représente le fait qu'ils sont fixes dans le repère associé à $K3$.

Le réseau objets-contraintes contient à présent un nœud $N3$ qui représente l'agrégat $K3$. Ce nœud a remplacé les nœuds correspondants à des objets de $K3$: $o5$, $o6$ et $N2$. Les contraintes internes à $K3$, $c3$, $c4$ et $c5$, ont disparu aussi, et les nouvelles contraintes placées entre $N3$ et ses objets frontières n'apparaissent pas puisque les objets frontières n'apparaissent pas dans le réseau. Les contraintes $c6$ et $c7$ qui étaient incidentes à des objets frontières de $K3$ ont été reportées sur $N3$. D'autre part, les contraintes $c1$ et $c2$, qui avaient permis l'extension de $K3$ sur $o1$, apparaissent à présent entre $N1$ et $N3$: elles représentent le fait que ces agrégats partagent l'objet $o1$.

La recherche d'un nouvel agrégat par un appel à l'algorithme `Minimal-Dense` dans ce réseau ne peut pas retourner un agrégat déjà identifié, puisque chaque agrégat précédemment identifié n'y est représenté que par un nœud : cela règle donc le problème de terminaison avec la condensation FA.

Problème d'extension Contrairement à CA, les extensions dans FA ne permettent pas d'inclure un ancien agrégat en une seule fois : pour pouvoir inclure un ancien agrégat K' dans K, il faut d'abord étendre K vers des objets frontières de K', puis étendre K sur $N_{K'}$ et finalement depuis $N_{K'}$ on peut inclure les derniers objets frontières de K'.

Alors, en dimension $d = 2$, il n'est plus possible d'inclure un agrégat entier que si sa frontière comporte un objet possédant 3 DDL. Rappelons que points et droites possèdent seulement 2 DDL en 2D, et qu'il sera donc impossible de procéder à un extension sur un agrégat entier en 2D.

En 3D, un objet repère possède 6 DDL, et tout objet rigide possède entre 3 et 6 DDL. Il n'est alors possible d'inclure un agrégat entier que si sa frontière contient un objet possédant 6 DDL, ou deux objets possédant 3 DDL. Rappelons qu'en 3D, les points, plans et droites, disposent respectivement de 3, 3 et 4 DDL ; il ne sera donc possible de procéder à l'inclusion complète d'un ancien agrégat que si sa frontière contient une paire de points ou plans. Or de telles inclusions sont fausses du point de vue de la rigidité :

- fixer 2 points dans un agrégat laisse à cet agrégat la possibilité de tourner sur l'axe défini par ces 2 points (cf. contre-exemple présenté à la figure 2.10, page 44),
- fixer 2 plans dans un agrégat laisse à cet agrégat la possibilité d'être translaté dans la direction de la droite d'intersection de ces 2 plans,
- fixer 1 point et 1 plan dans un agrégat laisse à cet agrégat la possibilité de tourner autour du point fixer selon l'axe défini par le vecteur normal à ce plan.

FA limite donc les extensions possibles en ne permettant pas toujours l'inclusion d'anciens agrégats entiers lorsqu'elle serait possible, et peut réaliser des extensions fausses, du point de vue de la rigidité, en 3D. Cette dernière remarque est en fait valable aussi pour CA et MFA car elle est liée à l'utilisation de la s_rigidité comme caractérisation de la rigidité.

Problème d'assemblage

L'assemblage avec FA consiste à utiliser la position des objets frontières F_K dans K et leurs doubles dans S' (le GCSP S après condensation de K par FA) afin de déterminer le déplacement à appliquer à tous les objets de K pour les positionner dans S'. Cet assemblage est correct si et seulement si F_K permet la définition d'un repère local (cf. section 4.5.1, page 116). Comme pour CA, ceci n'est vérifié que dans le cas où la s_rigidité correspond à la rigidité, comme nous le verrons au chapitre suivant.

5.5.4 Condensation par MFA

MFA (*Modified Frontier Algorithm*) est le dernier opérateur de condensation proposé par Hoffmann *et al.* [Hoffmann *et al.*, 2000]. Il est aussi basé sur la notion d'objets frontières, mais effectue une mise à jour sensiblement différente de FA.

En terme de mise à jour du graphe objets-contraintes, l'opération MFA consiste à remplacer le sous-graphe induit par les sommets internes I_K de l'agrégat K par un unique sommet N_K, et à reporter les arêtes qui étaient incidentes à des sommets de I_K directement

sur N_K ; le poids du sommet N_K est égal au poids du sous-graphe induit par I_K. Le sommet N_K n'est introduit que si l'ensemble des sommets internes de K contient au moins 2 éléments ; en cas contraire, l'agrégat K n'est pas condensé du tout par MFA.

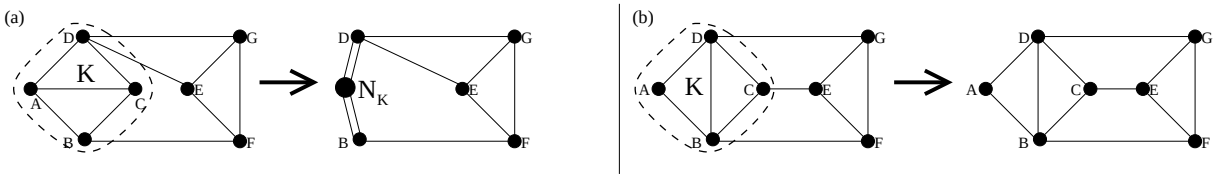

FIG. 5.10 – Exemples d'application de l'opération de condensation MFA

La figure 5.10 présente 2 exemples d'application de MFA pour la condensation d'un agrégat K dans un GCSP S. On peut constater que lorsque l'agrégat K ne contient aucun objet interne, la condensation par MFA est l'identité, c-a-d. qu'elle ne fait rien.

MFA au niveau du GCSP

Dans le cas de MFA, le sommet unique N_K représente le sous-GCSP K_I induit par les objets internes I_K de K. Ces objets ne peuvent être remplacés par un objet repère que si le sous-GCSP K_I qu'ils induisent est bien-s_rigide, comme pour CA ; par exemple, si les objets internes d'un agrégat en 3D sont réduits à une paire de points, ils ne permettent pas de définir un repère local. On ne peut donc pas toujours représenter l'introduction du sommet N_K par l'introduction d'un objet repère dans le GCSP.

Sitharam *et al.* utilisent MFA dans le logiciel FRONTIER, mais n'ont pas soulevé ce problème.

Nous proposons la solution suivante :

- Si K_I, le sous-GCSP induit par les objets internes de I_K, est bien-s_rigide, alors N_K est un nouvel objet repère introduit dans le GCSP. Dans ce cas, il faut reporter les contraintes qui portaient sur des objets de I_K directement sur N_K, comme cela a été expliqué pour CA (cf. page 143). De plus, les objets frontières doivent être dupliqués, comme pour FA, afin de pouvoir être résolus dans plusieurs agrégats indépendamment (cf. page 147). Les contraintes qui étaient incidentes à des objets frontières sont alors reportées sur les nouvelles instances de ces objets.
- Sinon l'agrégat K n'est pas condensé du tout, et sera donc résolu entièrement dans un prochain agrégat. Ceci est relativement gênant, car il peut exister des sous-GCSP K_I sous-s_rigides de taille arbitrairement grande.

Pour déterminer si K_I est bien-s_rigide, il suffit de vérifier que $DDL(K_I) = \frac{d(d+1)}{2}$. En effet, K étant bien-s_rigide et K_I étant un sous-GCSP de K, K_I est nécessairement bien- ou sous-s_rigide et tous ces sous-GCSP le sont aussi. Pour déterminer s'il est bien- ou sous-s_rigide, il suffit alors de vérifier qu'il possède globalement le bon nombre de DDL (cf. définition 32, page 60).

L'algorithme réalisant cette opération de condensation est présenté à l'algorithme 14.

Algorithme 14 MFA ($K = (O_K, C_K)$: agrégat, $S = (O, C)$: le GCSP) **retourne** $S' = (O', C')$ un GCSP équivalent

$F_K \leftarrow \{o \in O_K | \exists c \in C \wedge \exists o' \in O \setminus O_K \wedge c$ porte sur o et $o'\}$
$I_K \leftarrow O_K \setminus F_K$
$C_{I_K} \leftarrow \{c \in C_K | c$ ne porte que sur des objets de $I_K\}$
Si $DDL((I_K, C_{I_K})) = \frac{d(d+1)}{2}$ **Alors**
  $O' \leftarrow (O \setminus I_K) \cup \{N_K\}$ *{retrait des objets internes de K et ajout de l'objet repère* N_K*}*
  $C' \leftarrow C \setminus C_{I_K}$ *{retrait des contraintes entre objets internes}*
  Pour chaque $o \in I_K$ **Faire**
    Pour chaque $c \in C_K$ **tel que** c porte sur o **Faire**
      $c' \leftarrow$`Traduire-Contrainte`(c, o, N_K) *{report de c sur* N_K*}*
      $C' \leftarrow C' - \{c\} \cup \{c'\}$
    Fin Pour
  Fin Pour
  Pour chaque $o \in F_K$ **Faire**
    $o' \leftarrow$ `Nouvelle-Instance`(o) *{duplication de o en* o'*}*
    $O' \leftarrow O' \setminus \{o\} \cup \{o'\}$ *{substitution de o par* o'*}*
    Pour chaque $c \in C'$ **tel que** c porte sur o **Faire**
      $c' \leftarrow$`Substituer-Objet`(c, o, o') *{substitution de o par* o' *dans c}*
      $C' \leftarrow C' - \{c\} \cup \{c'\}$
    Fin Pour
  Fin Pour
Sinon
  {Aucune condensation n'est faite, car on ne sait pas remplacer les objets internes}
Fin Si
Return $S' = (O', C')$

Cette opération combine des sous-parties des 2 opérations de condensation précédemment présentées et est ainsi de complexité $O(n * m)$, n représentant toujours le nombre d'objets frontières de K et m le nombre de contraintes incidentes à ces objets frontières. A nouveau, on considère que la traduction de contraintes, la duplication d'objets et la substitution d'objets dans des contraintes se font en temps constant.

La correction de MFA a été démontrée par Hoffmann *et al.* dans [Hoffmann *et al.*, 2000].

Problèmes de MFA

MFA sans effet MFA pose le même problème que la méthode FA : parfois il ne condense pas du tout l'agrégat K. Cela se produit non seulement lorsque $|I_K| < 2$, mais aussi lorsque I_K ne peut être remplacé par un repère local, c-a-d. lorsqu'il est sous-s_rigide.

Dans ces cas, nous proposons la même solution que pour FA : ne pas tenir compte de la découverte de K dans le plan d'assemblage, puisqu'il sera de toute façon re-résolu entièrement dans un agrégat de taille supérieur dans une itération future.

Notons que cela engendre aussi le problème de terminaison signalé pour FA : rien n'assure que les prochaines fusions n'identifieront pas à nouveau les agrégats laissés inchangés. Ce problème est discuté au paragraphe qui suit, car il est en fait plus général.

Problème de terminaison Tout comme FA et pour les mêmes raisons, MFA peut amener la phase de planification à ne pas terminer. Afin d'éviter qu'un ancien agrégat puisse être identifié à nouveau lors d'une itération suivante, nous proposons la même solution que pour FA : maintenir le réseau objets-contraintes à l'aide de l'algorithme `MaintenanceRéseau` (cf. algorithme 13, page 149).

Problème du repère local Nous avons déjà expliqué que déterminer si un ensemble d'objets permet la définition d'un repère est un problème difficile dans le cas général (cf. section 4.5.1, page 116). Nous proposons une solution à ce problème à la section 6.4, page 191.

Problème d'assemblage L'assemblage avec MFA consiste, comme celui de FA, à utiliser la position des objets frontières F_K dans K et leurs doubles dans S' (le GCSP S après condensation de K par MFA) afin de déterminer le déplacement à appliquer à tous les objets de K pour les positionner dans S'. Cet assemblage est correct si et seulement si F_K permet la définition d'un repère local (cf. section 4.5.1, page 116). Comme pour CA, ceci n'est vérifié que dans le cas où la s_rigidité correspond à la rigidité, comme nous le verrons au chapitre suivant.

De plus, MFA laisse les contraintes entre les objets frontières dupliqués. Ceux-ci sont donc résolus dans S' indépendamment des objets frontières résolus dans K. Si les configurations des objets frontières sont incompatibles, l'assemblage ne pourra pas avoir lieu et aucune solution parasite ne sera donc engendrée pour $K \cup S'$. En revanche, ces solutions incompatibles seront générées séparément pour K et S', ce qui représente une perte de temps au moment de la résolution.

Pour résoudre ce problème, il faut établir un lien entre le objets frontières dans K et leurs copies dans S'. Ce lien peut être établi par des contraintes supplémentaires qui induisent une seule configuration dans S' par configuration dans K. Par exemple, on peut introduire des contraintes imposant que les objets frontières respectent dans S' la même topologie que dans K (contraintes d'orientation, ...). Ces contraintes ne sont pas introduites pour la planification, mais stockées dans un ensemble des contraintes annexes qui sera utilisé lors de la phase d'assemblage pour invalider au plus tôt les solutions incompatibles.

5.5.5 Comparaison des 3 opérations de condensation

Nous récapitulons ici les différences entre les 3 opérations de condensation proposées par Hoffmann *et al.* :

- CA remplace l'agrégat par un objet repère ; ceci est toujours possible, mais peut s'avérer incorrect en présence de sur-s_rigidités. De plus l'assemblage peut engen-

drer des solutions parasites qui sont éliminées au prix d'un test supplémentaire des contraintes initiales du GCSP sur les solutions assemblées. CA ne pose pas de problème de terminaison, ni d'extension, mais ne permet pas le partage d'objets entre agrégats.

- FA remplace les objets internes d'un agrégat par un objet repère, retire toutes les contraintes de l'agrégat et fixe chaque objet frontière relativement au nouvel objet repère ; ceci est toujours possible et correct. L'assemblage peut se baser sur les seuls objets frontières et ne produit pas de solutions parasites. Cependant, FA pose un problème de terminaison, réglé au prix du maintien du réseau utilisé par la fusion par une mise à jour particulière. FA permet le partage d'objets frontières, mais ne permet d'en partager qu'un à la fois de façon correcte, ce qui empêche la plupart des extensions sur des agrégats entiers.
- MFA remplace les objets internes par un objet repère sur lequel sont reportées les contraintes qui étaient incidentes à des objets internes ; cela n'est possible que si les objets internes peuvent définir un repère local et que le poids du sous-agrégat qu'ils induisent est bien le même que celui d'un objet repère. L'assemblage peut être réalisé à partir des objets frontières et n'engendre pas de solutions parasites. Cependant, MFA pose un problème de terminaison, pouvant être réglé comme celui de FA. MFA permet le partage d'objets frontières et ne limite pas l'extension.

Illustrons ces différences sur un exemple ; la figure 5.11 présente un GCSP en 2D constitué de 9 points liés par 15 contraintes de distances, représentées par des segments. Cette figure illustre la décomposition par CA, FA et MFA de ce GCSP et à chaque fois le plan résultant. Nous avons considéré que les sous-GCSP retournés par l'opération de fusion sont (dans l'ordre) : AB, DE, GI, et $N_{K1}N_{K2}N_{K3}$ dans le cas de CA et FA.

On peut constater que la décomposition de ce GCSP par CA et FA nécessite 4 itérations, c-a-d. 4 opérations de fusion. A l'inverse, la décomposition par MFA de ce même GCSP ne nécessite que 3 itérations ; ceci illustre le fait que l'opération MFA permet plus d'extensions que CA et FA. D'autre part, on peut constater que FA est équivalent à CA, mais si le second agrégat identifié par fusion avait été GI au lieu de DE, CA aurait permis l'extension sur N_{K1} qui à son tour aurait permis d'inclure directement D, puis E, et ainsi résoudre tout le GCSP en 2 itérations. Pour FA, découvrir GI à la seconde fusion n'améliore pas la décomposition, car FA est incapable d'étendre vers des agrégats entiers. MFA lui, aurait permis comme CA de résoudre tout le GCSP en 2 itérations : c'est le fait que les contraintes entre les objets frontières demeurent qui fait la différence avec FA.

Quant au plan d'assemblage produit, on peut remarquer que celui de CA forme un arbre renversé d'agrégats, car tout objet et tout agrégat ne peut être intégré qu'une seule fois dans un agrégat d'ordre supérieur. Dans le cas de FA et MFA, les agrégats peuvent se recouvrir partiellement, et un même objet peut être résolu dans plusieurs agrégats ; le plan forme alors un DAG. Dans tous les cas, l'unique puits du plan est le dernier agrégat identifié, celui qui regroupe l'ensemble du GCSP[5].

[5]Dans le cas d'un GCSP sous-rigide, le plan est une forêt de DAG donc les puits représentent chacun un sous-GCSP rigide maximal

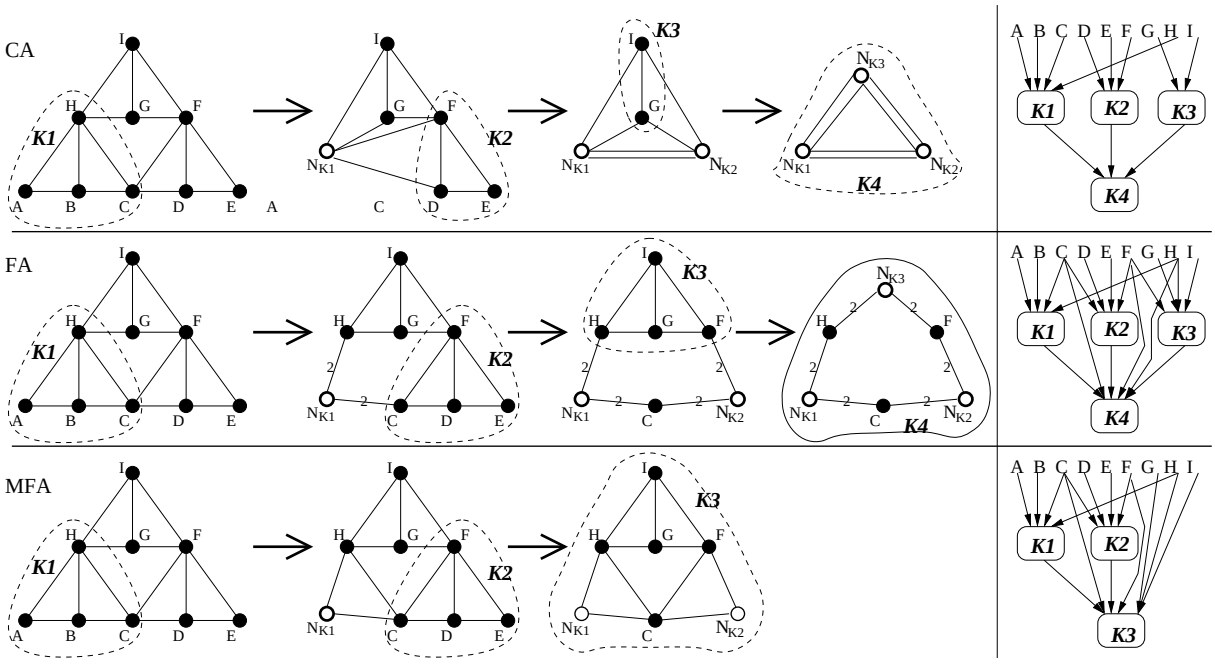

FIG. 5.11 – Comparaison des condensations par CA, FA et MFA

De notre point de vue, le choix se situe entre CA et MFA : FA est moins performant que CA et MFA pour les extensions, son seul avantage étant sa plus grande *robustesse*. Or maximiser le nombre d'étapes d'extension est intéressant pour trois raisons :

- cela abaisse le coût de la phase de planification, l'opération d'extension étant de complexité pratique inférieure à celle de la fusion ;
- cela réduit le nombre d'itérations nécessaires pour décomposer un GCSP ;
- cela affine la décomposition car les sous-systèmes produits par extension résolvent toujours un unique objet géométrique, alors que ceux produits par fusion résolvent toujours au moins 2 objets géométrique, et peuvent en résoudre un nombre arbitraire.

MFA permet généralement plus d'extensions que CA ; En fait, toute extension qui était possible dans un GCSP reste possible au fil des mises à jour de ce GCSP par MFA. Intuitivement, cette propriété est assez évidente : MFA laisse toutes les contraintes pertinentes du GCSP, c-a-d. les contraintes portant sur des objets pouvant encore être résolus dans de futurs agrégats, en place. Cependant, MFA peut être incapable de produire certaines extensions sur des agrégats entiers contrairement à CA.

MFA, comme FA, duplique les objets frontières et les contraintes et peut donc agrandir sensiblement la taille totale du GCSP à résoudre. De plus la condensation par MFA n'est possible que si l'ensemble des objets internes permet la définition d'un repère local et induit un sous-GCSP ayant $\frac{d(d+1)}{2}$ DDL. Ceci constitue une sérieuse limite à l'utilisation de MFA.

5.6 Méthode CA+MFA

Nous proposons dans cette section une amélioration directe de la phase de planification de la méthode HLS. Cette amélioration utilise :

- une opération d'élimination des sur-s_rigidités du GCSP traité,
- l'opération de fusion présentée en section 5.3 (page 125),
- l'opération d'extensions présentées en section 5.4 (page 137),
- l'opération de condensation CA+MFA, combinaison des opérations CA et MFA, que nous allons présenter maintenant.

La planification consiste alors à commencer par appliquer l'opération d'élimination des sur-s_rigidité afin d'obtenir un GCSP exempt de sur-s_rigidité, puis à appliquer itérativement la séquence fusion-extensions-condensation jusqu'à ce que la fusion échoue à identifier un nouveau sous-GCSP bien-s_rigide.

Nous allons décrire la phase d'élimination des sur-s_rigidités puis décrire la nouvelle opération de condensation, que nous appelons CA+MFA, avant de fournir une petite comparaison avec les 3 condensations proposées par Hoffmann *et al.*.

5.6.1 Elimination des sur-s_rigidités

La phase d'élimination des sur-s_rigidités consiste à reconstruire incrémentalement le GCSP contrainte par contrainte en vérifiant à chaque étape l'apparition d'une sur-s_rigidité. Si une sur-s_rigidité apparaît, elle est alors forcément due à la dernière contrainte introduite ; celle-ci est alors supprimée du GCSP. Pour vérifier l'apparition de sur-s_rigidité, on utilise à chaque étape l'algorithme `Dense` (algorithme 8, page 131). En effet, cet algorithme peut identifier un sous-GCSP sur-s_rigide en temps polynomial s'il en existe un ; il suffit pour cela d'utiliser la valeur de surcharge $\frac{d(d+1)}{2}$ au lieu de $\frac{d(d+1)}{2}+1$.

Algorithme 15 Eliminer-Sur-S_Rigidités ($S=(O,C)$: un GCSP ; d : la dimension de l'espace géométrique considéré) **retourne** $S'=(O',C')$ un GCSP équivalent à S ne contenant plus de sur-s_rigidité

$O' \leftarrow O$
$C' \leftarrow \emptyset$
Pour chaque $c \in C$ **Faire**
  $C' \leftarrow C' \cup \{c\}$
  Si `Dense`$(S', \frac{d(d+1)}{2}-1) \neq (\emptyset,\emptyset)$ **Alors**
    $C' \leftarrow C' \setminus \{c\}$ *{retrait de c qui introduisait une sur-s_rigidité dans S}*
    $Red \leftarrow Red \cup \{c\}$ *{stockage de c parmi les contraintes R-redondantes (ou inconsistantes)}*
  Fin Si
Fin Pour

Les contraintes ainsi supprimées sont stockées dans l'ensemble *Red*. Notons que ces contraintes peuvent être de véritables R-redondances, ou bien inconsistantes, la caractéri-

sation structurelle ne permettant pas de distinguer ces deux types de contraintes. Lors de la phase d'assemblage, on peut utiliser les contraintes stockées afin de filtrer les solutions produites par la résolution du GCSP épuré des contraintes sur-s_rigides. Si cela conduit à un ensemble de solutions vide, on saura que ces contraintes engendrent l'inconsistance ; sinon, les solutions restantes satisferont les contraintes R-redondantes.

L'algorithme 15 présente l'opération d'élimination des sur-s_rigidités. Cet algorithme possède une complexité en $O(m * C_{Dense})$, où m est le nombre de contraintes du GCSP initial. Il est correct et complet car toutes les contraintes sont examinées et chaque contrainte enlevée introduisait bien une R-redondance.

5.6.2 Condensation par CA+MFA

L'opération de condensation CA+MFA consiste à utiliser MFA autant que possible, et CA lorsque MFA ne peut pas condenser. Plus précisément, considérons un GCSP S dans lequel un agrégat K a été identifié par fusion puis agrandi par extensions.

Nous avons vu à la section 5.5.4 que MFA ne peut pas condenser K si les objets internes I_K de K ne permettent pas la définition d'un repère local, ou si le sous-GCSP K_I induit par ces objets internes ne possède pas exactement les $\frac{d(d+1)}{2}$ DDL d'un objet repère.

Nous proposons donc d'utiliser CA pour condenser l'agrégat K lorsque cette condensation n'est pas possible par MFA. Ainsi, cela évitera le problème de MFA signalé à la page 153 : le fait que MFA consiste parfois à ne rien faire entraîne que le sous-GCSP bien-s_rigide identifié sera amené à être résolu à nouveau entièrement dans un futur agrégat. CA+MFA ne posera pas ce problème, puisque CA permettra toujours de tenir compte des agrégats identifiés pour simplifier la résolution du GCSP. Ceci sera illustré à la section suivante.

Du point de vue de l'assemblage, rappelons que MFA peut engendrer des solutions partielles parasites et qu'il faut introduire des contraintes annexes si l'on souhaite éviter la génération inutile de ces solutions partielles (cf. section 5.5.4, page 153).

L'utilisation de MFA implique les problèmes listés à la section 5.5.4, page 153 : le problème d'identité disparaît grâce à l'utilisation conjointe de CA, le problème de fusion peut être réglé en maintenant le réseau objets-contraintes (cf. algorithme 13, page 149) ; quant au problème du repère local, il reste aussi difficile que pour MFA.

L'opération CA+MFA est correcte car :

- nous avons démontré à la page 145, que CA est correct si le GCSP est exempt de sur-s_rigidités ; ceci est assuré par l'application préliminaire de l'opération d'élimination des sur-s_rigidités.
- Hoffmann *et al.* ont démontré que MFA est correct.

Enfin, la complexité de l'opération est semblable à celle de MFA : $O(n * m)$, n représentant toujours le nombre d'objets frontières de l'agrégat à condenser et m le nombre de contraintes incidentes à ces objets frontières.

5.6.3 Comparaison

Intuitivement, CA+MFA est meilleur que MFA puisqu'il permet de condenser tous les agrégats identifiés et évite ainsi que tous les objets d'un agrégat ne soient résolus en une fois lors d'une itération future.

Afin d'illustrer ceci, considérons le GCSP présenté à la figure 5.10-b. Ce GCSP en 2D, constitué de 7 points et 11 contraintes de distance, n'est pas condensé du tout par MFA ; sa planification nécessite 3 itérations, les 2 premières produisant les agrégats $ABCD$ et EFG qui ne peuvent pas être condensés par MFA, et la dernière produisant l'agrégat $ABCDEFG$ qui résout l'ensemble du GCSP en une seule fois (cf. figure 5.12).

Avec CA+MFA, une première itération identifie l'agrégat $ABCD$ qui est condensé par CA en un objet repère $N1$; à la seconde itération, l'opération de fusion identifie l'agrégat FG qui est étendu sur E puis sur $N1$, résolvant ainsi le reste du GCSP (cf. figure 5.12).

Il faut donc seulement 2 itérations à CA+MFA pour décomposer ce GCSP, et les sous-systèmes à résoudre en une fois sont de taille au plus 3, alors qu'il faut 3 itérations à MFA pour décomposer le même GCSP, le système entier de 11 équations étant finalement résolu en une seule fois à la fin.

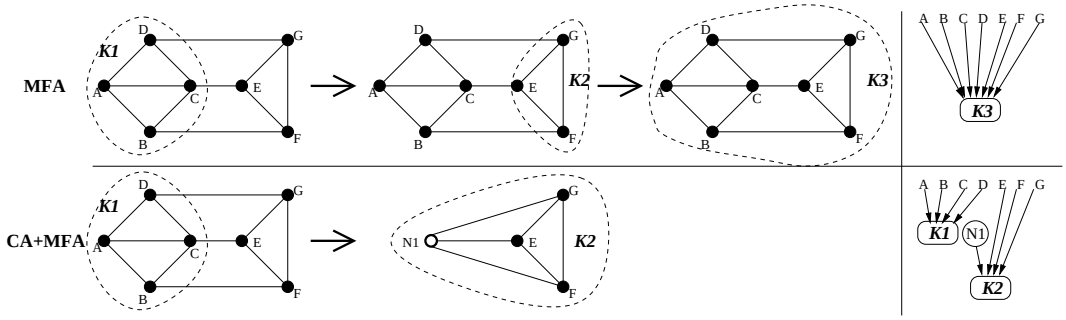

FIG. 5.12 – GCSP illustrant l'intérêt de CA+MFA par rapport à MFA seul

CA+MFA est aussi meilleur que CA car il peut permettre, comme MFA, de sauter des itérations de la phase d'assemblage en maximisant le nombre d'extensions possibles. Ceci a déjà été illustré sur le GCSP présenté à la figure 5.11, page 156.

Une comparaison plus poussée des différentes méthodes de décomposition sera proposée au chapitre D qui regroupe des expérimentations de décompositions et de résolutions sur des GCSP en 2D et 3D.

5.7 Conclusion sur la méthode HLS

La méthode de rigidification récursive proposée par Hoffmann, Lomonosov et Sitharam présente l'avantage d'être la première méthode de décomposition géométrique structurelle applicable à tout GCSP en dimension quelconque. Les opérations qu'elle met en jeu (Fusion, Extensions et Condensation) sont toutes polynomiales, ce qui rend le temps de

décomposition (phase de planification) négligeable en regard du temps de résolution (phase d'assemblage). Nous avons proposé des solutions personnelles aux problèmes que pose la méthode HLS lorsque l'on cherche à la mettre en œuvre pour la décomposition effective de GCSP. Nous avons aussi proposé une amélioration directe de la méthode qui consiste à combiner les différentes opérations de condensation qu'elle propose.

Malgré tout, l'utilisation de la s_rigidité comme caractérisation de la rigidité engendre des problèmes à tous les niveaux de la méthode : la correction de la méthode est toute relative car l'opération de fusion, l'opération d'extensions, l'opération de condensation et la méthode d'assemblage peuvent toutes produire des résultats incorrects du point de vue géométrique.

Ainsi l'exemple de la **double banane** présenté à la figure 2.10, page 44, peut être rigidifié par la méthode HLS, et ce quelle que soit l'opération de condensation utilisée, ce qui illustre bien la correspondance que nous avons établie à la section 2.3.3, page 59, entre la s_rigidité et la caractérisation structurelle de la rigidité générique en 3D par les conditions de Laman.

Dans les chapitres qui suivent, nous proposons de revenir sur les problèmes posés par une approche géométrique structurelle. Nous définissons le concept de *degré de rigidité* qui nous permettra d'introduire une nouvelle caractérisation plus tout à fait structurelle de rigidité, la *rigidité structurelle étendue*. Cette nouvelle forme de rigidité tire parti des informations géométriques pour affiner son analyse structurelle. Nous montrerons que, s'il s'agit toujours d'une approximation ne correspondant pas parfaitement à la rigidité, elle permet cependant de lever certaines des différences qui existent entre la s_rigidité et la rigidité d'un GCSP.

Nous définissons alors une nouvelle méthode de rigidification suivant le schéma de la méthode HLS mais basée sur notre nouvelle caractérisation de la rigidité et utilisant de nouveaux opérateurs de fusion, d'extensions et de condensation. Enfin, nous proposons une phase d'assemblage basée sur la résolution par intervalles.

Chapitre 6

Limites des approches géométriques structurelles

Sommaire

Dans ce chapitre, nous tentons d'apporter des réponses aux problèmes posés par l'emploi d'une méthode de décomposition géométrique structurelle. Ces méthodes ont en commun d'utiliser la propriété de rigidité des sous-GCSP et de procéder à l'assemblage de sous-GCSP par déplacements. Elles posent deux problèmes majeurs :

– Caractérisation de la rigidité : les méthodes structurelles se basent généralement sur une caractérisation similaire à la s_rigidité (cf. définition 32, page 60) qui peut être

trompée par la présence de R-redondances, de singularités ou d'inconsistances dans le GCSP.
- Possibilité d'assembler : pour pouvoir assembler deux sous-GCSP, il faut qu'ils partagent suffisamment d'objets géométriques pour que la position de ces objets dans les repères locaux de ces 2 sous-GCSP permette de déterminer un déplacement. Nous avons vu à la section 4.5.1, page 116, que cela revient à résoudre le *problème du repère local*, c-a-d. à décider si un ensemble d'objets géométrique permet la définition d'un repère local.

Les méthodes géométriques structurelles travaillent ainsi souvent sous l'hypothèse d'absence de singularités et de R-redondances. Or ces deux hypothèses sont irréalistes : un ingénieur utilisant un outil de CAO n'aura pas toujours conscience d'introduire une R-redondance, et les singularités sont omniprésentes dans les GCSP issus d'applications réelles (alignement de points, mécanismes ayant des pièces de même longueur, ...).

Nous commençons ce chapitre en discutant du problème des singularités, des R-redondances et des inconsistances. Nous expliquons que les contraintes booléennes[1] permettent de spécifier les singularités d'un GCSP de façon explicite. Nous proposons quelques techniques de détection de R-redondances et d'inconsistances.

Nous proposons [Jermann *et al.*, 2002a; 2002b] alors le concept de *degré de rigidité*. Intuitivement, le degré de rigidité définit le nombre de degrés de liberté qu'un sous-GCSP doit posséder pour être rigide.

Ce concept nous permet de répondre aux deux problèmes majeurs des méthodes de décomposition géométrique structurelle :

- Le degré de rigidité nous permet de définir la *rigidité structurelle étendue*, une nouvelle caractérisation plus tout à fait structurelle de la rigidité. Cette nouvelle caractérisation reste néanmoins une approximation dans le cas général : si elle tient compte des singularités introduites par les contraintes booléennes d'un GCSP, elle peut toujours être trompée par d'autres singularités, des R-redondances et des inconsistances. Cependant, nous n'avons pas trouvé de cas de GCSP générique, non-R-redondant et consistant où notre caractérisation se trompe, et nous pensons qu'elle doit, dans ce cas, correspondre très fidèlement à la rigidité.
- Le degré de rigidité permet de répondre au problème du repère local, et ainsi de déterminer si l'assemblage de 2 sous-GCSP est possible ou pas.

Nous verrons que le calcul du degré de rigidité est difficile dans le cas général, mais qu'il peut s'avérer simple dans certaines classes de GCSP ; d'autre part, ce calcul ne sera pas toujours nécessaire dans notre méthode de rigidification récursive présentée aux chapitres suivants.

[1] Rappelons que les contraintes booléennes sont des contraintes géométriques sans paramètre valué, comme l'incidence, le parallélisme, *etc.*

6.1 Singularités, R-redondances et inconsistances

Nous avons à plusieurs reprises dans ce mémoire évoqué les problèmes posés par la présence de singularités, de R-redondances ou d'inconsistances : dès lors que l'on considère une approche structurelle, la caractérisation de la rigidité peut être trompée si le GCSP contient des contraintes singulières, R-redondantes ou inconsistantes.

Dans cette section, nous allons caractériser plus formellement ces 3 types de contraintes et indiquer les moyens de déterminer si un GCSP contient de telles contraintes.

6.1.1 Singularités

La terme de singularité a plusieurs significations :
- En algèbre, un système de n équations est singulier s'il permet de déterminer plus de n inconnues.
- En géométrie, un GCSP est singulier si ses solutions sont des configurations géométriques singulières (cf. définition 20, page 37).

Les singularités algébriques posent problème dans le compte des DDL d'un GCSP. En effet, on ne peut pas compter les DDL des contraintes et des objets en comptant simplement le nombre d'équations qui les représentent (cf. section 2.3.2, page 55).

Les singularités géométriques posent problème pour déterminer si un GCSP est rigide, ou si un ensemble d'objets géométriques permet de définir un repère local. En effet, une configuration d'un GCSP peut être rigide tout en étant sous-s_rigide s'il s'agit d'une configuration singulière (cf. table 2.4, page 61). D'autre part, un ensemble d'objets géométriques en configuration singulière ne permet pas de définir tous les déplacements de l'espace géométrique et ne peut définir un repère ou servir pour un assemblage (cf. section 4.5.1, page 116).

Nous allons voir que les contraintes booléennes permettent de représenter explicitement les singularités d'un GCSP. Il suffit donc de tenir compte de ces contraintes dans les méthodes de décomposition géométrique structurelle pour régler les problèmes liés aux singularités.

Nous proposons à la section 6.2 le concept de *degré de rigidité* qui permet de prendre en compte les contraintes booléennes pour caractériser la rigidité et pour répondre au problème du repère local.

Contraintes booléennes et degrés de liberté

Les contraintes booléennes (incidences, parallélismes, égalité de distances ou d'angles, ...) sont omniprésentes dans les constructions humaines. Il est important de les distinguer des contraintes valuées car elles ne retirent pas le même nombre de DDL, et elles permettent de représenter les singularités d'un GCSP de façon explicite.

Une solution souvent proposée par les méthodes de décomposition géométrique consiste à modéliser les contraintes booléennes par des contraintes valuées associées à des valeurs particulières : l'incidence est une distance valant 0, le parallélisme est un angle valant 0,

etc.. Nous avons expliqué à la section 2.3.2, page 56, que cette solution rend le calcul des degrés de liberté faux, puisque les contraintes booléennes ne retirent pas toujours le même nombre de DDL que les contraintes valuées correspondantes. Par exemple, une contrainte de distance entre 2 points en 2D retire 1 DDL, alors qu'une contrainte d'incidence entre 2 points en 2D retire 2 DDL.

Ceci s'explique par le fait que la contrainte valuée permettant de représenter une contrainte booléenne est généralement singulière (algébriquement). Par exemple, l'équation $(x_A - x_B)^2 + (y_A - y_B)^2 = 0$, représentation de la contrainte d'incidence entre deux points A et B en 2D par la contrainte Distance(A, B)=0, constitue une singularité algébrique : elle permet de fixer 2 inconnues ; on peut s'en convaincre en constatant que cette contrainte d'incidence peut aussi s'écrire comme un système de 2 équations non algébriquement redondant (permettant donc de fixer 2 inconnues) : $(x_A = x_B) \wedge (y_A = y_B)$.

Représenter les contraintes booléennes par des contraintes valuées singulières peut donc fausser le compte des DDL dans un GCSP ; or le calcul des degrés de liberté est utilisé dans la caractérisation structurelle de la rigidité d'un GCSP. Il est donc important de bien distinguer les contraintes booléennes des contraintes valuées afin de ne pas introduire d'erreur dans la s_rigidité.

Elimination algébriste des singularités

Une solution parfois envisagée par les algébristes pour éliminer les singularités d'un système d'équations consiste à le perturber afin d'obtenir un système *similaire* mais générique [Emiris and Canny, 1995].

Cette solution intéressante pose cependant plusieurs problèmes lorsque l'on souhaite l'utiliser dans un GCSP. Pour s'en convaincre, considérons le GCSP bien-rigide en 2D constitué de deux droites A et B et des contraintes Parallélisme(A, B) et Distance(A, B). La perturbation du système d'équations correspondant engendre le GCSP en 2D constitué de deux droites A' et B' et des contraintes Angle(A', B') et Distance(A', B') ; en effet, la contrainte de distance reste une contrainte de distance, mais lorsque le membre de droite d'une contrainte de parallélisme change, celle-ci devient une contrainte d'angle.

Ceci pose deux problèmes :

- Du point de vue du compte des DDL, nous venons d'expliquer que les contraintes booléennes ne retirent généralement pas le même nombre de DDL que les contraintes valuées ; on peut cependant pratiquer le compte des DDL sur le système non-perturbé afin d'éviter ce genre de problèmes.
- Du point de vue géométrique, le GCSP perturbé n'a pas de sens : deux droites formant un angle ne peuvent se trouver à distance prescrite ; la notion même de distance n'a plus de sens ! Alors, la résolution du GCSP perturbé ne satisfera certainement pas des relations *très proches* de celles du système original.

Il nous semble donc préférable de traiter correctement les singularités à l'aide des contraintes booléennes afin de conserver une sémantique géométrique claire.

Contraintes booléennes et rigidité

Par nature, les contraintes booléennes introduisent des singularités géométriques dans un GCSP. En effet, un cas particulier des singularités géométriques sont les configurations non-générales (cf. définition 18, page 35) : un ensemble d'objets est en configuration non-générale s'il ne remplit pas un espace géométrique de dimension maximale, c-a-d. qu'il existe une autre configuration de ces objets qui permet de décrire un espace géométrique de dimension strictement supérieure.

A la section 2.1.4, page 35, nous donnions des exemples de configurations non-générales à base de points : en 3D, un ensemble de 3 points est en configuration non-générale s'ils sont confondus ou alignés. Ces configurations singulières de 3 points correspondent à des contraintes booléennes : la contrainte d'incidence permet de faire coïncider des points ou de les rendre alignés en les plaçant tous sur une même droite.

La présence d'une contrainte booléenne dans un GCSP implique donc que toutes les configurations-solutions de ce GCSP seront singulières. Dès lors, proposer des méthodes de décomposition géométrique travaillant sous hypothèse de généricité du GCSP semble vain, les contraintes booléennes étant nécessaires dans la plupart des applications (architecture, CAO, mécanique, ...).

Traiter les singularités à l'aide des contraintes booléennes

En autorisant l'utilisation des contraintes booléennes, on peut se placer dans l'hypothèse où toute singularité est décrite explicitement dans les GCSP. En effet, puisque l'on distingue contraintes booléennes et contraintes valuées, nous pouvons interdire d'utiliser des contraintes valuées algébriquement singulières ; d'autre part, toutes les singularités souhaitables dans les solutions d'un GCSP peuvent être mentionnées explicitement sous forme de contraintes booléennes, en définissant de nouvelles contraintes si besoin est.

Les problèmes majeurs posés par les méthodes de décomposition géométrique structurelle sont liés au fait qu'elles ne traitent pas les contraintes booléennes de façon adéquate : ces contraintes introduisent des singularités de façon explicite (comme nous venons de le voir), mais la plupart des méthodes structurelles ignorent cette information. Il doit cependant être possible d'en tenir compte ; pour cela il faudrait définir une caractérisation de rigidité utilisant les contraintes booléennes et pas seulement un compte des degrés de liberté des GCSP.

Dans la suite de ce chapitre, nous allons voir que si les contraintes booléennes sont prises en compte, il est alors possible de définir une nouvelle caractérisation de la rigidité qui n'est pas trompée par ces singularités explicites. Il est également possible de répondre au problème du repère local et ainsi déterminer si l'assemblage de 2 sous-GCSP est possible. Cette caractérisation passe par la définition du concept de *degré de rigidité* qui tient compte des contraintes booléennes du GCSP (cf. section 6.2, page 175).

6.1.2 Redondances

Les redondances se définissent aussi de différentes façons :

- En algèbre, un système de n équations est A-redondant s'il permet de fixer moins de n inconnues. On dit aussi qu'une équation est redondante dans un système d'équations si elle ne permet pas de fixer de nouvelle variable du système. Nous avons vu à la section 1.4.2 (page 21) que cette définition est plus générale que le sens classique de redondance. En effet, pour nous, une contrainte peut être redondante même si elle invalide certaines solutions du GCSP considéré.
- En géométrie, un GCSP rigide est R-redondant (géométriquement redondant) s'il est possible de retirer une contrainte tout en conservant la rigidité du GCSP. On dit aussi qu'une contrainte est redondante dans un GCSP si les propriétés géométriques qu'elle impose (incidence, non incidence, parallélisme, ...) sont déductibles des contraintes du GCSP. Là encore, la R-redondance est plus générale que la notion classique de redondance : une contrainte R-redondante ne change pas la rigidité, mais peut éliminer des solutions du GCSP considéré.

Redondances algébrique et géométrique recouvrent en fait une seule et même notion : un ensemble de contraintes/équations n'est pas indépendant et ne permet pas de déterminer autant d'inconnues qu'il contient d'équations. On dira alors qu'une contrainte/équation est redondante si elle appartient à un ensemble de contraintes/équations redondantes.

R-redondance partielle

Le fait de travailler au niveau objets-contraintes au lieu du niveau variables-équations induit une autre forme de R-redondance : la *R-redondance partielle*. Une contrainte géométrique est partiellement R-redondante si au moins l'une des équations qui la représente est algébriquement redondante alors qu'au moins une autre équation qui la représente n'est pas algébriquement redondante.

Nous illustrons le concept de R-redondance partielle sur le GCSP représentant un rectangle en 3D :

- Objets : 4 points A, B, C et D et 4 droites (AB, BC, CD et AD).
- Contraintes : Incidence(A, AB), Incidence(B, AB), Incidence(B, BC), Incidence(C, BC), Incidence(C, CD), Incidence(D, CD), Incidence(A, AD), Incidence(D, AD), Distance(A, B), Distance(B, C), Parallélisme(AB, CD), Parallélisme(BC, AD), Angle$(AB, AD)=\frac{\pi}{2}$, Angle$(AB, BC)=\frac{\pi}{2}$, Angle$(BC, CD)=\frac{\pi}{2}$.

Ce GCSP est représenté à la figure 6.1. On s'aperçoit que l'une des 2 contraintes de parallélisme est R-redondante vis à vis de ce système :

Parallélisme$(AB, CD) \wedge$ Angle$(AB, AD)=\frac{\pi}{2} \wedge$ Angle$(AB, BC)=\frac{\pi}{2} \wedge$ Angle$(BC, CD)=\frac{\pi}{2}$ $\Rightarrow$ Parallélisme(BC, AD).

Il est plus difficile de se rendre compte que la contrainte Parallélisme(AB, CD) est partiellement R-redondante. Tout d'abord, on peut constater que si elle est retirée, le GCSP n'est plus rigide : on peut *tirer* le point D hors du plan défini par les points A, B et C tout en maintenant les contraintes de perpendicularité. Cette contrainte n'est donc pas

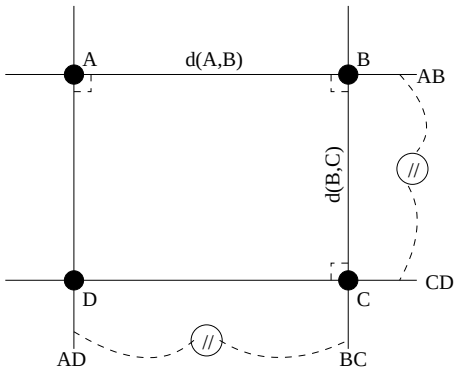

FIG. 6.1 – Contraintes R-redondantes et partiellement R-redondantes

entièrement R-redondante car elle est nécessaire à la rigidité du GCSP. En revanche, elle est partiellement une conséquence des autres contraintes ; afin de le comprendre, on peut constater que la contrainte Angle$(BC, CD)=\frac{\pi}{2}$ est, elle, R-redondante :

Parallélisme$(AB, CD) \wedge$ Angle$(AB, BC)=\frac{\pi}{2} \Rightarrow$ Angle$(BC, CD)=\frac{\pi}{2}$

Or une contrainte d'angle ne retire qu'un seul DDL, alors que la contrainte de parallélisme retire 2 DDL en 3D. La R-redondance du GCSP a pu être éliminée en retirant 1 seul DDL, donc la contrainte Parallélisme(AB, CD) n'était que partiellement R-redondante.

Problèmes posés par les R-redondances

Proposer des méthodes de décomposition géométrique travaillant sous hypothèse d'absence de R-redondances est peu réaliste : par exemple, les GCSP issus de d'applications en CAO sont produits par des ingénieurs qui n'ont pas toujours conscience d'introduire des contraintes R-redondantes dans leur modélisation d'un GCSP. De plus, ils peuvent sciemment introduire de telles contraintes afin de filtrer l'ensemble des solutions d'un GCSP en imposant des restrictions ; en effet, une contrainte R-redondante ne change pas la rigidité du GCSP, mais elle peut éliminer certaines de ses solutions. De telles contraintes, tout en étant R-redondantes, ne sont pas redondantes (cf. définition 11, page 21) et peuvent donc être utiles : elles doivent être prises en compte dans la résolution du GCSP.

Si l'on souhaite utiliser une méthode de décomposition structurelle, les R-redondances posent pourtant un problème : elles faussent le compte du nombre de DDL dans un GCSP car elles retirent des DDL qui sont en réalité déjà retirés par d'autres contraintes (cf. section 2.3.2, page 58). La caractérisation structurelle de la rigidité s'en trouve donc faussée.

Si l'on est capable d'identifier les contraintes R-redondantes, on peut les retirer[2] du GCSP et ainsi utiliser une méthode de décomposition géométrique structurelle sans crain-

[2]On ne peut pas retirer les contraintes partiellement R-redondantes car elles sont utiles à la rigidité du GCSP ; on peut en revanche abaisser le nombre de DDL qu'elles fixent afin de prendre en compte leur R-redondance partielle.

dre que les R-redondances ne faussent la caractérisation structurelle de la rigidité. Lors de la résolution du GCSP décomposé, il faut alors être capable de prendre en compte ces contraintes afin de les respecter. Nous proposons à la section 8.3.4, page 236, plusieurs façons de tenir compte des contraintes R-redondantes lors de la phase d'assemblage de notre nouvelle méthode de rigidification récursive.

Nous présentons à la section 6.1.4 quatre approches pour la détection et l'élimination des R-redondances dans un GCSP.

6.1.3 Inconsistances

Enfin, un GCSP est inconsistant s'il n'admet aucune solution, c-a-d. qu'il est sur-contraint. Une contrainte c est dite inconsistante avec un GCSP $S = (O, C)$ qui ne l'est pas si $S' = (O, C \cup \{c\})$ est inconsistant.

La présence d'une inconsistance fait systématiquement basculer un GCSP dans la catégorie des GCSP sur-rigides. Il est donc encore plus difficile d'identifier les inconsistances que les R-redondances. La seule solution générale consiste à résoudre le GCSP afin de déterminer si oui ou non, il admet des solutions.

Cependant, les quatre méthodes de détection de R-redondances que nous proposons permettent aussi d'identifier *certaines* inconsistances :

- l'approche structurelle identifie des contraintes sur-s_rigides, c-a-d. des contraintes *en trop* du point de vue de la s_rigidité ; elle ne permet cependant pas de déterminer s'il s'agit de R-redondances ou d'inconsistantes, et peut donc identifier certaines inconsistances ;
- l'approche géométrique permet de déterminer certaines inconsistances géométriques : si les règles qu'elle utilise permettent d'inférer la propriété géométrique $\neg c$, alors la contrainte géométrique c est contradictoire et donc inconsistante ;
- l'approche formelle peut établir l'inconsistance dans certains cas : si le GCSP est sur-contraint dans les complexes, alors il l'est aussi forcément dans les réels ;
- l'approche probabiliste, à l'instar de l'approche structurelle, identifie des contraintes *en trop* sans déterminer s'il s'agit de R-redondances ou d'inconsistances.

Il est aussi possible de ne pas tenir compte du problème de l'inconsistance : une inconsistance fausse effectivement la caractérisation structurelle de la rigidité. Cependant, un GCSP inconsistant n'admet pas de solution, et ce, que sa décomposition soit correcte ou pas. Produire une décomposition correcte n'a alors que peu d'importance.

Qui plus est, la décomposition peut permettre de cerner où se situe une inconsistance : si l'un des sous-systèmes produits n'admet jamais de solution, alors il est inconsistant en lui-même et on peut le retourner à un utilisateur comme explication de l'inconsistance du GCSP. Il peut donc être utile de produire une décomposition, même incorrecte, d'un GCSP inconsistant.

6.1.4 Détection de R-redondances et d'inconsistances

Il existe de nombreuses façons de détecter des contraintes R-redondantes ou inconsistantes. Nous présentons quatre approches qui nous semblent intéressantes et complémentaires : une approche structurelle, une approche à base de règles, une approche formelle et enfin une approche probabiliste.

Ces approches suivent toutes le schéma de l'algorithme `Eliminer-Sur-S_Rigidités` (page 157) : les contraintes sont introduites une à une dans le GCSP ; à chaque introduction, on teste si une R-redondance ou une inconsistance apparaît dans le GCSP. Si tel est le cas, c'est que la dernière contrainte introduite en est la cause. Elle est alors retirée et stockée dans un ensemble séparé de contraintes qui sera utilisé lors de la résolution du GCSP.

La différence entre les quatre approches réside donc dans la manière dont est testée l'apparition d'une R-redondance ou d'une inconsistance. Dans les paragraphes qui suivent, nous décrivons plus en détail ces quatre approches et discutons leurs avantages et limites respectives. Nous verrons aussi qu'il est possible et profitable d'utiliser ces quatre approches conjointement.

Approche structurelle

Nous avons expliqué à la section 5.6.1, page 157 comment l'algorithme `Dense` peut être utilisé pour identifier et éliminer les contraintes qui introduisent une sur-s_rigidité dans un GCSP (cf. algorithme 15, page 157) : il suffit de distribuer une surcharge valant $\frac{d(d+1)}{2}$ au lieu de $\frac{d(d+1)}{2}+1$ dans l'algorithme `Dense`.

Une sur-s_rigidité représente le fait qu'il y a une contrainte en trop vis à vis du compte des degrés de liberté. Les contraintes retirées par l'algorithme `Eliminer-Sur-S_Rigidités` sont alors des contraintes R-redondantes ou inconsistances. Il est impossible de faire la distinction entre ces deux catégories de contraintes structurellement.

De plus, cette méthode structurelle d'identification de R-redondances peut se tromper : la s_rigidité est trompée par la présence de singularités. Nous avons présenté à la section 2.3.3, page 62, des exemples de GCSP possédant moins de $\frac{d(d+1)}{2}$ DDL mais étant pourtant rigides. Par exemple, un simple segment en 3D constitue un GCSP sur-s_rigide ; toutes les contraintes de distance portant entre des points en 3D sont donc retirées par l'algorithme `Eliminer-Sur-S_Rigidités` !

Nous proposons au chapitre suivant l'algorithme `ES_Rigide` (page 202), basé sur notre nouvelle caractérisation de la rigidité et qui permettra de lever ce type de problèmes. Il suffit alors de remplacer l'appel à la fonction `Dense` par un appel à `ES_Rigide` dans l'algorithme `Eliminer-Sur-S_Rigidités` pour obtenir un algorithme correct permettant de détecter structurellement des contraintes *sur-es_ rigides*.

Toutefois, cet algorithme ne permet toujours pas de distinguer R-redondances et inconsistances. De plus, il demeure des R-redondances qui ne peuvent pas être détectées structurellement, comme celle présente dans le GCSP présenté à gauche dans la figure 2.10 : la fameuse *double-banane* reste hors de portée des méthodes structurelles.

L'approche structurelle pour la détection de R-redondances a une complexité en temps

polynomiale, (sous réserve que l'algorithme `ES_Rigide` soit lui-même de complexité polynomiale).

Approche à base de règles

L'approche à base de règles consiste à utiliser une *base de faits* représentant des propriétés géométriques et une *base de règles* contenant des théorèmes géométriques.

Les contraintes d'un GCSP sont introduites une à une dans la base de faits. La *base de règles* contient des théorèmes de géométrie qui permettent d'inférer des prédicats géométriques supplémentaires à partir de la base de faits.

Nous avons vu que l'on peut considérer que les contraintes valuées sont toujours génériques (cf. section 6.1.1, page 165). Or des contraintes valuées génériques ne permettent jamais l'apparition d'un théorème de géométrie métrique (Pythagore, Thalès, triangles plats, *etc.*). Ainsi, la présence de contraintes valuées entre des objets permet de déduire que ces objets sont en configuration générique. Ceci peut être représenté par l'introduction, dans la base de faits, de la négation de contraintes booléennes ; par exemple, la présence des contraintes Distance(A, B), Distance(A, C) et Distance(B, C) entre trois points A, B et C implique que ces trois points ne peuvent pas être alignés ; ces contraintes permettent donc l'introduction de la relation $\neg$ Alignés(A, B, C) dans la base de faits. D'autre part, chacune de ces 3 contraintes permet d'assurer que les points A, B et C sont deux à deux distincts, ce qui se traduit par l'introduction dans la base de faits de 3 relations géométriques : $\neg$ Incidence(A, B), $\neg$ Incidence(A, C) et $\neg$ Incidence(B, C).

Le mécanisme d'inférence utilise les théorèmes de la base de règles pour compléter la base de faits à chaque fois qu'une nouvelle contrainte est introduite dans celle-ci. C'est un algorithme de point fixe qui génère toutes les déductions possibles à partir de la base de règles.

Avant d'introduire une nouvelle contrainte c dans la base de faits, on vérifie si celle-ci ne la contient pas déjà. Si tel est le cas, c'est que la contrainte c est une R-redondance. On peut vérifier aussi que la base de faits ne contient pas $\neg c$, car la contrainte c serait alors inconsistante. Dans un cas comme dans l'autre, c est retirée du GCSP ; dans le cas où c est R-redondante, elle peut être stockée dans un ensemble des contraintes R-redondantes.

Cette approche n'est pas complète : il est impossible, dans le cas général, de proposer un ensemble fini de théorèmes capable de produire toutes les inférences possibles. De plus, si certaines règles d'inférences produisent des objets géométriques nouveaux, l'algorithme qui applique toutes les règles jusqu'à un point fixe peut alors ne pas terminer ; ce problème était aussi évoqué pour l'un des agents à base de règles proposé par Mathis *et al.* (cf. section 4.4.3, page 111).

Cependant, en pratique, l'utilisation de cette méthode est intéressante car elle est très bien adaptée au domaine et peut éliminer de façon efficace les R-redondances les plus courantes. Elle peut de plus être complète pour certaines classes de GCSP.

Par exemple, dans la classe de GCSP ({Points, Droites},{Distances, Angles, Parallélismes}) en 2D, la seule règle suivante permet de vérifier que toute contrainte de parallélisme n'est pas une conséquence du reste du GCSP :

Droite ?$(D_1) \wedge$ Droite ?$(D_2) \wedge$ Droite ?$(D_3) \wedge$ Parallélisme$(D_1, D_3) \wedge$ Parallélisme(D_2, D_3) $\Rightarrow$ Parallélisme(D_1, D_2)

En revanche, dès que l'on introduit la contrainte d'incidence dans cette classe, la méthode n'est plus complète ; par exemple, le théorème de Pappus, illustré en figure 6.2, peut entraîner l'incidence d'un point à une droite.

Théorème 13 Théorème de Pappus
Soient trois points A, B et C sur une droite D_1 et trois points E, F et G sur une droite D_2. Soient H le point se trouvant à l'intersection des droites (AF) et (BE), I le point se trouvant à l'intersection des droites (AG) et (CE), et J le point se trouvant à l'intersection des droites (CF) et (BG). Alors, H, I et J sont alignés.

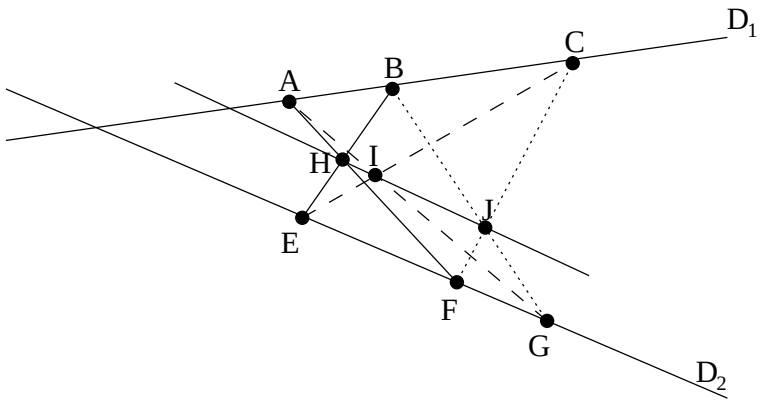

FIG. 6.2 – Illustration du théorème de Pappus

Il peut aussi exister des théorèmes de taille arbitraire impliquant un parallélisme ou une incidence, même si l'on considère que les contraintes valuées sont génériques.

Cette méthode ne peut pas non plus identifier les contraintes partiellement R-redondantes, car elles ne peuvent pas être entièrement déduites depuis les autres contraintes du GCSP.

Elle est de complexité polynomiale si la base de règles n'engendre pas de nouveaux objets. Nous ne proposons pas de base de règles détaillée car il nous semble qu'une base de règles doit être adaptée aux GCSP d'une classe d'application particulière : une règle utile pour un type de GCSP donné peut être totalement superflue et inutile dans une autre classe de GCSP ; il serait alors dommage et coûteux de produire des inférences inutiles en utilisant une base de règles inadaptée.

Approche formelle

Pour déterminer si une contrainte est une conséquence d'un ensemble de contraintes, on peut utiliser les techniques de démonstration automatique de théorèmes géométriques,

basées sur les méthodes formelles présentées à la section 3.1. L'utilisation de telles méthodes a été illustrée pour la caractérisation algébrique de la rigidité (cf. section 2.1.2, page 29).

Les contraintes du GCSP sont toujours introduites une à une, et chaque nouvelle contrainte est testée pour déterminer si elle est R-redondante ou inconsistante avec les contraintes déjà présentes dans le GCSP.

En travaillant au niveau des équations et non des contraintes géométriques, cette approche peut permettre d'identifier les contraintes partiellement R-redondantes : il suffit d'introduire les équations modélisant une contrainte une par une au lieu de les introduire toutes en une seule fois.

Les méthodes de démonstration automatique ont cependant l'inconvénient de n'être applicables que si toutes les équations du GCSP sont algébriques. De plus, leur complexité en temps et en espace rend leur utilisation difficile en pratique pour des GCSP de grande taille.

Approche probabiliste

L'approche probabiliste a été proposée initialement dans [Lamure and Michelucci, 1998]. Cette approche, déjà brièvement exposée en section 2.1.7, repose sur un calcul probabiliste du jacobien du GCSP. Le calcul du jacobien permet de déterminer si un GCSP est de rang maximal ou pas, c-a-d. s'il est R-redondant ou pas : si le jacobien est nul, le rang n'est pas maximal, sinon il l'est. Tout comme l'approche structurelle, cette approche ne permet pas de distinguer R-redondances et inconsistances.

Le calcul du jacobien est effectué de façon probabiliste car la fonction du jacobien ne peut généralement pas être produite formellement : on peut calculer formellement la matrice jacobienne d'un GCSP, mais son déterminant, le jacobien, peut comporter un nombre exponentiel de termes.

L'idée de Lamure et Michelucci consiste à évaluer la matrice jacobienne en certaines configurations aléatoires des objets du GCSP. En une configuration donnée, le jacobien peut alors être calculé numériquement très rapidement. Si sa valeur est non nulle, alors le jacobien du GCSP est non identiquement nul, et le GCSP est *génériquement* de rang maximal, c-a-d. qu'il ne contient pas de contraintes R-redondantes ou inconsistantes si elles sont toutes génériques.

Ce résultat repose sur un argument probabiliste : le jacobien est soit identiquement nul, soit il ne s'annule que pour certaines configurations particulières des objets géométriques. Le sous-espace des configurations du GCSP qui annulent le jacobien est alors de dimension inférieure à la dimension de l'espace de toutes les configurations des objets du GCSP. La probabilité qu'une configuration tirée aléatoirement dans l'espace de toutes les configurations annule le jacobien est alors *nulle*[3].

Cette approche présente deux inconvénients :

– En pratique, il est impossible de procéder à un véritable tirage aléatoire de chaque paramètre de chaque objet géométrique sur les réels.

[3]Tout comme la probabilité de tirer un point sur une droite donnée dans le plan $\mathbb{R}^2$, ou encore de tirer un nombre entier dans $\mathbb{R}$ est nulle...

– La fiabilité des calculs numériques mis en jeu ne permet pas toujours d'assurer que la valeur obtenue est nulle (ou non nulle). On peut procéder à un calcul par intervalles, plus sûr, mais on ne peut alors décider que le jacobien est non nul que si 0 n'appartient pas à son intervalle de valeurs.

Pour pallier ces deux problèmes, Lamure et Michelucci proposent d'effectuer plusieurs tirages aléatoires : il suffit que l'un d'entre eux puisse être certifié non nul pour que le jacobien soit non identiquement nul.

L'approche est intéressante et rapide, mais elle comporte une limitation importante : la génération de configurations aléatoires pour les objets géométriques du GCSP ne peut prendre en compte les contraintes booléennes. En effet, les paramètres de deux droites tirés aléatoirement ne placeront *jamais* ces droites en position parallèle, même si la contrainte de parallélisme est explicitée dans le GCSP.

Le calcul du jacobien est alors effectué sur des configurations génériques qui n'ont pas les propriétés des configurations-solutions du GCSP. En particulier, il est possible que le jacobien soit non-identiquement nul, mais s'annule en toutes les solutions du GCSP, indiquant alors la présence d'une R-redondance ou d'une inconsistance dans le GCSP due aux contraintes booléennes du GCSP.

Nous avons imaginé qu'en effectuant des tirages pseudo-aléatoires de configurations satisfaisant uniquement les contraintes booléennes du GCSP, il serait possible de remédier à ce problème. Ainsi, le jacobien serait toujours évalué en des points vérifiant les relations géométriques induites par les contraintes booléennes. Nos recherches préliminaires en ce sens sont malheureusement assez négatives : la génération pseudo-aléatoire que nous avons imaginée fait perdre la propriété probabiliste de l'approche qui devient une simple heuristique permettant parfois de détecter des R-redondances, et ce moyennant un coût important.

Le problème qui se pose est le suivant :

Problème 2 Problème du système paramétré
Donnée : *Un CSP $P = (X, D, C)$ et un sous-ensembles $X' \subset X$ de variables qui sont des paramètres, c-a-d. $P' = (X \setminus X', D, C)$ est structurellement bien-contraint.*
Question : *Existe-t-il une instanciation de X' tel que P' est bien-contraint ?*

En effet, si l'on souhaite résoudre le GCSP réduit à ses contraintes booléennes, on se trouve confronté à un CSP structurellement sous-contraint. Il devient alors nécessaire de sélectionner un certain nombre de variables qui seront des paramètres (1) et d'affecter une valeur à ses paramètres (2) afin de pouvoir résoudre les variables restantes de sorte que les contraintes booléennes soient toutes satisfaites. Le premier sous-problème peut être résolu en temps polynomial par une approche de type couplage maximum. Le second correspond au problème 2 que nous venons de formuler : trouver une instanciation partielle qui rende le CSP satisfiable nous semble être un problème difficile.

Une étude plus approfondie de cette méthode appartient donc aux perspectives de la thèse.

Il existe d'autres utilisations plus poussées de la méthode probabiliste, qui permettent d'étudier plus précisément les types de mouvements génériquement admis par un GCSP.

Cette approche est similaire à l'approche algébrique pour la vérification de la rigidité infinitésimale (cf. section 2.1.3, page 32).

Limites de la détection

On dispose donc de plusieurs approches possibles pour la détection des R-redondances et inconsistances dans un GCSP, chacune avec ses limites et propriétés propres. L'utilisation conjointe de plusieurs méthodes pourrait être la meilleure solution, apte à tirer parti des avantages de chacune. Par exemple, il peut être intéressant de commencer par une détection structurelle polynomiale, suivie d'une détection à base de règles bien adaptée au domaine et en temps polynomial ; cette dernière a l'avantage de produire toutes les déductions possibles dans un GCSP. Si elle termine en ayant produit toutes les relations possibles entre tous les objets du GCSP, on est alors assuré que la détection a été complète. Sinon, on peut avoir recours à la méthode formelle si l'on souhaite pousser la détection des contraintes R-redondantes ou inconsistantes plus loin.

Il existe cependant des R-redondances qui demeurent hors de portée de toutes ces méthodes de détection *a priori*. En effet, il est possible qu'une contrainte s'avère R-redondante pour certaines configurations d'un GCSP uniquement, mais pas pour toutes les configurations d'un GCSP. Nous verrons un exemple de R-redondance de ce type à la section 6.2.1, page 176. Ces contraintes qui deviennent parfois R-redondantes ne peuvent pas être retirées du GCSP, car il existe des configurations pour lesquelles elles ne sont pas R-redondantes mais nécessaires.

Si un GCSP contient ce type de contraintes, nous verrons qu'une planification unique du GCSP ne peut être satisfaisante, car la décomposition peut s'avérer incorrecte lors de la phase d'assemblage dans certains sous-espaces de recherche, tout en étant correcte dans d'autres. Ceci constitue une limite forte des approches à base de décomposition.

Conclusions

Nous avons à nouveau signalé les problèmes posés par la présence de singularités, R-redondances et inconsistances pour la décomposition géométrique structurelle de GCSP. Ces 3 types de contraintes faussent la caractérisation structurelle de la rigidité, et les singularités sont importantes pour décider si un ensemble d'objets permet la définition d'un repère local.

Nous avons expliqué comment les contraintes booléennes permettent d'introduire de façon explicite les singularités souhaitées d'un GCSP ; ceci permet de travailler sous l'hypothèse que les contraintes valuées sont toujours génériques.

Nous avons présenté quatre méthodes de détection de R-redondances et d'inconsistances, dont l'utilisation simultanée permet d'éliminer la plupart des R-redondances et inconsistances.

Nous avons aussi expliqué que les inconsistances représentent un problème sémantique *si grave* dans un GCSP que la décomposition n'a de toute façon pas de sens : peu importe alors qu'elle soit incorrecte, puisque le GCSP n'aura de toute façon pas de solution.

Nous allons maintenant introduire le concept de degré de rigidité qui permettra de tenir compte des contraintes booléennes et donc des singularités du GCSP. Ce concept nous permettra alors de définir une nouvelle caractérisation de la rigidité, et de répondre au problème du repère local.

6.2 Degré de rigidité

Le degré de rigidité d'un ensemble d'objets géométriques O' dans un GCSP $S = (O, C)$ définit le nombre de degrés de liberté attendu du sous-GCSP $S' = (O', C')$ induit par O' s'il est bien-rigide. Pour définir le degré de rigidité, on utilise le concept de *rigidification* :

Définition 44 Rigidification d'un ensemble d'objets géométriques
Soit O un ensemble d'objets géométriques et C_R un jeu de contraintes non R-redondant *portant sur les objets de O. On dit que C_R est une* **rigidification** *de O si et seulement si le GCSP (O, C_R) est bien-rigide. On dit alors que O* **est rigidifié** *par C_R.*

Considérons par exemple l'ensemble d'objets O constitué d'un point A et d'une droite D en 3D. Cet ensemble peut être rigidifié par la contrainte Incidence(A, D) ; en effet, un point sur une droite en 3D constitue un GCSP rigide. L'ensemble O peut aussi être rigidifié par la contrainte distance(A, B) : un point à distance fixe d'une droite constitue bien un GCSP rigide en 3D. Notons que la valeur de la distance importe peu dans le fait qu'un segment est rigide. En fait, les contraintes valuées d'une rigidification peuvent prendre toute valeur générique.

On constate qu'il peut exister plus d'une rigidification possible pour un même ensemble d'objets géométriques. Qui plus est, les GCSP résultant de différentes rigidifications d'un même ensemble d'objets peuvent posséder un nombre de degrés de liberté différent. Par exemple, le point et la droite en 3D rigidifiés par une incidence constitue un GCSP ayant 5 DDL, alors que rigidifiés par une contrainte de distance, ils produisent un GCSP possédant 6 DDL.

Il existe donc un ensemble de valeurs possibles pour le degré de rigidité d'un ensemble d'objets. Cependant, on considère les ensembles d'objets géométriques *dans le contexte* d'un GCSP donné. Toutes les rigidifications ne sont pas compatibles avec le GCSP considéré. Par exemple, dans un GCSP contenant la contrainte Distance(A, B), la rigidification Incidence(A, B) n'a pas de sens puisqu'elle contredit les contraintes du GCSP. Il faut donc considérer les rigidifications d'ensembles d'objets *géométriquement consistantes* avec le GCSP qui sert de contexte à l'étude de la rigidité.

Définition 45 Consistance géométrique
Soit O un ensemble d'objets géométriques et C et C' deux ensembles de contraintes géométriques sur O. C et C' sont **géométriquement consistants**, *noté* G-consistants, *si et seulement si $I_g(C \cup C') \perp$, I_g étant un opérateur complet d'inférence géométrique.*

Intuitivement, deux ensembles de contraintes géométriques sont G-consistants s'ils n'induisent pas des propriétés géométriques contradictoires. Nous avons vu à la section 6.1.4, page 170, que les contraintes valuées peuvent être considérées comme contradictoires avec les contraintes booléennes correspondantes : Distance = ¬ Incidence, Angle = ¬ Parallélisme. D'autre part, les contraintes valuées seront toujours considérées G-consistantes entre elles indépendamment de leurs valeurs.

Il faut donc utiliser une rigidification G-consistante avec les contraintes du GCSP si l'on veut que le concept de degré de rigidité ait un sens :

Définition 46 Degré de Rigidité
Le degré de rigidité (DDR) d'un ensemble d'objets O' dans un GCSP $S = (O, C)$, noté $DDR(O', S)$, vaut $DDL(S_R)$ où $S_R = (O', C_R)$ résulte de la rigidification de O' par C_R et C_R est G-consistante avec C.

Notons que la rigidification C_R utilisée pour définir le DDR doit être G-consistante par rapport à *l'ensemble* des contraintes du GCSP, et pas uniquement par rapport aux contraintes portant sur O'. En effet, certaines relations géométriques sur les objets de O' peuvent dépendre des contraintes existant entre d'autres objets du GCSP. Par exemple, si O' est composé de deux droites D_1 et D_2 en 2D, et s'il existe par ailleurs dans le GCSP une droite D_3 liée à D_1 et D_2 par les contraintes Parallélisme(D_1, D_3) et Parallélisme(D_2, D_3), alors on peut déduire la relation booléenne Parallélisme(D_1, D_2) uniquement si l'on considère l'ensemble du GCSP et pas seulement les contraintes portant entre D_1 et D_2.

Ainsi, pour déterminer le DDR d'un ensemble d'objets géométriques, il faut produire une rigidification de cet ensemble d'objets qui soit G-consistante avec le GCSP considéré. Ceci pose deux problèmes principaux :

- Produire toutes les rigidifications d'un ensemble d'objets quelconques est un problème difficile
- S'assurer de la G-consistance de deux ensembles de contraintes est aussi un problème difficile

En fait, il existe un autre problème lié au concept même de degré de rigidité : nous avons vu qu'il peut exister plusieurs rigidifications pour un même ensemble d'objets dans un même GCSP ; comme les différentes rigidifications d'un même ensemble d'objets sont généralement contradictoires entre elles, il en existe généralement une seule G-consistante avec les contraintes du GCSP. Cependant, il peut s'avérer que même des rigidifications contradictoires entre elles s'avèrent toutes G-consistantes dans le GCSP considéré. Si cela se produit, l'ensemble d'objets peut alors disposer d'un DDR multiple. Nous allons exposer ce problème et ses raisons dans la section suivante, avant de revenir au problème du calcul du DDR.

6.2.1 Degré de rigidité multiple

La consistance géométrique de deux ensembles de contraintes établit que ces deux ensembles ne sont pas géométriquement inconsistants. Elle n'établit pas que ces deux en-

sembles sont R-redondants. Il existe alors des GCSP où deux rigidifications G-inconsistantes entre elles sont pourtant toutes deux G-consistantes avec l'ensemble des contraintes du GCSP.

Considérons le GCSP présenté à la figure 6.3. Ce GCSP en 2D est constitué de 4 points A, B, C et D et d'une droite E, liés par les contraintes :
Incidence(A, E), Incidence(B, E), Incidence(C, E), Distance(A, B), Distance(A, D), EgalitéDistance(A, B, B, C), EgalitéDistance(A, D, D, C).

Considérons maintenant le DDR des points A et C ; une paire de points en 3D peut être rigidifiée par une distance ou par une incidence. Ces deux rigidifications sont contradictoires entre elles, pourtant elles sont toutes deux G-consistantes avec le GCSP puisque celui-ci admet 2 configurations pour les points A et C, présentées toutes deux à la figure 6.3.

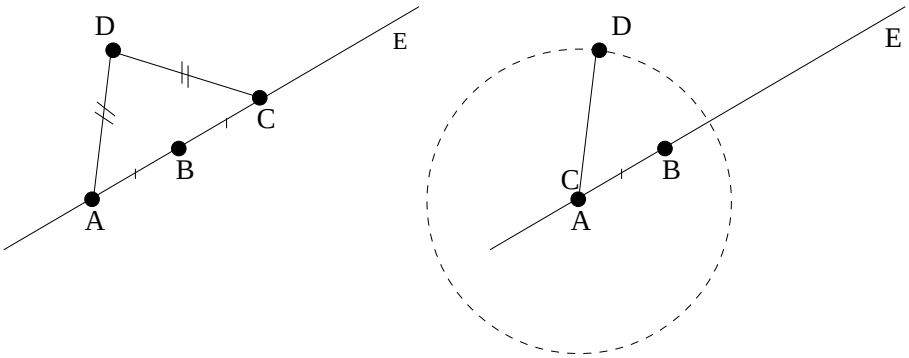

FIG. 6.3 – Exemple de GCSP à DDR multiple

Or, la rigidification par une distance implique $DDR(\{A, C\}, S) = 3$, alors que celle par une incidence implique $DDR(\{A, C\}, S) = 2$. l'ensemble $\{A, C\}$ admet donc un DDR multiple, provenant du fait que le GCSP possède des configurations où A et C sont incidents, et d'autres configurations où ils ne le sont pas.

Les ensembles d'objets à DDR multiple engendrent de multiples problèmes :

- ils peuvent changer la rigidité du GCSP ; on peut constater sur le GCSP présenté à la figure 6.3 que les configurations où les points A et C sont confondus sont sous-rigides car le point D peut tourner autour de $A = C$; en revanche, les configurations où A et C sont distincts sont rigides.
- cela signifie que des contraintes peuvent être R-redondantes dans certaines configurations et pas dans les autres ; par exemple, dans le GCSP de la figure 6.3, la contrainte EgalitéDistance(A, D, D, C) est R-redondante lorsque A et C sont confondus uniquement.

Cependant, le problème est plus général car un GCSP contenant un sous-ensemble d'objets à DDR multiple peut posséder certaines configurations-solutions bien-rigides, d'autres sous-rigides et d'autres encore sur-rigides. Dans ce cas, le GCSP n'est pas globalement rigide (cf. section 2.3.1, page 54) : il n'est pas surprenant qu'aucune méthode de décomposition ne permette de le décomposer de façon correcte.

Dans la suite de ce chapitre, nous allons donc considérer que le DDR de chaque sous-ensemble d'objets dans un GCSP est unique. Nous expliquerons dans une section séparée les problèmes que peut causer la présence de DDR multiples pour :

- le calcul du DDR d'un ensemble d'objets par les DDR-minimaux (cf. section 6.2.4, page 186),
- la caractérisation de la rigidité par la es_rigidité (cf. section 6.3.3, page 190),
- la réponse apportée au problème du repère local (cf. section 6.4.1, page 192),
- et, au chapitre suivant, la décomposition d'un GCSP.

6.2.2 Calcul du DDR : rigidifications

Comme nous l'avons déjà indiqué, le calcul du degré de rigidité d'un ensemble O d'objets géométriques dans un GCSP S revient à identifier la rigidification de O G-consistante avec S. Ceci implique de résoudre deux sous-problèmes assez difficiles :

- Identifier les rigidifications possibles de O.
- Décider si deux ensembles de contraintes sont G-consistants.

Dans cette section, nous nous intéressons au premier problème. Nous allons en fait ramener le problème de l'identification des rigidifications d'un ensemble quelconque d'objets géométriques à celui de la génération des *sous-ensembles DDR-minimaux* de cet ensemble d'objets. Pour ce faire, nous allons nous baser sur des propriétés particulières du degré de rigidité. Remarquons tout d'abord la propriété suivante :

Proposition 15
Soit $S = (O, C)$ un GCSP et $O'' \subsetneq O' \subset O$. Alors $DDR(O', S) \geq DDR(O'', S)$.

Nous démontrons cette proposition par contradiction. Rappelons que le DDR d'un ensemble d'objets représente son nombre de DDL lorsque celui-ci est rigide. Or le nombre de DDL d'un GCSP représente le nombre de déplacements indépendants qu'admettent les objets d'un GCSP. Si $DDR(O', S) < DDR(O''S)$, cela signifie que O'' autorise plus de déplacements que n'en autorise O', alors que $O'' \subset O'$. Notons $C_{O'}$ et $C_{O''}$ les rigidifications respectives de O' et O''. Il est impossible que O' permette moins de déplacements qu'un sous-ensemble de ses objets géométriques, sauf si $C_{O'}$ introduit une sur-rigidité, ce qui est contraire à la définition de rigidification. Donc $DDR(O', S) \geq DDR(O'', S)$. □

La proposition 15 nous permet d'établir la relation suivante :

$$DDR(O, S) = max_{O' \subseteq O} DDR(O', S).$$

Ainsi, pour établir le degré de rigidité d'un ensemble d'objets, on peut s'intéresser au DDR de ses sous-ensembles. Nous introduisons alors le concept d'*ensemble DDR-minimal* afin de simplifier ce calcul.

Ensemble DDR-minimal

Intuitivement, un ensemble d'objets géométriques constitue un ensemble DDR-minimal s'il ne contient aucun sous-ensemble d'objets ayant le même DDR. Plus formellement :

Définition 47 Ensemble DDR-minimal
Soit $S = (O, C)$ un GCSP et $O' \subset O$. O' est un ensemble DDR-minimal si et seulement si $\forall O'' \subsetneq O'$, $DDR(O'', S) < DDR(O', S)$. A chaque DDR-minimal O', on associe sa rigidification $C_{O'}$ G-consistante dans S.

Considérons par exemple le GCSP S présenté à la figure 6.3. Les ensembles d'objets $\{A, B\}$ est un ensemble DDR-minimaux : étant donnée la contrainte Distance(A, B), la seule rigidification possible est une distance, ce qui implique $DDR(\{A, B\}, S) = 3$. En revanche, l'ensemble $\{A, B, E\}$ n'est pas DDR-minimal : sa rigidification G-consistante avec S correspond aux contraintes {Incidence(A, E), Incidence(B, E), Distance(A, B)} ; ceci confère à cet ensemble d'objets $DDR(\{A, B, E\}, S) = 3$; or nous avons vu que $DDR(\{A, B\}, S) = 3$ et $\{A, B\} \subset \{A, B, E\}$.

On peut alors simplifier le problème du calcul du DDR d'un ensemble d'objets à l'aide de la proposition suivante :

Proposition 16
Soit $S = (O, C)$ un GCSP et M l'ensemble des ensembles DDR-minimaux de O. Alors, $DDR(O, S) = max_{O' \in M} DDR(O', S)$.

Cette proposition découle directement de la définition 47 et de la proposition 15 : puisque le DDR d'un ensemble d'objets est le plus grand DDR de ces sous-ensembles, alors il est le plus grand DDR de ses sous-ensembles DDR-minimaux.

Le calcul du DDR d'un ensemble d'objets se ramène donc au calcul du DDR de ses sous-ensembles DDR-minimaux ; de plus, une fois tous les ensembles DDR-minimaux d'un GCSP connus, on peut donner le DDR de n'importe quel sous-ensemble d'objets dans ce GCSP. Nous verrons au chapitre suivant que le concept d'ensemble DDR-minimal et la proposition 16 constituent des arguments cruciaux pour la définition d'un nouvel opérateur de fusion. En ce qui concerne le calcul du DDR d'un ensemble d'objets géométriques, le ramener au calcul de ses sous-ensembles DDR-minimaux simplifie le problème car le nombre d'objets dans un ensemble DDR-minimal est borné :

Proposition 17
Le nombre d'objets géométriques dans un ensemble DDR-minimal est borné par $\frac{d(d-1)}{2} + 1$ en dimension d.

Pour démontrer cette proposition, il faut constater que tout objet indépendant du repère global, c-a-d. rigide, possède au moins d DDL en dimension d, représentant d déplacements de l'espace géométrique.

D'autre part, dans un ensemble DDR-Minimal O contenant $k+1$ objets, chaque objet en plus du premier confère au moins 1 DDL de plus à ce DDR-minimal ; autrement, O ne serait pas un ensemble DDR-minimal puisqu'il contiendrait des objets qui ne lui confèreraient aucun DDL supplémentaire, c-a-d. inutiles à sa valeur de DDR[4]. Il en résulte que si O

[4] Cet argument repose sur l'hypothèse, assez raisonnable, que tout sous-ensemble d'objets est rigidifiable.

est bien un ensemble DDR-minimal à k objets, il possède au moins $d + k$ DDL. Or, la proposition 18 établit que la valeur du DDR de tout ensemble d'objets en dimension d est bornée par $\frac{d(d+1)}{2}$. Ainsi, $d + k \leq \frac{d(d+1)}{2}$ et donc $k \leq \frac{d(d-1)}{2}$. □

Proposition 18
Le degré de rigidité de tout sous-ensemble d'objets dans tout GCSP est au plus $\frac{d(d+1)}{2}$ *en dimension d.*

La proposition 18 découle du simple fait que le DDR est le nombre de DDL d'un GCSP bien-rigide. Or, nous avons montré que le nombre de DDL d'un GCSP bien-rigide est borné par $\frac{d(d+1)}{2}$ en dimension d (cf. corollaire 1, page 40). □

Ainsi, la taille des ensembles DDR-minimaux est bornée par $\frac{d(d-1)}{2} + 1$, ce qui nous permet de déduire le corollaire suivant.

Corollaire 2 Nombre d'ensembles DDR-minimaux
Dans un GCSP contenant n objets, il existe au plus $C^n_{\frac{d(d-1)}{2}+1}$, *c-a-d.* $O(n^{\frac{d(d-1)}{2}+1})$, *sous-ensembles DDR-minimaux.*

Il existe donc un nombre polynomial de sous-ensembles DDR-minimaux, ce qui nous permet de définir un algorithme de calcul polynomial pour produire le DDR de tout ensemble d'objets géométriques : il suffit de produire son ensemble de sous-ensembles DDR-minimaux et de prendre le maximum de leurs DDR.

Modèle DDR-minimal

Le nombre d'objets d'un ensemble DDR-minimal étant borné, il est possible de lister tous les *modèles DDR-minimaux* pour une classe de GCSP donnée. Intuitivement, un modèle de DDR-minimal est un ensemble d'objets géométriques qui correspond à un ensemble d'objets DDR-minimal dans un GCSP donnée. Par exemple, un triplet de points est un modèle DDR-minimal en 3D ; en effet, dans le GCSP en 3D contenant 3 points, A, B, C et les contraintes Distance(A, B), Distance(A, C) et Distance(B, C), l'ensemble d'objets $\{A, B, C\}$, qui est un triplet de points, est DDR-minimal. Plus formellement :

Définition 48 Modèle DDR-minimal
Un ensemble $M = \{o_1, o_2, ..., o_n\}$ *d'objets géométriques est un* **modèle de DDR-minimal** *si et seulement si il existe un GCSP* $S = (O, C)$ *contenant un ensemble DDR-minimal* $O' = \{o'_1, o'_2, ..., o'_n\}$ *tel que* $\forall i \in [1..n]$, o'_i *est de même* type[5] *que* o_i.

Pour la classe de GCSP ({Points, Droites},{Distances, Angles, Incidences, Parallélismes}) en 2D par exemple, une paire de points constitue un modèle DDR-minimal. En effet, dans tout GCSP de cette classe contenant une paire de points A et B, l'ensemble d'objets $\{A, B\}$ est DDR-minimal : nous avons déjà vu que cet ensemble d'objets peut

[5]Le type d'un objet géométrique est par exemple Point, Droite, Plan, Cercle, Ellipse, ...

être rigidifié soit par une distance, soit par une incidence ; dans les deux cas, cet ensemble d'objets ne contient aucun sous-ensemble possédant même DDR et est donc DDR-minimal.

Plus généralement, toute paire d'objets rigidifiables constitue un modèle DDR-minimal car elle ne contient aucun sous-ensemble d'objets.

Revenons maintenant sur le triplet de points en 3D. Dans un GCSP dont les contraintes n'impliquent pas que ces points sont alignés, ce triplet est un ensemble DDR-minimal : un triplet de points non alignés peut être rigidifié par 3 contraintes de distance, rigidification lui conférant DDR=6 ; or les paires de points ne disposent que d'un DDR=5 et le triplet est donc bien un ensemble DDR-minimal. Alors un triplet de points est un modèle DDR-minimal de toutes les classes de GCSP incluant des points et des contraintes de distance en 3D.

Proposition 19 Nombre de modèles DDR-minimaux
Dans une classe de GCSP en dimension d proposant N types d'objets géométriques différents, il existe au plus $C^N_{\frac{d(d-1)}{2}+1}$ *modèles DDR-minimaux.*

Cette proposition découle directement de la proposition 17 ; en effet, puisque la taille des ensembles DDR-minimaux est bornée, le nombre de modèles DDR-minimaux est fonction du nombre de types d'objets géométriques disponibles dans la classe de GCSP considérée.

Par ailleurs, nous avons démontré par énumération que :

- Pour la classe de GCSP ({Points, Droites},{Distances, Angles, Incidences, Parallélismes}) en 2D, il existe 3 modèles DDR-minimaux : {Point1,Point2}, {Point,Droite} et {Droite1,Droite2}.
- Pour la classe de GCSP ({Points, Droites, Plans},{Distances, Angles, Incidences, Parallélismes}) en 3D, il existe 11 modèles DDR-minimaux : {Point1,Point2}, {Point,Droite}, {Point,Plan}, {Droite1,Droite2}, {Droite,Plan}, {Plan1,Plan2}, {Point1,Point2,Point3}, {Point1,Point2,Plan}, {Point,Droite,Plan}, {Point,Plan1,Plan2} et {Plan1,Plan2,Plan3}.

La démonstration par énumération est proposée en annexe C.

Rigidifications des modèles DDR-minimaux

A chaque modèle DDR-minimal, on associe l'ensemble des rigidifications qu'il peut admettre. Ceci est possible car la taille d'un modèle DDR-minimal est bornée. De plus, l'ensemble des rigidifications pouvant porter sur un ensemble fini d'objets géométriques est de cardinalité bornée :

Proposition 20 Nombre de rigidifications
Dans une classe de GCSP en dimension d proposant M types de contraintes géométriques différentes, il existe moins de $M^{n*\frac{d(d+1)}{2}}$ *rigidifications pour un ensemble d'objets à n éléments.*

Cette proposition est due au fait qu'une contrainte retire au moins 1 DDL alors qu'un objet possède au plus $\frac{d(d+1)}{2}$ DDL en dimension d. Ainsi, si l'on place $n * \frac{d(d+1)}{2}$ contraintes

non R-redondantes sur un ensemble de n objets, il est certain que le GCSP engendré est sur-rigide car il possède au plus 0 DDL, c-a-d. aucun déplacement. Comme une rigidification engendre un GCSP bien-rigide, il est donc certain qu'elle comporte moins de $n * \frac{d(d+1)}{2}$ contraintes. Enfin, comme il est possible de choisir les contraintes parmi un ensemble de M types de contraintes, on peut en déduire qu'il existe moins de $M^{n*\frac{d(d+1)}{2}}$ rigidifications. □

En fait, nous avons démontré, toujours par énumération, que :

- Pour la classe de GCSP ({Points, Droites},{Distances, Angles, Incidences, Parallélismes}) en 2D, il existe 6 rigidifications compatibles avec les modèles DDR-minimaux de cette classe :
  - {Point1,Point2}→ {Distance(Point1,Point2)} ou {Incidence(Point1,Point2)}
  - {Point,Droite} → {Distance(Point,Droite)} ou {Incidence(Point,Droite)}
  - {Droite1,Droite2} → {Distance(Droite1,Droite2), Parallélisme(Droite1, Droite2)} ou {Angle(Droite1,Droite2)}

  Nous donnons la démonstration de ce résultat à l'annexe C.
- Pour la classe de GCSP ({Points, Droites, Plans},{Distances, Angles, Incidences, Parallélismes}) en 3D, il existe 18 rigidifications compatibles avec les modèles DDR-minimaux de cette classe ; nous donnons la liste complète de ces 18 rigidifications et la preuve d'exhaustivité à l'annexe C.

Calcul des DDR-minimaux

Etant donné que le nombre de modèles DDR-minimaux est borné, et étant donné que le nombre de rigidifications que l'on peut leur associer est borné, nous proposons de générer une fois pour toute et de stocker dans une table les modèles DDR-minimaux et les rigidifications associées pour la classe de GCSP considérée : $Modeles$ la *table des modèles DDR-minimaux*, et $Rigidifications$, la *table des rigidifications* associées à des modèles de DDR-minimaux. Ces tables nous permettent de calculer l'ensemble des DDR-minimaux d'un GCSP S de la façon décrite à l'algorithme 16.

L'algorithme `DDR-Minimaux` produit d'abord l'ensemble des sous-ensembles d'objets du GCSP qui correspondent à des modèles DDR-minimaux de la table $Modeles$. Ensuite, il tente d'associer à chaque sous-ensemble O' engendré une rigidification en s'aidant de la table des rigidifications $Rigidifications$. S'il existe une rigidification $C_{O'}$ G-consistante avec les contraintes du GCSP considéré, alors cette rigidification fait de O' un ensemble DDR-minimal. S'il n'existe aucune rigidification G-consistante, alors O' n'est pas un ensemble DDR-minimal de S.

Cet algorithme est complet pour peu que les tables utilisées le soient. Comme nous l'avons expliqué, il est tout-à-fait possible pour une classe de GCSP donnée d'engendrer la table complète des modèles DDR-minimaux et des rigidifications associées. La complexité de cet algorithme est en $O(N * M * C)$ où :

- N est le nombre de sous-ensembles d'objets du GCSP S qui sont candidats à être des ensembles DDR-minimaux ; nous avons vu que la taille d'un ensemble DDR-minimal est bornée par $\frac{d(d-1)}{2} + 1$ en dimension d ; si le GCSP contient n objets géométriques,

Algorithme 16 DDR-Minimaux ($S = (O, C)$: un GCSP ; d : la dimension du GCSP S) **retourne** $DDRM$: ensemble des ensembles DDR-minimaux et de leurs rigidifications associées

$DDRM \leftarrow \emptyset$
$DDRP \leftarrow$ `SousEnsembles(`$O, Modeles$`)` *{engendre l'ensemble des sous-ensembles de O qui correspondent à un modèle DDR-minimal}*
Pour chaque $O' \in DDRP$ **Faire**
  $DDRP \leftarrow DDRP \setminus \{O'\}$
  $RP \leftarrow$ `Associer-Rigidifications(`$O', Rigidifications$`)` *{engendre l'ensemble des rigidifications compatibles avec O'}*
  $C_{O'} \leftarrow$ `UnElément(`RP`)`
  Tant que $\neg$ `G-consistants ?(`$C_{O'}, C$`)` **et** $RP \neq \emptyset$ **Faire**
    $RP \leftarrow RP \setminus \{C_{O'}\}$
    $C_{O'} \leftarrow$ `UnElément(`RP`)`
  Fin Tant que
  Si $RP \neq \emptyset$ **Alors** *{$C_{O'}$ est une rigidification G-consistante avec le GCSP}*
    $DDRM \leftarrow DDRM \cup \{(O', C_{O'})\}$ *{O' est un ensemble DDR-minimal lorsqu'il est rigidifié par $C_{O'}$}*
  Fin Si
Fin Pour
Return $DDRM$

le nombre de candidats est alors borné par $C_n^{\frac{d(d-1)}{2}+1}$, c-a-d. $O(n^{\frac{d(d-1)}{2}+1})$.

- M est le nombre de rigidifications pouvant être associées à un modèle DDR-minimal. Nous avons vu que si le GCSP admet m types de contraintes différentes, alors le nombre de rigidifications possibles pour un modèle DDR-minimal contenant n objets est inférieur à $m^{n*\frac{d(d+1)}{2}}$ en dimension d. Comme n est pour sa part borné par $\frac{d(d-1)}{2}+1$, M peut être considéré comme une constante.
- C est la complexité du test de G-consistance.

Ainsi, si la complexité du test de G-consistance est polynomiale, la complexité de la génération de l'ensemble de tous les ensembles DDR-minimaux d'un GCSP est elle aussi polynomiale.

6.2.3 Calcul du DDR : G-consistance

La définition de la consistance géométrique (cf. définition 45, page 175) indique que deux ensembles de contraintes sont G-consistants s'ils n'induisent pas de propriétés géométriques contradictoires. Afin d'établir la G-consistance, il suffit de disposer d'un mécanisme d'inférence géométrique I_g correct et complet qui permette de décider si un ensemble de contraintes implique la négation d'une autre contrainte.

Vérifier qu'une contrainte est conséquence d'un ensemble de contraintes relève dans le cas général de la démonstration automatique en géométrie. Les méthodes de démonstra-

tion automatique géométrique sont basées sur des approches formelles qui rendent leur utilisation difficile dans des GCSP de grande taille.

Cependant, nous avons présenté à la section 6.1.4, page 170, une méthode d'inférence géométrique à base de règles. Cette méthode est généralement polynomiale mais n'est pas complète dans le cas général, bien qu'elle puisse l'être pour certaines classes de GCSP : si la méthode permet d'inférer toutes les relations géométriques *pertinentes*, c-a-d. importantes pour décider de la G-consistance, entre les objets géométriques, alors la méthode est complète. Sinon, on peut en venir à une méthode formelle.

Notons cependant qu'une méthode formelle peut aussi se trouver dans l'impossibilité de conclure s'il se trouve un DDR multiple dans le GCSP ; en effet, dans ce cas, la contrainte c sera une conséquence de C dans certaines configurations, alors que $\neg c$ sera une conséquence de C dans d'autres configurations. Ni c, ni $\neg c$ ne sont donc des conséquences globales du GCSP.

La complexité du mécanisme d'inférence utilisé est cruciale puisqu'elle conditionne la complexité de l'algorithme `DDR-Minimaux` et celle du calcul du DDR d'un ensemble d'objets quelconques.

6.2.4 Calcul du DDR à partir des DDR-minimaux

Une façon de calculer le DDR d'un ensemble O d'objets géométriques consiste ainsi à calculer l'ensemble des sous-ensembles DDR-minimaux. En effet, la proposition 16 (page 179) indique qu'il suffit ensuite de prendre le maximum des DDR de ces sous-ensembles DDR-minimaux. On peut donc calculer le DDR de tout ensemble d'objets en temps égal au temps de calcul de ces sous-ensembles DDR-minimaux.

Cependant, on peut améliorer cette complexité : pour calculer le DDR d'un ensemble d'objets, il est inutile en général de générer tous ses sous-ensembles DDR-minimaux. En effet, puisque le DDR d'un ensemble d'objets est le maximum des DDR de ses sous-ensembles DDR-minimaux, on peut chercher à générer d'abord les DDR-minimaux correspondant à des modèles DDR-minimaux associés à une rigidification leur offrant un maximum de DDL.

Ensuite, parmi les modèles de DDR-minimaux de plus grand DDR potentiel, il peut en exister qui possèdent le même DDR quelle que soit la rigidification qui leur est associée. C'est par exemple le cas du modèle {Point,Droite} en 2D : qu'il soit rigidifié à l'aide d'une distance ou d'une incidence, le DDR-minimal engendré possèdera toujours DDR=3. Ces modèles sont très intéressants car ils ne nécessitent pas de vérifier la G-consistance : quelle que soit la rigidification choisie, le DDR sera le même et il est donc inutile de déterminer quelle est la *bonne* rigidification.

Enfin, lorsque l'appel au mécanisme d'inférence est nécessaire, il est possible dans un premier temps de le limiter au mécanisme à base de règles. En effet, si celui-ci permet de décider que la rigidification associée à un modèle DDR-minimal de plus grand DDR est G-consistante, le calcul est terminé.

Classes de GCSP faciles

Il existe aussi des classes de GCSP où le DDR d'un ensemble d'objets est facile. Parmi celles-ci, on peut citer :

- les systèmes à barres. Rappelons que les systèmes à barres sont constitués de points et de contraintes de distance uniquement. Si l'on considère que les contraintes de distances sont toujours génériques, une rigidification d'un ensemble de points dans un système à barres en dimension d peut toujours être construite comme suit : rigidifier d points par $\frac{d(d-1)}{2}$ distances placées de façon non R-redondantes (c-a-d. de sorte à produire le graphe complet sur les d points), puis rigidifier chaque point supplémentaire à l'aide de d distances le liant aux d premiers points rigidifiés. Le degré de rigidité de tout ensemble O de points est ainsi déterminé uniquement par le nombre de points qu'il contient, indépendamment de son contexte S : $DDR(O,S) = \frac{d(d+1)}{2}$ si $|O| > d$, $DDR(O,S) = d * |O| - \frac{|O|(|O|-1)}{2}$ si $|O| \leq d$.
- Les mécanismes. Les mécanismes se situent à l'exact opposé des systèmes à barres : ils sont constitués de pièces mécaniques qui sont toutes des solides dotés du plus grand nombre de DDL[6] : $\frac{d(d+1)}{2}$ en dimension d. La proposition 15 (page 178) assure donc que le DDR de tout ensemble O de pièces mécaniques dans tout mécanisme S vaut alors toujours $DDR(O,S) = \frac{d(d+1)}{2}$.
- Les GCSP de la classe ({Droite},{Distance,Angle,Parallélisme}) en 2D et ({Droite, Plan},{Distance,Angle,Parallélisme}) en 3D. En effet, les seules singularités autorisées dans les GCSP de cette classe sont des relations de parallélisme, et ces relations sont soient explicites, soient toujours déductibles par des inférences simples (cf. section 6.1.4, page 170).
- Plus généralement, les GCSP composés uniquement de contraintes valuées génériques possèdent toujours un DDR facile à calculer. En effet, dans des GCSP constitués uniquement de contraintes valuées génériques, toutes les propriétés géométriques entre objets géométriques sont connues par avance puisque les configurations sont toujours génériques : deux points ne sont jamais confondus, deux plans jamais parallèles, un point jamais incident à une droite, trois points jamais alignés, ... On peut donc retirer de la table des rigidifications associées aux modèles DDR-minimaux toutes les rigidifications qui impliquent une relation géométrique contraire à la généricité : incidence, parallélisme, alignement, ... Calculer le DDR revient alors à choisir n'importe lesquelles des rigidifications restantes pour les modèles DDR-minimaux : des rigidifications par contraintes valuées génériques ne sont jamais inconsistantes.
  Qui plus est, les GCSP de cette classe sont exempts de toute singularité : les contraintes booléennes n'y sont pas autorisées et les contraintes valuées sont toutes génériques. La méthode probabiliste présentée à la section 6.1.4, page 172, permet alors d'identifier toutes les contraintes R-redondantes de n'importe quel GCSP. Seules le inconsistances peuvent encore poser problème dans la caractérisation de la rigidité.
  Toutefois, constatons que peu d'applications pratiques se rangent, à notre connais-

[6] Sous l'hypothèse que les pièces mécaniques ne possèdent pas d'axe de symétrie.

sance, dans cette catégorie de GCSP, car, comme nous l'avons souligné au début de ce chapitre, les contraintes booléennes sont courantes et même nécessaires à la conception de la majorité des constructions humaines.

Il n'est donc pas toujours nécessaire d'avoir recours au calcul de la G-consistance afin de déterminer le DDR d'un ensemble d'objets.

Calcul du DDR en présence de DDR multiples

Nous avons signalé qu'un sous-ensemble d'objets peut admettre plusieurs DDR si plusieurs rigidifications sont G-consistantes avec le GCSP considéré. Cela pose un problème dans le calcul des DDR-minimaux : un DDR-minimal peut lui aussi posséder un DDR multiple.

Dans l'algorithme `DDR-Minimaux`, nous arrêtons l'examen des rigidifications d'un ensemble d'objets candidat à être DDR-minimal dès qu'une rigidification G-consistante avec le GCSP a été identifiée. Pour traiter le DDR multiple, il faut remplacer la boucle `Tant que` par une boucle `Pour chaque` et stocker ainsi toutes les rigidifications G-consistantes d'un DDR-minimal.

Ceci modifie le calcul du DDR d'un ensemble d'objets à partir de ses sous-ensembles DDR-minimaux. En effet, la proposition 16 (page 179) ne tient plus : le DDR d'un ensemble d'objets n'est plus égal au plus grand DDR parmi ses sous-ensembles DDR-minimaux puisque ses sous-ensembles DDR-minimaux possèdent plus d'une valeur de DDR.

Une combinatoire apparaît, qui peut conférer plusieurs DDR à l'ensemble d'objets O considéré : pour chaque sous-ensemble DDR-minimal à DDR multiple, un point de choix est effectué. Ceci permet de produire toutes les combinaisons de rigidifications des DDR-minimaux pour O. Parmi celles-ci, seules celles qui sont G-consistantes doivent être considérées. Chaque combinaison restante définit une valeur de DDR possible pour l'ensemble O : c'est le maximum des DDR des ensembles DDR-minimaux d'une combinaison.

Considérons à nouveau l'exemple de GCSP présenté à la figure 6.3, page 177. Si l'on souhaite calculer le DDR du sous-ensemble $\{A, C, E\}$ des objets de ce GCSP, on va devoir produire tous les DDR-minimaux et combiner leur différentes valeurs de DDR possibles. Nous avons présenté aux sections précédentes, page 180 et page 181 les modèles DDR-minimaux et les rigidifications associées pour la classe de GCSP ({Points, Droites},{Distances, Angles, Incidences, Parallélismes}) en 2D. Nous allons utiliser l'algorithme `DDR-Minimaux` pour produire tous les DDR-minimaux de ce GCSP.

On doit donc considérer toute paire de points, toute paire de droites, et tout couple point-droite comme un candidat DDR-minimal de ce GCSP. La table ci-dessous présente le résultat de l'algorithme sur ce GCSP. On y voit, pour chaque ensemble DDR-minimal, la (ou les) rigidification(s) associée(s) et le DDR (ou les multiples DDR) en résultant.

Sous-ensembles	Rigidifications	DDR
$\{A, C\}$	Incidence(A, C)	2
	Distance(A, C)	3
$\{A, E\}$	Incidence(A, E)	3
$\{C, E\}$	Incidence(C, E)	3

Il existe donc 2 combinaisons de rigidifications des DDR-minimaux :

- (Incidence$(A, C) \wedge$ Incidence$(A, E) \wedge$ Incidence(C, E)),
- (Distance$(A, C) \wedge$ Incidence$(A, E) \wedge$ Incidence(C, E)).

Ces deux combinaisons sont parfaitement G-consistantes : rien dans les contraintes Incidence(A, E) et Incidence(C, E) ne permet d'invalider l'une ou l'autre des rigidifications possibles pour $\{A, C\}$. Le DDR de $\{A, C, E\}$ associé à la première combinaison est $max(2, 3, 3) = 3$, celui de la seconde combinaison et $max(3, 3, 3) = 3$. Notons au passage que, dans cet exemple, la présence d'un sous-ensemble DDR-minimal à DDR multiple n'a pas influé sur le DDR de l'ensemble $\{A, C, E\}$.

Il est ainsi toujours possible de calculer l'ensemble des DDR possibles pour un ensemble d'objets géométriques même en présence de DDR multiple dans ses sous-ensembles DDR-minimaux. Cependant, la méthode devient alors exponentielle dans le nombre de sous-ensembles DDR-minimaux à DDR multiples, puisqu'il faut effectuer toutes les combinaisons de rigidifications G-consistantes.

Bien sûr, on peut toujours utiliser les améliorations proposées à la page 184 pour obtenir une complexité pratique bien meilleure.

Qui plus est, on peut utiliser des *filtrages* sur les combinaisons à générer, à base de constatations très simples : la plus petite valeur de DDR pour un sous-ensemble DDR-minimal à DDR multiple représente une borne inférieure pour tous les autres sous-ensembles DDR-minimaux à DDR-multiples. Par exemple, considérons un ensemble d'objets O contenant 3 sous-ensembles DDR-minimaux $O1$, $O2$ et $O3$ tous trois à DDR multiples : $DDR(O1) = \{1, 2\}$, $DDR(O2) = \{1, 3\}$ et $DDR(O3) = \{3, 4\}$. Alors, comme $min(DDR(O3)) > max(DDR(O1))$, $O1$ n'est pas utile dans le calcul du DDR de O et il est inutile de générer les combinaisons correspondant au choix de son DDR. De même, comme $min(DDR(O3)) \geq max(DDR(O2))$, $O2$ est aussi inutile pour le calcul de $DDR(O)$. Il en résulte que sans générer aucune combinaison ni utiliser aucun test de G-consistance, on peut établir avec certitude que $DDR(O) = DDR(O3) = \{3, 4\}$. Ce type de filtrages simples, inspiré des méthodes de consistances partielles de programmation par contraintes en domaines finis, permet généralement des améliorations importantes de la complexité pratique.

6.2.5 Conclusion sur le DDR

Nous avons proposé le concept de degré de rigidité, qui représente le nombre de degrés de liberté attendu d'un sous-GCSP pour qu'il soit rigide. Nous avons expliqué les différents problèmes que posent ce concept. En particulier, certains sous-ensembles d'objets peuvent admettre un DDR multiple. Nous avons vu qu'il est malgré tout possible de calculer le DDR de tout ensemble d'objets géométriques. Nous verrons que cela occasionne cependant d'importantes limites dans l'utilisation que nous ferons de ce concept par la suite.

Nous avons proposé une méthode de calcul du DDR d'un ensemble d'objets quelconques. Nous avons montré que le calcul du DDR peut se ramener au calcul des ensembles DDR-minimaux, qui peut être effectué en temps polynomial sous réserve que l'on puisse vérifier la consistance géométrique de contraintes en temps polynomial.

Nous allons maintenant voir en quoi le concept de DDR, et la discussion que nous avons proposée sur les singularités, les R-redondances et les inconsistances dans la section 6.1 vont nous permettre de lever certaines limites des méthodes de décomposition géométrique structurelle.

6.3 Rigidité structurelle étendue

La notion de DDR permet de déterminer le nombre de degrés de liberté attendu par un ensemble d'objets géométriques dans le contexte d'un GCSP particulier pour qu'il soit rigide. Ainsi, il permet de définir une nouvelle forme de rigidité, similaire à la rigidité structurelle (cf. définition 32, page 60), mais qui prend en compte cet aspect géométrique du problème.

Définition 49 Rigidité structurelle étendue
Un GCSP $S = (O, C)$ en dimension d est **bien-es_rigide** *si et seulement si $DDL(S) = DDR(O, S)$ et $\forall S' = (O', C')$ sous-GCSP de S, $DDL(S') \geq DDR(O', S)$.*
Il est **sur-es_rigide** *si et seulement si $\exists S' = (O', C')$ sous-GCSP de S tel que $DDL(S') < DDR(O', S)$.*
Il est **sous-es_rigide** *si et seulement si $DDL(S) > DDR(O, S)$ et $\forall S' = (O', C')$ sous-GCSP de S, $DDL(S') \geq DDR(O', S)$.*

La es_rigidité se définit comme un compte des degrés de liberté dans le GCSP mais qui est comparé à une valeur particulière pour chaque sous-GCSP, le degré de rigidité, au lieu d'être seulement comparé à $\frac{d(d+1)}{2}$.

6.3.1 Comparaison avec la s_rigidité

Nous avons vu que la rigidité structurelle est équivalente à la rigidité uniquement dans le cas générique et non R-redondant, conditions imposées par la façon de compter le nombre de degrés de liberté (cf. section 2.3.2, page 55). La rigidité structurelle étendue est plus forte que la rigidité structurelle puisqu'elle peut prendre en compte les singularités introduites par les contraintes booléennes. La es_rigidité est donc strictement meilleure que la s_rigidité : dans le cas où le DDR vaut $\frac{d(d+1)}{2}$, elles sont équivalentes, mais dans le cas où le DDR est inférieur, la s_rigidité est trompée alors que la es_rigidité reste correcte.

Comparons rigidité, s_rigidité et es_rigidité sur le GCSP présenté à la figure 6.4. Ce GCSP en 3D est constitué d'une droite A et de 5 points B, C, D, E et F liés par les contraintes suivantes :
Incidence(A, B), Incidence(A, C), Incidence(A, D), Incidence(A, E), Distance(C, D), Distance(C, F), Distance(D, E), Distance(D, F), Distance(E, F).

Ce GCSP est sur-rigide dans le cas général : le point F doit être placé à l'intersection de 3 sphères de rayons différents centrés sur 3 points, C, D et E, alignés sur une même droite, A ; ce problème n'admet généralement pas de solution, sauf pour des valeurs particulières des contraintes de distance, c-a-d. dans un cas singulier. Supposons que ce GCSP

se trouve en configuration générique, mis à part les singularités forcées par les contraintes booléennes qu'il contient. Ce GCSP est R-redondant : sa sur-rigidité provient de la sur-détermination d'une partie des objets géométriques. En effet, en retirant par exemple la contrainte Distance(E, F), on obtiendrait un GCSP bien-rigide.

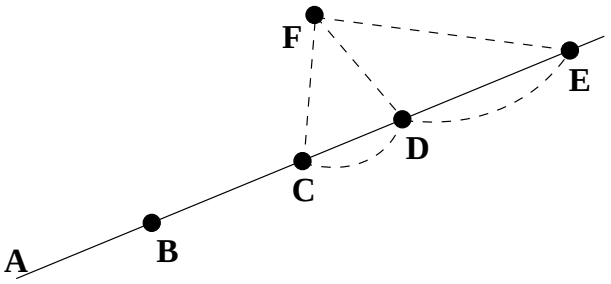

FIG. 6.4 – GCSP illustrant les différences entre rigidité, s_rigidité et es_rigidité.

Le tableau 6.1 présente une comparaison de la rigidité, la s_rigidité et la es_rigidité sur quelques sous-GCSP de ce GCSP. La colonne intitulée s_rigidité bis représente l'utilisation de la s_rigidité où l'on ne considère pas les sous-GCSP triviaux, c-a-d. contenant moins de $d+1$ objets en dimension d (cf. définition 34, page 62).

sous-GCSP	Rigidité	DDL	s_rigidité	s_rigidité+	DDR	es_rigidité
CD	bien	5	sur	trivial	5	bien
CDF	bien	6	sur	trivial	6	bien
BCD	sous	7	sur	trivial	5	sous
ACDE	bien	5	sur	sur	5	bien
ABCD	sous	6	sur	bien	5	sous
ACDEF	sur	5	sur	sur	6	sur
ABCDE	sous	6	sur	sur	5	sous
ABCDEF	sur	6	sur	sur	6	sur

TAB. 6.1 – Comparaison de la rigidité, de la s_rigidité et de la es_rigidité

Cet exemple illustre ce que nous venons d'expliquer : la es_rigidité correspond mieux à la rigidité que la s_rigidité car elle tient compte des singularités introduites par les contraintes booléennes dans son compte des degrés de liberté.

6.3.2 Limites de la es_rigidité

En revanche, la es_rigidité reste une approximation de la rigidité : elle peut être trompée par la présence de R-redondances, de contraintes valuées singulières ou d'inconsistances. Par exemple, le système à barres présenté comme contre-exemple de la caractérisation de

Laman en 3D (cf. figure 2.10, page 44) est bien-es_rigide mais n'est pas rigide. Le tableau 6.2 présente les problèmes que peuvent causer les R-redondances et les contraintes valuées singulières. On peut constater que ce tableau correspond à la table 2.4. Le lecteur peut se référer aux exemples déjà proposés pour illustrer chaque cas de divergence à la section 2.3.3. La différence entre rigidité structurelle et rigidité structurelle étendue réside dans le type de singularités qui pose problème.

		rigidité		
		sur	bien	sous
rigidité	sur	-	R-redondance	R-redondance
structurelle	bien	inconsistance	-	R-redondance
étendue	sous	inconsistance	singularité*	-

$*$: uniquement les singularités qui ne sont pas dues aux contraintes booléennes.

TAB. 6.2 – Raisons des différences entre la es_rigidité et la rigidité

Nous avons expliqué qu'il est légitime de considérer que les GCSP ne contiennent pas d'autres singularités que celles introduites par des contraintes booléennes (cf. section 6.1.1, page 165) ; dans ce cas, les inconsistances et les R-redondances qui ne sont pas des sur-es_rigidités peuvent créer une différence entre la es_rigidité et la rigidité. En revanche, il est peu réaliste de supposer que les GCSP issus d'une application réelle soient dépourvus de R-redondances. C'est pourquoi il est important de proposer des mécanismes capables de détecter, voire d'éliminer automatiquement, les R-redondances présentes dans un GCSP (cf. section 6.1.4, page 169).

Ainsi, on peut espérer qu'après élimination des R-redondances et des inconsistances, la es_rigidité correspond exactement à la rigidité.

6.3.3 Es_rigidité et DDR multiple

Un GCSP $S = (O, C)$ ne peut être bien-es_rigide que si $DDR(O, S)$ est unique ; en effet, le nombre de DDL d'un GCSP ne change pas et ne peut donc pas prendre un ensemble de valeurs différentes. Ainsi, la seule façon pour que $DDL(S) = DDR(O, S)$ est que $DDR(O, S)$ soit unique. Dans le cas contraire, le GCSP sera considéré sur-es_rigide si $DDL(S) < max(DDR(O, S))$, sinon il sera considéré sous-es_rigide.

Par ailleurs, pour qu'un GCSP soit sur-es_rigide, il suffit qu'il contienne un sous-GCSP $S' = (O', C')$ tel que $DDL(S') < max(DDR(O', S))$. En effet, si $DDL(S') < max(DDR(O', S))$, cela signifie qu'il existe des configurations de O' pour lesquelles S' sera sur-es_rigide ; comme on s'intérèsse à la rigidité globale des GCSP (rigidité en toutes les solutions, cf. définition 28, page 54), ces cas doivent être considérés globalement sur-es_rigides.

Cependant, nous avons vu que si le GCSP contient des sous-ensembles d'objets à DDR multiples, il peut *changer de rigidité* ; c'est le cas par exemple du GCSP présenté à la

figure 6.3, page 177. Ce GCSP est sous-rigide, puisqu'il admet des sous-configurations sous-rigides et que la rigidité d'un GCSP établit sa rigidité globale, c-a-d. la rigidité de toutes ses solutions (cf. définition 28, page 54). Il devrait donc être détecté sous-es_rigide, mais tel n'est pas le cas : il est détecté bien-es_rigide, et ce quelle que soit la valeur de DDR choisie pour son sous-ensemble $\{A, C\}$ à DDR multiple. En fait, pour pouvoir détecter la sous-es_rigidité qui devrait apparaître, il faudrait être capable de remplacer la contrainte EgalitéDistances(A, B, B, C) qu'il contient par la disjonction des rigidifications que cette contrainte permet sur $\{A, C\}$: Incidence$(A, C) \vee$ Distance(A, C). Il faudrait de plus être capable de constater que la contrainte EgalitéDistance(A, D, D, C) devient R-redondante lorsque l'on choisit la rigidification Incidence(A, C). Tout cela est bien entendu impossible structurellement et constitue une limite de la caractérisation structurelle de la rigidité : la présence de DDR multiples peut fausser toute approche structurelle.

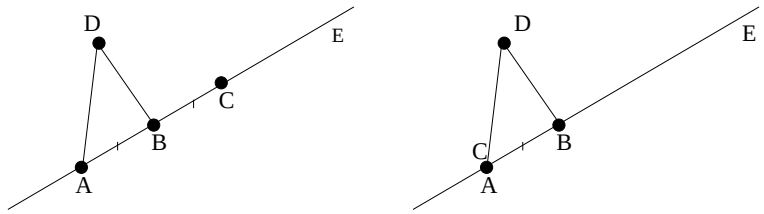

FIG. 6.5 – GCSP bien-rigide contenant un sous-ensemble d'objet à DDR multiple

Nous montrons maintenant qu'il existe aussi des GCSP bien-rigides contenant des sous-ensembles d'objets à DDR multiples. Considérons par exemple le GCSP présenté à la figure 6.5. Ce GCSP, qui ressemble à l'exemple de la figure 6.3, page 177, possède le même sous-ensemble $\{A, C\}$ à DDR multiple, mais demeure bien-rigide quel que soit le choix de DDR fait pour cet ensemble. De plus, aucune des contraintes qu'il contient ne devient R-redondante si l'on choisit que A et C soient confondus. La es_rigidité est elle aussi respectée quel que soit le choix de DDR fait pour $\{A, C\}$.

La présence de DDR multiples engendre donc des problèmes particulièrement ardus dans la caractérisation structurelle de la rigidité, mais remarquons que ces problèmes sont liés à toute approche structurelle, et pas seulement à la es_rigidité. Autrement dit, ce n'est pas l'utilisation du concept de DDR qui pose problème, mais plutôt les particularités de certains GCSP qui se traduisent par la présence de DDR multiples.

6.4 Le problème du repère local

Nous avons exposé le problème du repère local à la section 4.5, page 116. Nous avons expliqué que ce problème est crucial pour les méthodes de décomposition géométrique puisqu'il faut s'assurer qu'un sous-GCSP rigide $S1$ pourra être assemblé de façon rigide avec le reste du GCSP S' par la suite. Pour ce faire, il faut que le sous-ensemble d'objets représentant $S1$ qui est résolu dans S' permette de définir le déplacement à effectuer pour

assembler $S1$ et S', et nous avons montré que cela n'est possible que si ce sous-ensemble d'objets permet la définition d'un repère local.

Pour qu'un ensemble O d'objets géométriques permette de fixer un repère local en dimension d, il faut qu'il permette de déterminer les $\frac{d(d+1)}{2}$ DDL correspondant aux déplacements de l'espace géométrique de dimension d. Cependant, il est difficile de distinguer les DDL d'un ensemble d'objets qui correspondent à des déplacements des DDL correspondant à des déformations.

Une solution existe : considérer le degré de rigidité de O. En effet, le degré de rigidité de O représente le nombre de DDL qu'aurait O s'il était bien-rigide. Comme tout DDL d'un sous-GCSP bien-rigide représente un déplacement, ceci permet de comptabiliser le nombre de déplacements indépendants que permet de représenter l'ensemble d'objets O. La proposition suivante formalise le lien entre le problème du repère local et le concept de DDR :

Proposition 21 Repère local et DDR
Un ensemble d'objets O' dans un GCSP S en dimension d permet de définir un repère local si et seulement si $DDR(O', S) = \frac{d(d+1)}{2}$.

Ainsi, le calcul du DDR d'un ensemble d'objets géométriques permet de déterminer si cet ensemble d'objets peut définir un repère local, et donc de décider si l'assemblage de sous-GCSP est possible ou pas.

6.4.1 Repère local et DDR multiple

Encore une fois, si l'ensemble d'objets O' considéré admet plusieurs DDR dans le GCSP S, on ne peut pas décider si l'assemblage sera possible ou pas. En effet, la présence d'un DDR-multiple implique qu'il existe des valeurs de $DDR(O', S)$ inférieures à $\frac{d(d+1)}{2}$. Ces choix de DDR ne permettront pas d'assembler les sous-GCSP dépendants de O'.

En revanche, si aucun des DDR multiples de O' ne vaut $\frac{d(d+1)}{2}$, on peut conclure à coup sûr que O' ne permettra jamais un assemblage.

6.5 Conclusion

Dans ce chapitre, nous avons discuté des limites propres aux méthodes de décomposition géométrique structurelle. Schématiquement, ces limites sont dues à la présence de contraintes singulières, R-redondantes ou inconsistantes.

Nous avons expliqué que les singularités peuvent être introduites de façon explicite sous forme de contraintes booléennes et nous avons proposé le concept de degré de rigidité qui permet de prendre en compte les contraintes booléennes, et donc les singularités d'un GCSP. Malheureusement, le calcul du degré de rigidité met en œuvre une étape de déduction automatique qui peut être assez coûteuse. De plus, nous avons montré que certains GCSP peuvent laisser le choix de plusieurs degrés de rigidité, choix qui peut modifier la rigidité du GCSP et faire apparaître de nouvelles R-redondances. Cependant, il existe

des classes de GCSP où le calcul du degré de rigidité reste simple, et nous verrons qu'il n'est pas toujours requis dans l'utilisation de ce concept par notre nouvelle méthode de rigidification récursive.

Nous avons présenté des techniques de détection de R-redondances et d'inconsistances qui, si elles sont incapables de détecter les contraintes pouvant devenir R-redondantes en cours de résolution, permettent d'éliminer les principales R-redondances d'un GCSP.

Enfin, nous avons proposé une nouvelle caractérisation de la rigidité : la rigidité structurelle étendue. Cette dernière est plus conforme à la rigidité que la s_rigidité mais peut encore être trompée par la présence de R-redondances, d'inconsistances ou de DDR multiples.

La question de l'assemblage, formulée comme le problème qui consiste à décider si un ensemble d'objets permet la définition d'un repère local, a aussi pu être résolue grâce au concept de degré de rigidité, sauf dans les cas de DDR multiples.

Nous avons donc clarifié les limites des approches structurelles et montré que ces limites ne peuvent pas être repoussées facilement : il faut faire un compromis entre la correspondance de la caractérisation structurelle approchée à la véritable rigidité et le coût des méthodes aptes à calculer cette approximation. Nous pensons que le gain en fiabilité dû à l'utilisation de la es_rigidité dans les méthodes structurelles peut justifier son sur-coût par rapport à la s_rigidité. De plus, nous allons voir au chapitre suivant qu'à l'instar de la s_rigidité, il est possible de caractériser les sous-GCSP bien- ou sur-es_rigide à l'aide d'un algorithme de flot.

Chapitre 7

Une nouvelle phase de planification

Sommaire

Forts des remarques et résultats introduits au chapitre précédent, nous proposons une nouvelle méthode de rigidification récursive, basée sur le schéma classique des méthodes de décomposition géométrique : une phase de planification décompose le GCSP en sous-GCSP rigides qui sont résolus et assemblés durant la phase d'assemblage. Ce chapitre expose la nouvelle phase de planification, notre nouvelle phase d'assemblage faisant l'objet du chapitre suivant.

Notre nouvelle phase de planification emprunte son schéma à celle de la méthode HLS (cf. chapitre 5). Le GCSP est décomposé en sous-GCSP par application de trois opérateurs : fusion, extensions et condensation. Nous avons cependant remplacé les trois opéra-

teurs par de nouveaux opérateurs, et introduit un opérateur supplémentaire : l'opérateur d'élimination des R-redondances/inconsistances.

La nouvelle opération de fusion [Jermann *et al.*, 2002a; 2002b] est réalisée par un algorithme de flot similaire à celui utilisé par la méthode HLS (cf. section 5.3, page 125). Cependant, il effectue la distribution de la surcharge de façon géométriquement correcte, et est basé sur le concept de degré de rigidité. Nous verrons qu'il ne nécessite pas l'utilisation d'une Look-up Table.

La nouvelle opération d'extension est similaire à celle de la méthode HLS. Elle est cependant basée sur la es_rigidité elle aussi. De plus, nous introduisons un nouveau type d'extension, l'*extension globale*, qui permet d'inclure un ancien agrégat entier en une seule extension dès que cela est possible. Ce nouveau concept évite que ne soient réalisées trop d'extensions inutiles et améliore la capacité de l'opération d'extension, comme nous l'expliquerons à la section 7.4.

La nouvelle opération de condensation [Jermann *et al.*, 2000a], que nous avons appelée FAR pour *Frontier Automatic Rigidification*, utilise le concept de rigidification (cf. définition 44, page 175) : l'agrégat identifié est remplacé par l'ensemble de ses objets frontières (cf. définition 43, page 146) rigidifiés. Nous expliquons en quoi cette opération est plus intéressante que CA, FA ou MFA à la section 7.5.

L'opération d'élimination des R-redondances/inconsistances est appliquée une seule fois au début de la planification. Elle assure, en théorie, que le GCSP décomposé est exempt de R-redondances et d'inconsistances. Nous verrons que cela permet de simplifier les opérations de fusion, d'extensions et de condensation tout en impliquant une plus grande adéquation de la es_rigidité pour caractériser la vraie rigidité.

Nous commençons par offrir un aperçu général de la phase de planification, suivi d'une section détaillée pour chacune des nouvelles opérations.

7.1 Aperçu général de la planification

Notre phase de planification est similaire à celle de la méthode HLS. Cependant, elle tient compte des remarques et des résultats présentés au chapitre précédent :

- La phase de planification commence par appliquer l'opérateur d'élimination des R-redondances/inconsistances `Eliminer-redondances`. L'opérateur stocke les contraintes R-redondantes qu'il retire dans un ensemble de contraintes annexes ; ces contraintes pourront être utilisées lors de la phase d'assemblage (cf. section 8.3.4, page 236).
- Ensuite, la planification consiste à appliquer itérativement la séquence fusion, extensions, condensation jusqu'à ce qu'un point fixe soit atteint ; le point fixe est toujours représenté par le fait que l'opération de fusion retourne un agrégat vide, signifiant qu'il n'existe plus aucun sous-GCSP es_rigide dans le GCSP. Cependant, nous renouvelons les trois opérations : nos nouvelles opérations sont basées sur la es_rigidité ; la nouvelle opération de fusion utilise toujours un algorithme de flot mais effectue la distribution de façon assez différente ; la nouvelle opération d'extensions introduit la notion d'*extension globale* ; enfin, notre opération de condensation est basée sur le

concept de *rigidification automatique*.

En théorie, notre phase de planification opère donc sur un GCSP exempt de R-redondances, d'inconsistances et de singularités autres que celles introduites par des contraintes booléennes et la es_rigidité correspond alors assez bien[1] à la rigidité ; ceci nous permet d'espérer que la décomposition produite par notre nouvelle phase de planification sera plus correcte que celle produite par la méthode HLS.

7.1.1 Exemple de planification

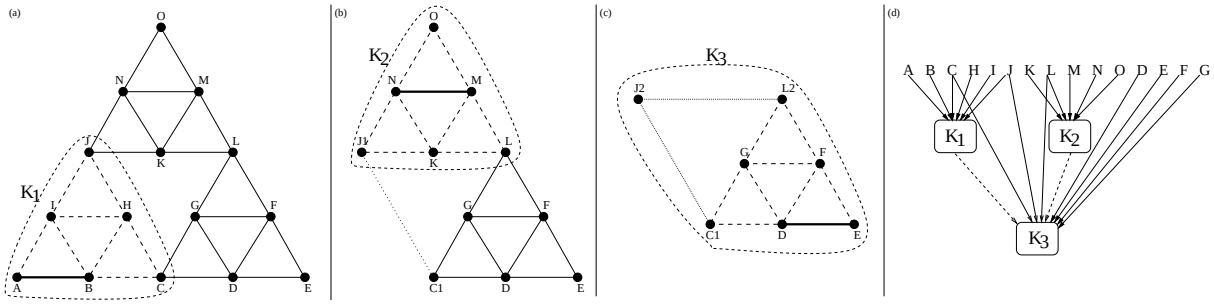

FIG. 7.1 – Déroulement de notre nouvelle phase de planification sur le GCSP *Ponts*.

Considérons la décomposition du GCSP *Ponts* par notre nouvelle phase de planification. La figure 7.1 présente le déroulement de cette phase sur le GCSP *Ponts*.

La première étape de fusion identifie le sous-GCSP $K_1 = AB$ (cf. figure 7.1-a). L'opération d'extensions lui ajoute alors un à un les objets géométriques I, H, C et J pour terminer sur l'agrégat $K_1 = ABCHIJ$ qui ne peut plus être étendu vers ses objets voisins (cf. figure 7.1-a). Le GCSP est alors mis à jour par la condensation FAR, qui consiste à dupliquer les *objets frontières* (cf. définition 43, page 146) de K_1 dans le GCSP et à leur imposer une *rigidification* (cf. définition 44, page 175). La frontière de K_1 est constituée des points C et J ; ils sont dupliqués en C_1 et J_1 qui sont rigidifiés par une contrainte de distance. K_1 est ensuite ajouté au plan d'assemblage P ; dans ce plan, il dépend des objets A, B, C, H, I et J et un arc établit cette dépendance.

L'étape de fusion suivante identifie $K_2 = MN$, ensuite étendu en $K_2 = J_1KLMNO$ par l'opération d'extensions (cf. figure 7.1-b). Notons que K_2 a pu être étendu sur J_1 puisque les objets frontières des anciens agrégats demeurent dans le GCSP. Le GCSP et le plan d'assemblage sont mis à jour pour prendre en compte le nouvel agrégat K_2 : ces objets frontières, J_1 et L, sont dupliqués en J_2 et L_2 qui sont rigidifiés par une nouvelle contrainte de distance.

La troisième étape de fusion identifie $K_3 = DE$ qui est étendu à $K_3 = C_1DEFGJ_2L_2$ par l'opération d'extensions (cf. figure 7.1-c). On constate que l'opération d'extensions a

[1]Nous n'avons pu produire aucun exemple de GCSP où la es_rigidité diffère de la rigidité et qui ne contienne pas une R-redondance ou une singularité

permis l'inclusion de tous les objets du GCSP à cette étape, ce qui conclut la phase de planification. De plus, l'inclusion de tous les objets frontières des agrégats K_1 et K_2 signifie que ces deux agrégats ont été inclus entièrement dans K_3. Notons que la planification termine en une itération de moins que la méthode HLS avec opération de condensation CA, FA ou MFA, ce qui illustre l'intérêt de FAR.

7.1.2 Limites de la nouvelle planification

On distingue trois catégories de limites à notre méthode de rigidification récursive :
- les limites liées au schéma de la méthode lui-même,
- celles liées à l'utilisation de la es_rigidité,
- et celles liées à la présence de sous-ensembles à DDR multiples

Limites liées au schéma de rigidification récursive

Notre nouvelle phase de planification impose les mêmes restrictions que la méthode HLS sur les GCSP qu'elle sait traiter :
- les objets géométriques doivent être rigides,
- les contraintes géométriques doivent être indépendantes du repère global.

Nous avons expliqué les raisons de ces limitations à la section 5.2, page 123. Brièvement, les objets doivent être rigides en eux-même car les dimensions variables faussent le compte de degré de liberté ; les contraintes doivent être indépendantes du repère global autrement le GCSP est sur-rigide. Nous avons aussi expliqué que si la première restriction constitue une véritable limite à l'emploi de la méthode, la seconde en revanche peut être réglée par l'introduction d'un objet repère dans le GCSP.

Limites liées à la es_rigidité

Nous avons vu à la section 6.3.2 (page 189) que la es_rigidité peut être trompée par la présence de singularités, de R-redondances ou d'inconsistances. Nous avons justifié l'absence de singularité autres que celles introduites sous forme de contraintes booléennes. L'absence de R-redondances et d'inconsistances est justifiée par l'utilisation d'une opération d'élimination appliquée au début de notre phase de planification.

Cependant, nous avons expliqué à la section 6.1.4, page 169, que même le mécanisme de détection le plus complet ne peut éliminer toutes les R-redondances et les inconsistances d'un GCSP en général. Ceci est dû au fait que certaines contraintes peuvent ne devenir R-redondantes qu'une fois le GCSP partiellement résolu, pour une sous-configuration de certains de ses objets géométriques (cf. figure 6.3, page 177).

On ne peut donc pas assurer que les GCSP traités seront entièrement exempts de R-redondances et d'inconsistances, et il est alors possible que la es_rigidité se trompe dans sa caractérisation de la rigidité.

Limites liées aux DDR multiples

La présence de sous-ensembles d'objets à DDR multiples met à mal toute caractérisation de la rigidité basée sur un compte de degré de liberté. En effet, comme nous l'avons vu à la section 6.3.3, page 190, on ne peut pas toujours décider si un sous-GCSP basé sur un ensemble d'objets à DDR multiple est bien-, sur- ou sous-es_rigide puisque son nombre de DDL ne varie pas alors que son DDR peut varier.

Nous avons vu à cette même section que la rigidité elle-même peut varier. Or une décomposition unique n'a de sens que si le GCSP est globalement rigide : si certains sous-GCSP sont parfois bien-rigides, parfois sous- ou sur-rigides, une décomposition unique n'a pas de sens.

La seule approche correcte consisterait alors à tenir compte de chaque possibilité séparément, c-a-d. à procéder à un point de choix durant la décomposition à chaque fois qu'un sous-GCSP peut admettre différentes es_rigidités. Ceci engendre malheureusement une combinatoire qui produit un nombre de décompositions exponentiel dans le nombre de sous-GCSP à DDR multiple.

Il ne nous semble pas envisageable de procéder à la génération d'une telle multitude de plans, *aussi supposerons-nous que les GCSP traités sont exempt de sous-ensembles d'objets à DDR multiples.*

Lorsque cette hypothèse n'est pas vérifiée, on peut décomposer le GCSP de façon heuristique en *choisissant* un seul plan parmi tous les plans possibles. Par exemple on peut choisir un unique plan en considérant que chaque sous-ensemble d'objets à DDR multiple admet pour unique DDR son plus grand DDR possible ; intuitivement, ce choix consiste à considérer que chaque sous-ensemble d'objets à DDR multiple se trouve dans la configuration la plus générique possible.

Afin de s'assurer que la décomposition correspondante à ce choix sera résolue de façon correcte, on peut introduire dans l'ensemble des contraintes R-redondantes, les rigidifications correspondantes aux choix de DDR faits pour les ensembles d'objets à DDR multiples. Lors de la résolution, la prise en compte de ces contraintes supplémentaires assurera que les solutions engendrées seront toutes conformes au choix de DDR faits durant la planification.

7.2 Détection des R-redondances

La première étape de notre phase de planification consiste à éliminer les R-redondances (et les inconsistances) du GCSP à décomposer. L'élimination des R-redondances peut être effectuée par une ou plusieurs des méthodes exposées à la section 6.1.4. Il en résulte que le GCSP soumis aux opérations de fusion, extensions et condensation sera (en théorie) exempt de R-redondances ; en pratique, nous avons expliqué qu'il existe des R-redondances indétectables *a priori*, car n'apparaissant qu'en cours de résolution. Notons par ailleurs que toute sur-es_rigidité est due à la présence d'une R-redondance ou d'une inconsistance ; en effet, une sur-es_rigidité représente le fait qu'il y a trop de contraintes du point de vue du compte des DDL. L'élimination des R-redondances et inconsistances élimine donc toute

sur-es_rigidité.

Par ailleurs, si l'élimination des R-redondances peut en général être incomplète (cf. section 6.1.4), l'élimination des sur-es_rigidités, elle, est complète et correcte : elle est basée sur l'approche structurelle de détection de R-redondances et utilise l'algorithme `ES_Rigide` que nous présentons à la section 7.3. On peut donc assurer que le GCSP traité est exempt de sur-es_rigidités ; ce fait sera utilisé à plusieurs reprises pour proposer des simplifications de la méthode.

Les contraintes R-redondantes éliminées sont regroupées dans un ensemble de contraintes annexes qui sera utilisé lors de la phase d'assemblage afin de filtrer les solutions produites pour le GCSP (cf. section 8.3.4, page 236).

7.3 Nouvelle opération de fusion

La nouvelle opération de fusion permet d'identifier un sous-GCSP bien-es_rigide minimal. Elle est effectuée par l'algorithme `Minimal-ES_Rigide` qui reprend les grandes lignes de l'algorithme `Minimal-Dense` (cf. section 5.3) avec deux modifications importantes :

- il utilise la es_rigidité comme caractérisation de la rigidité,
- il distribue la surcharge de façon géométriquement correcte dans le réseau.

A l'instar de l'algorithme `Minimal-Dense`, `Minimal-ES_Rigide` est constitué de deux étapes : d'abord, l'algorithme `ES_Rigide` identifie un sous-GCSP bien- ou sur-es_rigide, ensuite une étape de minimisation en extrait un sous-GCSP minimal bien- ou sur-es_rigide.

7.3.1 Algorithme `ES_Rigide`

L'algorithme `ES_Rigide` travaille sur le réseau objets-contraintes (cf. définition 41, page 128). Le principe de l'algorithme est aussi de distribuer un flot maximum dans ce réseau après y avoir introduit une surcharge qui permet d'identifier un déficit en DDL dans un sous-GCSP. La différence avec l'algorithme `Dense` se trouve dans la façon d'introduire la surcharge.

Rappelons que l'algorithme `Dense` est incorrect du point de vue géométrique car il distribue la surcharge de flot sur les contraintes qui peuvent être liées à n'importe quels objets. En fait, il faut comprendre que la surcharge appliquée représente une tentative de placer un repère local ; or, il existe des ensembles d'objets ne permettant pas la définition d'un repère local, comme une paire de points en 3D, ou une paire de droites parallèles en 2D. (cf. section 5.3.2, page 136).

Afin d'appliquer la surcharge de façon géométriquement correcte, nous proposons donc de la distribuer directement sur les sous-ensembles d'objets permettant la définition d'un repère local. Pour ce faire, nous introduisons une contrainte virtuelle R qui servira uniquement à la distribution de la surcharge. L'arc liant la source du réseau au nœud représentant R a une capacité égale à la surcharge que l'on souhaite distribuer ; un arc de capacité infinie lie R à chacun des objets géométriques sur lesquels on veut appliquer la surcharge.

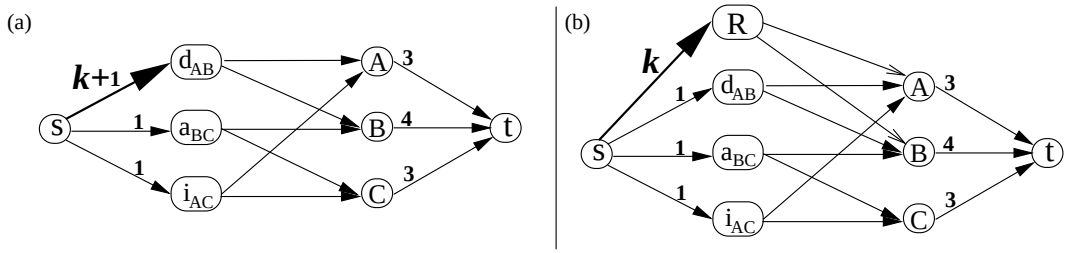

FIG. 7.2 – Différence entre la distribution de la surcharge par `Dense` et par ES_Rigide

Pour illustrer cette différence avec l'algorithme `Dense`, considérons à nouveau l'exemple de GCSP présenté à la figure 5.3, page 132. Ce GCSP en 3D est constitué d'un point A, d'une droite B et d'un plan C liés par les contraintes Distance(A, B), Angle(B, C) et Incidence(A, C). La figure 7.2 présente la différence entre la distribution d'une surcharge k dans ce GCSP selon l'algorithme `Dense` (cf. figure 7.2-a) et selon l'algorithme `Minimal-ES_Rigide` (cf. figure 7.2-b). On constate que la source est liée à la contrainte virtuelle R comme à toute autre contrainte ; la capacité de cette arc représente la surcharge à distribuer.

Valeur de la surcharge

L'identification d'un sous-GCSP sur-$k$dense (c-a-d. possédant moins de k DDL) est basée sur la constatation suivante : si un arc issu de la source n'est pas saturé après le calcul d'un flot maximum dans le réseau surchargé par une capacité k, alors c'est que le GCSP ne peut absorber k DDL supplémentaires, c-a-d. qu'il contient un sous-GCSP sur-$k$dense (cf. section 5.3.1, page 128). Hoffmann *et al.* ont par ailleurs démontré que le sous-GCSP sur-$k$dense ainsi identifié est induit par l'ensemble des objets géométriques traversé lors de la dernière recherche d'un chemin augmentant dans le réseau, ou, autrement dit, l'ensemble des objets géométriques atteignables par un parcours du graphe résiduel depuis la contrainte non-saturée.

Nous utilisons le même principe dans l'algorithme `ES_Rigide` : l'arc liant la source à la contrainte virtuelle R est distribué en dernier lors du calcul du flot maximum ; s'il ne peut être saturé, c'est qu'il existe un sous-GCSP sur-$k$dense, induit par l'ensemble des objets géométriques atteignables en partant de R dans le graphe résiduel.

Nous établissons le lien entre densité et es_rigidité dans la proposition suivante :

Proposition 22 Densité et es_rigidité
Soit $S = (O, C)$ un GCSP en dimension d. Alors :

- *S est bien-es_rigide si et seulement si S est bien-$DDR(O, S)$dense et $\forall S' = (O', C')$ sous-GCSP de S, S' est bien- ou sous-$DDR(O', S)$dense.*
- *S est sur-es_rigide si et seulement si $\exists S' = (O', C')$ sous-GCSP de S tel que S' est sur-$DDR(O', S)$dense.*

– *S est sous-es_rigide si et seulement si S est sous-$DDR(O,S)$dense et $\forall S' = (O', C')$ sous-GCSP de S, S' est bien- ou sous-$DDR(O', S)$dense.*

Cette proposition peut être aisément vérifiée à partir des définitions de densité et de es_rigidité : la $k$densité représente le fait que le compte des DDL est inférieur (sur-dense), égal (bien-dense) ou supérieur (sous-dense) à la valeur k ; la proposition 22 n'est donc qu'une reformulation de la définition de la es_rigidité (cf. définition 49, page 188).

Ainsi, pour identifier un sous-GCSP sur-es_rigide, il faut utiliser une surcharge égale au DDR de l'ensemble d'objets O' auquel est liée la contrainte R, alors qu'une surcharge égale à DDR+1 permettra d'identifier des sous-GCSP bien- ou sur-es_rigides.

Algorithme 17 ES_Rigide (S : un GCSP) **retourne** S' : sous-GCSP bien- ou sur-es_rigide

$S' \leftarrow (\emptyset, \emptyset)$
$M \leftarrow$ `DDR-Minimaux(`S`)` *{M contient l'ensemble des DDR-minimaux de S}*
trouve $\leftarrow$ *Faux*
$G_S = (\{s, t, R\} \cup V, E \cup \{s \rightarrow R\}, w) \leftarrow$ `RéseauObjetsContraintes(`S`)` *{le réseau objets-contraintes de S contient la contrainte virtuelle R et l'arc $s \rightarrow R$}*
Tant que $\neg$*trouve* **et** $M \neq \emptyset$ **Faire**
  $O' \leftarrow$ `UnElement(`M`)` ; $M \leftarrow M \setminus \{O'\}$ *{sélection d'un ensemble DDR-minimal}*
  $E \leftarrow E \cup \{R \rightarrow o | o \in O'\}$ *{liaison de R à O' par des arcs de capacité $w(R \rightarrow o) = +\infty$}*
  $w(s \rightarrow R) \leftarrow DDR(O', S) + 1$ *{ou $DDR(O', S)$ pour l'algorithme* `Sur-ES_Rigide`*}*
  flots $\leftarrow$ `FlotMaximum(`G`)`
  trouve $\leftarrow$ `NonSaturé ?(`$s \rightarrow R, flots$`)`
  $E \leftarrow E \setminus \{R \rightarrow o | o \in O'\}$ *{restauration du réseau pour la prochaine itération}*
Fin Tant que
Si *trouve* **Alors**
  $O' \leftarrow$ `SommetsCheminAugmentant(`$G, R, flots$`)`
  $S' \leftarrow$ `SousGCSPInduit(`O', S`)`
Fin Si
Return S'

Application de la surcharge

L'algorithme `Dense` appliquait la surcharge sur chaque contrainte du GCSP successivement. Afin d'assurer que notre algorithme identifie un sous-GCSP bien- ou sur-es_rigide s'il en existe un, il faudrait normalement lier la contrainte virtuelle R successivement à tous les sous-ensembles d'objets en adaptant la surcharge au DDR de l'ensemble considéré à chaque fois. Nous proposons de ne lier R qu'aux seuls sous-ensembles DDR-minimaux (cf. définition 47, page 179) du GCSP considéré. Cette modification est cruciale, puisque l'ensemble de tous les sous-ensembles d'objets est de taille exponentielle alors que l'ensemble de sous-ensembles DDR-minimaux est de taille polynomiale (cf. proposition 2, page 180) : c'est cette propriété qui rend le nombre d'application de la surcharge polynomial.

L'algorithme `ES_Rigide` (cf. algorithme 17) consiste donc à lier la contrainte virtuelle R successivement à tous les DDR-minimaux du GCSP, en adaptant la surcharge au sous-ensemble O' considéré à chaque itération. Le calcul de l'ensemble des sous-ensembles DDR-minimaux est effectué par la fonction `DDR-minimaux` présentée à l'algorithme 16, page 183.

Propriétés et complexité

Nous allons maintenant démontrer que l'algorithme `ES_Rigide` est correct, complet et polynomial sous réserve que le calcul de l'ensemble des DDR-minimaux soit accompli en temps polynomial.

Afin de démontrer que cet algorithme est correct et complet, nous allons démontrer la proposition suivante :

Proposition 23 `ES_Rigide` **correct et complet**
Soit $S' \subset S$ le sous-GCSP retourné par un appel à `ES_Rigide`*(S). Alors, S' est bien- ou sur-es_rigide, ou bien S' est le sous-GCSP vide et S est sous-es_rigide.*

Correction de `ES_Rigide` **:** Considérons que $S' = (O', C')$ est un sous-GCSP non-vide retourné par l'appel à `ES_Rigide`(S). Ce sous-GCSP résulte de la distribution de la surcharge sur un sous-ensemble DDR-minimal O''. Par définition de l'algorithme, la surcharge distribuée était alors $DDR(O'', S)$. Toujours par définition de l'algorithme, S' vérifie nécessairement $O'' \subset O'$. Alors, la proposition 15 (page 178) assure que $DDR(O', S) \geq DDR(O'', S)$. De plus, le fait que le calcul du flot maximum n'ai pu distribuer toute la surcharge signifie que S' possède moins de DDL que la surcharge appliquée : $DDL(S') \leq DDR(O'', S)$. En combinant ces deux inégalités, on obtient donc $DDL(S') \leq DDR(O', S)$, qui est une condition suffisante pour que S' soit bien- ou sur- es_rigide (cf. définition 49, page 188).□

Complétude de `ES_Rigide` **:** Le lemme suivant permet d'établir qu'il est suffisant de lier R aux sous-ensembles DDR-minimaux du GCSP pour assurer la complétude de l'algorithme :

Lemme 2
Soient S un GCSP et $O'' \subset O'$ deux sous-ensembles d'objets vérifiant $DDR(O'', S) = DDR(O', S)$. Soit G' (resp. G'') le réseau objets-contraintes dans lequel la contrainte virtuelle R est liée à O' (resp. O''). Alors le flot maximum distribué dans chaque arc de G' est supérieur ou égal au flot maximum dans ceux de G''.

Démontrons tout d'abord ce lem avant d'expliquer pourquoi le corollaire 3 ci-dessous en découle. Le GCSP S et la surcharge distribuée étant les mêmes, la seule différence entre G' et G'' se trouve dans les arêtes liant R à des nœuds-objets. Puisque $O'' \subset O'$, on peut dire que $G'' \subset G'$, c-a-d. que G'' et G' contiennent les mêmes nœuds mais G' contient les arcs $R \rightarrow o$ pour $o \in O' \setminus O''$ en plus de ceux de G''. Rappelons que par construction du

réseau objets-contraintes, l'arc liant une contrainte à un objet est de capacité infinie. Qui plus est, les arcs issus de la source, et ceux allant dans le puits, sont les mêmes pour G' et G''. Le flot maximum de G' ne peut donc qu'être supérieur ou égal au flot maximum dans G'' car il dispose de plus d'arcs pour faire passer la même capacité initiale. □

Corollaire 3 Distribution DDR-minimaux suffisante
La distribution d'une surcharge valant $DDR(O', S)+1$ (resp. $DDR(O', S)+1$) sur chaque sous-ensemble DDR-minimal O' d'un GCSP S suffit à identifier un sous-GCSP bien-es_rigide (resp. sur-es_rigide) s'il en existe un.

Il résulte du lemme 2 et de la proposition 16 (page 179) qu'il est suffisant de distribuer le flot sur les sous-ensembles d'objets DDR-minimaux (corollaire 3). En effet, par la proposition 16, tout sous-ensemble O' d'objets non DDR-minimal contient un sous-ensemble O'' d'objets DDR-minimal ayant même DDR ; Par le lemme 2, il est alors inutile de distribuer la surcharge sur O', puisque si l'arc liant la source à R ne peut être saturé, il ne le pourra pas non plus lorsque R sera lié au DDR-minimal O''. □

Complexité de `ES_Rigide` : Dans la discussion qui suit au sujet de la complexité, on notera n (resp. m) le nombre d'objets (resp. de contraintes) géométriques dans le GCSP.

La complexité de l'algorithme `ES_Rigide` dépend de la complexité du calcul de l'ensemble M des DDR-minimaux. Nous avons montré à la section 6.2.2, page 182, que la complexité de l'algorithme `DDR-Minimaux` est en $O(n^{\frac{d(d-1)}{2}+1} * C)$, où C est la complexité du mécanisme d'inférence qui établit la G-consistance des rigidifications pouvant être associées aux modèles DDR-minimaux ; C peut être exponentielle si des méthodes formelles sont utilisées pour démontrer la G-consistance ; cependant, nous avons présenté une méthode polynomiale (incomplète) à base de règles pour effectuer cette même opération.

Ensuite, pour chaque DDR-minimal dans M, l'algorithme effectue une mise à jour du réseau (en temps constant selon la proposition 17, page 179) et un calcul de flot maximum en $O(n^2 * (n + m))$. Le nombre de DDR-minimaux est en $O(n^{\frac{d(d-1)}{2}+1})$ (cf. proposition 2, page 180). Cette étape est donc de complexité $O(n^2 * (n + m) * n^{\frac{d(d-1)}{2}+1})$ en pire cas.

La complexité totale est alors $O(n^{\frac{d(d-1)}{2}+1} * (C + n^2 * (n + m)))$. On constate que la complexité de l'algorithme `ES_Rigide` dépend donc fortement de la complexité du mécanisme d'inférence mis en jeu pour établir la G-consistance : si celui-ci est polynomial, alors l'algorithme `ES_Rigide` l'est aussi. De plus, dans le cas polynomial, C est généralement négligeable devant la complexité du calcul du flot maximum, et l'algorithme est alors en $O(n^{\frac{d(d-1)}{2}+3} * (n + m))$.

Exemple d'exécution

Pour illustrer cet algorithme, considérons son exécution sur le GCSP présenté à la figure 6.4, page 189. Ce GCSP en 3D est constitué d'une droite A et de 5 points B, C, D, E et F liés par les contraintes :

Incidence(A, B), Incidence(A, C), Incidence(A, D), Incidence(A, E), Distance(C, D), Distance(C, F), Distance(D, E), Distance(D, F), Distance(E, F).

Rappelons que ce GCSP est sur-rigide dans le cas général. La rigidité et la es_rigidité de certains de ses sous-GCSP sont présentées à la table 6.1, ainsi que les valeurs de DDL et DDR.

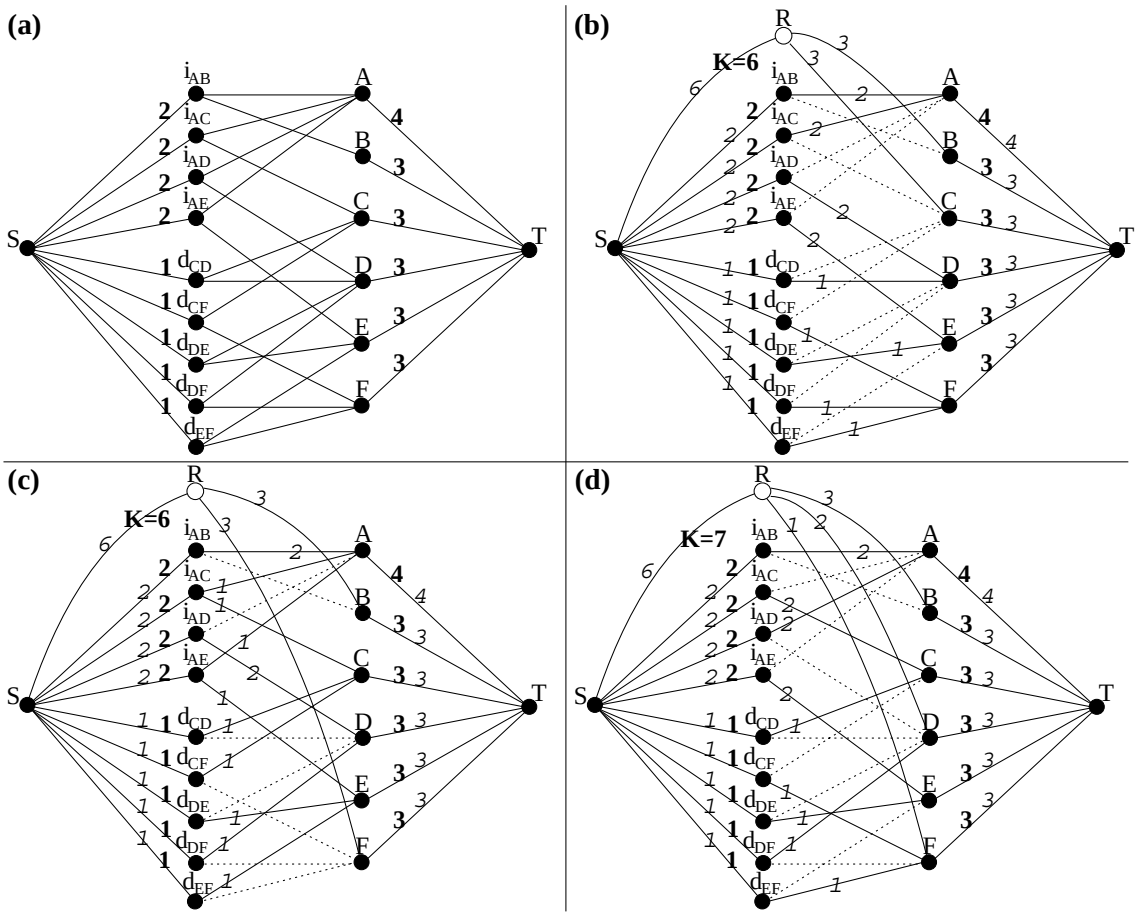

FIG. 7.3 – Illustration du déroulement de l'algorithme `ES_Rigide`

La figure 7.3 présente l'exécution de l'algorithme `ES_Rigide` sur ce GCSP. Considérons l'ensemble des DDR-minimaux $M = \{\{B, C\}, \{B, F\}, \{B, D, F\}, ...\}$ généré par un appel à la fonction `DDR-minimaux`. La figure 7.3-a représente le réseau objets-contraintes de ce GCSP.

A la première étape, $O' = \{B, C\}$ et $DDR(\{B, C\}, S) = 5$. La contrainte virtuelle R est alors liée à B et C et la capacité de l'arc $s \rightarrow R$ est ajustée à 6. La figure 7.3-b représente la distribution de flot maximum dans le réseau de cette étape. L'arc $s \rightarrow R$ est saturé, ainsi que tous les arcs issus de la source, ce qui indique qu'aucun sous-GCSP bien- ou sur-es_rigide n'a pu être identifié à cette étape.

A la seconde étape, $O' = \{B, F\}$ et $DDR(\{B, F\}, S) = 5$. R est liée à B et F et la surcharge vaut 6. La figure 7.3-c présente la distribution de flot maximum dans le résultat de cette étape. On constate que tous les arcs issus de la source sont saturés, et aucun sous-GCSP bien- ou sur-es_rigide n'a donc été identifié encore.

A la troisième étape, $O' = \{B, D, F\}$ and $DDR(\{B, D, F\}, S) = 6$. R est liée à B, D et F et la surcharge vaut 7. La distribution de flot maximum, présentée en figure 7.3, ne peut saturer l'arc $s \rightarrow R$ ce qui indique qu'un sous-GCSP bien- ou sur-es_rigide a été trouvé. Ce GCSP est induit par l'ensemble des objets traversés par un parcours partant de R dans le graphe résiduel du réseau ; ces objets sont A, B, C, D, E et F et le sous-GCSP est donc le GCSP entier. Il s'agit bien d'un GCSP sur-es_rigide car il contient le sous-GCSP $ACDEF$ qui est sur-es_rigide (cf. table 6.1).

Amélioration des performances

La complexité de l'algorithme peut être améliorée de plusieurs façons. Tout d'abord une première amélioration générale consiste à utiliser une version incrémentale de l'algorithme. En effet, on peut constater que le réseau sur lequel est effectué le calcul d'un flot maximum est toujours le même aux liaisons de la contrainte virtuelle R près. On peut donc calculer une fois pour toute le flot maximum dans le réseau non surchargé ; distribuer une surcharge valant k nécessite alors au maximum k calculs de chemins augmentants ; or la surcharge représente le DDR d'un ensemble d'objets géométriques, et k est donc borné par $\frac{d(d+1)}{2}$ en dimension d (cf. proposition 18, page 180). La complexité de la distribution à chaque itération est alors en $O(n * (n + m))$, c-a-d. qu'elle est améliorée d'un facteur linéaire. Par exemple, pour la classe ({Points, Droites, Plans},{Distances, Angles, Incidences, Parallélismes}) en 3D, la complexité de l'algorithme est en fait $O(n^3 * (C + n * (n + m))$ en pire cas.

La complexité pratique peut encore être améliorée par l'utilisation d'heuristique pour le calcul de l'ensemble des DDR-minimaux. Constatons tout d'abord qu'il est inutile de précalculer l'ensemble de tous les ensembles DDR-minimaux par avance : on peut se contenter de proposer un nouvel ensemble DDR-minimal à chaque itération et ainsi limiter le nombre de DDR-minimaux engendrés en moyenne, puisque l'algorithme termine aussitôt qu'il a identifié un premier sous-GCSP bien- ou sur- es_rigide.

Afin d'éviter l'appel au mécanisme d'inférence coûteux visant à établir la G-consistance de la rigidification associée à un modèle DDR-minimal, on peut aussi utiliser les heuristiques proposées à la section 6.2.4, page 184 : mieux vaut utiliser en premier les modèles DDR-minimaux dont le DDR est constant quelle que soit la rigidification qui leur est associée ; ces modèles ne nécessitent en effet pas l'appel au mécanisme d'inférence. Ensuite, certaines rigidifications associées à des modèles DDR-minimaux peuvent être certifiées G-consistantes en temps polynomial à l'aide du mécanisme d'inférence à base de règles.

Ces optimisations combinées permettent d'obtenir un algorithme d'une complexité pratique bien inférieure, car ayant recours le moins souvent possible à des opérations coûteuses.

Utilisations de l'algorithme `ES_Rigide`

L'algorithme `ES_Rigide` peut être utilisé en place de l'algorithme `Dense` pour répondre aux principaux problèmes liés à la rigidité : décider si un GCSP est rigide, identifier un sous-GCSP bien-rigide, identifier un sous-GCSP sur-rigide et minimiser un sous-GCSP bien- ou sur-rigide. Nous exposons ici comment utiliser cet algorithme pour résoudre ces différents problèmes.

Décider de la rigidité : Nous proposons d'utiliser la caractérisation par es_rigidité afin de décider si un GCSP est rigide ou pas. Décider si un GCSP est rigide revient alors à compter son nombre de DDL et vérifier qu'il est égal à son DDR, puis à s'assurer qu'il ne contient aucun sous-GCSP sur-es_rigide (cf. définition 49, page 188).

La première condition est vérifiée par un simple compte de DDL et un calcul du DDR du GCSP considéré. La seconde condition peut être vérifiée à l'aide de l'algorithme `ES_Rigide`. En effet, utilisé avec une surcharge valant DDR et non pas DDR+1, cet algorithme permet d'identifier un sous-GCSP sur-es_rigide s'il en existe un. Vérifier qu'un GCSP ne contient aucun sous-GCSP sur-es_rigide revient donc simplement à vérifier que le sous-GCSP retourné par un appel à `ES_Rigide`(S) avec surcharge valant DDR (au lieu de DDR+1) est vide.

Identifier un sous-GCSP bien-rigide : Afin d'identifier un sous-GCSP bien-rigide en se basant sur la caractérisation par es_rigidité, on peut utiliser directement l'algorithme `ES_Rigide` tel qu'introduit à la page 202. En effet, l'application de cet algorithme avec une surcharge valant DDR+1 permet de retourner un sous-GCSP bien- ou sur-es_rigide. Il suffit alors de s'assurer que le sous-GCSP retourné est bien-es_rigide (et pas sur-es_rigide) : un compte des DDL du sous-GCSP identifié, et une comparaison à son DDR déterminent s'il est bien-es_rigide ou sur-es_rigide.

L'algorithme `ES_Rigide` est prolongé jusqu'à ce qu'il retourne un sous-GCSP bien-es_rigide et non pas sur-es_rigide ; ceci est accompli en modifiant la condition d'arrêt de l'algorithme : il faut que le sous-GCSP identifié vérifie DDL=DDR pour qu'il soit retourné par l'algorithme.

Identifier un sous-GCSP sur-rigide : Ce problème est directement résolu par l'application de l'algorithme `ES_Rigide` avec une surcharge valant DDR au lieu de DDR+1, comme il a déjà été expliqué.

Minimiser un sous-GCSP bien- ou sur-rigide : L'étape de minimisation est semblable à celle proposée par Hoffmann *et al.* pour l'algorithme `Dense` (cf. algorithme 9, page 133).

Elle consiste à essayer de retirer un à un les objets du sous-GCSP sur- ou bien-es_rigide identifié par un appel à l'algorithme `ES_Rigide`, puis à calculer un nouveau sous-GCSP bien ou sur-es_rigide plus petit dans le sous-GCSP privé d'un objet. Chaque objet étant

examiné une fois au plus, l'opération de minimisation entraîne un sur-coût linéaire à l'algorithme `ES_Rigide`. l'algorithme `Minimal-ES_Rigide`, consiste donc simplement à remplacer l'appel à l'algorithme `Dense` par un appel à l'algorithme `ES_Rigide` dans l'algorithme 9, page 133. Il est de complexité $O(n^{\frac{d(d-1)}{2}+2} * (C + n * (n + m)))$.

7.3.2 Algorithme `ES_Rigide` pour l'opération de fusion

L'algorithme `Minimal-ES_Rigide` permet d'identifier un sous-GCSP minimal bien- ou sur-es_rigide, c-a-d. ne contenant que des sous-GCSP sous-es_rigides. Comme signalé à la section 7.3.1, il peut aussi être légèrement modifié pour n'identifier que des sous-GCSP sur-es_rigides et servir pour la détection de R-redondances et d'inconsistances (cf. section 6.1.4, page 169).

Nous proposons d'utiliser l'algorithme `Minimal-ES_Rigide` pour réaliser l'opération de fusion de notre nouvelle méthode. La phase de planification commence par éliminer les R-redondances du GCSP à décomposer, aussi l'algorithme ne peut-il identifier que des sous-GCSP bien-es_rigides (cf. section 7.2).

Cependant, l'algorithme `Minimal-ES_Rigide` peut identifier des sous-GCSP es_rigides possédant moins de $\frac{d(d+1)}{2}$ DDL. Ceci pose problème pour une méthode de rigidification récursive car l'assemblage ne peut être réalisé que si les agrégats partagent des sous-ensembles d'objets ayant DDR=$\frac{d(d+1)}{2}$ (cf. section 6.4, page 191). Or, si l'agrégat en lui-même ne vérifie pas DDR=$\frac{d(d+1)}{2}$, la proposition 16 (page 179) assure qu'il ne contient aucun sous-ensemble d'objets le vérifiant. Les sous-GCSP identifiés par fusion mais possédant moins de $\frac{d(d+1)}{2}$ DDL ne pourront donc pas être assemblés par rigidification récursive. Par exemple, un triplet de points alignés peut constituer un sous-GCSP rigide, mais celui-ci ne permet pas de définir un repère local et ne peut donc être utilisé pour les méthodes de rigidification récursive.

Afin de n'identifier que les sous-GCSP pertinents, nous proposons de modifier l'algorithme `ES_Rigide` de la façon suivante : l'ensemble M contient uniquement les ensembles DDR-minimaux ayant DDR=$\frac{d(d+1)}{2}$. Pour ce faire, c'est la fonction `DDR-minimaux` (cf. algorithme 16, page 183) qui est modifiée : elle ne considère que les modèles DDR-minimaux pouvant atteindre DDR=$\frac{d(d+1)}{2}$, et seulement les rigidifications associées qui leurs confèrent ce DDR maximum.

L'algorithme `Minimal-ES_Rigide` ainsi modifié ne produit donc que des sous-GCSP bien-es_rigides permettant de définir un repère local. De plus, si le GCSP est exempt de R-redondances/inconsistances (grâce à l'étape d'élimination des R-redondances) et de singularités (autres que celles représentées par ses contraintes booléennes), les sous-GCSP produits par cet algorithme sont bien-rigides (car la es_rigidité correspond alors à la rigidité).

Ceci constitue l'intérêt de notre opération de fusion comparée à celle de la méthode HLS : l'utilisation de la es_rigidité, l'élimination des R-redondances et le traitement des singularités liées aux contraintes booléennes assurent une plus grande justesse de l'opération, c-a-d. que le sous-GCSP qu'elle identifie a plus de chance d'être vraiment rigide que

celui retourné par l'opération de fusion de la méthode HLS.

7.4 Nouvelle opération d'extensions

L'opération d'extensions procède du même principe que celle de la méthode HLS (cf. section 5.4, page 137) : il s'agit d'inclure un maximum d'objets voisins de l'agrégat courant de façon rigide, en considérant ces voisins un par un seulement.

Notre nouvelle opération présente cependant deux différences majeures :

- elle est basée sur la es_rigidité,
- elle utilise l'*extension globale* pour inclure tous les objets frontières d'un ancien agrégats en une seule fois lorsque cela est possible.

Algorithme 18 Extensions ($S = (O, C)$: un GCSP ; d : la dimension ; $K = (O_K, C_K)$: un agrégat de S ; P : le plan d'assemblage) **retourne** $K' = (O'_K, C'_K)$: un agrégat de S contenant K

$K' \leftarrow K$
$Voisins \leftarrow \{o \in O \setminus O_K | \exists c \in C$ portant sur $o \cup O'$ et $O' \subset O_K\}$ *{Les objets candidats à l'extension sont pris dans le voisinnage de K'}*
Tant que $Voisins \neq \emptyset$ **Faire**
  $v \leftarrow$ `UnObjet(`$Voisins$`)` ; $Voisins \leftarrow Voisins \setminus \{v\}$
  $Liens \leftarrow \{c \in C | c$ porte sur $o \cup O'$ et $O' \subset O_K\}$
  Si $DDL(v) = DDL(Liens)$ **Alors** {extension possible}
    $O'_K \leftarrow O'_K \cup \{v\}$
    $C'_K \leftarrow C'_K \cup Liens$
    $Voisins \leftarrow Voisins \cup \{o \in O \setminus O'_K | \exists c \in C$ portant sur $\{v, o\} \cup O'$ et $O' \subset O'_K\}$
    Pour chaque $K'' \in$ `Agrégats-Contenant(`v, P`)` **tel que** $DDR(O'_K \cap O''_K, S) = \frac{d(d+1)}{2}$ et $F''_K \not\subseteq O'_K$ **Faire**
      $K' \leftarrow$ `Extension-Globale(`K', K''`)`
      $Voisins \leftarrow (Voisins \setminus F''_K) \cup \{o \in O \setminus (O'_K \cup O''_K) | \exists c \in C$ portant sur $\{o\} \cup O'$ et $O' \subset (O'_K \cup O''_K)\}$
    Fin Pour
  Fin Si
Fin Tant que
Return $K' = (O'_K, C'_K)$

L'opération d'extensions, décrite à l'algorithme 18, prend en entrée l'agrégat K produit par l'opération de fusion, ainsi que le GCSP et la dimension de l'espace géométrique considéré. K est renommé en K' dans l'algorithme afin de bien distinguer l'agrégat résultant de l'opération de fusion (K) de celui construit durant l'opération d'extension (K').

Chaque objet v voisin de l'agrégat K' est examiné tour à tour ; si l'extension vers cet objet voisin est *possible*, v et toutes les contraintes *Liens* le liant à K' sont intégrées dans l'agrégat courant. Le voisinnage est alors mis à jour de façon incrémentale, en intégrant les

objets voisins de l'objet v qui vient d'être intégré. Intuitivement, une extension est possible si l'agrégat en résultant est bien-es_rigide.

Lorsqu'un nouvel objet v est inclus par extension, la fonction `Agrégats-Contenant` extrait du plan d'assemblage P tous les anciens agrégats contenant v. Chaque ancien agrégat K'' est examiné tour à tour, et, si cela est possible, il est inclus en une seule étape par une extension globale dans K'. Intuitivement, une extension globale de K' sur K'' consiste à positionner tous les objets frontières de K'' dans K' par déplacement ; elle est possible si K' contient suffisamment d'objets de K'' pour permettre la définition de ce déplacement. Lorsqu'une extension globale de K' sur K'' est réalisée, le voisinnage de K' est mis à jour : les objets frontières de K'' sont retirés du voisinnage de K', alors que tout leur propre voisinnage lui est ajouté.

Nous allons maintenant détailler les deux étapes difficiles de l'extension, à savoir le test déterminant si une extension est possible et la réalisation d'une extension globale.

7.4.1 Extension possible

L'extension de K' sur v est *possible* uniquement si l'agrégat étendu est bien-es_rigide.

Vérifier qu'un sous-GCSP $K' \cup \{v\} = (O'_K \cup \{v\}, C'_K \cup Liens)$ est bien es_rigide dans un GCSP S consiste à s'assurer de deux conditions :

1. vérifier que son nombre de $DDL(K' \cup \{v\}) = DDR(O'_K \cup \{v\}, S)$,
2. vérifier que $K' \cup \{v\}$ ne contient aucun sous-GCSP sur-es_rigide.

Cependant, le DDR de tout agrégat retourné par l'opération de fusion vaut $\frac{d(d+1)}{2}$ (cf. section 7.3.2) et la proposition 15 (page 178) assure que l'ajout d'objets géométriques ne peut que faire croître le DDR, qui est par ailleurs borné par $\frac{d(d+1)}{2}$ (cf. proposition 18). Ainsi, $DDR(O'_K \cup \{v\}, S) = \frac{d(d+1)}{2}$, et $K' \cup \{v\}$ est donc bien-es_rigide si et seulement si $DDL(K' \cup \{v\}) = DDL(K') + DDL(v) - DDL(Liens) = \frac{d(d+1)}{2}$, c-a-d. $DDL(v) = DDL(Liens)$.

Afin d'assurer qu'un sous-GCSP ne contient aucun sous-GCSP sur-es_rigide, il faudrait normalement faire appel à l'algorithme `Sur-ES_Rigide` (cf. algorithme 17, page 202). Cependant cette opération est rendue inutile par la phase préliminaire d'élimination des R-redondances : comme nous l'avons expliqué à la section 7.2, toute sur-es_rigiditié correspond à une R-redondance ou une inconsistance structurellement identifiable et peut donc être éliminée. Le GCSP initial ne contient donc pas de sous-GCSP sur-es_rigide et, en l'absence de R-redondances et d'inconsistances, FAR n'introduit pas de nouvelle sur-es_rigidité. Ceci rend donc inutile la vérification de la seconde condition.

C'est pourquoi, pour déterminer si une extension est possible, il est suffisant de vérifier que $DDL(v) = DDL(Liens)$.

7.4.2 Extension globale

Un nouvel objet v inclus par extension dans l'agrégat courant K' peut déjà appartenir à d'anciens agrégats ; en effet, nous verrons que l'opération de condensation FAR laisse la

frontière des anciens agrégats dans le GCSP, ce qui permet, comme avec FA et MFA, le partage d'objets frontières entre agrégats.

On peut alors se demander s'il ne serait pas possible d'inclure en une seule fois dans K' tout un ancien agrégat K'' contenant v ? Pour le déterminer, il suffit de considérer l'ensemble des objets actuellement partagés par K' et K'' (noté $O'_K \cap O''_K$). Si cet ensemble d'objets permet de réaliser l'assemblage de K' et K'', alors on peut procéder à une extension globale de K' sur K'' : tous les objets frontières de K'' sont inclus en une seule fois dans K'. Cette extension globale de K' sur K'' n'utilise pas les contraintes présentes dans le GCSP ; elle correspond à réaliser l'assemblage de K'' et K' (cf. section 8.4, page 237) :

- déterminer le déplacement qui permet de positionner K'' dans K' à l'aide des objets qu'ils partagent (dont le dernier objet inclus, v),
- puis appliquer ce déplacement à tous les objets frontières de K'' afin de les positionner dans K'.

Le test déterminant si un ensemble d'objets $O'_K \cap O''_K$ partagé par deux agrégats K' et K'' permet leur assemblage est le suivant : $DDR(O'_K \cap O''_K, S) = \frac{d(d+1)}{2}$. En effet, nous avons vu qu'il faut, pour que l'assemblage soit possible, que les objets partagés permettent la définition d'un repère local ; or, nous avons proposé une réponse au problème du repère local avec la proposition 21, page 192 : un ensemble d'objets O permet la définition d'un repère local si et seulement si $DDR(O, S) = \frac{d(d+1)}{2}$. Bien entendu, il est inutile de procéder à l'extension globale si tous les objets frontières de l'agrégat K'' ont déjà été inclus par extensions dans K', ce qui est symbolisé dans l'algorithme 18 par le test $F''_K \nsubseteq O'_K$ où F''_K est la frontière de K''.

La correction d'une extension globale est assurée par la phase d'élimination des R-redondances ; en effet, si une extension globale introduit une sur-es_rigidité dans K', c'est que le GCSP contient une sur-es_rigidité ; or, la phase d'élimination des R-redondances/inconsistances assure que le GCSP est exempt de sur-es_rigidité (cf. section 7.2).

Exemple d'extension globale

Nous illustrons le principe et l'intérêt de l'extension globale sur le GCSP présenté à la figure 7.4-a. Ce GCSP en 2D est constitué de points et distances représentées par des segments dans cette figure. L'agrégat courant K' qui y est représenté est en phase d'extension ; il existe des contraintes le liant aux objets frontières A et B d'un ancien agrégat K'', ce qui place A et B dans le voisinnage de K'. L'agrégat K'' a pour frontière l'ensemble d'objets $\{A, B, C, D, E, F, G\}$.

Notre nouvelle opération d'extensions procède d'abord à l'inclusion de A. A est un point et n'est donc pas suffisant pour permettre l'extension globale de K' sur K'' en 2D. L'extension suivante inclut B dans K'. K' et K'' partagent maintenant une paire de points, ensemble d'objets permettant la définition d'un repère local : $DDR(\{A, B\}, S) = 3$ car A et B ne sont pas incidents. On procède donc à l'extension globale de K' sur K'' : les objets C, D, E, F et G sont globalement inclus en une seule étape ; ils seront résolus dans K' par déplacement à partir de leurs positions dans K'' ; ce déplacement sera calculé grâce aux positions relatives de A et B dans K' et K'' (cf. figure 7.4-c).

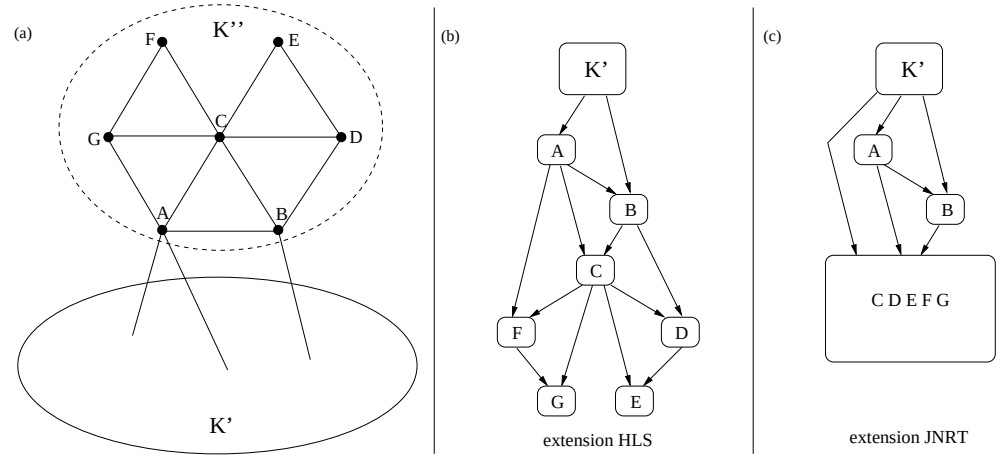

FIG. 7.4 – GCSP illustrant l'intérêt de l'extension globale

Lors de la phase d'assemblage, l'extension globale de K' sur K'' représente la résolution d'un seul système de trois équations linéaires à trois inconnues pour le calcul du déplacement[2], puis à la simple application du déplacement ainsi calculé à tous les objets frontières de K''.

En comparaison, l'opération d'extensions proposée par Hoffmann *et al.* aurait procédé de la façon suivante : elle aurait inclus tout d'abord A, puis B ; à partir de là, les contraintes portant entre les objets frontières de K'' auraient permis d'inclure par extension C, puis D, puis E, puis F et enfin G. Ceci représente cinq extensions, chacune générant un sous-système de deux équations quadratiques pour la résolution (cf. section 8.2, page 228), comme représenté à la figure 7.4-b.

L'extension globale permet donc de limiter le nombre d'extensions sur les objets appartenant à la frontière d'anciens agrégat au strict nécessaire : dès que l'extension globale est possible, elle est appliquée ; comme elle correspond en phase d'assemblage à un système d'équations très simple à résoudre, elle est toujours préférable à l'extension sur les objets frontières un à un.

7.4.3 Intérêt de la nouvelle extension

Comme nous l'avons déjà dit, la différence entre notre opération d'extensions et celle proposée par Hoffmann *et al.* réside dans deux points essentiels :

- notre extension est basée sur la es_rigidité alors que celle proposée par Hoffmann *et al.* repose sur la s_rigidité ;
- notre extension introduit la notion d'extension globale qui permet de réaliser de façon immédiate l'inclusion d'un ancien agrégat entier dès qu'elle est possible.

[2]Déterminer un déplacement en 2D consiste à déterminer un angle de rotation et un vecteur de translation.

Ceci confère plusieurs avantages à notre opération d'extensions vis à vis de celle de la méthode HLS.

Tout d'abord, la es_rigidité est plus *juste* que la s_rigidité ; grâce à la phase d'élimination des R-redondances et au traitement des singularités par les contraintes booléennes, la es_rigidité correspond assez bien à la rigidité et nos extensions sont donc généralement géométriquement correctes ; ce n'est pas toujours le cas de l'opération d'extensions de la méthode HLS, qui peut, par ailleurs, empêcher des extensions correctes (cf. section 5.4.2, page 139). Notons qui plus est que notre test déterminant si une extension est possible peut être vérifié en temps constant (puisque, dans un GCSP en dimension d exempt de sur-es_rigidité, le nombre de contraintes dans l'ensemble *Liens* est inférieur à $\frac{d(d-1)}{2}$).

D'autre part, le concept d'extension globale permet de limiter le nombre d'extensions réalisées et ainsi d'éviter la multiplication du nombre de sous-systèmes engendrés. Cette notion prend tout son sens lorsqu'un ancien agrégat K'' possédant un grand nombre d'objets frontières peut être inclus par extension globale dans l'agrégat courant K'. L'opération d'extensions de la méthode HLS, en revanche, procède toujours à autant d'extensions que les contraintes du GCSP le lui permette.

Qui plus est, l'extension globale permet de décider si un ancien agrégat est entièrement inclus dans l'agrégat courant ou pas : si l'extension globale de K' sur K'' n'est pas possible, c'est que K'' n'est pas entièrement inclus dans K'. L'opération d'extensions de la méthode HLS ne permet de se rendre compte qu'un ancien agrégat K'' est inclus entièrement dans l'agrégat courant K' que si tous les objets frontières de K'' ont été inclus dans K' un à un. Ainsi, si jamais tous les objets frontières de K'' ne peuvent pas être inclus par extension dans K', cette opération conclut que K'' n'était pas entièrement inclus dans K' même si suffisamment d'objets frontières de K'' ont été inclus pour permettre de procéder à l'assemblage de K'' et K'.

Notre opération d'extension est donc plus puissante et plus souvent correcte que celle de la méthode HLS. L'extension globale peut cependant s'avérer plus coûteuse puisqu'elle peut nécessiter le calcul du DDR de certains sous-ensembles d'objets. Toutes les heuristiques présentées à la section 6.2.4, page 184, peuvent être utilisées pour éviter de rendre ce calcul prohibitif.

7.5 Nouvelle opération de condensation

La nouvelle opération de condensation, appelée FAR pour *Frontier Automatic Rigidification*, repose, comme FA et MFA, sur la notion d'objets frontières. Cependant, nous avons constaté qu'il est inutile d'introduire un objet repère afin de pouvoir maintenir la rigidité de l'ensemble des objets frontières d'un agrégat condensé. Cette constatation repose sur la proposition suivante :

Proposition 24
Dans un GCSP bien-rigide en dimension d, les objets frontières F_K d'un agrégat K vérifient $DDR(F_K, S) = \frac{d(d+1)}{2}$.

Nous démontrons cette proposition par contradiction : soit S un GCSP bien-rigide et $K = (O', C')$ un agrégat ayant un ensemble F_K d'objets frontières tel que $DDR(F_K, S) < \frac{d(d+1)}{2}$; alors $DDR(F_K, S) < DDR(O', S)$ puisque tout agrégat identifié par fusion et extensions possède $\frac{d(d+1)}{2}$ DDL. Cela signifie que K dispose de déplacements qui ne peuvent pas être décrits par F_K ; autrement dit, fixer F_K est insuffisant pour fixer tous les déplacements de K. Or, les seuls objets de K pouvant encore être positionnés dans S', le GCSP S après condensation de K, sont ses objets frontières F_K. Alors S ne peut pas être bien-rigide, car l'assemblage de K avec S' laissera certains déplacements libres pour K qui représenteront des déformations pour S. □

Par exemple, en 3D, si un agrégat ayant DDR=6 possède seulement 2 points comme objets frontières, il admettra toujours une rotation sur l'axe défini par ces 2 points ; c'est le cas du GCSP présenté à la figure 2.10, page 44.

L'opération de condensation FAR, décrite à l'algorithme 19, distingue donc deux cas pour la condensation d'un agrégat K en fonction de sa frontière F_K :

1. $DDR(F_K, S) = \frac{d(d+1)}{2}$. La condensation consiste alors à retirer tous les objets internes de K ainsi que toutes les contraintes de K, à dupliquer les objets frontières de K afin qu'ils puissent être résolus indépendamment dans de futurs agrégats et à rigidifier automatiquement les copies des objets frontières de K par des *contraintes interfaces*.

2. $DDR(F_K, S) < \frac{d(d+1)}{2}$. Alors, la proposition 24 assure que le GCSP est sous-rigide et que K ne pourra pas être assemblé de façon rigide avec le reste du GCSP. K constitue donc une partie maximale bien-es_rigide, et devient un puits du plan d'assemblage, c-a-d. qu'aucun agrégat futur ne contiendra K ; rappelons que dans un GCSP sous-es_rigide, le plan d'assemblage est en fait un ensemble de plans d'assemblages de parties maximales bien-es_rigides.

   La condensation de K consiste alors simplement à retirer intégralement K du GCSP, et à dupliquer ses objets frontières pour qu'ils puissent être résolus indépendamment dans d'autres sous-GCSP maximaux bien-es_rigides.

   Notons que l'absence de sur-es_rigidités dans le GCSP assure que K ne pourra pas être inclus par une extension globale dans un futur agrégat puisque ses objets frontières ne vérifient pas $DDR(F_K, S) = \frac{d(d+1)}{2}$.

La complexité en temps de FAR est en $O(n_K * m_K + \mathrm{R} + C)$ où n_K est le nombre d'objets frontières dans K et m_K est le nombre de contraintes de S' incidentes à des objets de K. On considère que la duplication d'un objet géométrique et la traduction d'une contrainte s'effectuent en temps constant : dans la majorité des contraintes géométriques, le nombre d'occurrences d'une même variable dépasse rarement 3. R est la complexité du mécanisme de rigidification automatique `Rigidification-Automatique` que nous décrivons ci-après. C est la complexité du test $DDR(F_K, S) = \frac{d(d+1)}{2}$ déterminant si F_K est un représentant de K ; ce test implique le calcul d'un DDR, dont la complexité est polynomiale si le test de G-consistance peut être effectué en temps polynomial (cf. section 6.2.4, page 184).

Algorithme 19 FAR ($S = (O, C)$: le GCSP ; d : la dimension ; $K = (O_K, C_K)$: agrégat ; P : plan d'assemblage) **retourne** (S', P'), $S' = (O', C')$ est un GCSP équivalent et P' est le plan d'assemblage mis à jour

$O' \leftarrow O \setminus O_K$
$C' \leftarrow C \setminus C_K$
$F_K \leftarrow \{o \in O_K | \exists c \in C \wedge \exists o' \in O \setminus O_K \wedge c$ porte sur o et $o'\}$
Pour chaque $o \in F_K$ **Faire**
  $o' \leftarrow$ `Nouvelle-Instance(`o`)` *{duplication de o en o'}*
  $O' \leftarrow O' \cup \{o'\}$ *{insertion de la copie o' de o}*
  Pour chaque $c \in C' \setminus C_K$ **tel que** c porte sur o **Faire**
    $c' \leftarrow$`Substituer-Objet(`c, o, o'`)` *{substitution de o par o' dans c}*
    $C' \leftarrow C' - \{c\} \cup \{c'\}$
  Fin Pour
Fin Pour
Si $DDR(F_K, S) = \frac{d(d+1)}{2}$ **Alors**
  $C' \leftarrow C' \cup$ `Rigidification-Automatique(`F_K, S, d`)`
Sinon
  {Indiquer que le GCSP est sous- ou sur-rigide à sa jonction avec l'agrégat K}
Fin Si
Return $((O', C')$, `Planifier(`K, P`)`$)$

7.5.1 Mécanisme de rigidification automatique

Le mécanisme de rigidification automatique a pour but de produire une rigidification de l'ensemble des objets frontières d'un agrégat condensé. Comme nous l'avons expliqué à la section 6.2.2, page 178, il n'est pas aisé de proposer un mécanisme capable d'exhiber toutes les rigidifications possibles pour un ensemble d'objets quelconques. De plus, pour rigidifier un ensemble d'objets correctement, il faut encore choisir une rigidification G-consistante avec le GCSP considéré.

Nous montrons dans cette section qu'il est possible de proposer une rigidification de l'ensemble des objets frontières automatiquement à l'aide de la table des modèles DDR-minimaux (cf. section 6.2.2, page 180) et d'une table de modèles de *rigidifications simples*.

Le principe de la rigidification automatique est le suivant :

1. rigidifier un sous-ensemble B_K de F_K permettant la définition d'un repère local, c-a-d. vérifiant $DDR(B_K, S) = \frac{d(d+1)}{2}$ (cf. proposition 21, page 192) ;

2. rigidifier chaque objet frontière o restant relativement à B_K à l'aide d'un modèle de rigidification simple.

Nous allons détailler ces deux points dans les paragraphes qui suivent.

Rigidification d'un ensemble B_K DDR-minimal ayant DDR=$\frac{d(d+1)}{2}$

Rappelons tout d'abord que nous n'appliquons le mécanisme de rigidification automatique que si F_K vérifie $DDR(F_K, S) = \frac{d(d+1)}{2}$. La proposition 15 (page 178) assure alors que F_K contient un sous-ensemble B_K DDR-minimal qui vérifie aussi $DDR(B_K, S) = \frac{d(d+1)}{2}$.

Or, la table des modèles DDR-minimaux associe à chaque modèle l'ensemble des modèles de rigidification qu'il peut admettre ; Pour rigidifier automatiquement B_K, il suffit donc d'identifier le modèle de rigidification associé qui est G-consistant avec les contraintes du GCSP (cf. section 6.2.3, page 183).

Certifier la G-consistance d'une rigidification peut être coûteux dans le cas général. Aussi est-il préférable de choisir, si possible, un sous-ensemble B_K correspondant à un modèle DDR-minimal à DDR constant, ou dont la rigidification peut être certifiée G-consistante en temps polynomial à l'aide du mécanisme d'inférence à base de règles. Ces heuristiques ont déjà été présentées pour le calcul du DDR d'un ensemble d'objets à la section 6.2.4 (page 184).

La proposition 21 (page 192) assure que le sous-ensemble B_K des objets frontières ainsi rigidifié permet de définir un repère local. Tout objet frontière de $K \setminus B_K$ peut donc être positionné de façon rigide relativement à B_K. Ce positionnement est effectué en utilisant les *modèles de rigidifications simples.*

Modèles de rigidification simple

Un modèle de rigidification simple est un ensemble de contraintes qui permet de rigidifier un objet géométrique relativement à un ensemble d'objets géométriques déjà rigidifiés. Intuitivement, une rigidification simple correspond à une extension d'un ensemble d'objets rigides vers un nouvel objet. Plus formellement :

Définition 50 Modèle de rigidification simple
Soit C' un ensemble de contraintes portant sur un ensemble d'objets $\{o\} \cup O'$. C' est une rigidification simple de o avec O' dans un GCSP $S = (O, C)$ si et seulement si $C' \cup C_R$ est une rigidification de $\{o\} \cup O'$ G-consistante avec C, C_R étant une rigidification de O' G-consistante avec C.

Un ensemble de contraintes C' est un modèle de rigidification simple si et seulement si il existe un GCSP dans lequel il est une rigidification simple.

Les rigidifications simples permettent donc de construire incrémentalement des rigidifications d'ensembles d'objets quelconques. Considérons par exemple O' comme étant une paire de points $\{A, B\}$ en 2D et $o = C$ étant un troisième point. Dans un GCSP S où l'on connaît la rigidification $C_R = \{\text{Distance}(A, B)\}$ G-consistante de $\{A, B\}$, construire une rigidification de $\{A, B, C\}$ G-consistante avec le GCSP peut se faire en ajoutant la rigidification simple $C'_1 = \{\text{Incidence}(A, C)\}$, ou $C'_2 = \{\text{Incidence}(B, C)\}$ ou encore $C'_3 = \{\text{Distance}(A, C), \text{Distance}(B, C)\}$.

Ces trois rigidifications simples correspondent à deux modèles de rigidification simple pour la paire (Point3, {Point1,Point2}) :
{Incidence(Point1,Point3)} et {Distance(Point1,Point3), Distance(Point2,Point3)}.

Le nombre de modèles DDR-minimaux étant borné (cf. proposition 19, page 181), le nombre de modèles de rigidifications simples qu'il nous faut considérer est aussi borné :

Proposition 25 Nombre de modèles de rigidifications simples
Dans une classe de GCSP en dimension d admettant N types d'objets géométriques différents et M types de contraintes géométriques différentes, il existe moins de $\mathrm{A}_N^k * \mathrm{A}_M^{k*\frac{d(d+1)}{2}} * k^{\frac{d(d+1)}{2}}$ *modèles de rigidifications simples rigidifiant un objet par rapport à un ensemble d'objets à k éléments.*

Pour démontrer cette proposition, il faut constater que le nombre maximum de DDL d'un objet géométrique est $\frac{d(d+1)}{2}$ alors que le nombre de DDL minimum d'une contrainte est 1. Il est donc impossible de placer plus de $\frac{d(d+1)}{2}$ contraintes de façon non R-redondante sur un objet pour le positionner de façon rigide vis à vis d'un ensemble d'objets O'.

Si cet ensemble d'objets O' possède k éléments, il existe alors $k^{\frac{d(d+1)}{2}}$ façons de répartir un jeu de contraintes[3] sur les objets de O'.

Comme il existe N types d'objets différents et M types de contraintes différentes, on obtient bien que le nombre de modèles de rigidification simples pouvant positionner un objet quelconque par des contraintes quelconques par rapport à k objets est $\mathrm{A}_N^k * \mathrm{A}_M^{k*\frac{d(d+1)}{2}} * k^{\frac{d(d+1)}{2}}$.
□

Le nombre des modèles DDR-minimaux étant lui-même borné (cf. proposition 19, page 181), il en résulte qu'il existe un nombre fini de modèles de rigidifications simples permettant la rigidification d'un objet géométrique quelconque vis à vis d'un modèle DDR-minimal quelconque dans une classe de GCSP donnée. Nous donnons la table de modèles de rigidifications simples pour les classes de GCSP ({Points, Droites},{Distances, Angles, Incidences, Parallélismes}) en 2D à l'annexe C.

Rigidifier la frontière avec les modèles de rigidification simple

Pour rigidifier les objets frontières restant d'un agrégat après la rigidification de B_K, il suffit donc de rigidifier chacun d'eux indépendamment vis à vis de B_K à l'aide d'un modèle de rigidification simple G-consistant avec le GCSP. On considérera que tous les modèles sont stockés dans la *table des modèles de rigidification simple*, appelée *RSimples* dans l'algorithme 20.

Cet algorithme produit une rigidification C_R de l'ensemble des objets frontières F_K de l'agrégat condensé F_K. Remarquons que l'ensemble de contraintes interfaces introduites par le mécanisme de rigidification récursive peut contenir des contraintes valuées. Une

[3] Cette valeur tient pour les contraintes binaires seulement ; dans le cas de contraintes n-aires, le nombre de façon de poser les contraintes est aussi borné mais dépend de l'arité des contraintes.

Algorithme 20 Rigidification-Automatique (F_K : la frontière de l'agrégat courant ; $S = (O, C)$: le GCSP ; d : la dimension) **retourne** C_R : une rigidification de F_K

Si $DDR(F_K, S) < \frac{d(d+1)}{2}$ **Alors**
  Return $\emptyset$ *{cet agrégat ne pourra être assemblé : inutile de le rigidifier}*
Sinon Si $DDL($`Sous-GCSP-Induit`$(F_K, S)) = \frac{d(d+1)}{2}$ **Alors** {la frontière était es_rigide}
  $C_R \leftarrow \{c \in C | c$ porte sur $O_c \subset F_K\}$ *{inutile de chercher une autre rigidification}*
Sinon
  $(B_K, C_R) \leftarrow$ `DDR-Minimal-Représentant`(F_K, S, d) *{production d'un ensemble DDR-minimal B_K vérifiant $DDR(B_K, S) = \frac{d(d+1)}{2}$ et sa rigidification associée C_R}*
  Pour chaque $o \in F_K \setminus B_K$ **Faire**
    $Rigidifications \leftarrow$ `Rigidifications`$(o, B_K, RSimples)$ *{produit l'ensemble des rigidifications simples possible pour o vis à vis de B_K}*
    Répéter
      $C' \leftarrow$ `UnElément`$(Rigidifications)$; $Rigidifications \leftarrow Rigidifications \setminus \{C'\}$
    Jusqu'à `G-consistant ?`(C', C)
    $C_R \leftarrow C_R \cup C'$
  Fin Pour
Fin Si
Return C_R

contrainte interface valuée est introduite de sorte que les valeurs qui lui sont associées correspondent aux valeurs calculées dans l'agrégat condensé.

Par exemple, considérons les copies C_1 et J_1 des objets frontières C et J de l'agrégat $K1$ du GCSP *Ponts* (cf. section 7.1.1, page 197) ; C_1 et J_1 sont rigidifiés par une contrainte interface de distance après condensation de $K_1 = ABCHIJ$. Cette contrainte interface est formulée de sorte que la distance entre C_1 et J_1 soit toujours la même que celle entre C et J dans K_1 : Distance(C_1, J_1)=Distance(C, J).

Ainsi, la résolution de K positionne les instances originales de C et J et permet de reporter directement la valeur correspondante pour la contrainte interface Distance(C_1, J_1). Ceci assure que l'assemblage sera possible, car on retrouvera les mêmes solutions pour C_1 et J_1 dans le GCSP restant que pour C et J dans K_1.

Propriétés et complexité

L'algorithme `Rigidification-automatique` est correct puisqu'il est toujours possible de rigidifier tout objet d'un sous-GCSP bien-rigide vis à vis d'un sous-ensemble de ses objets permettant la définition d'un repère local.

Il est complet si les tables de modèles de DDR-minimaux, de modèles de rigidifications pour DDR-minimaux et de modèles de rigidifications simples sont complètes.

Sa complexité est en $O(n * C)$ où n représente le nombre d'objets géométriques de F_K et C la complexité du mécanisme d'inférence utilisé pour certifier la G-consistance. Nous

justifions cette complexité par le fait que la boucle `Répéter` interne n'est effectuée qu'un nombre borné de fois, indépendant de la taille de F_K ou de S. En effet, nous avons montré que le nombre de modèles de rigidification simple est borné par une fonction de la taille du DDR-minimal considéré (cf. proposition 25, page 217). La taille de B_K est elle-même bornée et indépendante de la taille de la frontière à rigidifier. On peut donc considérer que la boucle `Répéter` interne est effectuée un nombre constant de fois.

7.5.2 Problèmes posés par FAR

FAR pose un ensemble de problèmes comparable à ceux posés par MFA. En effet, les objets frontières demeurent dans le GCSP et l'opération peut consister à ne rien faire si tous les objets d'un agrégat sont dans sa frontière. Comme nous l'avons vu pour FAR, cela pose un problème de terminaison.

Problème de terminaison Lorsque tous les objets d'un agrégat sont dans sa frontière, FAR est l'opération identité. De plus, même lorsqu'il condense un agrégat, rien n'empêche que l'opération de fusion identifie le sous-GCSP condensé sempiternellement.

A ce problème, nous proposons de répondre de la même façon que pour FA et MFA : le réseau objets-contraintes doit être maintenu et mis à jour durant la condensation afin d'éviter qu'un ancien agrégat puisse être retourné par l'opération de fusion. La mise à jour adéquate a été décrite de façon détaillée à l'algorithme 13, page 149.

Pour FAR, il faut cependant y apporter une légère modification : si F_K ne vérifie pas $DDR(F_K, S) = \frac{d(d+1)}{2}$, ce qui correspond à la découverte d'une sous- ou sur-rigidité dans le GCSP comme nous l'avons expliqué en début de section 7.5, la mise à jour du réseau consiste alors à retirer tous les objets internes et toutes les contraintes de l'agrégat K condensé, et à dupliquer les objets frontières de K dans le réseau. Elle reflète donc, dans cet unique cas, l'exacte mise à jour du GCSP.

Problème d'assemblage L'assemblage avec FAR consiste, comme celui de FA et MFA, à utiliser la position des objets frontières F_K dans K et leurs doubles dans S' (le GCSP S après condensation de K par FAR) afin de déterminer le déplacement à appliquer à tous les objets de K pour les positionner dans S'. Cet assemblage est correct puisqu'il ne sera possible que si F_K vérifie $DDR(F_K, S) = \frac{d(d+1)}{2}$.

Cependant, FAR introduit des contraintes interfaces entre les objets frontières dupliqués. Ceux-ci sont donc résolus dans S' indépendamment[4] des objets frontières résolus dans K. Si les configurations des objets frontières sont incompatibles, l'assemblage ne pourra pas avoir lieu et aucune solution parasite ne sera donc engendrée pour $K \cup S'$. En revanche, ces solutions incompatibles seront générées séparément pour K et S', ce qui représente une perte de temps au moment de la résolution.

[4]pas tout à fait indépendamment, puisque nous avons vu que les valeurs associées aux contraintes interfaces valuées sont *mesurées* sur les objets frontières dans K.

Pour résoudre ce problème, il faut, comme pour MFA, établir un lien entre les objets frontières dans K et leurs copies dans S'. Ce lien peut être établi par des contraintes supplémentaires qui induisent une seule configuration dans S' par configuration dans K : contraintes imposant que les objets frontières respectent dans S' la même topologie que dans K, contraintes d'orientation d'angles, ... Ces contraintes ne sont pas introduites pour la planification, mais stockées dans un ensemble des contraintes annexes qui sera utilisé lors de la phase d'assemblage pour invalider au plus tôt les solutions incompatibles.

Correction de FAR

Pour démontrer que FAR est correct, il faudrait démontrer que les quatre points de la proposition 14 (page 142) transposée à la es_rigidité, sont vérifiés.

Ceci est malheureusement impossible, du moins en s'appuyant uniquement sur une démonstration structurelle basée sur le compte de DDL. En effet, le schéma de preuve que nous avons proposé pour l'opération de condensation CA (page 145) ne peut pas être simplement adapté pour FAR car la es_rigidité considère le DDR et non pas la valeur unique $\frac{d(d+1)}{2}$; or, on ne peut pas établir d'égalité entre le DDR de différents sous-ensembles d'objets.

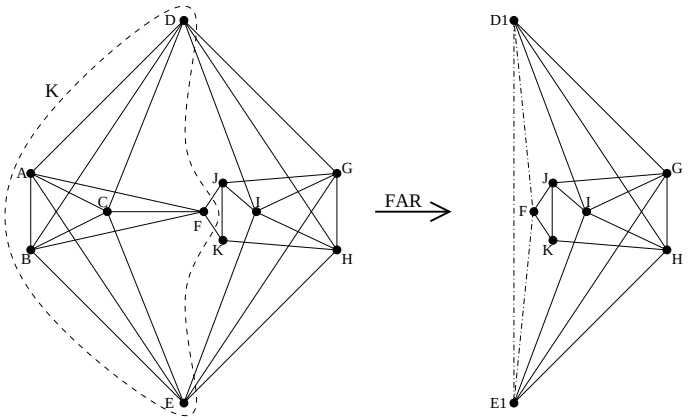

FIG. 7.5 – Exemple de GCSP **R-redondant** pour lequel FAR est incorrect

L'impossibilité de produire une démonstration purement structurelle est confirmée par le GCSP présenté à la figure 7.5. Ce GCSP en 3D est constitué de 11 points liés par 27 contraintes de distances représentées par des segments sur cette figure. On peut constater que ce GCSP est bien-es_rigide. Nous allons voir que FAR y fait apparaître une sur-es_rigidité, ce qui rend la condensation incorrecte vis à vis de la es_rigidité sur ce GCSP.

La partie gauche de la figure 7.5 présente le GCSP initial et la sélection d'un agrégat K bien-es_rigide produit par fusion et extension. Sa condensation par FAR est présentée à droite de la figure 7.5. On peut constater que le mécanisme de rigidification automatique a introduit trois nouvelles contraintes interfaces de distances entre les points D, E et F à la frontière de K.

C'est l'apparition de la contrainte de distance entre D et E qui provoque l'apparition d'une sur-es_rigidité. En effet, le sous-GCSP $DEGHI$ était auparavant bien-es_rigide mais l'apparition de la contrainte Distance(D, E) lui retire un DDL supplémentaire et le rend sur-es_rigide.

Ainsi, il existe des GCSP ne contenant aucune sur-es_rigidité dans lesquels FAR peut faire apparaître des sur-es_rigidités : FAR est donc généralement incorrect vis à vis de la es_rigidité.

Cependant, ce GCSP ne constitue pas un contre-exemple à l'opération de condensation FAR. En effet, bien qu'il soit bien-es_rigide, il est en réalité sur-rigide et contient une R-redondance : le sous-GCSP $ABCDEGHI$ correspond à la *double-banane* qui nous a servi de contre-exemple à la plupart des méthodes structurelles (cf. figure 2.10, page 44) ; nous avons à maintes reprises expliqué que ce sous-GCSP contient une R-redondance et est génériquement sur-rigide car les valeurs associées aux contraintes de distance ne permettent génériquement pas la coïncidence des points D et E. Or nous avons admis comme hypothèse que notre méthode travaille sur des GCSP exempts de R-redondances, et ce GCSP ne constitue donc pas un contre-exemple valide à notre opération de condensation. Nous ne sommes parvenu à exhiber aucun contre-exemple non R-redondant démontrant que FAR est incorrect.

Qui plus est, l'opération FAR a fait apparaître la sur-rigidité de ce GCSP sous forme de sur-es_rigidité ; la même chose s'est produite sur tous les contre-exemples que nous avons tenté de produire. FAR semble donc permettre de réparer les erreurs de la caractérisation approximative de la rigidité en rendant structurellement décelables des sur-rigidités qui étaient invisibles initialement.

Nous ne sommes pas encore parvenu à formaliser cette propriété ni à en produire une démonstration, mais cela constitue l'une des perspectives les plus prometteuses de la thèse : si l'opération FAR, ou une amélioration de cette opération, permet de transformer toute R-redondance présente dans un GCSP en sur-es_rigidité, notre méthode de planification ne sera plus limitée que par la présence d'inconsistances, dont nous avons expliqué qu'elles constituent un problème hors de la compétence de toute méthode de décomposition, et qui peut être traité lors de la phase d'assemblage (cf. section 6.1.3, page 168).

Ajout de l'agrégat au plan

L'opération de condensation a aussi pour tâche d'introduire l'agrégat K produit par fusion et extensions dans le plan d'assemblage. La mise à jour du plan d'assemblage est très semblable à celle proposée dans la méthode HLS : K est introduit sous la forme d'un nœud N_K dans le plan ; des arcs lient tous les agrégats et objets regroupés dans K au nœud N_K. Tout comme celui de FA et MFA, le plan produit est un DAG d'agrégats.

Chaque agrégat conserve toujours la trace des opérations qui l'ont produit : opération de fusion et distribution de flot ayant permis l'identification d'un sous-GCSP minimal bien-es_rigide ; chaque extension et extension globale ayant été produite durant l'opération d'extensions.

Nous verrons au chapitre suivant que ce plan est utilisé en phase d'assemblage pour

produire un *DAG de blocs*, similaire en un sens à celui produit par la décomposition équationnelle structurelle DM-CFC (cf. section 4.1.1).

7.5.3 Intérêt de FAR

Les opérations de condensation CA, FA et MFA proposées par Hoffmann *et al.*, et notre proposition d'une opération CA+MFA combinant CA et MFA, peuvent très bien être utilisées avec les nouvelles opérations de fusion et d'extensions que nous avons présentées dans ce chapitre. Prouver leur correction lorsqu'elles sont utilisées avec la es_rigidité n'est pas difficile : il suffit d'adapter la démonstration proposée au chapitre 5, page 145, pour montrer que CA, FA et MFA sont corrects avec la es_rigidité.

Cependant, l'opération de condensation FAR nous paraît meilleure que ces quatre opérations. La figure 7.6 regroupe les décompositions par CA, FA, MFA, CA+MFA et FAR du GCSP *Ponts* dans une même figure afin d'illustrer notre propos.

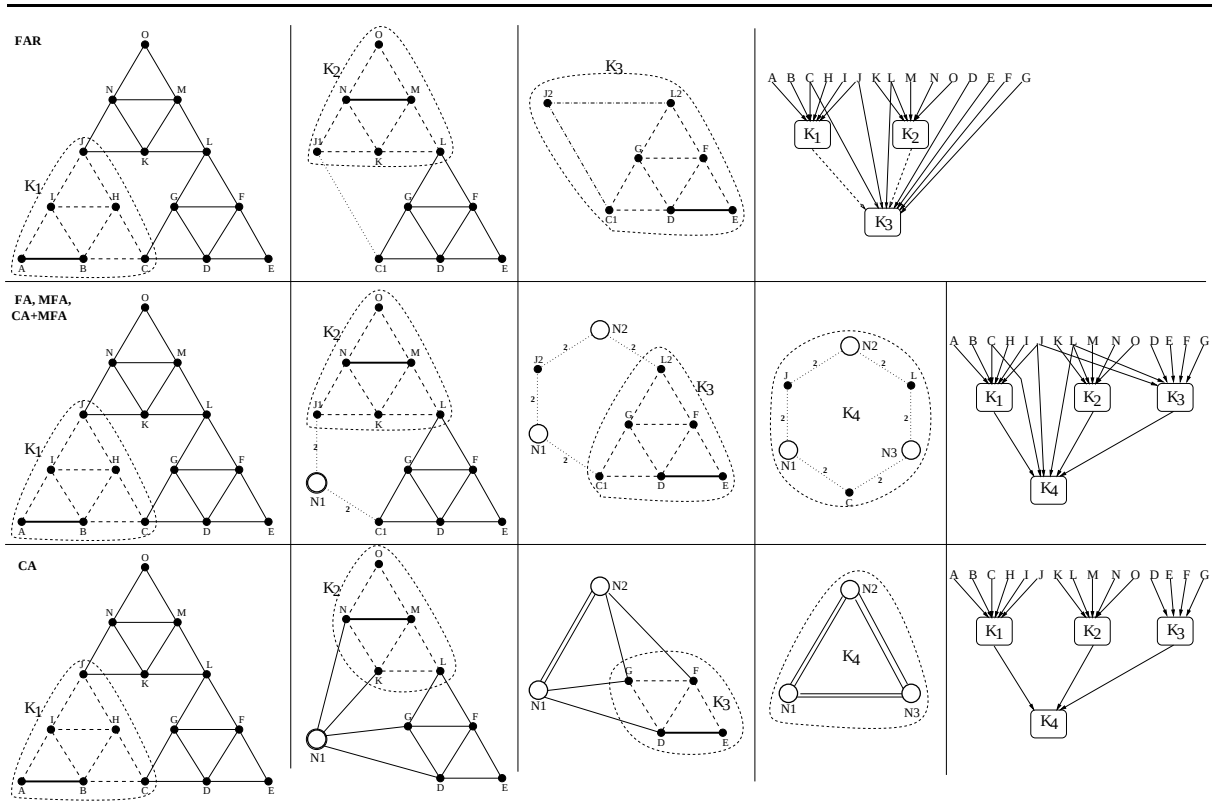

FIG. 7.6 – Comparaison de toutes les opérations de condensation sur le GCSP *Ponts*

On peut constater que seul FAR permet, sur ce GCSP, de gagner une itération et d'éviter la production d'une fusion de grande taille en dernière itération : le plus gros sous-système à résoudre en une fois contient respectivement 12 équations pour FA, MFA et CA+MFA, 6 équations pour CA et 2 équations pour FAR.

Ceci s'explique de la façon suivante : avec FA, MFA ou CA+MFA, les contraintes entre les objets frontières d'un agrégat restent inchangées, et le sous-GCSP induit est donc bien ou sous-es_rigide ; avec FAR, la frontière est rigidifiée et le sous-GCSP induit par les objets frontières est donc toujours bien-es_rigide après condensation. Ainsi, il existe généralement plus de contraintes entre les objets frontières avec FAR qu'avec les autres opérations de condensation. Or ce sont ces contraintes qui permettent de réaliser ces extensions intéressantes qui permettent d'inclure plusieurs agrégats en une seule étape : nous avons vu que FA, qui retire toutes les contraintes entre les objets frontières de l'agrégat condensé, ne permet pas ce genre d'extensions.

D'autre part, FAR utilise une vérification afin de déterminer si un agrégat pourra, par la suite, être assemblé de façon rigide avec le reste du GCSP : l'opération FAR ne produit une rigidification automatique que si l'ensemble des objets frontières F_K de l'agrégat K identifié vérifie $DDR(F_K, S) = \frac{d(d+1)}{2}$. Ceci permet à notre phase de planification de déceler des sur- ou sous-rigidités et de ne pas être trompée par la présence de certaines R-redondances difficilement identifiables.

Considérons encore une fois le GCSP *double-banane* présenté à la figure 2.10, page 44. Nous avons expliqué que CA, FA, MFA et même CA+MFA peuvent planifier ce GCSP R-redondant et génériquement sur-rigide comme s'il était bien-rigide.

Or FAR permet d'identifier que ce GCSP contient un problème. En effet, après que l'une des *bananes* ait été identifiée bien-es_rigide par fusion et extensions, la condensation par FAR n'introduira aucune rigidification car la frontière, constitué de deux points, ne permet pas la définition d'un repère local en 3D. Le GCSP sera alors coupé en deux et chaque *banane* sera résolue individuellement et retournée comme un sous-GCSP maximal bien-es_rigide. La présence d'une sous- ou sur-rigidité sera signalé à la jonction des deux bananes.

Notons que cette vérification d'assemblage possible peut être introduite dans les autres opérations de condensation et n'est donc pas spécifique à FAR : CA, FA, MFA et CA+MFA seraient alors capable d'identifier le problème de la *double-banane* eux aussi.

7.6 Conclusion

Nous avons présenté dans ce chapitre la phase de planification d'une nouvelle méthode de rigidification récursive. Cette phase de planification tire parti des résultats exposés au chapitre 6 pour essayer de lever un maximum des limitations des méthodes de décomposition géométrique purement structurelles. Pour ce faire :

- elle utilise la es_rigidité pour caractériser la rigidité dans un GCSP. La es_rigidité permet de prendre en compte les singularités d'un GCSP introduites sous forme de contraintes booléennes grâce au concept de degré de rigidité.
- elle utilise un opérateur d'élimination de R-redondances/inconsistances qui sont les autres sources d'erreur des méthodes structurelles. Cet opérateur peut s'avèrer être incomplet, et il peut donc demeurer certaines R-redondances dans les GCSP traités.

Nous avons présenté de nouveaux opérateurs de fusion, d'extensions et de condensation tirant avantage de l'utilisation de la es_rigidité, du concept de degré de rigidité et de l'absence de sur-es_rigidité due à l'opérateur d'élimination des R-redondances/inconsistances.

Nous avons exposé les avantages, en termes de justesse géométrique et de puissance de nos nouvelles opérations, mais il ne faut pas oublier que l'utilisation du degré de rigidité peut entraîner une complexité exponentielle dans le pire cas pour toutes ces opérations.

Nous avons identifié au chapitre 6 des classes de problèmes pratiques pour lesquelles le calcul du DDR et le test de G-consistance peuvent toujours être effectués de façon complète en temps polynomial. Pour les autres classes de GCSP, nous avons proposé des approches heuristiques du calcul du DDR et de la vérification de G-consistance permettant parfois d'éviter la complexité exponentielle. Dans tous ces cas, les nouvelles opérations de fusion, d'extensions et de condensations sont aussi performantes que puissantes.

Chapitre 8

Une nouvelle phase d'assemblage

Sommaire

Ce chapitre présente la phase d'assemblage que nous proposons pour notre nouvelle méthode de rigidification récursive [Jermann *et al.*, 2000a; 2000b]. La phase d'assemblage consiste à ramener la résolution du GCSP décomposé à la résolution d'un DAG de blocs, puis à résoudre ce DAG de blocs par la méthode proposée par Bliek, Neveu et Trombettoni [Bliek *et al.*, 1998].

Le plan produit en phase de planification est transformé en un DAG de blocs d'équations de la façon suivante : chaque opération de fusion correspond à un bloc ; chaque extension correspond à un bloc. Un bloc d'équations calcule ses variables de sortie à partir des valeurs de ses variables d'entrée ; il dépend donc des blocs calculant les valeurs de ses variables d'entrée, ce qui est modélisé par un arc. L'utilisation de notre nouvelle opération d'extensions entraîne l'introduction de nouvelles contraintes, les contraintes interfaces, dont il faut générer les équations correspondantes.

La résolution du DAG s'effectue par un algorithme de retour-arrière dans le DAG de blocs. Le bloc courant est résolu par les techniques de résolution par intervalles (cf.

section 3.2.3), ce qui permet l'obtention de toutes ses solutions. Le mécanisme de retour-arrière permet de remettre en cause le choix d'une solution d'un bloc précédent lorsque le bloc courant n'admet pas de solutions. Il permet aussi d'obtenir toutes les solutions du GCSP initial. Nous présentons plusieurs stratégies de propagation, et différents mécanismes de retour-arrière pouvant être utilisés pour la résolution du DAG de blocs.

La dernière étape de la résolution consiste à replacer tous les sous-GCSP dans un unique repère local, ce qui nécessite une passe d'unification des repères locaux, partant du repère local des agrégats puits du plan d'assemblage et remontant vers les sources du plan.

8.1 Aperçu général

Les opérations de fusion et d'extension de la phase de planification génèrent des sous-systèmes d'équations, appelés *blocs*, qui peuvent être résolus en séquence. Le bloc correspondant à l'une de ces opérations contient toutes les contraintes utilisées par l'opération. Prenons un exemple de fusion et d'extension dans le GCSP *Ponts* (cf. figure 7.1, page 197) : la fusion des points A et B produit un bloc B_1 contenant la contrainte Distance(A, B) ; l'extension vers le point I produit un bloc B_2 contenant les contraintes Distance(A, I) et distance(B, I).

Chaque bloc résout un ensemble de variables de sortie et attend en entrée un ensemble de valeurs pour ses variables d'entrée. Cette relation de dépendance établit un ordre partiel pour la résolution des blocs que nous représentons par des arcs : il y a un arc entre le bloc B_1 et le bloc B_2 car B_2 attend en entrée les coordonnées des points A et B pour calculer celles du point I.

Le première étape de la phase d'assemblage consiste à produire ce DAG de blocs en générant les blocs correspondant à chaque étape de fusion et à chaque extension et en établissant leurs dépendances. Le DAG du GCSP *Ponts* est présenté en figure 8.1.

La résolution du GCSP consiste alors à résoudre le DAG de blocs. Pour cela, nous utilisons la méthode de résolution proposée par Bliek, Neveu et Trombettoni [Bliek *et al.*, 1998]. Elle consiste à résoudre chaque bloc par intervalles (cf. section 3.2.3, page 73) dans un ordre compatible avec le DAG. A chaque fois qu'une solution d'un bloc est obtenue, on tente de résoudre le bloc suivant. Lorsqu'un bloc n'a pas ou plus de solutions, un mécanisme de retour-arrière permet de remonter au bloc précédent et de remettre en cause la solution qui y a été proposée. Ce même mécanisme assure la complétude de la résolution : lorsque le dernier bloc est résolu, cela produit une solution du GCSP entier. Le mécanisme de retour-arrière peut alors être utilisé afin d'entraîner la recherche de la solution suivante ; ceci permet une résolution complète d'un GCSP décomposé qui n'est pas proposée dans la plupart des méthodes de décomposition géométriques présentées au chapitre 4.

Lorsqu'une solution est obtenue pour le GCSP entier, on doit procéder à l'assemblage, qui consiste à unifier les repères locaux. En effet, nous avons expliqué que les blocs d'un même agrégat sont résolus dans un repère local qui est choisi dans le bloc de fusion de l'agrégat. Les valeurs des variables calculées dans les blocs d'un agrégat dépendent donc du repère local choisi.

La dernière étape de la phase d'assemblage consiste ainsi à unifier tous les repères locaux afin d'obtenir les valeurs de tous les paramètres géométriques dans un seul repère. Cette étape est effectuée en calculant les changements de repère qui doivent être appliqués pour assembler les différents agrégats.

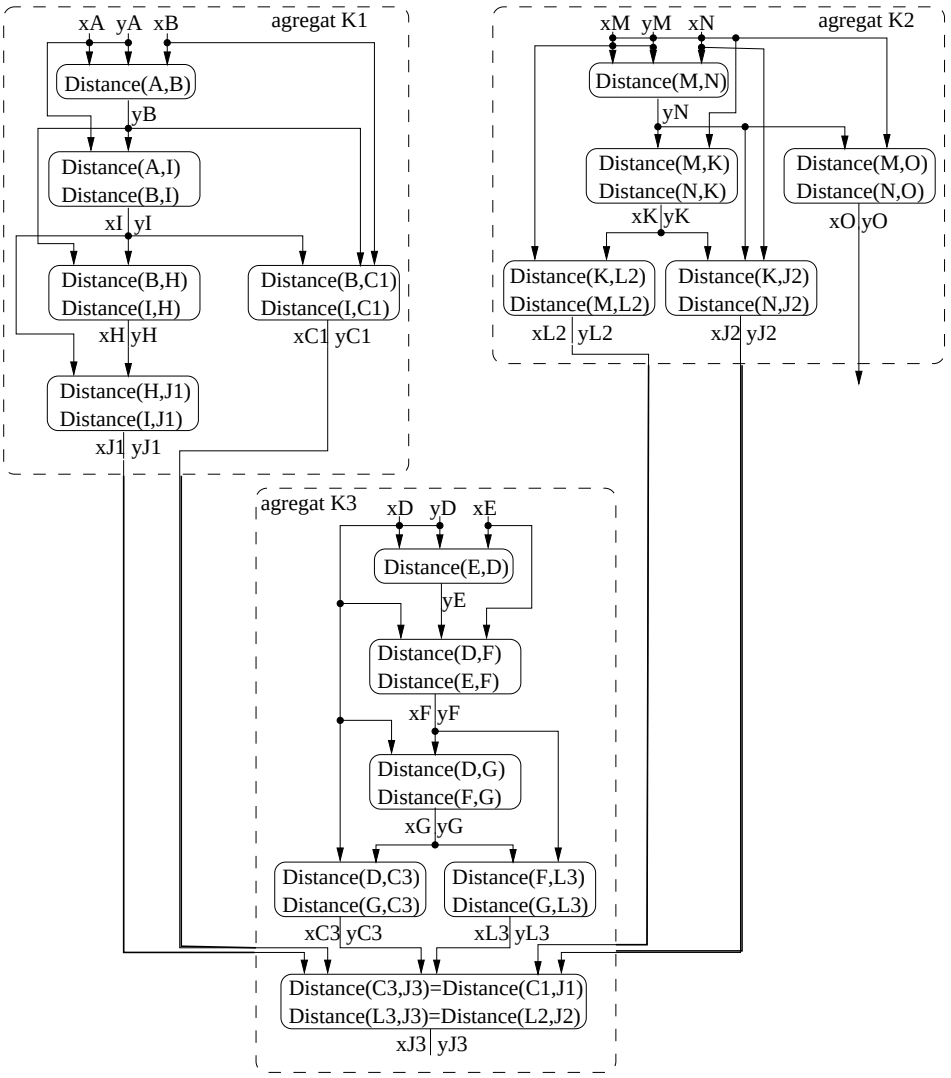

FIG. 8.1 – DAG de blocs correspondant au GCSP *Ponts*

Le schéma général de la phase d'assemblage est donc le suivant :

1. DAG $\leftarrow$ `GénérationDAG(`S, P`)`
2. $\mathbb{S} \leftarrow$ `RésolutionDAG(`DAG`)`
3. $\mathbb{S} \leftarrow$ `UnificationRepères(`$P, \mathbb{S}$`)`

A la fin de la phase d'assemblage, $\mathbb{S}$ contient l'ensemble des solutions du GCSP S. P représente le plan d'assemblage produit pour le GCSP S en phase de planification.

Dans les sections qui viennent, nous allons développer chacune des étapes de la phase d'assemblage : génération du DAG de blocs, résolution du DAG de blocs et unification des repères locaux.

8.2 Génération du DAG de blocs

Dans la phase d'assemblage, chaque sous-système bien-contraint issu de la phase de planification constitue un *bloc* de résolution. Un bloc B est constitué d'un ensemble E_B d'équations portant sur un ensemble V_B de variables. Parmi ces variables, on distingue 2 sous-ensembles : V_B^e les *variables d'entrée* du bloc et V_B^s les *variables de sortie* du bloc. Les variables de sortie V_B^s sont les variables du bloc qui seront déterminées par la résolution des équations E_B du bloc B ; les variables d'entrée V_B^e doivent déjà être instanciées pour que le bloc B puisse l'être. Intuitivement, elles représentent des paramètres du bloc qui n'est bien-contraint que lorsque ces paramètres sont tous déterminés.

Par exemple, considérons un bloc B constitué de l'équation $x + y = 0$. Ce bloc possède deux variables ($V_B = \{x, y\}$) et son ensemble d'équations est réduit à une équation ($E_B = \{x + y = 0\}$). Ce bloc permet de résoudre une seule variable, et on peut donc choisir soit $V_B^e = \{x\}$ et $V_B^s = \{y\}$, soit $V_B^e = \{y\}$ et $V_B^s = \{x\}$. Il est possible que l'ensemble des blocs impose l'un de ces choix ; par exemple, s'il existe un bloc qui ne permet de résoudre que la variable x, alors le bloc B doit forcément résoudre la variable y. Ceci donne un aperçu des liens existants entre les blocs : un bloc produisant une variable en sortie est lié à tous les blocs attendant la valeur de cette variable en entrée.

Les blocs sont déduits de la phase de planification du GCSP :

- une opération de fusion engendre un bloc, dit *de fusion* ;
- une extension (ou une extension globale) engendre un bloc, dit *d'extension*.

Pour produire le bloc correspondant à une étape de fusion ou d'extension, on considère le sous-GCSP correspondant à cette étape : les contraintes utilisées dans le bloc correspondent à toutes les contraintes utilisées durant l'étape. Ce sont donc les contraintes portant sur les objets identifiés par l'opération de fusion dans le cas d'un bloc de fusion, ou les contraintes utilisées pour intégrer le nouvel objet dans le cas d'un bloc d'extension.

Au niveau variables-équations, un bloc est constitué des paramètres modélisant les objets géométriques qu'il résout, et de l'ensemble des équations modélisant les objets (relations internes, cf. définition 5, page 16) et les contraintes (relations externes, cf. définition 6, page 17) qu'il contient.

Remarquons que les objets frontières peuvent appartenir à plusieurs agrégats. Nous avons expliqué qu'ils sont alors dupliqués durant les opérations de condensation qui mettent à jour le GCSP. Cette duplication signifie que leurs paramètres aussi sont dupliqués et substitués dans les équations modélisant des contraintes reportées sur l'objet dupliqué le cas échéant.

Par exemple, dans le GCSP *Ponts*, le point J apparaît dans les agrégats K_1, K_2 et K_3. Les variables qui le définissent, x_J et y_J sont alors dupliquées dans chacun de ces agrégats : dans K_1, J sera représenté par les variables x_{J_1}, y_{J_1} ; dans K_2, J_1 sera représenté par x_{J_2},

y_{J_2} ; et dans K_3, J_2 sera représenté par x_{J_3}, y_{J_3}.

Ainsi, plusieurs instances des objets frontières sont résolues au cours de la phase d'assemblage. Lors de la phase d'unification des repères locaux, les agrégats seront assemblés en utilisant le fait que toutes les instances d'un objet géométrique représentent en fait le même objet. Ceci permettra de calculer les changements de repères qui font coïncider les différentes instances des objets géométriques.

Nous avons expliqué que les contraintes interfaces introduites par le mécanisme de rigidification automatique utilisé dans la condensation FAR (cf. section 7.5.1, page 215) peuvent, s'il s'agit de contraintes valuées, faire intervenir plusieurs instances des mêmes objets géométriques afin d'assurer que les propriétés métriques sont respectées entre toutes les instances d'objets appartenant à la frontière d'un même agrégat.

Par exemple, dans le GCSP *Ponts*, les variables x_{J_1}, y_{J_1} (resp. x_{J_2}, y_{J_2}) sont calculées lors de la résolution des blocs de l'agrégat K_1 (resp. K_2). Lors de la résolution des blocs de l'agrégat K_3, le bloc calculant x_{J_3}, y_{J_3} contient les deux contraintes interfaces liant J_2 à C_1 et J_2 à L_2. Ces contraintes sont des contraintes de distance, et pour refléter la distance entre ces points résultant des résolutions des agrégats K_1 et K_2, les contraintes interfaces qui ont été introduites lors de la condensation de K_1 et lors de celle de K_2 sont Distance(J_2, L_2) = Distance(J_1, L) et Distance(J_2, C_1) = Distance(J, C). Ces équations utilisent donc les valeurs calculées pour les variables de C, J et L dans K_1 et K_2.

8.2.1 Bloc de fusion

Une étape de fusion produit un sous-GCSP bien-es_rigide, c-a-d. un sous-système d'équations bien-contraint dans un repère local ; une telle étape correspond donc à un bloc dont les variables d'entrée permettent de fixer un repère local.

Par exemple, l'étape de fusion produisant l'agrégat $K_1 = AB$ engendre un bloc B_1 contenant l'équation Distance(A, B) ; l'ensemble de ses variables est donc x_A, y_A, x_B et y_B. Les points ne possèdent pas de relations internes dans la modélisation que nous utilisons.

Dans un bloc de fusion, on peut choisir différentes variables d'entrée, mais ce choix doit permettre la définition d'un repère local. En dimension d, il faut donc choisir $\frac{d(d+1)}{2}$ variables d'entrée parmi les variables d'un ensemble d'objets permettant de définir un repère local.

La distribution du flot issu de la contrainte virtuelle R dans l'exécution de l'algorithme `ES_Rigide` qui a produit la fusion considérée représente une aide pour choisir les variables d'entrée. En effet, chaque unité de flot distribué depuis la contrainte virtuelle R sur un des objets auquel elle est liée représente la sélection d'une variable parmi les variables modélisant un objet.

Par exemple, le flot issu de R pour l'étape de fusion produisant l'agrégat $K_1 = AB$ vaut 3. Cette valeur peut être répartie de n'importe quelle façon entre les points A et B, par exemple 2 vers A et 1 vers B ; dans ce cas, cela indique que l'on peut choisir 2 variables d'entrée parmi les variables modélisant A et 1 variable d'entrée parmi celles modélisant B.

Ce flot ne permet cependant pas toujours de décider l'ensemble exact de variables à choisir. Par exemple, en 3D, si le flot issu de R est distribué sur 3 points à raison de 2 par point, cela indique qu'il faut choisir 2 variables parmi les variables modélisant chacun de

ces points ; cependant, choisir les variables x et y de chacun de ces points représente un choix invalide : ces variables ne permettent pas de définir un repère de l'espace géométrique car elles sont dépendantes : elles ne définissent pas une base de l'espace géométrique.

Afin de résoudre ce problème, on peut ajouter une colonne supplémentaire à la table des modèles DDR-minimaux qui indique les choix de variables valides et invalides parmi les variables modélisant les objets d'ensembles DDR-minimaux pour définir un repère local. Notons que ces modèles de sélection de variables d'entrée valides ne peuvent être proposés que pour les modèles DDR-minimaux ayant DDR=$\frac{d(d+1)}{2}$, qui sont les seuls permettant la définition d'un repère local (cf. proposition 21, page 192).

Par exemple, si l'on considère le modèle DDR-minimal {Point1, Point2, Point3} en 3D, chaque point étant représenté par ses trois coordonnées cartésiennes (x,y,z), les schémas de sélection de variables d'entrée *invalides* sont : {x1,y1,z1,x2,y2,z2}, {x1,y1,x2,y2,x3,y3}, {x1,z1,x2,z2,x3,z3}, {y1,z1,y2,z2,y3,z3} ; tout autre schéma est valide.

Une fois les variables d'entrée d'un bloc de fusion choisies, les variables restantes sont toutes des variables de sortie.

8.2.2 Bloc d'extension

Une étape d'extension intègre un objet géométrique à l'aide de contraintes permettant de fixer exactement cet objet à partir d'objets déjà fixés. Cela constitue un sous-système bien contraint : toutes les variables représentant l'objet géométrique inclus par extension peuvent être déterminées par les contraintes utilisées par l'extension.

Par exemple, l'étape d'extension qui inclut le point I à l'agrégat K_1 engendre un bloc B_2 contenant les équations Distance(A, I) et Distance(B, I). Les variables de ce bloc sont x_A, y_A, x_B, y_B, x_I et y_I.

Le cas des extensions globales a été brièvement décrit à la section 7.4.3, page 212 : une extension globale inclut tous les objets frontières d'un agrégat en une seule étape. Cette étape correspondra à un unique bloc. Les équations de ce bloc permettent de calculer un changement de repère (ou déplacement) à partir d'un ensemble d'objets dont on connaît la configuration dans les repères locaux de deux agrégats différents K et K'. Ensuite, ce changement de repère est appliqué à tous les objets frontières de K' (si l'extension globale s'est faite de K sur K') afin de les positionner dans le repère local de K.

Dans un bloc d'extension, les variables de sortie sont toujours les variables modélisant l'objet qui est inclus par cette extension. Par exemple, pour l'extension de K_1 sur I dans le GCSP *Ponts*, les variables de sortie du bloc B_2 sont x_I et y_I. Les autres variables sont des variables d'entrée.

8.2.3 Liens entre les blocs

Un DAG de blocs est un graphe orienté acyclique dont les nœuds sont des blocs. Il existe un arc d'un bloc A vers un bloc B dans ce DAG si le bloc B possède une variable d'entrée v qui est une variable de sortie du bloc A. Cet arc représente une relation de précédence pour la résolution des blocs : A doit être résolu avant B.

La génération des blocs de fusion et des blocs d'extension induits par le plan d'assemblage d'un GCSP engendre un DAG de blocs : les variables d'entrée et de sortie de blocs établissent les liens entre les blocs dans un même agrégat (et même entre agrégats à cause des contraintes interfaces valuées qui établissent des liens entre les différentes instances des objets géométriques). Ce DAG fournit une relation de précédence entre les blocs. Cette relation constitue un ordre partiel pour la résolution des blocs.

La figure 8.1 représente le DAG de blocs ainsi engendré par la planification du GCSP *Ponts*. Ce DAG illustre bien les différents points que nous avons évoqués pour la génération des blocs correspondant à des fusions ou à des extensions, et la liaison des blocs entre eux.

Nous décrivons maintenant les différentes méthodes de résolution d'un DAG de blocs que nous proposons, toutes basées sur les techniques de résolution par intervalles.

8.3 Résolution du DAG de blocs

Le schéma général de la méthode de résolution du DAG de blocs est celui de la méthode proposée par Bliek, Neveu et Trombettoni [Bliek *et al.*, 1998] :

- Une séquence B de blocs représentant un ordre total sur les blocs compatible avec le DAG est produite par un simple tri topologique [Cormen *et al.*, 1990] du DAG en $O(n+m)$.
- Chaque bloc est résolu par intervalles ; différentes stratégies de propagation peuvent être appliquées durant cette résolution.
- Lorsqu'une solution d'un bloc est obtenue, on peut résoudre le bloc suivant dans la séquence B.
- Si un bloc de la séquence n'admet pas (ou plus) de solutions, un mécanisme de retour-arrière sélectionne un bloc déjà résolu de la séquence B pour lequel il faut chercher une nouvelle solution. Divers mécanismes de retour-arrière peuvent être proposés, depuis le plus simple qui sélectionne simplement le bloc précédent dans la séquence (retour-arrière chronologique), jusqu'au plus sophistiqué qui utilise une gestion de *no-goods* et la structure du DAG afin d'identifier le bloc de retour le plus adéquat.
- A chaque fois que le dernier bloc de la séquence B est résolu avec succès, une solution complète, constituée de toutes les valeurs calculées pour tous les paramètres des objets géométriques, est produite. Le mécanisme de retour-arrière peut alors être relancé afin de trouver la prochaine solution.
- La résolution du DAG est terminée lorsque toutes les solutions complètes ont été identifiées.

Ce schéma de résolution assure la complétude de la résolution du DAG. Nous allons en étudier les différentes étapes.

Nous proposons aussi plusieurs nouvelles stratégies de propagation durant la résolution d'un bloc, plusieurs mécanismes de retour-arrière et quelques heuristiques permettant d'accélérer la résolution.

8.3.1 Stratégies de résolution

Chaque bloc est résolu par intervalles. Nous avons décrit les grandes lignes de ce type de méthodes à la section 3.2.3 ; elle repose sur l'utilisation de l'arithmétique d'intervalles pour les calculs, et des opérations de filtrage et de découpage d'intervalles pour la résolution.

La stratégie de résolution consiste à tirer parti de la structure particulière du GCSP décomposé pour appliquer les opérations de filtrage et de découpage. Afin de bien comprendre cela, il faut considérer le GCSP entier S, c-a-d. toutes les équations et toutes les variables réelles du problème. Un bloc B représente un sous-système du GCSP entier, c-a-d. un sous-ensemble d'équations et un sous-ensemble de variables (les variables de sortie du bloc). Un agrégat K est constitué d'un ensemble de blocs et représente donc aussi un sous-système du GCSP entier. Nous proposons 3 façons de résoudre un bloc d'équations B dans un agrégat K dans le GCSP S :

1. *Résolution par blocs* ; dans ce cas le filtrage et le découpage sont limités aux équations et aux variables de sortie du bloc B. Aucune propagation n'est effectuée sur le reste du GCSP. C'est la méthode minimale pour résoudre le bloc B ; elle est purement locale dans le sens où elle ne tient pas du tout compte du reste du GCSP.
2. *Résolution par blocs avec propagation globale* ; dans ce cas, le découpage reste cantonné aux variables de sorties de B mais le filtrage s'effectue sur l'ensemble des équations et variables des blocs *atteignables* à partir de B.
3. *Résolution globale* ; dans ce cas, la séquence de blocs est utilisée uniquement comme une heuristique de choix de variable pour l'opération de découpage : seules les variables du bloc courant sont soumises à la dichotomie jusqu'à ce que tous les intervalles correspondants soient suffisamment petits. Le filtrage, lui, s'effectue sur le système complet.

Les 2 premières stratégies se distinguent par la puissance de leur filtrage :
- La résolution par blocs est une résolution locale puisqu'elle ne tient compte que des informations internes au bloc courant. Cette résolution peut donc produire des solutions inconsistantes avec le reste du GCSP.
- La résolution par blocs avec propagation globale permet de prendre en compte le maximum d'informations disponibles par filtrage ; en effet, dans cette résolution, chaque réduction du domaine de l'une des variables du bloc courant entraîne une propagation dans le reste du GCSP.

Dans la seconde stratégie, nous proposons d'utiliser tous les blocs *atteignables* pour la propagation. Un bloc A est atteignable à partir du bloc courant B s'il existe un chemin de B à A dans le graphe *non-orienté* de blocs correspondant au DAG dans lequel les blocs déjà résolus ont été retirés. L'ensemble des blocs atteignables représente l'ensemble maximal des blocs pouvant être utilisés pour propager des informations durant la résolution de B ; les blocs déjà résolus ne peuvent pas fournir plus d'information qu'ils n'en donnent, et pour les blocs non-atteignables, il n'existe pas de séquence de propagation permettant d'utiliser les contraintes ou les variables qu'ils contiennent.

Si cette propagation réduit le domaine d'une variable à l'intervalle vide, c'est que le sous-espace de recherche exploré dans le bloc courant est inconsistant. La branche courante de la résolution de B est alors abandonnée, évitant la génération de solutions inutiles pour le bloc B.

Cependant, la propagation a un coût qui peut être élevé, surtout si l'on utilise une consistance forte comme la 3B ou la 4B et il peut alors être plus rapide de résoudre le bloc sans propagation globale.

Pour tirer parti de la propagation à moindre coût, on pourrait imaginer divers degrés de propagation, par exemple limiter la propagation aux équations et variables de l'agrégat K courant. Il est cependant malaisé d'estimer la validité d'une propagation limitée, aussi nous ne présenterons des expérimentations que pour les 3 stratégies proposées ci-dessus.

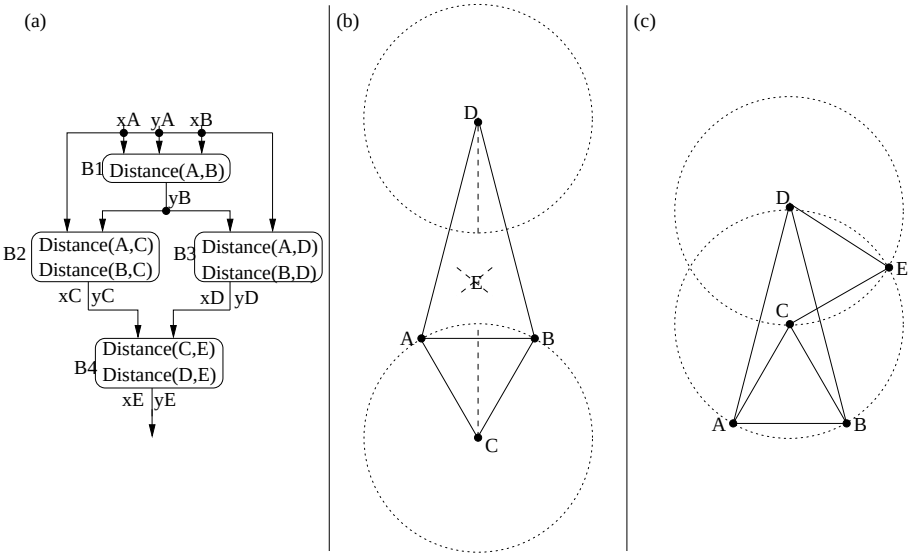

FIG. 8.2 – GCSP illustrant l'intérêt de la propagation

Illustrons la différence entre les 2 premières stratégies sur l'exemple présenté à la figure 8.2. Ce GCSP en 2D est constitué de 5 points A, B, C, D,et E liés par 7 contraintes de distances : Distance(A, B)=1, Distance(A, C)=1, Distance(A, D)=2, Distance(B, C)=1, Distance(B, D)=2, Distance(C, E)=1, Distance(D, E)=1. La décomposition de ce GCSP produit la séquence de blocs $< B_1, B_2, B_3, B_4 >$ présentés en figure 8.2-a. Admettons que les blocs B_1 et B_3 aient été résolus. Le bloc courant est alors B_2, qui résout les variables x_C et y_C en utilisant les équations Distance(A, C)=1, et Distance(B, C)=1. Ce bloc admet deux solutions qui conduisent aux 2 configurations présentées en figures 8.2-b et 8.2-c. Dans l'une de ces configurations, les points C et D se trouvent de part et d'autre du segment AB, ce qui conduit le bloc B_4 suivant à ne disposer d'aucune solution : le point E ne peut être à la fois à distance 1 de deux points séparés par une distance supérieure à 2.

La stratégie de résolution par bloc ne peut pas détecter cette inconsistance à venir lors de la résolution de B_3 et produit donc inutilement la configuration de la figure 8.2-b. A

l'inverse, la stratégie de résolution par blocs avec propagation globale peut détecter que l'une des solutions pour le point D est inutile : la propagation dans le bloc B_4 durant la résolution de B_3 conduira à un domaine vide pour les paramètres de E dans le sous-espace de recherche conduisant à la mauvaise configuration pour le point D.

La troisième stratégie est intéressante car elle ne nécessite pas de gérer les blocs de résolution par un mécanisme de retour-arrière entre blocs. Elle est proche de la seconde stratégie puisqu'elles effectuent toutes deux des propagations sur le GCSP restant. Cependant, cette troisième stratégie ne permet pas de remplacer les variables d'un bloc par des valeurs constantes lorsque ce bloc est résolu. Cette simplification est en revanche possible dans les 2 premières stratégies : les variables d'un bloc résolu peuvent être considérées comme des valeurs constantes (intervalles, ou valeur ponctuelle si l'on utilise l'*heuristique du point milieu*) dans la résolution de tous les blocs avals, ce qui améliore les performances des opérations de filtrage. De plus, cela permet d'utiliser l'*heuristique du point milieu* (cf. section 8.3.3) qui ne peut pas être utilisée dans la dernière stratégie.

Nous proposons une évaluation comparée de ces différentes stratégies sur quelques exemples de GCSP au chapitre D.

8.3.2 Mécanismes de retour-arrière

Lorsqu'un bloc n'admet pas de solution, ou que l'ensemble de ses solutions a été épuisé, il faut alors remettre en cause la solution choisie dans l'un des blocs précédents afin de poursuivre la résolution. C'est le principe du mécanisme de retour-arrière. Etant donné la structure de la décomposition, plusieurs mécanismes intelligents peuvent être mis à collaboration : GBJ [Dechter, 1990], GPB [Bliek, 1998].

Nous ne proposons pas une étude comparative entre ces différents mécanismes. Sur nos exemples, nous proposons d'utiliser le mécanisme le plus simple, BT, ou bien le mécanisme GPB, qui est celui qui est le plus à même de profiter de la structure du DAG car il utilise un ordre partiel sur les blocs.

8.3.3 Amélioration des performances

Pour améliorer les performances de la résolution, plusieurs heuristiques propres aux mécanismes de filtrage ou de découpage peuvent être utilisés : accélération des convergences lentes par compilation des cycles de filtrage, heuristiques de découpage plus sophistiquées que la simple dichotomie, ... Ces heuristiques sont générales et applicables à tout problème résolu par les techniques de résolution par intervalles.

Nous proposons ici l'*heuristique du point milieu* qui est spécifique à la résolution par intervalles d'un DAG de blocs. Elle consiste à remplacer les intervalles solutions issus de la résolution d'un bloc par une constante réelle au lieu d'un intervalle constant. La constante réelle utilisée comme valeur pour les variables résolues est calculée comme étant le milieu de l'intervalle solution rendu par la résolution du bloc.

Par exemple, si la résolution par intervalles d'un bloc B produit l'intervalle-solution $[0.99, 1.01]$ pour sa variable de sortie x, l'heuristique du point milieu consiste à remplacer

toute occurrence de x dans les équations des autres blocs par la constante $\frac{0.99+1.01}{2} = 1$.

Autrement dit, cette heuristique simplifie les équations des blocs en amont du bloc courant en :

1. remplaçant les variables de sorties par des constantes,
2. remplaçant les intervalles-solutions par des nombres réels.

L'utilisation de ILOG Solver 4.0 [ILO, 1997] ne nous a pas permis d'évaluer l'exact intérêt de chacune de ces deux simplifications séparément, car il ne gère pas les intervalles constants.

Notons cependant que cette heuristique ne peut être utilisée qu'avec les stratégies de résolution par blocs car il est impossible de remplacer une variable par une constante au cours de la résolution d'un GCSP complet (Stratégie 3).

Si la première simplification (Variable→Constante) est toujours correcte avec les stratégies de propagation 1 et 2, le passage d'un intervalle-solution à une valeur réelle peut faire perdre des solutions et rendre la méthode incorrecte.

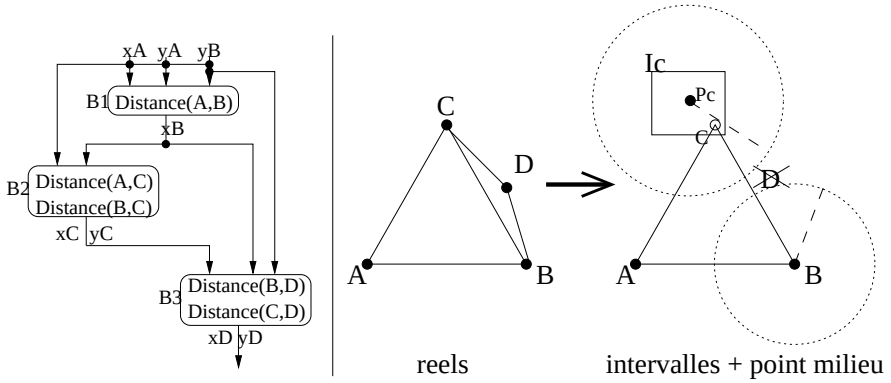

FIG. 8.3 – Problèmes liés à l'utilisation de l'heuristique du point milieu

Considérons par exemple le GCSP en 2D constitué de quatre points A, B, C et D et liées par cinq contraintes : Distance(A,B)=1, Distance(A,C)=1, Distance(B,C)=1, Distance(B,D)=0,5 et Distance(C,D)=0,51. Le plan d'assemblage produit par la décomposition de ce GCSP correspond à une séquence de 3 blocs $< B_1, B_2, B_3 >$ tels que présentés en figure 8.3-a. Admettons que le bloc B_1 soit résolu ($x_A = y_A = y_B = 0, x_B = 1$) et que la solution du bloc B_2 produise les intervalles solutions $x_C \in [0.48, 0.50]$ et $y_C \in [0.86, 0.90]$, c-a-d. la boîte I_C encadrant la solution réelle de C. L'heuristique du point milieu consiste à remplacer cette boîte par le point heuristique $P_C = (0.49, 0.88)$.

Malheureusement, cette position du point C rend le bloc B_3 inconsistant : le cercle centré sur P_C de rayon 0.51 n'a pas d'intersection avec le cercle centré sur B de rayon 0.5. La figure 8.3-b illustre le problème.

Remarquons cependant que cet exemple illustre un cas limite où la précision des intervalles-solutions est si grossière et les contraintes si proches de l'inconsistance que la moindre approximation peut faire perdre des solutions. En pratique, nous avons toujours

pu résoudre les GCSP décomposés de façon complète et correcte en utilisant l'heuristique du point milieu.

Il est de plus possible d'utiliser l'heuristique de façon correcte :

- Dans un premier temps, on résout le DAG de blocs en utilisant l'heuristique ;
- A chaque fois qu'une solution complète s du GCSP est produite, on vérifie si elle est correcte. Pour ce faire, il suffit d'effectuer une passe de propagation d'intervalles sur les contraintes du GCSP où chaque variable est associée à son intervalle-solution dans s. Si cette passe réduit le domaine d'une variable à l'intervalle vide, alors la solution s est incorrecte et est éliminée. Sinon s est correcte et elle est conservée dans l'ensemble $\mathbb{S}$ des solutions du GCSP.
- Lorsque la résolution complète du GCSP est terminée, $\mathbb{S}$ contient uniquement des solutions correctes du GCSP. En revanche, il peut ne pas contenir toutes les solutions correctes du GCSP (cf. exemple d'incorrection de l'heuristique ci-dessus). Si $\mathbb{S}$ est vide, on peut alors craindre que ce soit à cause de l'utilisation de l'heuristique, et reprendre une résolution complète du DAG sans utiliser l'heuristique.

8.3.4 Utilisation des contraintes R-redondantes

Nous avons expliqué à la section 7.2 que les contraintes R-redondantes et inconsistantes sont éliminées du GCSP pour la phase de planification. Les contraintes R-redondantes ainsi retirées sont stockées dans un ensemble de contraintes annexes. Pour la phase d'assemblage, ces contraintes annexes doivent être prises en compte afin d'invalider certaines solutions du GCSP.

La façon la plus simple et la moins efficace de prendre en compte ces contraintes est de vérifier que les solutions complètes produites pour le GCSP les satisfont. Ceci représente une approche *"générer puis tester"* puisque tout l'effort de résolution a été fourni avant que l'on décide que certaines solutions sont inadéquates.

Une approche plus pertinente consiste à tester ces contraintes aussi tôt que possible : une contrainte R-redondante peut être testée dès que tous les objets sur lesquels elle porte sont résolus. Ceci permet d'invalider des solutions de certains blocs plus tôt.

Enfin, l'approche qui utilise au mieux ces contraintes annexes consiste à effectuer la propagation sur ces contraintes. Ceci permet d'invalider au plus tôt les zones de l'espace de recherche qui ne sont pas compatibles avec ces contraintes.

Nous proposons donc d'adjoindre les contraintes annexes aux blocs correspondants[1] avant la résolution du DAG, ce qui permet de tirer au mieux parti de ces contraintes durant la résolution.

Comme nous l'avons expliqué à la section 6.1.4, page 169, certaines contraintes identifiées lors de la phase de d'élimination des R-redondances peuvent en réalité être inconsistantes et non R-redondantes ; par exemple, la méthode de détection structurelle ne peut distinguer ces deux types de contraintes. Si un bloc auquel nous avons adjoint des con-

[1] c-a-d. qu'une contrainte annexe est ajoutée au bloc le plus haut qui regroupe tous les objets sur lesquels elle porte.

traintes annexes n'admet jamais de solution, on peut relaxer les contraintes annexes de ce bloc afin d'éviter les problèmes dus aux éventuelles inconsistances parmi les contraintes annexes.

8.4 Unification des repères locaux

Une fois la phase de résolution terminée, il faut unifier les différents repères locaux utilisés pour la résolution de chaque agrégat. C'est cette étape qui peut être qualifiée d'assemblage à proprement parler.

Un puits K du plan d'assemblage représente un sous-GCSP S' rigide maximal constitué de tous les objets agrégés dans les agrégats en amont de S'. Pour réaliser l'assemblage, il faut positionner tous les objets de S' dans le repère local r de K. La position de tous les objets agrégés dans K est connue dans r. Les agrégats qui ont été regroupés dans K sont représentés de façon différente selon l'opération de condensation utilisée. Le principe d'assemblage général est cependant commun quelle que soit l'opération de condensation utilisée ; pour déterminer la position de l'ensemble des objets d'un agrégat K' dans l'agrégat K il faut :

1. Calculer le déplacement à appliquer ; ceci se fait en considérant la position des objets représentant K' dans K par rapport à la position des objets *équivalents* dans K'. C'est cette étape qui est différente selon l'opération de condensation utilisée.
2. Appliquer le déplacement calculé au point 1 à chaque objet de K'. On obtient ainsi sa position dans K.

Afin d'éviter de calculer et appliquer plusieurs déplacements à chaque objet géométrique, nous proposons de procéder à l'assemblage dans l'ordre inverse du plan d'assemblage. Considérons par exemple trois agrégats K'', K' et K. K est un puits du plan d'assemblage, K' est lié directement à K et K'' est lié uniquement à K'.

L'assemblage que nous proposons consiste d'abord à positionner K' dans K par l'assemblage en deux étapes décrites ci-dessus. Ainsi, les coordonnées de tous les objets de K' sont connues dans le repère local r de K ; en particulier, les objets représentant K'' dans K' sont maintenant positionnés directement dans K. On utilise alors ces positions qui viennent juste d'être calculées afin de produire, à l'étape 1 de l'assemblage, le déplacement qui positionnera directement K'' dans K.

Nous décrivons maintenant la façon de calculer le déplacement selon que les agrégats sont représentés par des objets repères (CA, CA+MFA) ou par leurs objets frontières (FA, MFA, CA+MFA et FAR).

Assemblage avec un objet repère Lorsque l'agrégat K' est représenté dans K par un objet repère $N_{K'}$, les solutions de K définissent la position de $N_{K'}$ dans le repère local de l'agrégat K. Le déplacement est alors immédiatement connu, puisque dans K' le repère local est positionné de façon standard (en position 0). Le déplacement à appliquer est donc celui qui fait passer le repère local de K' en position $N_{K'}$.

Par exemple, considérons un GCSP en 2D où l'agrégat K' composé de 3 points A, B et C est représenté dans l'agrégat K par son objet repère $N_{K'}$. Admettons que l'on veuille assembler la solution $A = (0,0)$, $B = (1,0)$ et $C = (0,1)$ avec la solution correspondante $N_{K'} = (p_{K'} = (1,1), v1_{K'} = (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}))$ dans K. Le déplacement à appliquer est alors déduit en considérant que A, B et C ont été positionnés dans le repère local $N_{K'} = (p_{K'} = (0,0), v1_{K'} = (0,1))$: le déplacement qui amène ce repère dans la position de $N_{K'}$ dans K est alors constitué d'une translation par le vecteur $(1,1)$ et d'une rotation d'angle $\frac{\pi}{4}$ dans le sens trigonométrique.

En appliquant ce déplacement aux points A, B et C, on obtient leurs positions dans K : $A = (1,1)$, $B = (1 + \frac{\sqrt{2}}{2}, 1 + \frac{\sqrt{2}}{2})$ et $C = (1 - \frac{\sqrt{2}}{2}, 1 + \frac{\sqrt{2}}{2})$.

Assemblage avec les objets frontières Lorsque l'agrégat K' est représenté dans K par sa frontière dupliquée $F'_{K'}$, on dispose de la position de deux instances de chaque objet frontière de K' : sa position dans K et celle dans K'. Ceci permet de calculer un déplacement amenant les objets frontières à coïncider, à condition que la frontière permette la définition d'un repère local. Ceci est assuré pour FAR, mais ne l'est pas pour FA, MFA et CA+MFA.

Considérons à nouveau l'exemple fourni au paragraphe précédent. L'agrégat K' est cette fois représenté dans K par ses objets frontières A et B. La position de A, B et C dans K' reste inchangée, alors que la position de A' et B' (les copies des objets A et B dans K) sont $A' = (1,1)$ et $B' = (1 + \frac{\sqrt{2}}{2}, 1 + \frac{\sqrt{2}}{2})$ dans le repère local de l'agrégat K.

Sachant que A et A' sont en fait un même objet géométrique à un déplacement près, on obtient l'équation $A' = t(r(A))$ où r est la rotation et t est la translation qui assemblent A et A'. Il en va de même pour B et B', avec les mêmes rotation et translation car A et B appartiennent à un sous-GCSP rigide et ne peuvent donc être déplacés indépendamment l'un de l'autre.

On dispose donc d'un système d'équations qui permet de déterminer la rotation r et la translation t [2] : t est la translation par le vecteur $(1,1)$ et r la rotation d'angle $\frac{\pi}{4}$ dans le sens trigonométrique.

En appliquant ce déplacement aux objets restant de K', on peut ainsi déterminer leur position dans K : $C = (1 - \frac{\sqrt{2}}{2}, 1 + \frac{\sqrt{2}}{2})$ dans K.

8.5 Conclusion

La phase d'assemblage que nous venons de décrire est très générale puisqu'elle est basée sur la résolution par intervalles qui permet de résoudre des sous-systèmes quelconques.

Elle permet la résolution complète d'un GCSP décomposé grâce à l'utilisation d'un mécanisme de retour-arrière. De plus, l'utilisation de schémas d'optimisation, comme le *Branch&Bound*, permettrait de prendre un compte un critère de sélection de solution. On

[2] A condition que la frontière permette la définition d'un repère local, sinon certains paramètres de translation ou rotation restent indéterminés.

pourrait alors produire les solutions satisfaisant au mieux certains critères, par exemple afin de produire les solutions les plus *proches* d'un croquis initial.

L'utilisation de mécanismes de propagation rend la résolution par intervalles plus efficace en pratique que les méthodes formelles ; elle peut, de plus, bénéficier d'heuristiques améliorant ses performances pratiques : accélération des convergences lentes, heuristique du point milieu, ...

En conclusion, notons aussi que c'est la première fois, à notre connaissance, que les techniques de résolution par intervalles sont employées pour la résolution d'un GCSP décomposé par une méthode géométrique. Des expérimentations poussées sur des problèmes issus d'applications réelles doivent encore valider cette approche, mais les expérimentations préliminaires présentées en annexe D sont encourageantes.

Conclusion

Dans ce chapitre, nous commençons par rappeler brièvement l'ensemble des contributions de la thèse en soulignant leurs avantages et leurs limites. Nous portons aussi un regard critique sur les contributions de la thèse et leurs implications générales. Nous présentons finalement quelques perspectives de travaux qui nous semblent intéressantes, et un bilan qui tire les conclusions du travail effectué.

Rappel des contributions

Les contributions de cette thèse portent sur l'analyse et la proposition de méthodes de rigidification récursive pour la décomposition et la résolution de GCSP en 2D et 3D.

Analyse de la méthode HLS

Nous avons ré-expliqué la méthode HLS proposée par Hoffmann, Lomonosov et Sitharam [Hoffmann *et al.*, 2000] en nous référant aux fondements de la théorie de la rigidité afin de pouvoir évaluer les limites de cette approche :

- L'approximation de la rigidité qu'elle utilise peut être fausse lorsqu'un GCSP contient des R-redondances, des inconsistances ou des singularités. Or les R-redondances sont omniprésentes dans les applications réelles et des singularités apparaissent dès que l'on spécifie des relations booléennes entre les objets géométriques.
- La méthode ne prend pas en compte les contraintes booléennes (incidences, parallélismes, ...) de façon générale et propose des solutions *ad-hoc* (s_rigidité bis, *Look-up Table*, ...) aux problèmes que cela engendre.
- L'opération de fusion s'effectue de façon géométriquement incorrecte puisqu'elle essaye de poser un repère sur des ensembles d'objets quelconques, ce qui n'est pas toujours possible.
- Les opérations de condensation sont incorrectes (CA) ou mettent en péril la terminaison de la planification (FA et MFA).

Nous avons explicité et analysé ces problèmes avant de proposer des solutions plus générales à chacun d'eux.

En particulier, nous avons proposé des solutions aux problèmes des opérations de condensation : nous avons identifié une condition suffisante pour que CA soit correcte,

nous avons proposé une modification de l'opération de fusion afin d'assurer la terminaison de FA et MFA (cf. section* 5.5).

Nous avons aussi expliqué qu'il peut être profitable de faire collaborer des opérations de condensations pour pallier les défauts de chacune ; ainsi, l'association de CA et MFA avec un mécanisme d'élimination des sur-s_rigidités produit une nouvelle méthode de rigidification récursive plus correcte et performante que CA, FA ou MFA seuls (cf. section* 5.6).

Analyse des limites des approches géométriques structurelles

Nous avons analysé plus finement les limites de toute approche géométrique structurelle :

- La caractérisation approchée de la rigidité utilisée par ces approches peut être trompée par la présence de R-redondances, de singularités ou d'inconsistances.
- Il est difficile de déterminer si l'assemblage de deux sous-GCSP rigides est possible.

Redondances, inconsistances et singularités

Nous avons exposé quelques façons de détecter des R-redondances et inconsistances (cf. section* 6.1.4) et expliqué pourquoi il est légitime de supposer que les GCSP sont exempts de singularités autres que celles introduites par des contraintes booléennes (cf. section* 6.1.1).

Degré de rigidité

Nous avons ensuite proposé le concept de degré de rigidité (DDR) d'un ensemble d'objets géométriques (cf. section* 6.2). Le DDR d'un ensemble d'objets géométriques indique le nombre de degrés de liberté que doit posséder un GCSP rigide basé sur ces objets. Nous avons alors explicité les difficultés du calcul du DDR :

- Il nécessite de tenir compte des relations géométriques induites par les contraintes du GCSP sur les objets géométriques considérés.
- Il peut ne pas être unique dans certains GCSP particuliers.

Nous avons proposé une méthode de calcul du DDR, basée sur le concept d'ensemble d'objets DDR-minimaux et sur l'utilisation d'un opérateur d'inférence géométrique. Nous avons aussi expliqué comment cette méthode peut permettre de produire l'ensemble des valeurs possibles lorsque le DDR est multiple et comment utiliser des heuristiques permettant d'éviter l'utilisation de mécanismes d'inférence trop coûteux (cf. section* 6.2.4).

Rigidité structurelle étendue et test d'assemblage

Ce concept de DDR nous a permis de répondre (partiellement) aux deux problèmes des méthodes de décomposition géométrique structurelle :

- Nous avons défini la rigidité structurelle étendue (cf. section* 6.3), caractérisation approchée de la rigidité basée sur le concept de DDR et permettant ainsi de prendre en compte les singularités introduites par les contraintes booléennes d'un GCSP.

Nous avons montré que cette nouvelle caractérisation peut encore être trompée par la présence de R-redondances ou d'inconsistances, et que l'existence d'un DDR multiple engendre une indétermination de la rigidité (qui peut, dans certains cas, être justifiée, car le GCSP peut effectivement posséder des solutions bien- sur- et sous-rigides).
– Le concept de DDR nous a permis de caractériser les ensembles d'objets permettant la définition d'un repère local, et ainsi de déterminer si un assemblage est possible ou pas (cf. section* 6.4).

Nouvelle méthode de rigidification récursive

En nous basant sur le schéma de la méthode proposée par Hoffmann *et al.*, nous avons proposé une nouvelle méthode de décomposition et de résolution de GCSP basée sur notre nouvelle caractérisation de la rigidité et sur les réponses apportées aux problèmes des méthodes géométriques structurelles.

Phase de planification

Nous avons proposé de commencer la phase de planification par l'application d'un opérateur d'élimination des R-redondances et inconsistances. Ceci, plus le traitement des singularités par des contraintes booléennes, nous permet d'espérer que notre nouvelle caractérisation de la rigidité correspond bien à la rigidité, puisque ses sources d'erreurs sont éliminées.

Pour réaliser l'étape de fusion, nous avons remplacé l'algorithme `Minimal-Dense` par l'algorithme `Minimal-ES_Rigide` (cf. section* 7.3). Cet algorithme utilise la es_rigidité et effectue la distribution de flot de façon géométriquement correcte.

Nous avons adapté l'opération d'extension à la es_rigidité et proposé le concept d'extension globale qui permet d'intégrer des agrégats entiers à moindre coût pour la planification et la résolution (cf. section* 7.4).

Enfin, nous avons proposé une nouvelle opération de condensation qui consiste à rigidifier automatiquement les objets frontières des agrégats (cf. section* 7.5). La rigidification automatique est un problème généralement difficile, mais nous avons présenté un mécanisme de rigidification automatique basé sur des modèles de rigidification. Notre nouvelle opération de condensation n'est pas correcte dans le cas général. Cependant, il nous semble que son *incorrection* permet justement de pallier les limites de la caractérisation approximative de la rigidité ; ceci reste à démontrer.

Phase d'assemblage

Nous avons expliqué comment ramener la décomposition générée en phase de planification à un DAG de blocs d'équations à résoudre (cf. section* 8.2).

Pour résoudre un tel DAG, on peut alors utiliser la méthode proposée par Bliek *et al.* [Bliek *et al.*, 1998]. Nous avons montré comment opère cette méthode et nous avons proposé trois nouvelles stratégies de propagation compatibles avec ce schéma de résolution

(cf. section* 8.3). Nous avons aussi proposé une heuristique pour améliorer les performances de la résolution. Cette heuristique peut faire perdre la complétude de la résolution dans certains cas.

Enfin, nous avons expliqué comment reconstituer des solutions du GCSP à partir des solutions de chaque agrégat en procédant à l'assemblage des différents agrégats par unification des repères locaux individuels dans lesquels ils sont résolus (cf. section* 8.4).

Perspectives

Les améliorations potentielles de la méthode que nous avons proposée ne manquent pas :

- Au niveau du concept de degré de rigidité dont le calcul, difficile dans le cas général, réclame donc une attention particulière.
- Au niveau de la nouvelle phase de planification, dont les propriétés de l'opération de condensation donnent à réfléchir.
- Au niveau de la phase d'assemblage, où la génération du DAG peut jouer un rôle crucial dans la résolution efficace du GCSP décomposé.

Nous explicitons dans les paragraphes qui suivent les améliorations que nous envisageons comme importantes ou potentiellement intéressantes à tous ces niveaux.

Degré de rigidité

Nous avons discuté de la difficulté potentielle du calcul du DDR dans le cas général. Cependant, nous avons fourni en annexe les moyens pratiques nécessaires à son calcul pour certaines classes de GCSP :

- ({Points,Droites},{Distances, Angles, Incidences, Parallélismes}) en 2D,
- ({Points, Droites, Plans},{Distances, Angles, Incidences, Parallélismes}) en 3D.

Il reste à démontrer s'il est possible ou non de proposer un mécanisme d'inférence complet pour ces classes de GCSP.

Un autre travail consistera à identifier des classes intéressantes de GCSP pour lesquels le calcul du DDR est plus facile ; pour cela il faudra s'intéresser de plus près à des domaines applicatifs particuliers, tels que la reconstruction de scènes en 3D, ou la biologie moléculaire qui nous semblent des domaines prometteurs.

Phase de planification

Nous avons expliqué à la section* 7.5.1 que la rigidification automatique est un problème délicat dans le cas général. Nous avons donné en annexe C les moyens pratiques pour l'effectuer pour les classes de GCSP ({Points,Droites},{Distances, Angles, Incidences, Parallélismes}) en 2D et ({Points, Droites, Plans},{Distances, Angles, Incidences, Parallélismes}) en 3D. Dans le cadre d'un développement pour un domaine applicatif particulier, il faudra spécialiser et compléter les modèles de rigidification que nous avons proposés.

La phase d'élimination des redondances peut utiliser les divers mécanismes de détection que nous avons présenté en section* 6.1.4 (page 169). Bien que nous ayons analysé ces diverses méthodes pour en évaluer l'intérêt, les limites, et le coût, nous n'avons pas encore proposé une véritable nouvelle méthode de détection de redondances. En particulier, la méthode structurelle et la méthode numérique probabiliste (page 172) nous semblent prometteuses pour les applications pratiques. La méthode structurelle peut bénéficier directement de la es_rigidité et ainsi augmenter sa fiabilité ; la méthode numérique probabiliste pourrait voir son champs d'application étendu en tenant compte des contraintes booléennes. La recherche sur ces deux approches complémentaires et polynomiales fait parti des perspectives prioritaires de la thèse en vue de pouvoir vérifier l'intérêt de la méthode sur des applications réelles, qui contiendront certainement des singularités et redondances.

Nous avons vu que l'opération de condensation semble posséder la propriété intéressante de rendre structurellement décelable des R-redondances qui ne l'étaient pas. Nous avons pensé qu'il serait possible de généraliser cette opération de condensation afin d'en proposer une nouvelle possédant clairement cette propriété et permettant donc un traitement correct (bien que de complexité exponentielle en pire cas) du problème de la décomposition géométriques de GCSP.

Ce nouvel opérateur de condensation pourrait consister à rigidifier automatiquement *à la demande* en cours d'extension par exemple, ce qui permettrait de maximiser la capacité d'extensions. Il pourrait aussi consister à rigidifier les objets frontières à l'aide de *contraintes n-aires* symbolisant leur appartenance à d'anciens agrégats et leur rigidité.

Phase d'assemblage

A la section* 8.2, nous avons proposé une façon de générer le DAG de blocs à partir des étapes de fusion et des extensions produites en phase de planification. Il est possible d'utiliser des informations plus générales afin d'engendrer un DAG contenant moins de blocs et d'équations.

En particulier, on pourrait changer la modélisation des agrégats si l'on s'aperçoit qu'en fait, un ancien agrégat entier a été inclus par une série d'extensions. Il est souvent dans ce cas préférable de résoudre l'agrégat entier en un seul bloc en tant qu'objet repère, même si la phase de planification utilise une représentation des agrégats par objets frontières ; en effet, le bloc résolvant un objet repère est constitué de $\frac{d(d+1)}{2}$ équations en dimension d, alors que la résolution de k objets frontières correspond au minimum à la résolution de $k * d$ équations (éventuellement en plusieurs blocs). Dès que $k \geq d$, le bloc résolvant un objet repère devient ainsi plus petit que l'ensemble des blocs résolvant les objets frontières.

Enfin, de plus amples expérimentations sont nécessaires afin de permettre d'analyser plus sérieusement l'intérêt des différentes stratégies de résolution, de l'heuristique du point milieu et des différents mécanismes de retour-arrière.

Bilan

Signalons tout d'abord que la principale contribution de la thèse, à savoir la proposition d'une méthode de décomposition géométrique apte à tenir compte des singularités[3] et des R-redondances, représente, à notre connaissance, la première tentative de réparation non structurelle d'une méthode structurelle.

En effet, le concept de degré de rigidité que nous avons introduit subsume les corrections *ad-hoc* généralement utilisées dans les méthodes structurelles : il permet de définir une caractérisation plus proche de la rigidité, et de déterminer si un assemblage sera possible ou pas.

Par ailleurs, cette thèse nous a permis de mieux cerner les limites des approches structurelles pour l'analyse de CSP (et pas seulement de GCSP). Ces limites sont principalement dues aux contraintes A-redondantes et aux singularités des CSP traités en pratique.

Cependant, ce mémoire illustre aussi la difficulté qu'il y a à réparer les approches structurelles : l'introduction de concepts géométriques peut conférer une complexité exponentielle à la planification, alors que les approches structurelles pures demeurent polynomiales en pire cas. Nous avons proposé plusieurs optimisations de la méthode, et des heuristiques permettant d'obtenir une complexité pratique sans doute acceptable. Cependant, seul le développement d'un outil logiciel, et son utilisation sur des problèmes issus d'applications réelles, permettrait d'estimer la complexité en pratique de l'approche et de juger l'intérêt de sa plus grande justesse.

En particulier, identifier des problèmes applicatifs pour lesquels le sur-coût engendré par une analyse plus précise des GCSP est amorti par la plus grande fiabilité et robustesse de la résolution est tout particulièrement intéressant et permettra de valider l'approche.

Dans le contexte applicatif, la méthode de rigidification récursive n'est pas suffisante : elle peut traiter des GCSP bien-rigides en dimension quelconque, mais est incapable de traiter des GCSP sous-rigides. Or les applications mécaniques et robotiques nécessitent d'être capables de traiter ce type de GCSP. Il faut donc envisager d'utiliser une autre méthode capable de traiter ce type de systèmes en conjonction avec la rigidification récursive. Parmi les méthodes adaptées à ce type de problème, on trouve le schéma OpenPlan, qui est basé sur la décomposition équationnelle structurelle DM-CFC. Une collaboration de OpenPlan avec la rigidification récursive est envisageable, mais reste à définir : un GCSP sous-rigide peut contenir des sous-parties bien-rigides, traitables par rigidification récursive ; le reste du GCSP pourrait être traité par un méthode plus générale.

Enfin, dans les CSP issus d'applications réelles, les contraintes géométriques ne représentent généralement qu'un aspect restreint d'un problème de conception plus vaste, contenant différentes catégories de contraintes : contraintes représentant des normes légales, contraintes financières liées aux coûts des matériaux, contraintes physiques liées aux possibilités des outils de manutention, contraintes mécaniques, contraintes électriques, *etc.*. On pourrait donc envisager un résolveur dédié aux contraintes géométriques comme une forme de *contrainte globale* à utiliser dans une méthode de résolution de CSP plus générale.

[3]Si elles sont toutes introduites sous forme de contraintes booléennes.

Annexes

Annexe A

Modélisation d'objets et de contraintes 2D et 3D

Cette annexe présente la modélisation que nous utilisons pour les objets et les contraintes géométriques usuelles en 2D et 3D. Cette modélisation est conforme à la définition des objets et contraintes géométriques proposées en section 1.2 (cf. définitions 5 et 6). Rappelons sommairement que chaque objet est défini par un ensemble P de paramètres géométriques, une relation interne I entre ces paramètres qui doit rendre la représentation unique, et une relation externe E définissant l'ensemble des points de l'espace géométriques appartenant à l'objet. Une contrainte géométrique se définit par l'ensemble O des objets sur lesquels elle porte, l'ensemble P de ces paramètres géométriques, une relation interne I sur ces paramètres et une relation externe E sur les paramètres des objets géométriques de O.

Objets géométriques

Pour plus de clarté, nous distinguons les modélisation en 2D et 3D des objets géométriques. Nous ne proposons les modélisations que des objets points, droites, plans et repères en 2D et 3D. Ces objets constituent en effet une base pour la définition d'objets plus complexes. Par exemple, un cercle, une ellipse ou encore une sphère (à rayons fixes) peuvent être représentés par leurs points centraux.

Objets géométriques en 2D

Point

- Modélisation : $(P = \{x_0, y_0\}, I = \emptyset, E = \{x = x_0, y = y_0\})$
- Explication : R.A.S.

Droite

- Modélisation : $(P = \{a, b, c\}, I = \{a^2 + b^2 = 1, (a > 0 \vee (a = 0 \wedge b > 0)\}, E = \{b * x - a * y - c = 0\})$
- Explication : les paramètres $(a\ b)$ représente le vecteur directeur de la droite, c sa distance algébrique orthogonale au point origine. La relation interne spécifie que le vecteur directeur doit être unitaire et orienté *vers la droite*, afin que chaque droite de l'espace 2D soit modélisée de façon unique. La relation externe spécifie que tout point appartenant à la droite se projette sur le point de la droite se trouvant à distance minimale de l'origine.

Repère

- Modélisation : $(P = \{x_0, y_0, a, b\}, I = \{a^2 + b^2 = 1\}, E = \{x = x_0, y = y_0\})$
- Explication : $\vec{i} = (a\ b)$ représente le vecteur des abscisses du repère, le vecteur des ordonnées se déduisant comme $\vec{j} = (-b\ a)$ afin que le repère soit orthogonal. Le point (x_0, y_0) est le point origine du repère. La relation interne spécifie que l'on ne considère que des repères orthonormés. La relation externe n'a pas vraiment de sens ici : nous avons choisi de dire que le seul point appartenant à un repère est son origine.

Objets géométriques en 3D

Point

- Modélisation : $(P = \{x_0, y_0, z_0\}, I = \emptyset, E = \{x = x_0, y = y_0, z = z_0\})$
- Explication : R.A.S.

Droite

- Modélisation : $(P = \{a, b, c, x_0, y_0, z_0\}, I = \{a^2 + b^2 + c^2 = 1, (a > 0) \vee ((a = 0) \wedge (b > 0)) \vee ((a = b = 0) \wedge (c > 0)), a * x_0 + b * y_0 + c * z_0 = 0\}, E = \{a * (y - y_0) - b * (x - x_0) = 0, b * (z - z_0) - c * (y - y_0) = 0\})$
- Explication : $\vec{v} = (a\ b\ c)$ est le vecteur directeur de la droite, $P_0 = (x_0, y_0, z_0)$ est un point sur la droite. Afin de rendre la représentation unique, le vecteur est normé et orienté, et on choisit que le point doit se trouver à distance minimale de l'origine o ; cette dernière condition se traduit par le fait que le vecteur $\overrightarrow{oP_0}$ doit être orthogonal au vecteur directeur de la droite.
  Les points $P = (x, y, z)$ appartenant à une droite doivent vérifier $\overrightarrow{P_0P}$ colinéaire à $\vec{v}$. Pour représenter cette colinéarité, plusieurs possibilités s'offrent à nous :
  - $\overrightarrow{P_0P} = k * \vec{v}$; ceci représente 3 équations, mais introduit une variable supplémentaire rendant cette représentation générique et non A-redondante.
  - $\overrightarrow{P_0P}.\vec{v} = ||\overrightarrow{P_0P}||$; cette représentation constitué d'une seule équation est inadéquate. Remarquons que $\overrightarrow{P_0P} = \overrightarrow{oP} - \overrightarrow{oP_0}$, et donc $\overrightarrow{P_0P}.\vec{v} = \overrightarrow{oP}.\vec{v} - \overrightarrow{oP_0}.\vec{v} = \overrightarrow{oP}.\vec{v}$.

Or poser l'équation $\overrightarrow{oP}.\vec{v} = d$ ne définit pas un point P unique, mais un plan entier de points (cf. modélisation d'un plan ci-dessous).
- $\overrightarrow{P_0P} \wedge \vec{v} = 0$. Cette représentation est constitué de 3 équations non-indépendantes. On peut cependant choisir n'importe quelle paire d'équations dans cette représentation pour modéliser l'ensemble des points appartenant à une droite. Cette modélisation est intéressante car elle n'introduit pas de variable supplémentaire, à l'inverse de la première.

Plan

- Modélisation : $(P = \{a, b, c, d\}, I = \{a^2 + b^2 + c^2 = 1, (a > 0) \vee ((a = 0) \wedge (b > 0)) \vee ((a = b = 0) \wedge (c > 0))\}, E = \{ax + by + cz - d = 0\})$
- Explication : $\vec{v} = (a\ b\ c)$ est le vecteur normal au plan, d est sa distance algébrique à l'origine. Le vecteur normal est normé et orienté par la relation interne du plan. La relation externe du plan symbolise le fait que tout point dont la projection sur la direction $(a\ b\ c)$ et de longueur algébrique d appartient au plan.

Repère

- Modélisation : $(P = \{x_0, y_0, z_0, a, b, c, d, e, f\}, I = \{a^2 + b^2 + c^2 = 1, d^2 + e^2 + f^2 = 1, a * d + b * e + c * f = 0\}, E = \{x = x_0, y = y_0, z = z_0\})$
- Explication : $\vec{i} = (a\ b\ c)$ représente le vecteur des abscisses du repère, $\vec{j} = (d\ e\ f)$ celui des ordonnées et celui des hauteurs, $\vec{k}$, se définit par le produit vectoriel $\vec{k} = \vec{i} \wedge \vec{j}$ afin que le repère soit orthogonal. Le point (x_0, y_0) est le point origine du repère. La relation interne spécifie que l'on ne considère que des repères orthonormés. La relation externe n'a pas vraiment de sens ici : nous avons choisi de dire que le seul point appartenant à un repère est son origine.

Reformulation d'un objet dans un objet repère

Nous présentons ici les reformulations des objets géométriques que nous avons modélisés en fonction des paramètres d'un objet repère dans lequel on connaît une valuation de leurs paramètres. Nous notons à l'aide d'apostrophes (') cette valuation. Constatons que tous les objets que nous avons modélisés sont représentés par des points, des vecteurs et des longueurs. Il suffit donc de savoir reformuler des points et des vecteurs pour pouvoir reformuler toutes nos modélisations d'objets géométriques, les longueurs étant invariantes par changement de repère.

Reformulation d'un point 2D

Considérons un point en 2D (x, y) que l'on souhaite reformuler relativement à un objet repère (x_0, y_0, a, b) dans lequel on connaît sa position (x', y'). On sait qu'un point s'exprime en fait comme $P = o + x' * \vec{i} + y' * \vec{j}$. En partant de ce principe, on peut donc écrire :

$$\begin{aligned} x &= x_0 + a * x' - b * y' \\ y &= y_0 + b * x' + a * y' \end{aligned}$$

Reformulation d'un vecteur en 2D

Considérons un vecteur en 2D $(a\ b)$ que l'on souhaite reformuler relativement à un objet repère (x_0, y_0, a_0, b_0) dans lequel on connaît son orientation $(a'\ b')$. On sait qu'un vecteur s'exprime comme $\vec{v} = a' * \vec{i} + b' * \vec{j}$ en 2D. En partant de ce principe, on peut donc écrire :

$$\begin{aligned} a &= a_0 * a' - b_0 * b' \\ b &= b_0 * a' + a_0 * b' \end{aligned}$$

Reformulation d'un point 3D

Considérons un point en 3D (x, y, z) que l'on souhaite reformuler relativement à un objet repère $(x_0, y_0, z_0, a, b, c, d, e, f)$ dans lequel on connaît sa position (x', y', z'). On sait qu'un point s'exprime en fait comme $P = o + x' * \vec{i} + y' * \vec{j} + z' * \vec{k}$. En partant de ce principe, on peut donc écrire :

$$\begin{aligned} x &= x_0 + a * x' + d * y' + (b * f - c * e) * z' \\ y &= y_0 + b * x' + e * y' + (a * f - c * d) * z' \\ z &= z_0 + c * x' + f * y' + (a * e - b * d) * z' \end{aligned}$$

Reformulation d'un vecteur en 3D

Considérons un vecteur en 2D $(a\ b\ c)$ que l'on souhaite reformuler relativement à un objet repère $(x_0, y_0, z_0, a_0, b_0, c_0, d_0, e_0, f_0)$ dans lequel on connaît son orientation $(a'\ b'\ c')$. On sait qu'un vecteur s'exprime comme $\vec{v} = a' * \vec{i} + b' * \vec{j} + c' * \vec{k}$. En partant de ce principe, on peut donc écrire :

$$\begin{aligned} a &= a_0 * a' + d_0 * b' + (b_0 * f_0 - c_0 * e_0) * c' \\ b &= b_0 * a' + e_0 * b' + (a_0 * f_0 - c_0 * d_0) * c' \\ c &= c_0 * a' + f_0 * b' + (a_0 * e_0 - b_0 * d_0) * c' \end{aligned}$$

Contraintes géométriques

Nous allons présenter ici la modélisation de quelques contraintes géométriques en 2D et 3D. Nous nous limiterons aux contraintes de distance, d'angle, d'incidence et de parallélisme qui constituent une base de relation pour les objets que nous avons modélisés. Nous commençons par présenter ces contraintes entre des objets en 2D avant d'introduire leurs équivalents en 3D.

Contraintes géométriques en 2D

Distance Point-Point

– **Objets :** Points p et q.

- **Paramètres :** d, un réel.
- **Relation interne :** $d \neq 0$. Cette relation empêche l'introduction de contraintes de distance singulières.
- **Relation externe :** $(p.x - q.x)^2 + (p.y - q.y)^2 = d^2$. Cette équation représente l'expression analytique classique de distance entre 2 points.

Distance Point-Droite

- **Objets :** Point p et Droite q.
- **Paramètres :** d, un réel.
- **Relation interne :** $d \neq 0$.
- **Relation externe :** $(q.a * p.y - q.b * p.x - q.c)^2 = d^2$. Cette équation définit la distance d'un point à une droite comme la norme du produit vectoriel du vecteur directeur de la droite par le vecteur liant l'origine au point p.

Distance Droite-Droite (uniquement si parallèles)

- **Objets :** Droites p et q
- **Paramètres :** d, un réel.
- **Relation interne :** Pas de relation interne : rendre 2 droites parallèles confondues n'est pas une contrainte singulière.
- **Relation externe :** $(p.c - q.c)^2 = d^2$. Cette relation fixe la distance entre 2 droites parallèles en fixant leurs distances à l'origine.

Angle Droite-Droite

- **Objets :** Droites p et q.
- **Paramètres :** a, un réel.
- **Relation interne :** $a \in [0, 2\pi]$.
- **Relation externe :** $p.a * q.a + p.b * q.b = cos(a)$. Les vecteurs directeurs des droites étant normés, on peut représenter la relation d'angle à l'aide du produit scalaire de ces vecteurs. Cependant, ceci ne permet pas de gérer les angles orientés.

Incidence Point-Droite

- **Objets :** Point p et Droite q.
- **Paramètres :** aucun.
- **Relation interne :** aucune.
- **Relation externe :** $q.b * p.x - q.a * p.y - q.c = 0$. Comme nous l'avons déjà expliqué pour la définition de la relation externe d'une droite en 2D, on peut représenter l'appartenance d'une droite à un point par le fait que la norme du produit vectoriel du vecteur directeur de la droite par le vecteur liant l'origine au point est la distance algébrique de l'origine à la droite.

Parallélisme Droite-Droite

- **Objets :** Droite p et q.
- **Paramètres :** aucun.
- **Relation interne :** aucune.
- **Relation externe :** $p.a = q.a$. étant donné que les vecteurs directeurs des droites sont normés et orientés, on peut imposer que leurs coordonnées soient égales. Notons qu'une seule équation suffit étant donné les relations internes des droites imposent que la seconde sera nécessairement vérifiée.

Contraintes géométriques en 3D

Distance Point-Point

- **Objets :** Points p et q.
- **Paramètres :** d, un réel.
- **Relation interne :** $d \neq 0$. Cette relation empêche l'introduction de contraintes de distance singulières.
- **Relation externe :** $(p.x - q.x)^2 + (p.y - q.y)^2 + (p.z - q.z)^2 = d^2$. Cette équation représente l'expression analytique classique de distance entre 2 points.

Distance Point-Droite

- **Objets :** Point p et Droite q.
- **Paramètres :** d, un réel.
- **Relation interne :** $d \neq 0$.
- **Relation externe :** $q.a * (p.y - q.y) - q.b * (p.x - q.x))^2 + (q.a * (p.z - q.z) - q.c * (p.x - q.x))^2 + (q.c * (p.y - q.y) - q.b * (p.z - q.z))^2 = d^2$. Cette équation définit la distance d'un point à une droite comme la norme du produit vectoriel du vecteur directeur de la droite par le vecteur liant l'origine au point p.

Distance Point-Plan

- **Objets :** Point p et Plan q.
- **Paramètres :** d, un réel.
- **Relation interne :** $d \neq 0$.
- **Relation externe :** $(q.a * p.x + q.b * p.y + q.c * p.z - q.d)^2 = d^2$. Cette équation définit la distance d'un point à une droite comme la norme du produit vectoriel du vecteur directeur de la droite par le vecteur liant l'origine au point p.

Distance Droite-Droite

- **Objets :** Droites p et q.
- **Paramètres :** d, un réel ; pp et pq deux points ; r une droite.

- **Relation interne :** aucune. ici, poser la distance comme étant nulle ne pose pas de problème de singularité. Cela permet de rendre 2 droites coplanaires.
- **Relation externe :** Incidence(pp, p), Incidence(pq, q), Incidence(pp, r), Incidence(pq, r), Angle$(p, r, \frac{\pi}{2})$, Angle$(q, r, \frac{\pi}{2})$, Distance(pp, pq, d). Cette contrainte étant de formulation assez compliquée, il nous a semblé plus judicieux de la décomposer en plusieurs étapes en introduisant des objets intermédiaires : on calcule les points pp et pq respectivement sur p et sur q qui sont à distance la plus proche (ils forment un vecteur orthogonal aux vecteurs directeurs des 2 droites), et on pose la relation de distance entre ces points. Cette modélisation peut être inadéquate lorsque les droites sont parallèles. lorsque tel est le cas, on peut recourir à une autre relation externe, qui consiste à prendre la distance entre les points les plus proches de l'origine de chaque droite (cf. modélisation des droites en 3D).

Distance Droite-Plan (uniquement si parallèles)

- **Objets :** Droite p et Plan q.
- **Paramètres :** d, un réel.
- **Relation interne :** $d > 0$.
- **Relation externe :** $(q.a * p.x + q.b * p.y + q.c * p.z - q.d)^2 = d^2$. Cette relation exprime le fait que le produit scalaire du vecteur normal au plan par le vecteur liant l'origine au point de la droite le plus proche de l'origine, soustrait de la distance du plan à l'origine, doit valoir la distance cherchée. Cette modélisation tire parti du fait que la contrainte ne peut être posée qu'entre des droites et plans parallèles.

Distance Plan-Plan (uniquement si parallèles)

- **Objets :** Plans p et q.
- **Paramètres :** d, un réel.
- **Relation interne :** $d > 0$.
- **Relation externe :** $(p.d - q.d)^2 = d^2$. Les plans étant nécessairement parallèles, c'est la différence de leurs distances à l'origine qui établit leur distance relative.

Angle Droite-Droite

- **Objets :** Droites p et q.
- **Paramètres :** a, un réel.
- **Relation interne :** $a \in [0, 2\pi]$.
- **Relation externe :** $p.a * q.a + p.b * q.b + p.c * q.c = cos(a)$. Cette relation établit l'angle entre 2 droites à partir de la formule du produit scalaire de leurs vecteurs directeurs. Elle ne permet pas de traiter les angles orientés.

Angle Droite-Plan

- **Objets :** Droite p et Plan q.

- **Paramètres :** a, un réel.
- **Relation interne :** $a \in [0, 2\pi]$.
- **Relation externe :** $p.a * q.a + p.b * q.b + p.c * q.c = sin(a)$. Cette relation établit l'angle entre une droite et un plan comme étant le complément à π de l'angle entre le vecteur directeur de la droite et la vecteur normal au plan. On utilise à nouveau la formule du produit scalaire.

Angle Droite-Plan

- **Objets :** Plans p et q.
- **Paramètres :** a, un réel.
- **Relation interne :** $a \in [0, 2\pi]$.
- **Relation externe :** $p.a * q.a + p.b * q.b + p.c * q.c = cos(a)$. Cette relation établit l'angle entre deux plans comme l'angle entre leurs vecteurs normaux. On utilise à nouveau la formule du produit scalaire.

Incidence Point-Droite

- **Objets :** Point p et Droite q
- **Paramètres :** aucun.
- **Relation interne :** aucune.
- **Relation externe :** $q.a * (p.y - q.y) - q.b * (p.x - q.x) = 0$ et $q.b * (p.z - q.z) - q.c * (p.y - q.y) = 0$. Cf. relation externe des droites en 3D.

Incidence Point-Plan

- **Objets :** Point p et Plan q
- **Paramètres :** aucun.
- **Relation interne :** aucune.
- **Relation externe :** $p.x * q.a + p.y * q.b + p.z * q.c - q.d = 0$. Cf. relation externe des plans en 3D.

Incidence Droite-Plan

- **Objets :** Droite p et Plan q.
- **Paramètres :** aucun.
- **Relation interne :** aucune.
- **Relation externe :** $p.a*q.a+p.b*q.b+p.c*q.c = 0$ et $q.a*p.x+q.b*p.y+q.c*p.z-q.d = 0$. Ceci représente le fait que droite et plan sont parallèles et que le point de la droite le plus proche de l'origine appartient au plan.

Parallélisme Droite-Droite

- **Objets :** Droites p et q.

- **Paramètres :** aucun.
- **Relation interne :** aucune.
- **Relation externe :** $p.a = q.a$ et $p.b = q.b$. Ceci représente l'égalité des vecteurs directeurs des droites, utilisant par là même le fait que ces vecteur sont normés et orientés.

Parallélisme Droite-Plan

- **Objets :** Droite p et Plan q.
- **Paramètres :** aucun.
- **Relation interne :** aucune.
- **Relation externe :** $p.a * q.a + p.b * q.b + p.c * q.c = 0$. Droite et plans sont parallèles si le produit scalaire du vecteur directeur de la droite par le vecteur normal au plan est nul.

Parallélisme Plan-Plan

- **Objets :** Plans p et q.
- **Paramètres :** aucun.
- **Relation interne :** aucune.
- **Relation externe :** $p.a = q.a$ et $p.b = q.b$. Deux plans sont parallèles si leur vecteurs normaux sont égaux ; ceci utilise le fait que ces vecteurs sont normés et orientés.

Annexe B

Le problème MINIMUM-DENSE

Article présenté aux JNPC 2002

Résumé Cet article traite de la complexité du problème MINIMUM-DENSE. La densité est une propriété structurelle d'un graphe modélisant un système géométrique. Elle est utilisée pour détecter des sous-systèmes rigides. Identifier un sous-graphe dense nécessite une analyse des degrés de liberté du système géométrique. Cette analyse est l'étape clé des nombreux algorithmes ayant pour but la détection de rigidité ou la résolution de systèmes géométriques [Dufourd *et al.*, 1998; Bouma *et al.*, 1995]. En particulier, la méthode de rigidification récursive [Kramer, 1992; Hoffmann *et al.*, 2000] utilise cette propriété pour identifier des sous-systèmes rigides à résoudre séparément avant de les assembler.

Le problème MINIMUM-DENSE, consistant à identifier un sous-graphe dense de taille minimum, a été prouvé NP-difficile dans le cas général [Hoffmann *et al.*, 1997]. Dans cet article, nous introduisons une restriction qui rend le problème traitable en temps polynomial. De plus, cette restriction peut être vérifiée en temps polynomial.

Introduction

La modélisation par contraintes est une approche prometteuse dans le domaine de la conception assistée par ordinateur. Elle permet de construire un objet en spécifiant, de façon déclarative, les propriétés géométriques qui le caractérisent.

Un système géométrique peut être transcrit en un ensemble d'équations non-linéaires généralement algébriques. La *rigidification récursive* est l'une des approches principales étudiées par la communauté des contraintes géométriques pour résoudre ces systèmes[1]. La rigidification récursive est une méthode constructive permettant de résoudre un système rigide en identifiant ses sous-parties rigides puis en les assemblant récursivement. Cette méthode opère au niveau géométrique et produit une séquence de petits sous-systèmes

[1]Les méthodes symboliques sont trop lentes en pratique pour attaquer de grands systèmes non-linéaires [Ruiz and M, 1996; Lazard, 1981].

d'équations appelés *blocs*. Chaque bloc de la séquence est alors résolu séparément, fournissant l'ensemble des configurations du sous-système géométrique correspondant. Ces configurations sont alors combinées pour former les solutions du système géométrique global.

Détecter un sous-système rigide de petite taille est l'étape clé de la méthode de rigidification récursive (cf. [Hoffmann *et al.*, 1997]). Différentes approches pour accomplir cette étape ont été proposées.

Les premières tentatives [Dufourd *et al.*, 1998; Kramer, 1992; Fudos and Hoffmann, 1997; Bouma *et al.*, 1995] étaient basées sur la reconnaissance de schémas spécifiques rigides au sein du système. Ces approches étaient limitées par leur répertoires de schémas recensés qui ne pouvaient couvrir toutes les instances de sous-systèmes rigides.

Des approches plus récentes [Lamure and Michelucci, 1998] sont basées sur une analyse des degrés de liberté du système, et l'utilisation d'une propriété structurelle pour la détection de rigidité.

L'approche proposée par Hoffmann et al. [Hoffmann *et al.*, 1997] est aussi basée sur cette propriété. Elle consiste à utiliser un algorithme de flot pour détecter un sous-graphe *minimal dense*. Intuitivement, un sous-graphe dense correspond à un sous-système géométrique (génériquement) rigide. Ce sous-graphe est minimal s'il ne contient aucun sous-graphe dense. L'algorithme de flot, appelé `Minimal_Dense` a une complexité $O(n^2(m+n))$ en pire cas, n étant le nombre de sommets (objets géométriques) du graphe et m son nombre d'arêtes (contraintes géométriques).

Détecter un sous-graphe dense de taille *minimum* est un problème prouvé NP-difficile dans le cas général [Hoffmann *et al.*, 1998]. Personne n'a, dès lors, tenté d'exhiber des sous-classes polynomiales de ce problème. Dans cet article, nous présentons une restriction qui définit une sous-classe de systèmes géométriques pour lesquels l'utilisation de l'algorithme `Minimal_Dense` permet de répondre en temps polynomial au problème MINIMUM-DENSE. De plus, cette restriction, qui impose que le graphe initial ne contienne aucun sous-graphe sur-dense, *est vérifiable en temps polynomial* (cf. fin de la section suivante).

Le plan de l'article est le suivant :

- La section **Le Problème** MINIMUM-DENSE définit la notion de densité. Par mesure de simplicité, nous ne ferons aucun lien avec le domaine géométrique dans cette section, la densité étant uniquement une propriété de graphe.
- La section **Complexité du problème** MINIMUM-DENSE démontre que, sous notre restriction, le problème MINIMUM-DENSE est décidable en temps polynomial par un algorithme basé sur `Minimal_Dense`.
- La section **Application à la Géométrie** établit le lien avec le domaine géométrique. Tout d'abord, nous expliquons comment un système géométrique peut être vu comme un graphe pondéré. Puis nous donnons alors une idée de la méthode de rigidification récursive. Nous présentons enfin le lien entre densité et rigidité. Une fois établi, ce lien nous permet d'étendre notre résultat de complexité du problème MINIMUM-DENSE au problème MINIMUM-STRUCTURELLEMENT-RIGIDE dont l'intérêt est motivé en fin de section.

Le Problème MINIMUM-DENSE

Une analyse des degrés de liberté d'un système géométrique est une analyse structurelle menée sur un graphe pondéré $G = (V, E, w)$ modélisant ce système. Le lecteur se référera à la section **Application à la Géométrie** pour comprendre comment ce graphe est produit à partir du système géométrique.

La fonction de pondération w associe un entier strictement positif à chaque sommet et chaque arête de G. Par extension, nous définissons cette fonction pour des sous-graphes $G' = (V', E', w)$ de G de la façon suivante : $w(G') = \sum_{v \in V'}(w(v)) - \sum_{e \in E'}(w(e))$.

La notion de densité, introduite dans [Hoffmann *et al.*, 1997], peut être définie de la façon suivante :

Définition 51 *Soient $G = (V, E, w)$ un graphe pondéré et W un entier (pas nécessairement positif). G est W***dense** *ssi $w(G) \leq W$.*

G est **exact-Wdense** *ssi $w(G) = W$.*

G est **sur-Wdense** *ssi $w(G) < W$.*

G est **sous-Wdense** *ssi $w(G) > W$.*

La figure B.1(a) présente un graphe pondéré 3dense.

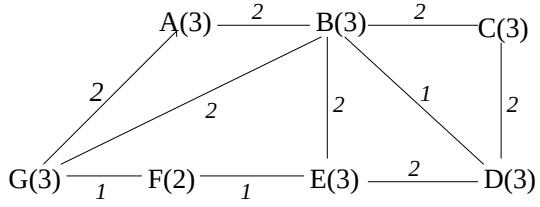

FIG. B.1 – (a) Un graphe pondéré (exact-)3dense. (b) Les sous-graphes $BCDE$ et ABG sont minimaux 3denses. ABG est minimum.

Définition 52 *Soient $G = (E, V, w)$ un graphe pondéré, W un entier et G' un sous-graphe de G.*

G' est **minimal Wdense** *ssi G' est Wdense et ne contient aucun sous-graphe Wdense.*

G' est **minimum Wdense** *ssi G' est Wdense il n'existe aucun sous-graphe Wdense de G ayant moins de sommets que G'.*

Le problème de trouver un sous-graphe minimum Wdense de G est appelé **Minimum Dense**.

La figure B.1(b) présente des exemples de sous-graphes minimaux et minimum 3denses.

La rigidification récursive produit une séquence de *blocs* d'équations. Un sous-graphe identifié comme dense correspondra à l'un de ces blocs. Il est donc souhaitable d'identifier les sous-graphes les plus petits possibles afin d'obtenir des blocs plus faciles à résoudre. Il semble donc intéressant de chercher à identifier des sous-graphes minimum Wdenses.

Hoffmann et al. ont prouvé dans [Hoffmann *et al.*, 1997] que ce problème est NP-difficile dans le cas général[2].

Notre résultat de polynomialité est basé sur la restriction suivante : *Le graphe initial ne contient aucun sous-graphe sur-Wdense*. Nous nommerons cette restriction *non-sur-dense* dans la suite de cet article. Cette restriction peut être vérifiée en temps polynomial : un appel à l'algorithme `Minimal_dense`, décrit dans [Hoffmann *et al.*, 1997], pour une densité $W-1$ détecte un sous-graphe sur-Wdense en temps polynomial s'il en existe un.

Appelons `MD` la classe générale (NP-difficile) du problème MINIMUM-DENSE et $\texttt{MD}_{nsd}$ la sous-classe du problème dont les instances vérifient la restriction *non-sur-dense*.

Complexité du problème MINIMUM-DENSE

Cette section prouve que $\texttt{MD}_{nsd}$ est dans P, c'est-à-dire que la restriction *non-sur-dense* rend le problème MINIMUM-DENSE décidable en temps polynomial.

Ce résultat repose essentiellement sur un autre résultat de polynomialité, celui du problème MINIMAL-DENSE. En effet, l'algorithme `Minimal_Dense` trouve un sous-graphe minimal Wdense en temps polynomial dans un graphe et pour un W quelconques.

La proposition suivante, alliée à l'algorithme `Minimal_Dense`, est la clé de la complexité de $\texttt{MD}_{nsd}$.

Proposition 26 *Soient G un graphe pondéré* **vérifiant la restriction** *non-sur-dense, et G_1, G_2 deux sous-graphes de G.*

Si $G1$ et $G2$ sont minimaux *Wdense, alors ils ne partagent aucune arête.*

Démonstration (par contradiction) Supposons que G_1 et G_2 partagent des arêtes. Puisque G_1 et G_2 sont minimaux denses, G_1 (resp. G_2) ne peut être un sous-graphe de G_2 (resp. G_1).

Soient :

- G_3 l'intersection de G_1 et G_2 ;
- V_3 l'ensemble des sommets de G_3, V_1 celui de $G_1 - G_3$ et V_2 celui de $G_2 - G_3$;
- $v_i = \sum_{v \in V_i} w(v) \quad (1 \leq i \leq 3)$;
- E_{ij} l'ensemble des arêtes joignant un sommet de V_i à un sommet de $V_j \quad (1 \leq i \leq j \leq 3)$;
- $e_{ij} = \sum_{e \in E_{ij}} w(e) \quad (1 \leq i \leq j \leq 3)$.

La figure B.2 illustre ces différents ensembles sur un exemple. Dans cet exemple, les sommets o_1, o_2, o_5 et o_6 sont dans G_1, les sommets o_3, o_4, o_5 et o_6 sont dans G_2 et les sommets o_5 et o_6 sont dans G_3. Il en résulte les ensembles suivants : $V_1 = \{o_1, o_2\}$, $V_2 = \{o_3, o_4\}$, $V_3 = \{o_5, o_6\}$, $E_{11} = \{c_1\}$, $E_{12} = \{c_6\}$, $E_{13} = \{c_4\}$, $E_{22} = \{c_2\}$, $E_{23} = \{c_5\}$ et $E_{33} = \{c_3\}$.

Les relations (1) et (2) établissent le fait que G_1 et G_2 sont minimaux Wdense :

[2]En utilisant une transformation polynomiale depuis Clique [Papadimitriou, 1994], le problème de décision correspondant a été prouvé NP-complet.

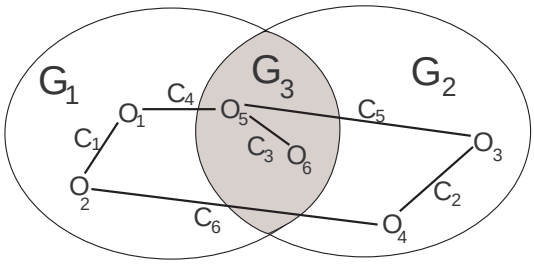

FIG. B.2 – Exemples de graphes G_1 et G_2 ayant pour intersection G_3.

$$\begin{array}{lll} (1) & v_1 + v_3 - e_{11} - e_{13} - e_{33} & = & W \\ (2) & v_2 + v_3 - e_{22} - e_{23} - e_{33} & = & W \end{array}$$

Notez que la restriction *non-sur-dense* empêche les poids de G_1 et de G_2 d'être inférieurs à W.

Par hypothèse, G_3 contient au moins une arête. Ceci implique que G_3 est sous-dense. En effet, si G_3 était Wdense, G_1 et G_2 ne seraient alors pas minimaux puisqu'ils contiendraient un sous-graphe dense plus petit. Le relation (3) exprime le fait que G_3 est sous-dense.

$$(3) \quad v_3 - e_{33} \quad = \quad W + k \quad (\mathbf{k > 0})$$

Le relation suivante correspond au calcul du poids de $G_1 \cup G_2$:

$$\begin{array}{lll} (4) & w(G_1 \cup G_2) & = & v_1 + v_2 + v_3 \\ & & & -e_{11} - e_{13} - e_{22} \\ & & & -e_{23} - e_{33} - e_{12} \end{array}$$

La relation (5) se construit à partir des 3 premières relations :

$$\begin{array}{lll} (5) : (1) + (2) - (3) & = & v_1 + v_2 + v_3 \\ & & -e_{11} - e_{13} - e_{22} \\ & & -e_{23} - e_{33} \\ & & = W - k \end{array}$$

En remplaçant la relation (5) dans la (4) on obtient :

$$(6) \quad w(G_1 \cup G_2) \quad = \quad W - k - e_{12}$$

Puisque $k > 0$ et $e_{12} \geq 0$, la relation (6) signifie que $G_1 \cup G_2$ est sur-dense ($w(G_1 \cup G_2) < W$), ce qui contredit l'hypothèse de la proposition 26.

□

La proposition 26 nous permet de concevoir l'algorithme polynomial `Minimum_Dense`, pour résoudre le problème MINIMUM-DENSE. En effet, les sous-graphes minimaux denses ne partageant pas d'arêtes sous la restriction *non-sur-dense*, on peut calculer l'ensemble de tous les sous-graphes minimaux denses, appelé `Minimaux-Denses` dans l'algorithme, de

Algorithme 21 Minimum_Dense (G : graphe pondéré ; W : entier) **retourne** G' : un sous-graphe minimum Wdense de G

$MinimauxDenses \leftarrow \emptyset$
$G' \leftarrow$ `Minimal_Dense`(G, W) *{Algorithme de flot de Hoffmann et al.}*
Tant que $G' \neq GrapheVide$ **Faire**
  $MinimauxDenses \leftarrow MinimauxDenses \cup G'$
  $G \leftarrow G-$ `Arêtes`(G')
  $G' \leftarrow$ `Minimal_Dense`(G, W)
Fin Tant que
Retourne `Min`$(MinimauxDenses)$ *{*`Min` *retourne l'élément de plus petite taille de l'ensemble passé en paramètre}*

manière gloutonne : L'algorithme `Minimal_Dense` est appelé itérativement pour trouver un sous-graphe minimal dense, dont les arêtes sont alors retirés du graphe principal avant la prochaine itération. La proposition 26 assure qu'aucun sous-graphe minimal dense n'est raté de cette façon. Le sous-graphe minimum dense est alors le plus petit dans l'ensemble `Minimaux-Denses`.

Complexité de $\texttt{MD}_{nsd}$ **:** l'algorithme `Minimum_Dense` est basé sur des appels à l'algorithme `Minimal_Dense` qui est $O(n^2(m+n))$. Le nombre d'appels à cet algorithme est borné par le nombre d'arêtes m dans le graphe initial. Ainsi, `Minimum_Dense` est $O(m \times n^2(m+n))$ et $\texttt{MD}_{nsd}$ est dans P.

Poids bornés : La transformation polynomiale utilisée dans [Hoffmann *et al.*, 1998] pour prouver que `MD` est NP-complet génère des instances de graphes dont les sommets et arêtes ont des poids non-bornés, dépendants de la taille du sous-graphe minimum dense. En pratique, lorsque le graphe pondéré correspond à un système géométrique, ces poids sont bornés. Il est intéressant de noter que cette hypothèse n'est pas utilisée dans notre preuve de polynomialité. En effet, l'algorithme `Minimal_Dense` demeure polynomial que les poids soient bornés ou pas, ceci par propriété de l'algorithme de flot [Ahuja *et al.*, 1993].

Application à la Géométrie

Informellement, un système géométrique est constitué d'un ensemble d'*objets géométriques* dont les positions et orientations doivent satisfaire un ensemble de *contraintes géométriques*. Un objet géométrique est représenté par un ensemble de paramètres géométriques. Une contrainte géométrique est un ensemble d'équations sur les paramètres des objets géométriques qu'elle contraint.

Un système géométrique peut être représenté par un *graphe objets-contraintes* $G = (O, C, w)$, graphe pondéré dont :

- chaque sommet $o \in O$ représente un objet géométrique. Le poids d'un sommet caractérise le nombre de degrés de liberté de l'objet associé, c-a-d, le nombre de paramètres qui doivent être déterminées pour fixer cet objet relativement à un repère global.
- une arête $c \in C$ représente une contrainte géométrique. Son poids correspond au nombre de degrés de liberté retirés par la contrainte (généralement égal au nombre d'équations modélisant la contrainte).

Par exemple, un point en 2D possède 2 degrés de liberté, et une ellipse 3. Les sommets correspondants auront respectivement poids 2 et 3. Une distance entre points retire 1 degré de liberté, et une coïncidence 2. Les arêtes correspondantes auront respectivement poids 1 et 2. Le poids d'un sous-graphe correspond alors au nombre de degrés de liberté du sous-système correspondant.

Rigidification récursive

La rigidification récursive est l'une des principales méthodes utilisée pour résoudre les systèmes géométriques. Elle produit une séquence de sous-systèmes rigides plus simples à résoudre que le système complet. Informellement, un système est rigide s'il est indéformable et déplaçable dans tout l'espace géométrique considéré.

Le schéma de la méthode de rigidification récursive est le suivant :

1. Trouver un sous-système rigide
2. Tant que possible, *étendre* ce sous-système[3]
3. Remplacer le sous-système courant par un sommet unique représentant un repère local associé à ce sous-système pour la résolution
4. Répéter les étapes 1 à 3 jusqu'au point fixe[4]

Lorsque le point fixe est atteint, chaque sous-système rigide est résolu dans le repère local qui lui est associé à l'étape 3 de la méthode. Ensuite, toutes les solutions des sous-systèmes sont combinées en des solutions du système global.

Résoudre un sous-système rigide dans son repère local revient à résoudre un ensemble de blocs d'équations. Un bloc d'équations correspond soit à l'étape 1 de la méthode, soit à *une* extension de l'étape 2 de la méthode. De façon général, un bloc d'équations comprenant moins d'équations sera plus facile à résoudre. C'est pourquoi il est crucial pour les performances de cette méthode que les blocs d'équations produits par les étapes 1 et 2 soient aussi petits que possibles.

Comme il sera expliqué à la section suivante, sous l'hypothèse où la restriction *non-sur-dense* est vérifiée, un sous-graphe $\frac{d(d+1)}{2}$dense *correspond* à un sous-système rigide. Il semble donc intéressant d'utiliser l'algorithme `Minimum_Dense` pour la détection de sous-systèmes rigides de taille minimum (étape 1 de la méthode de rigidification récursive).

[3]Une extension d'un sous-système rigide consiste à lui ajouter l'un de ses objets géométriques voisins de telle sorte que le sous-système résultant soit encore rigide.

[4]Un point fixe est atteint lorsque l'étape 1 échoue, c'est à dire qu'il n'existe plus aucun sous-système rigide dans le système.

Rigidité structurelle

La méthode de rigidification récursive fait appel à la propriété de *rigidité structurelle*. La rigidité structurelle est une généralisation de la caractérisation [Laman, 1970] de la rigidité pour les systèmes à barres en 2D (systèmes constitués uniquement de points et distances). Elle est considéré comme une bonne approximation de la rigidité dans le cas général. Cette notion est très proche de la notion de densité :

Définition 53 *Soit G le graphe objets-contraintes correspondant à un système géométrique S.*

En dimension d, S est **structurellement rigide** *ssi G est $\frac{d(d+1)}{2}$dense et ne contient aucun sous-graphe sur-$\frac{d(d+1)}{2}$dense.*

Il est immédiat de vérifier que, sous l'hypothèse *non-sur-dense*, un graphe objets-contraintes $\frac{d(d+1)}{2}$dense correspond toujours à un système structurellement rigide.

Considérons que le graphe présenté à la figure B.1 correspond à un système géométrique en 2D. On peut constater que ce système est structurellement rigide puisque le graphe est 3dense et vérifie la restriction. Les 3 degrés de liberté du graphe représentent le fait que les objets de ce système sont rigides relativement les uns aux autres et peuvent être déplacés (translations et rotation) conjointement.

Complexité du problème Minimum Structurellement Rigide

Les résultats de complexité obtenus pour les classes `MD` et $\texttt{MD}_{nsd}$ du problème MINIMUM-DENSE peuvent être appliqués aux classes correspondantes du problème Minimum Structurellement Rigide, le problème de la recherche d'un sous-système structurellement rigide de taille minimum. Soient `SR` la classe la plus générale de ce problème, et $\texttt{SR}_{nsd}$ la sous-classe de `SR` sous la restriction *non-sur-dense*.

- `SR` est aussi NP-difficile. La preuve de NP-difficulté de `MD` [Hoffmann *et al.*, 1998] s'appliquait uniquement pour une densité $W = 0$, alors que la propriété de rigidité structurelle considère $W = \frac{d(d+1)}{2}$ en dimension $d > 0$. Cependant, la transformation polynomiale depuis Clique peut être généralisée à tout W.
- Comme expliqué plus haut, sous la restriction *non-sur-dense*, la rigidité structurelle et la $\frac{d(d+1)}{2}$-densité sont équivalentes. Puisque `Minimum_Dense` peut trouver un sous-graphe minimum Wdense pour tout W, $\texttt{SR}_{nsd}$ est aussi dans P.

Intérêt du problème Minimum Structurellement Rigide

Comme expliqué auparavant, chaque étape de la méthode de rigidification récursive produit un bloc d'équations qu'il faudra résoudre. Ceci explique l'importance de la taille des sous-graphes identifiés par l'étape 1 de cette méthode. Il semble donc intéressant d'utiliser l'algorithme `Minimum_Dense` pour accomplir cette étape quand cela est possible.

En pratique, des heuristiques sont parfois associées à l'algorithme `Minimal_Dense` pour l'accomplissement de l'étape 1 de la rigidification récursive. Ces heuristiques ne peuvent

pas toujours être simplement prises en compte si c'est l'algorithme `Minimum_Dense` qui accomplit cette étape.

Notamment, une heuristique communément associée à l'algorithme `Minimal_Dense` vise à assurer la correction de l'étape 3 de la rigidification récursive. En effet, cette étape remplace le sous-système identifié par un repère local. Assurer qu'un sous-système peut être remplacé par un repère local est un problème difficile dans le cas général. L'heuristique visant à répondre à cette question est la suivante : un sous-système est remplaçable par un repère uniquement s'il contient au moins $d+1$ objets en dimension d.

L'algorithme `Minimal_Dense` peut prendre en compte cette heuristique pour ne produire que des sous-graphes denses contenant au moins k objets, pour un k quelconque. Il en résulte que les sous-graphes retournés par cette version de `Minimal_Dense` ne sont pas nécessairement minimaux, puisqu'ils doivent contenir au moins k objets. Ainsi, la proposition 26 sur laquelle est basée notre résultat de complexité et notre algorithme `Minimum_Dense` ne peut s'appliquer pour ces sous-graphes, c-a-d, ils peuvent partager des arêtes. L'algorithme `Minimum_Dense` ne peut donc tenir compte de cette heuristique pour produire des sous-graphes *minimum dense de taille au moins k*.

Cependant, bien qu'intéressante en pratique pour les systèmes à barres [Whiteley, 1987; Hendrickson, 1992; Lamure and Michelucci, 1998], cette heuristique n'est ni nécessaire (cf. exemple 13) ni suffisante (cf. exemple 14) pour assurer qu'un système peut être remplacé par un repère local. C'est pourquoi nous lui préférons le critère de taille minimum des blocs à résoudre, penchant en faveur de l'utilisation de l'algorithme `Minimum_Dense`.

Exemple 13 *Soit un système géométrique en 3D constitué de 2 droites liées par 2 contraintes : une contrainte d'angle et une contrainte de distance*[5]*. Le graphe objets-contraintes correspondant est constitué de 2 sommets de poids 4, liés par 2 arêtes de poids 1. Ce sous-système est bien* 6*dense, structurellement rigide et rigide en 3D. De plus, ce système peut être remplacé par un repère local : il suffit de prendre le vecteur directeur de l'une des droites, le vecteur basé sur les points les plus proches de ces 2 droites et le produit vectoriel de ces 2 premiers vecteurs pour obtenir un repère local valide pour remplacer ce système. Cependant, ce sous-système ne vérifie par l'heuristique proposée pour assurer qu'il est bien remplaçable par un repère local : il ne contient que 2 objets au lieu des 4 préconisés par l'heuristique pour un système en 3D.*

Exemple 14 *Soit le système géométrique suivant en 3D :*

- *Objets : 3 Points* (P_1, P_2, P_3) *et 1 droite* (L)
- *Contraintes : 3 Incidences* ($P_1 \in L, P_2 \in L, P_3 \in L$) *et 2 Distances* ($||P_1P_2|| = d_1, ||P_1P_3|| = d_2$)

Ce système est bien rigide en 3D. Cependant, il ne peut être remplacé par un repère local. En effet, il existe une infinité de repères locaux dans lesquels tous ces objets ont les mêmes coordonnées locales : il suffit que l'un des vecteurs du repère local soit colinéaire avec la

[5] Une contrainte d'angle entre deux droites en 3D se calcule à partir de l'angle entre leurs vecteurs directeurs ; une contrainte de distance entre droites en 3D établit la distance des deux points les plus proches de deux droites non-sécantes.

droite L. Pourtant, ce sous-système vérifie l'heuristique puisque le système comporte bien 4 objets en 3D.

Conclusion

De façon générale, détecter un sous-graphe dense de taille minimum est un problème NP-difficile (MD). Cependant, cet article présente une restriction définissant une sous-classe polynomiale de ce problème ($\texttt{MD}_{nsd}$). Cette restriction, qui impose que le graphe complet ne contienne aucun sous-graphe sur-dense, peut aussi être vérifiée en temps polynomial. Ces résultats devraient encourager la communauté des contraintes géométriques à utiliser l'algorithme `Minimum_Dense`, particulièrement pour améliorer les méthodes de rigidification récursive existantes.

Enfin, cet article a illustré l'étroite relation entre les problèmes MINIMAL DENSE et MINIMUM-DENSE, et l'importance de notre restriction pour l'obtention de sous-classes polynomiales.

Annexe C

Tables pour le calcul du DDR

Dans cette annexe, nous regroupons les différentes tables concernant le calcul du degré de rigidité (cf. section 6.2, page 175) concernant les classes de GCSP ({Points, Droites}, {Distances, Angles, Incidences, Parallélismes}) (notée C^1_{GCSP} dans la suite de cette annexe) en 2D et ({Points, Droites, Plans}, {Distances, Angles, Incidences, Parallélismes}) (notée C^2_{GCSP} dans la suite de cette annexe).

Modèles DDR-minimaux et rigidifications associées

La proposition 19 (page 181) établit le fait que le nombre de modèles DDR-minimaux est fini pour une classe de GCSP donnée. Ce nombre est, approximativement, en $C^N_{\frac{d(d-1)}{2}+1}$ pour une classe de GCSP en dimension d proposant N types d'objets géométriques différents. Pour la classe C^1_{GCSP} (resp. C^2_{GCSP}), on doit donc s'attendre à avoir au plus 4 (resp. 64) modèles DDR-minimaux. Nous allons montrer par énumération que cette classe dispose en réalité de 3 (resp. 11) modèles DDR-minimaux différents.

Notre démonstration par énumération repose sur le schéma suivant : Nous construisons toutes les paires d'objets de la classe considérée et leur associons toutes les rigidifications possibles. Rappelons que toute paire d'objets rigidifiable constitue un modèle DDR-minimal. Ensuite, un objet est introduit dans chaque paire d'objet n'ayant pas encore un DDR maximum de toutes les façon rigides possibles. Si un triplet ainsi généré possède un DDR supérieur à ceux de ces sous-ensembles, il s'agit d'un nouveau modèle DDR-minimal. On applique à nouveau ce principe jusqu'à ce qu'il n'existe plus aucun sur-ensemble des modèles actuels qui soit un nouveau modèle DDR-minimal.

Cette procédure incrémentale est validée par les propositions 15 et 18 (pages 178 et 180). La proposition 17 (page 179), pour sa part, assure qu'il suffit d'effectuer $\frac{d(d-1)}{2}$ itérations de cette procédure pour identifier tous les modèles DDR-minimaux.

Elle permet de plus d'identifier les rigidifications pouvant être associées à chaque modèle DDR-minimal.

Classe C^1_{GCSP}

1. {Point p, Point p} : {Incidence(p,q)}$\rightarrow DDR = 2$ ou {Distance(p,q)}$\rightarrow DDR = 3$
2. {Point p, Droite q} : {Incidence(p,q)}$\rightarrow DDR = 3$ ou {Distance(p,q)}$\rightarrow DDR = 3$
3. {Droite p, Droite q} : {Parallélisme(p,q), Distance(p,q)}$\rightarrow DDR = 2$ ou {Angle(p,q)}$\rightarrow DDR = 3$
4. {Droite p, Droite q, Point r} : Ce triplet contient la paire {Point r, Droite p} qui est déjà de DDR maximum et ne peut donc être un modèle DDR-minimal.

Classe C^2_{GCSP}

1. {Point p, Point q} : {Incidence(p,q)}$\rightarrow DDR = 3$ ou {Distance(p,q)}$\rightarrow DDR = 5$
2. {Point p, Droite q} : {Incidence(p,q)}$\rightarrow DDR = 5$ ou {Distance(p,q)}$\rightarrow DDR = 6$
3. {Point p, Plan q} : {Incidence(p,q)}$\rightarrow DDR = 5$ ou {Distance(p,q)}$\rightarrow DDR = 6$
4. {Droite p, Droite q} : {Parallélisme(p,q), Distance(p,q)}$\rightarrow DDR = 5$ ou {Angle(p,q), Distance(p,q)}$\rightarrow DDR = 6$
5. {Droite p, Plan q} : {Parallélisme(p,q), Distance(p,q)}$\rightarrow DDR = 5$ ou {Angle(p,q)}$\rightarrow DDR = 6$
6. {Plan p, Plan q} : {Parallélisme(p,q), Distance(p,q)}$\rightarrow DDR = 3$ ou {Angle(p,q)}$\rightarrow DDR = 5$

   Pour les triplets d'objets, nous considérons sans perte de généralité que les objets ne sont pas confondus : en cas contraire, ils se ramènent toujours à une paire d'objets déjà étudiée.
7. {Point p, Point q, Point r} : {Distance(p,q), Distance(p,r), Alignement(p,q,r)}$\rightarrow DDR = 5$ ou {Distance(p,q), Distance(p,r), Distance(q,r)}$\rightarrow DDR = 6$
8. {Point p, Point q, Droite r} : ce triplet ce ramène au modèle 2 ; si les points sont sur la droite il a $DDR = 5$, sinon $DDR = 6$. Il ne s'agit donc pas d'un modèle DDR-minimal.
9. {Point p, Point q, Plan r} : si l'un des 2 points n'est pas incident au plan, ce triplet se ramène au modèle 3 et n'est pas un modèle DDR-minimal. Sinon, il peut être rigidifié par {Incidence(p,r), Incidence(q,r), Distance(p,q)}$\rightarrow DDR = 6$ et constitue un modèle DDR-minimal.
10. {Point p, Droite q, Droite r} : ce triplet se ramène au modèle 2 ou 4 car les 2 droites ne peuvent pas être confondues par hypothèse sur les triplets considérés ; alors il a toujours $DDR = 6$ mais contient toujours une paire ayant même DDR : ce n'est pas un modèle DDR-minimal.

11. {Point p, Droite q, Plan r} : même dans le cas {Incidence(p,q),Incidence(q,r)}, ce triplet possède $DDR = 6$. Il s'agit d'un modèle DDR-minimal dont la valeur ne varie jamais et qui ne nécessite donc pas l'appel au mécanisme de vérification de la G-consistance.
12. {Point p, Plan q, Plan r} : si les plans sont parallèles, ce triplet se ramène au modèle 3. Sinon, il peut être rigidifié par {Incidence(p,q), Incidence(p,r), Angle(q,r)}$\rightarrow$ $DDR = 6$ ou {Incidence(p,q), Distance(p,r), Angle(q,r)}$\rightarrow$ $DDR = 6$ ou {Distance(p,q), Distance(p,r), Angle(q,r)}$\rightarrow$ $DDR = 6$. Dans tous les cas, il s'agit d'un modèle DDR-minimal puisqu'il offre $DDR = 6$ alors que les paires d'objets qu'il contient offrent au plus $DDR = 5$.
13. {Droite p, Droite q, Droite r} : ce triplet se ramène toujours au modèle 4 : si toutes les droites sont parallèles, il a $DDR = 5$, sinon $DDR = 6$; dans tous les cas, il ne s'agit pas d'un nouveau modèle DDR-minimal.
14. {Droite p, Droite q, Plan r} : ce triplet se ramène toujours au modèle 4 ou 5 : si les droites sont parallèles et parallèles au plan, il a $DDR = 5$, sinon $DDR = 6$; dans tous les cas, il ne s'agit pas d'un nouveau modèle DDR-minimal.
15. {Droite p, Plan q, Plan r} : ce triplet se ramène toujours au modèle 5 ou 6 : si les plans sont parallèles et parallèles à la droite, il a $DDR = 5$, sinon $DDR = 6$; dans tous les cas, il ne s'agit pas d'un nouveau modèle DDR-minimal.
16. {Plan p, Plan q, Plan r} : si les trois plans sont parallèles, ou que leurs droites d'intersections sont deux à deux parallèles, ce triplet se ramène aux modèles 6 ou 15. Sinon, il se ramène au modèle 6. Dans tous les cas, il n'est pas un nouveau modèles DDR-minimal.

    Le seul quadruplet ne contenant pas un triplet ayant déjà DDR=6 est :
17. {Plan p, Plan q, Plan r, Plan s} : si les plans sont tous parallèles, ou que leurs droites d'intersections sont deux à deux parallèles, ce quadruplet se ramène aux modèles 6 ou 15. Sinon, il se ramène au modèle 6. Dans tous les cas, il n'est pas un nouveau modèles DDR-minimal.

Modèles de rigidification simple

Nous donnons ici les modèles de rigidifications simples pour la classe de GCSP C^1_{GCSP}. Cette classe dispose de 3 modèles DDR-minimaux : {Point p, Point p}, {Point p, Droite q} et {Droite p, Droite q}. Nous allons montrer comment fixer un point ou une droite relativement à chacun de ces 3 modèles lorsqu'il est associé à une rigidification qui lui confère $DDR = 3$. Rappelons en effet que le mécanisme de rigidification automatique (cf. section 7.5.1, page 215) n'opère la rigidification de l'ensemble des objets frontières d'un agrégat à condenser qu'à partir d'un sous-ensemble DDR-minimal de ces objets possédant un DDR maximum (3 en 2D, 6 en 3D).

Nous n'avons pas encore produit tous les modèles de rigidification simples pour la classe de GCSP ({Points, Droites, Plans},{Distances, Angles, Incidences, Parallélismes}) en 3D.

Rigidification simple d'un point

({Point p, Point p}, Point r) : la paire de points (p, q) ne peut être rigidifiée que par une distance, autrement elle ne constitue pas un modèle DDR-minimal à $DDR = 3$. On peut alors rigidifier le point r par {Incidence(p, r)} ou {Distance(p, r), Distance(q, r)}.

({Point p, Droite q}, Point r) : que le point p soit incident ou non à la droite q ne change pas le DDR de la paire (p, q). On peut rigidifier le point r indépendamment de la rigidification retenue pour (p, q) par {Incidence(p, r)} ou {Distance(p, r), Distance(q, r)} ou {Distance(p, r), Incidence(q, r)}.

({Droite p, Droite q}, Point r) : la paire (p, q) ne constitue un modèle DDR-minimal de $DDR = 3$ que si p et q sont rigidifiés par un angle. On peut alors rigidifier r par {Incidence(p, r), Incidence(q, r)} ou
{Distance(p, r), Distance(q, r)} ou {Distance(p, r), Incidence(q, r)}.

Rigidification simple d'une droite

({Point p, Point p}, Droite r) : la paire de points (p, q) ne peut être rigidifiée que par une distance, autrement elle ne constitue pas un modèle DDR-minimal à $DDR = 3$. On peut alors rigidifier la droite r par {Incidence(p, r), Incidence(q, r)} ou {Incidence(p, r), Distance(q, r)} ou {Distance(p, r), Distance(q, r)}.

({Point p, Droite q}, Droite r) : que le point p soit incident ou non à la droite q ne change pas le DDR de la paire (p, q). On peut rigidifier la droite r indépendamment de la rigidification retenue pour (p, q) par {Incidence(p, r), Angle(q, r)} ou {Incidence(p, r), Parallélisme(q, r)} ou {Distance(p, r), Angle(q, r)} ou {Distance(p, r), Parallélisme(q, r)}.

({Droite p, Droite q}, Droite r) : la paire (p, q) ne constitue un modèle DDR-minimal de $DDR = 3$ que si p et q sont rigidifiés par un angle. On peut alors rigidifier r par {Angle(p, r), Angle(q, r)} ou {Parallélisme(p, r), Distance(p, r)} ou {Incidence(p, r)}.

Annexe D

Résultats expérimentaux

Cette annexe présente des résultats expérimentaux préliminaires sur quelques exemples didactiques de GCSP. Dans ces expérimentations, nous comparons la décomposition équationnelle DM-CFC (cf. section 4.1.1) réalisée par l'algorithme OpenPlan [Bliek *et al.*, 1998], la décomposition HLS utilisant CA, FA, MFA (cf. chapitre 5), notre amélioration CA+MFA (cf. section 5.6) et notre nouvelle méthode de rigidification récursive (cf. chapitre 7), que nous notons **FAR** dans ce chapitre. Chacune de ces décompositions est suivie de la phase d'assemblage que nous avons décrite au chapitre 8 afin de permettre une meilleure comparaison[1]. Pour ce faire, le DAG de blocs issu de la décomposition DM-CFC est résolu par la méthode de Bliek, Neveu et Trombettoni [Bliek *et al.*, 1998], et le plan produit par la méthode HLS est utilisé comme le plan produit par notre méthode : il sert à générer un DAG de blocs (cf. section 8.2) qui est alors résolu par des techniques d'intervalles (cf. section 8.3).

Pour la phase d'assemblage, nous comparons les différentes stratégies proposées en section 8.3.1 : résolution par blocs (B), résolution par blocs avec propagation globale (P) et résolution globale (G). Une quatrième résolution sert de témoin : la résolution par intervalles du GCSP non décomposé (ND) ; cela permet d'illustrer l'intérêt de la décomposition. Pour chaque stratégie où cela est possible, l'heuristique du point milieu est utilisée.

Description des GCSP

Nous avons considéré 4 exemples en 2D et 4 exemples en 3D.

Exemples en 2D

- Le GCSP Minimal (cf. figure D.1-a) est constitué de 7 points liés par 11 contraintes de distance, fixées de sorte qu'il n'admette que 8 solutions.
- Le GCSP *Ponts* (cf. figure D.1-b) est constitué de 15 points et 27 contraintes de distance ajustées de sorte qu'il n'admette que 128 solutions.

[1] Rappelons que la phase d'assemblage n'a jamais été formellement décrite par Hoffmann *et al.*.

- Le GCSP Tangent (cf. figure D.1-c) est constitué de 6 points et 4 droites liés par 6 contraintes de distance, 3 contraintes d'angle et 8 contraintes d'incidence. Il admet 32 solutions. Ce GCSP est tiré de l'article [Fudos and Hoffmann, 1997].
- Le GCSP Parall (cf. figure D.1-d) est constitué de 4 points et 4 droites liés par 13 contraintes : 2 distances, 1 angle, 8 incidences et 2 parallélismes ; il admet 4 solutions.

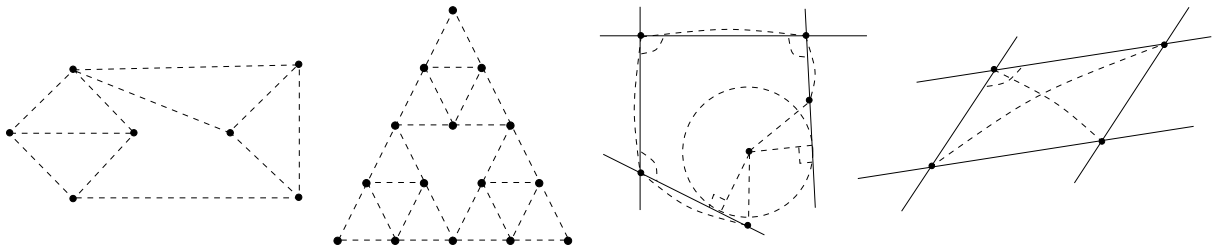

FIG. D.1 – GCSP en 2D : (a) Minimal, (b) Ponts, (c) Tangent et (d) Parall

Exemples en 3D

- Le GCSP Tétra *généralise* le GCSP Ponts en 3D (cf. figure D.2-a). Il est constitué de 9 points liés par 24 contraintes de distances, fixées de sorte qu'il n'admette que 128 solutions.
- Le GCSP Droites (cf. figure D.2-b) est constitué de 7 points et 4 droites, liés par 10 contraintes de distance, 10 incidences et 1 angle.
- Le GCSP Paral3D (cf. figure D.2-c) est constitué de 5 points et 4 droites ; il admet deux solutions.
- Le GCSP Sablier (cf. figure D.2-d) est constitué de 7 points et 2 plans, liés par 13 contraintes de distances, 6 incidences et 1 parallélisme. Il admet 2 solutions.

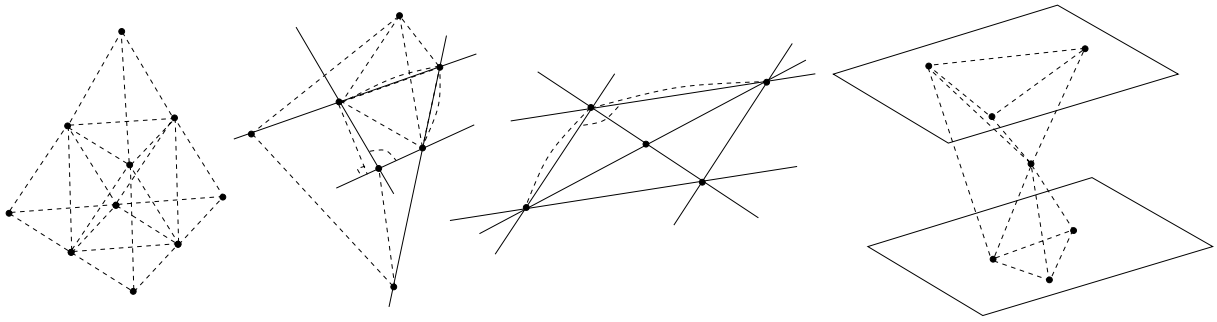

FIG. D.2 – GCSP en 3D : (a) Tétra, (b) Droites, (c) Paral3D, (d) Sablier

Comparaison des décompositions

Pour comparer les différentes méthode de décomposition, nous avons retenu 3 critères : TP la taille du GCSP non décomposé en nombre de variables, NB le nombre de blocs constituant le DAG résultant de chaque décomposition, et TB, la taille, en nombre d'équations, du plus gros bloc du DAG résultant de chaque décomposition. Ces trois informations permettent d'obtenir une évaluation de la finesse de la décomposition :

- Plus le nombre NB de blocs augmente, et plus la taille moyenne des blocs du DAG diminue. Cependant, rappelons que les décompositions géométriques introduisent toutes des variables et des équations supplémentaires dans le GCSP : l'opération de condensation remplace un agrégat par un objet-repère sur lequel sont reportés des contraintes dans le cas de HLS, et par un nouvel ensemble de contraintes entre les objets frontières dans le cas de FAR. Ainsi, le nombre de bloc seul ne permet pas de juger de la finesse de la décomposition.
- C'est pourquoi la taille TP du GCSP est aussi un critère important. Pour la décomposition DM-CFC, ce nombre est égal à la taille réel du GCSP puisque cette décomposition n'introduit pas de variables/équations supplémentaires. La taille du problème TP ne correspond pas exactement au nombre de degrés de liberté des objets des GCSP tels que présentés à la section précédente ; en effet, la modélisation des objets et contraintes présentée en annexe A n'est pas toujours minimale, c'est-à-dire qu'elle implique parfois plus de variables et d'équations que nécessaire. Par exemple, la modélisation utilisée pour une droite en 2D implique 4 variable et 2 équations (relation interne) alors qu'une droite pourrait se modéliser à l'aide de 2 variables seulement. Cette modélisation a cependant été choisie car elle simplifie l'expression des contraintes impliquant une droite.
- Enfin, la taille TB du plus gros bloc conditionne aussi la qualité de la décomposition, car le plus gros bloc sera généralement celui qui capitalisera le plus de temps CPU au moment de la résolution. Ainsi, si TB est proche de TP/NB, cela signifie que la décomposition est plus uniforme.

Rappelons que les décompositions géométriques fixent un repère par agrégat maximal identifié (fusion + extensions). Fixer un repère correspond à 3 blocs (resp. 6 blocs) de taille 1 en 2D (resp. 3D). On peut considérer que ces blocs, qui consistent simplement à affecter une valeur à une variable, sont résolus en temps constant négligeable devant celui de tout autre bloc. Nous avons noté entre parenthèses dans la colonne NB le nombre de blocs de ce type pour chaque décomposition géométrique. La décomposition équationnelle fixe toujours une seule fois le repère de façon semblable.

Le tableau D.3 présente ces jauges pour chacun des 8 GCSP et pour chacune des 6 décompositions. Ces résultats ont été produits en utilisant uniquement un prototype de décomposition équationnelle DM-CFC. Afin de simuler les décompositions géométriques à l'aide de ce prototype, nous avons procédé de la façon suivante :

1. Calcul manuel d'une décomposition en agrégats.
2. Génération manuelle du système d'équations correspondant à chaque agrégat.

3. Décomposition DM-CFC du système d'équation correspondant à chaque agrégat.

Ce procédé a permis d'obtenir des décompositions parfois plus fines que la simple décomposition géométrique seule. Il illustre bien le fait qu'une décomposition géométrique peut être suivie d'une décomposition équationnelle. Dans le tableau D.3, nous avons présenté en gras la décomposition qui nous paraît la plus fine : celle qui minimise la différence entre TP/NB et TB.

	DM-CFC				CA				FA				MFA				CA+MFA				FAR			
	TP	NB	NU	TB	TP	NB	NU	TB	TP	NB	NU	TB	TP	NB	NU	TB	TP	NB	NU	TB	TP	NB	NU	TB
Minimal	14	7	3	6	18	12	6	4	30	21	9	6	18	13	6	6	30	19	9	6	**18**	**13**	**6**	**2**
Ponts	30	11	3	14	42	26	12	9	**54**	**33**	**12**	**2**	54	31	12	10	54	31	12	10	38	25	9	14
Tangent	**28**	**17**	**3**	**2**	40	26	12	4	X	X	X	X	48	28	9	4	46	29	12	4	42	25	9	4
Parall	24	12	3	10	**36**	**24**	**12**	**6**	24	12	3	10	24	12	3	10	36	24	12	6	24	12	3	10
Tetra	**30**	**13**	**6**	**9**	48	28	18	12	30	13	6	9	30	13	6	9	48	28	18	12	66	45	30	9
Droites	45	14	6	15	54	24	12	9	45	14	6	15	45	14	6	15	X	X	X	X	**54**	**26**	**12**	**6**
Paral3D	**51**	**19**	**6**	**8**	X	X	X	X	51	19	6	8	51	19	6	8	X	X	X	X	72	22	12	32
Sablier	29	14	6	10	38	20	12	9	29	14	6	10	29	14	6	10	38	20	12	9	**39**	**25**	**12**	**4**

FIG. D.3 – Comparaison des décompositions DM-CFC, CA, FA, MFA, CA+MFA et FAR

Analyse des résultats

On peut tout d'abord constater que les décompositions géométriques augmentent la taille du système d'équations correspondant au GCSP de façon non négligeable, allant parfois jusqu'à la doubler (cf. GCSP Minimal, décomposition FA ; GCSP Tétra, décomposition FAR). Ainsi, la comparaison en terme de "finesse" de la décomposition doit-elle être contrebalancée par la comparaison entre les temps de résolution donnés plus loin dans cette annexe : la résolution d'une décomposition plus fine d'un système d'équations n'est pas nécessairement meilleure que la résolution d'une décomposition moins fine d'un système d'équations deux fois plus petit. Cependant, dans l'augmentation de taille des GCSP décomposés géométriquement, il faut tenir compte du nombre de blocs unitaires qui correspondent à poser des repères pour chaque agrégat maximal identifié. Ces blocs se résolvent en effet en temps constant négligeable et influent donc peu sur la résolution du GCSP décomposé. Par exemple, la décomposition FAR du GCSP Tétra conduit à la résolution d'un nombre total de 66 variables réparties en 45 blocs, soit plus du double du GCSP non-décomposé (30 variables réparties en 13 blocs) ; cependant, sur ces 66 variables, 30 sont résolus en temps constant par fixation des 5 repères locaux correspondant au 5 agrégats identifiés par FAR ; cette décomposition est donc "équivalente", en un sens, à une décomposition d'un GCSP de 36 variables réparties en 15 blocs, qui se compare plutôt favorablement à la décomposition DM-CFC de 24 (30-6) variables en 7 (13-6) blocs.

Les décompositions qui obtiennent les meilleurs résultats sur ces exemples selon le critère de taille du plus gros bloc (TB) comme le critère de "finesse" ($TB - TP/NB$) sont

les décomposition DM-CFC et FAR. Il semble donc que les décompositions géométriques proposées par Hoffmann *et al.* s'avèrent finalement moins efficaces qu'une décomposition équationnelle pourtant moins adaptée aux GCSP, du moins sur ces quelques exemples. Cependant, la décomposition FAR, elle, surclasse la décomposition équationnelle sur 3 exemples, y est équivalente sur 2 autres, et est surclassée par la décomposition équationnelle sur 3 exemples.

Temps de résolution

Dans cette section, nous exposons les temps de résolutions de quelques-unes des décompositions des GCSP 2D et 3D présentés à la section précédente. Ces expérimentations ont été effectuées sur un PC équipé d'un processeur pentium4 cadencé à 2.2GHz et doté de 512Mo de mémoire vive.

Protocole d'expérimentations

La résolution d'un GCSP décomposé, que la stratégie de résolution soit par blocs (B), par bloc avec propagation globale (P), ou globale avec heuristique d'instanciation (G), est interrompue au bout d'1h de temps CPU. La résolution du GCSP non décomposé, elle, est interrompue après 4H de calcul.

Prototype utilisé

Le prototype utilisé pour la résolution est basé sur une implantation de la décomposition DM-CFC effectuée par Christian Bliek, et améliorée depuis par Bertrand Neveu. Cette implantation effectue la décomposition DM-CFC du CSP passé en entrée, produit le DAG de blocs correspondant et utilise Ilog Solver v4.0 [ILO, 1997] pour la résolution de chaque bloc. Les mécanismes de propagation utilisés sont liés à la Box-consistance et à la 2B-consistance.

Nous avons ajouté une couche à ce prototype existant pour traiter les GCSP décomposés. Cette couche permet une formulation plus intuitive des GCSP et de leurs décompositions géométriques. Elle produit un système d'équations correspondant au GCSP ou à l'une des décompositions géométriques du GCSP. Ce système d'équations est alors décomposé par DM-CFC à l'aide du code initial de Bliek et Neveu. Ceci résulte en une décomposition parfois plus fine que la décomposition géométrique initiale, et démontre que les décompositions géométriques peuvent être suivies de décompositions équationnelles permettant d'affiner le DAG résultant.

Comparaison des temps de résolution

Nous avons regroupé dans le tableau D.4 les temps de résolution (en secondes) pour les différentes stratégies et décompositions. Le meilleur temps pour chaque GCSP apparaît

en gras dans la ligne correspondante à chaque GCSP. Le prototype n'est pas suffisamment complet pour avoir permis d'expérimenter de façon fiable tous les GCSP présentés aux figures D.1 et D.2. Les résultats obtenus pour les décompositions FA, MFA et FAR ne sont pas complètement représentatifs, car l'état actuel du prototype ne permet pas de prendre en compte des contraintes annexes, nécessaires pour ces 3 décompositions afin d'éviter la duplication des solutions. Ainsi, il n'est pas rare que les décompositions FA, MFA et FAR trouvent 2 fois plus de solutions, voir jusqu'à 256 fois plus pour la décomposition MFA ! Ceci nuit naturellement aux performances de ces 3 décompositions.

	DM-CFC			CA			FA			MFA			CA+MFA			FAR			ND
	P	B	G	P	B	G	P	B	G	P	B	G	P	B	G	P	B	G	
Minimal	0,23	0,2	0,46	0,1	0,05	0,22	0,76	0,2	3,46	8,75	8,55	3601	8,75	8,55	3601	0,14	**0,02**	0,17	0,63
Ponts	10	5,97	30,9	44,7	43,2	867	9,78	4,19	80,9	3601	3601	3601	3601	3601	3601	**1,77**	2,98	30,4	92,8
Tangent	2,36	**0,08**	120	5,32	0,17	179	19,9	0,36	350	16,2	2,34	3601	9,94	0,59	848	2	**0,08**	9,21	145
Parall	0,42	**0,04**	0,96	0,58	**0,04**	1,5	0,42	**0,04**	0,96	0,42	**0,04**	0,96	0,58	**0,04**	1,5	0,42	**0,04**	0,96	1,02
Tetra	27,4	9,04	112	3601	3601	3601	27,4	9,04	112	27,4	9,04	112	3601	3601	3601	3,02	**2,83**	3601	12000
Sablier	1,81	1,57	8,6	79,9	78,8	720	1,81	1,57	8,6	1,81	1,57	8,6	79,9	78,8	720	0,64	**0,01**	1,12	229

FIG. D.4 – Comparaison des résolutions avec les décompositions DM-CFC, CA, FA, MFA, CA+MFA et FAR

Analyse des résultats

On peut constater que la décomposition FAR se classe très bien parmi les autres méthodes de décomposition puisqu'elle obtient toujours le meilleurs temps, parfois ex-aequo avec une autre méthode. Il s'avère donc que, malgré le fait que la décomposition équationnelle DM-CFC produit une décomposition plus fine que FAR sur 3 exemples, la résolution reste au pire équivalente en temps CPU. Ces résultats préliminaires sont donc très encourageants, surtout que l'on peut voir que le comportement de la méthode est aussi bon sur les GCSP 2D que sur les GCSP 3D.

De façon plus générale, on peut tirer quelques conjectures de ces premières expérimentations :

- La résolution d'un GCSP décomposé est quasiment toujours meilleur que la résolution du même GCSP non décomposé.
- La résolution par bloc est quasiment toujours meilleure que la résolution globale avec heuristique d'instanciation.
- L'intérêt de la propagation n'apparaît pas sur ces exemples, mais serait probablement plus clair si les GCSP étaient plus grands et admettaient moins de solutions (par exemple, si l'on introduisait des contraintes R-redondantes). Cependant, cette constatation peut s'expliquer d'une certaine façon : la décomposition vise à découper un GCSP *en des points de faible couplage*, alors que la propagation vise à transmettre de l'information dans le GCSP via les contraintes ; on peut imaginer que les *points de faible couplage* véhiculent peu d'information de par leur nature même. D'autres

stratégies de propagation tenant mieux compte de la décomposition pourraient alors être envisagées.
- Certaines décompositions (FA, MFA et FAR principalement) dupliquent les solutions car notre prototype ne permet pas d'introduire des contraintes annexes/R-redondantes. Cela nuit bien entendu aux performances des résolutions avec ces décompositions.

De plus amples expérimentations sont prévues dans les perspectives de la thèse. Elles permettront de valider l'approche sur des GCSP de taille plus importante en 3D, et d'obtenir une meilleure analyse des apports des différentes stratégies de résolutions, de l'heuristique du point milieu et de l'impact du mécanisme de retour-arrière utilisé.

Glossaire

agrégat : sous-GCSP rigide ; terme issu des méthode de rigidification récursive qui *agrègent* peu à peu des sous-parties rigides de GCSP.
A-redondance : contrainte dont le retrait ne change pas la catégorie de l'espace des solutions du GCSP : il reste bien-, sous- ou sur- contraint.
arête-séparation (2-, 3-, 3-double-, 7-) : opérations structurelles de construction de systèmes à barres rigides initialement proposées par Henneberg.

bien-$k$dense : se dit d'un graphe pondéré dont la somme des poids des sommets, moins la somme des poids des arêtes est égale à 0.
bien-contraint : CSP possédant un ensemble dénombrable de solutions.
bien-es_rigide : propriété des GCSP admettant exactement DDR DDL et dont tout sous-GCSP admet au moins DDR DDL.
bien-rigide : propriété des GCSP dont les solutions admettent (ou d'une solution d'un GCSP qui admet) les déplacements mais pas de déformation.
bien-s_contraint : CSP carré, c-a-d. constitué d'autant d'équations que de variables.
bien-s_rigide : propriété des GCSP admettant exactement $\frac{d(d+1)}{2}$ DDL en dimension d et dont tout sous-GCSP en admet au moins autant.
bisection : dichotomie.
bloc : sous-système d'équations structurellement bien-contraint (n équations pour n inconnues).

CA : *Condensing Algorithm*; méthode de décomposition géométrique structurelle dite de rigidification récursive proposée par Hoffmann *et al.* [Hoffmann *et al.*, 1997].
CAO : Conception Assistée par Ordinateur.
CFC : Composante Fortement Connexe.
cluster : agrégat.
condensation : opération des méthodes de décomposition géométrique structurelle dites de rigidification récursive.
conditions de Laman : caractérisation de systèmes à barres rigides en 1D et 2D.
configuration : ensemble de positions et d'orientations pour les objets d'un GCSP.
configuration-solution : configuration satisfaisant les contraintes d'un GCSP.
configuration générale : configuration *linéairement* indépendante.

configuration générique : configuration *algébriquement* indépendante.
conflit : instanciation contradictoire d'une même variable ou inconsistance d'un instanciation partielle des variables d'un CSP.
conformation : configuration.
connexité (2-, 3-, 5-, 6-) : notion issue de la théorie des graphes ; intuitivement, existence d'un (de k pour la k-connexité) chemins reliant n'importe quelle paire de sommets dans un graphe.
consistance globale : état d'un CSP dont toute combinaison de valeurs issues des domaines de ses variables satisfait les contraintes.
consistance partielle/locale (2B-, 3B-, bound-, box-) : propriété de consistance *locale*, liée à un sous-ensemble de contraintes généralement.
consistant : satisfiable ; c-a-d. qu'il existe au moins une solution.
contrainte booléenne : contrainte géométrique ne nécessitant pas la spécification d'une valeur (ex : incidence, parallélisme, symétrie).
contrainte géométrique : contrainte restreignant les positions, orientations et/ou dimensions d'objets géométriques (ex : distance, angle, incidence, parallélisme).
contrainte valuée : contrainte géométrique nécessitant la spécification d'une valeur (ex : distance, angle, homothétie).
CSP : *Constraint Satisfaction Problem* ; parfois noté PSC pour Problème de Satisfaction de Contraintes.

$\frac{\mathbf{d(d+1)}}{\mathbf{2}}$: nombre de déplacements indépendants admis dans un espace géométrique de dimension d.
DAG : *Directed Acyclic Graph* ; graphe orienté sans circuits en français.
DDL : Degré De Liberté.
DDR : Degré De Rigidité.
DDR-minimal : ensemble minimal d'objets possédant un degré de rigidité donné ; tout sous-ensemble d'objets possède un degré de rigidité inférieur (cf. section 6.2.2, page 178).
décomposition équationnelle : méthode visant à découper un CSP en plusieurs sous-CSP.
décomposition géométrique : méthode visant à découper un GCSP en plusieurs sous-GCSP en s'appuyant sur des propriétés géométriques ; la plupart des méthodes géométriques utilisent, sous une forme ou une autre, le concept de rigidité.
déformable (in-) : se dit d'un GCSP qui admet(n'admet pas) des déformations.
déformation : mouvement qui n'est pas un déplacement.
dégénéré : singulier.
degré de mobilité : nombre de déformations (mouvements hors déplacements) dans un mécanisme.
densité : propriété d'un graphe pondéré ; cf. bien-/sous-/sur-$k$dense.
déplaçable (in-) : se dit d'un GCSP qui admet (n'admet pas) les déplacements.
déplacement : mouvement qui correspond à la composition d'une rotation et d'une translation.
DM : décomposition équationnelle structurelle de Dulmage&Mendelsohn.

DM-CFC : décomposition équationnelle structurelle en composantes bien-/sur-/sous-contraintes (DM), puis en composantes fortement connexes (CFC).
domaine : ensemble des valeurs pouvant être affectées à une variable d'un CSP.

es_rigidité : nouvelle caractérisation structurelle de la rigidité basée sur un compte des degrés de liberté et le calcul du degré de rigidité.
extension (2-, 3-) : opérations structurelles de construction de systèmes à barres rigides initialement proposées par Henneberg.
extension : opération constructive heuristique des méthodes de décomposition géométrique structurelle dites de rigidification récursive.
extension-globale : nouvelle opération constructive proposée dans cette thèse.

FA : *Frontier Algorithm*; méthode de décomposition géométrique structurelle dite de rigidification récursive proposée par Hoffmann *et al.* [Hoffmann *et al.*, 2000].
FAR : *Frontier Automatic Rigidification*; nouvelle opération de condensation proposée dans cette thèse. Par extension, nom donné à la nouvelle méthode de décomposition géométrique introduite dans la thèse.
FCSP : *Functional Constraint Satisfaction Problem*; problème de satisfaction de contraintes fonctionnelles en français; formalisme de contraintes issu du domaine de propagation locale.
filtrage : mécanisme visant à éliminer des sous-parties de l'espace de recherche ne contenant pas de solutions.
finesse : caractéristique d'une décomposition qui *mesure* sa qualité; la finesse dépend principalement de la taille du plus gros bloc, mais aussi du nombre de blocs et de la talle du problème.
FRONTIER : logiciel de CAO produit par Sitharam *et al.* [Oung *et al.*, 2001].
frontière : ensemble des objets d'un agrégat qui sont liés par des contraintes à des objets extérieurs à l'agrégat.
fusion : opération constructive principale des méthodes de décomposition géométrique structurelle dites de rigidification récursive.

GBJ : *Graph-based Back Jumping*; mécanisme de retour-arrière intelligent tirant parti de la structure du problème.
G-consistance : consistance géométrique; se dit d'un ensemble de contraintes géométriques qui n'est pas inconsistant.
GCSP : *Geometric Constraint Satisfaction Problem*; problème de satisfaction de contraintes géométriques.
GEOTHER : logiciel de démonstration automatique de théorèmes géométriques produit par Wang [Wang, 2002].
GPB : *Generalized Partial-order Backtracking*; mécanisme de retour-arrière intelligent tirant parti de la structure du problème et de l'existence d'un ordre partiel d'instanciation des variables.
GPDOF : *General(ized) Propagation of Degrees Of Freedom*; algorithme de propagation

des degrés de libertés pour le rétablissement de la consistance d'un FCSP.
graphe objets-contraintes : représentation graphique de la structure d'un GCSP où les sommets sont les objets et les arêtes (hyper-arêtes) sont les contraintes.
graphe résiduel : graphe représentant les capacités restantes dans un réseau après distribution d'un flot.
graphe variables-équations : représentation graphique de la structurel d'un CSP où les sommets sont les variables et les arêtes (hyper-arêtes) sont les équations.

HLS : nom donné dans cette thèse à l'ensemble des méthodes de décomposition géométrique structurelle proposées par Hoffmann, Lomonosov et Sitharam (cf. chapitre 5.

instanciation : affectation de valeurs aux variables d'un CSP.
intervalle : sous-ensemble continu de valeurs réelles comprises entre une borne inférieure a et une borne supérieure b ; noté $[a, b]$, $[a, b[$, $]a, b]$ ou encore $]a, b[$ selon que les bornes sont inclues ou exclues.
intervalle-solution : intervalle résultant de la résolution par intervalle d'un CSP ; cet intervalle représente l'encadrement à précision prescrite d'une, plusieurs, ou aucune solution (dans le cas indiscernable).
irréductible : se dit d'un sous-CSP qui ne peut être décomposé.
isostatique (2-, 3-) : état des systèmes à barres qui sont exactement rigides, c-a-d. dont la suppression d'une barre quelconque entraîne la perte de la rigidité.

Look-Up Table : répertoire de modèles de sous-GCSP bien-rigides et de règles de construction de ces modèles utilisé dans le logiciel FRONTIER.

MFA : *Modified Frontier Algorithm* ; méthode de décomposition géométrique structurelle dite de rigidification récursive proposée par Hoffmann *et al.* [Hoffmann *et al.*, 2000].
MM : *Maximum Matching* ; couplage maximum.
modélisation : représentation sous forme d'équations et de variables d'un objet ou d'une contrainte géométrique, et à la sémantique géométrique associée et permettant de l'interpréter graphiquement.
mouvement : fonction continue qui associe à un temps $t \in [0, 1]$ un ensemble de positions, orientations et dimensions aux objets d'un GCSP.

nœud-contrainte : nœud représentant une contrainte dans le réseau objets-contraintes associé à un GCSP.
nœud-objet : nœud représentant un objet dans le réseau objets-contraintes associé à un GCSP.
No-Goods : conflit rencontré lors de l'instanciation de variables dans la résolution d'un CSP.

objet géométrique : ensemble de points de l'espace géométrique liés par une définition en extension (liste) ou en compréhension (relation mathématique).

OpenPlan : algorithme de décomposition équationnelle structurelle similaire à DM-CFC, mais assurant que la taille du plus grand bloc est minimale (cf. section 4.1.1, page 91).

paramétrisation : ensemble des variables associées à la modélisation d'un objet ou d'une contrainte géométrique.
PC : Propagation de Conflits ; famille d'algorithmes de maintien de consistance issus du domaine de propagation locale.
PDOF : *Propagation of Degrees Of Freedom* ; famille d'algorithmes de maintien de consistance issus du domaine de propagation locale.
phase d'analyse (de planification) : première étape des méthodes de décomposition géométrique qui vise à produire la décomposition.
phase d'assemblage : seconde étape des méthodes de décomposition géométrique qui vise à résoudre la décomposition engendrée à l'étape de planification.
planification : action de produire un plan, c-a-d. une décomposition.
programmation par contraintes : discipline de recherche qui propose un formalisme pour la modélisation des problèmes et un ensemble de méthodes algorithmiques pour leur résolution.
propagation d'intervalles : méthode algorithmique qui utilise une forme de consistance partielle pour opérer un filtrage des domaines d'un CSP.

réalisable : dont il existe une réalisation.
réalisation : solution d'un GCSP.
redondance : contrainte superflue, c-a-d. dont le retrait ne change pas l'ensemble des solutions d'un CSP.
règle-et-compas : méthodes à base de règles permettant de reconstruire des figures géométriques en 2D.
relation externe : relation mathématique représentant une contrainte ou un objet géométrique.
relation interne : relation mathématique que doivent vérifier les paramètres modélisant une contrainte ou un objet géométrique.
réseau objets-contraintes : réseau représentant la structure d'un GCSP et permettant de calculer, par un algorithme de calcul de flot maximum, une distribution optimale des degrés de liberté des contraintes sur les degrés de liberté des variables.
retour-arrière : mécanisme combinatoire permettant de remettre en cause un choix et d'explorer l'ensemble des choix possibles dans un arbre de décision.
rigidification automatique : technique consistant à placer automatiquement des contraintes entre des objets géométriques de façon à les rendre bien-rigide.
rigidification récursive : méthode de décomposition géométrique utilisant la propriété de rigidité et fonctionnant en 2 phases : analyse puis assemblage récursif.
rigidification simple : technique consistant à placer automatiquement des contraintes depuis un sous-GCSP bien-rigide vers un objet afin de former un nouveau sous-GCSP bien-rigide.
rigidité : propriété des GCSP dont les solutions admettent (ou d'une solution d'un GCSP

qui admet) les déplacements mais pas de déformation.
rigidité infinitésimale : notion de rigidité liée aux mouvements infinitésimaux d'un GCSP.
r-methode : méthode de résolution qui permet de satisfaire une ou plusieurs contraintes en calculant une ou plusieurs variables à partir des valeurs associées à un ensemble de variables d'entrée.
R-redondance : contrainte dont le retrait ne change pas la rigidité du GCSP : il reste bien-, sous- ou sur-rigide.

s_rigidité : caractérisation structurelle de la rigidité basée sur un compte des degrés de liberté.
saturé : dans un réseau où circule un flot, un arc est saturé si le flot qui le parcourt est égal à sa capacité.
singularité : présence d'une dépendance algébrique qui confère des propriétés inhabituelles à un CSP.
singularité géométrique : présence d'une dépendance qui confère des propriétés géométriques inhabituelles à un GCSP. Par exemple, alignement de 3 points, parallélisme de 2 droites, *etc.*
singulier : se dit d'un GCSP ou CSP qui comporte une singularité ; contraire de indépendant.
sommet-contrainte : sommet du graphe variables-équations ou objets-contraintes qui représente une contrainte.
sommet-objet : sommet du graphe objets-contraintes qui représente un objet.
sommet-variable : sommet du graphe variables-équations qui représente une variable.
sous-$k$dense : se dit d'un graphe pondéré dont la somme des poids des sommets, moins la somme des poids des arêtes est supérieure à 0.
sous-contraint : CSP possédant un ensemble indénombrable de solutions.
sous-es_rigide : propriété des GCSP admettant plus de DDR DDL et dont tout sous-GCSP admet au moins DDR DDL.
sous-rigide : propriété des GCSP dont les solutions admettent (ou d'une solution d'un GCSP qui admet) les déplacements et des déformations.
sous-s_contraint : CSP constitué de moins d'équations que de variables.
sous-s_rigide : propriété des GCSP admettant plus de $\frac{d(d+1)}{2}$ DDL en dimension d et dont tout sous-GCSP en admet au moins autant.
squelette : structure d'un système à barres.
sur-$k$dense : se dit d'un graphe pondéré dont la somme des poids des sommets, moins la somme des poids des arêtes est inférieure à 0.
surcharge : capacité supplémentaire provenant de la source introduite dans le réseau objets-contraintes pour la détection de sous-GCSP bien- ou sur-rigides.
sur-contraint : CSP ne possédant aucune solution.
sur-es_rigide : propriété des GCSP dont au moins un des sous-GCSP admet moins de DDR DDL.

sur-rigide : propriété des GCSP dont les solutions n'admettent pas (ou d'une solution d'un GCSP qui n'admet pas) les déplacements.
sur-s_contraint : CSP constitué de plus d'équations que de variables.
sur-s_rigide : propriété des GCSP dont au moins un sous-GCSP admet moins de $\frac{d(d+1)}{2}$ DDL en dimension d.
système à barres : GCSP constitué exclusivement de points et de distances.
systèmes équivalents : systèmes d'équations ayant le même ensemble de solutions.

trivial : se dit d'un GCSP contenant au plus d objets en dimension d.

unification des repères locaux : opération finale de la phase d'assemblage des méthodes de rigidification récursive qui vise à ré-unifier l'ensemble des repères locaux utilisés durant la résolution.

voisinnage : notion de graphe ; deux sommets sont voisins s'ils sont liés par une arête.

Liste des définitions et théorèmes

Liste des figures

Liste des tableaux

Liste des algorithmes

Bibliographie

[Ahuja *et al.*, 1993] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows*. Prentice-Hall, 1993.

[Ait-Aoudia *et al.*, 1993] S. Ait-Aoudia, R. Jegou, and D. Michelucci. Reduction of constraint systems. In *Compugraphic*, Alvor, Portugal, 1993.

[Angeles, 1982] J. Angeles. *Spatial Kinematic Chains*. Springer-Verlag, Berlin, 1982.

[Benhamou *et al.*, 1994] F. Benhamou, D. McAllester, and P. Van Hentenryck. CLP(intervals) revisited. In *International Symposium on Logic Programming, ILPS'94*, pages 124–138, 1994.

[Bliek *et al.*, 1998] C. Bliek, B. Neveu, and G. Trombettoni. Using graph decomposition for solving continuous CSPs. In *cp98 : proceedings of the Fourth International Conference on Principles and Practice of Constraint Programming*, volume 1520 of *LNCS*, pages 102–116, 1998.

[Bliek, 1998] C. Bliek. Generalizing dynamic and partial order backtracking. In *Fifteenth National Conference on Artificial Intelligence, AAAI98*, 1998.

[Bondyfalat, 2000] D. Bondyfalat. *Interaction entre symbolique et numérique : application à la vision artificielle*. Thèse de doctorat, Université de Nice - Sophia Antipolis, 2000.

[Borning and Freeman-Benson, 1995] A. Borning and B. Freeman-Benson. The OTI constraint solver : A constraint library for constructive interactive graphical user interfaces. In *First International Conference on Principles And Practice of Constraint Programming, CP'95*, volume 976 of *LNCS*, pages 624–628, 1995.

[Borning, 1979] A. Borning. *ThingLab : A Constraint-Oriented Simulation Laboratory*. PhD thesis, Stanford University, 1979.

[Bouma *et al.*, 1995] W. Bouma, I. Fudos, C.M. Hoffmann, J. Cai, and R. Paige. Geometric constraint solver. *Computer Aided Design*, 27(6) :487–501, 1995.

[Buchberger, 1985] B. Buchberger. An algebraic method in ideal theory. In *Multidimensional System Theory*, pages 184–232. Reidel Publishing Co., 1985.

[Chou, 1988] S.C. Chou. *Mechanical Theorem Proving*. Reidel Publishing Co., 1988.

[Connelly, 2002] R. Connelly. Communication personnelle sur les configurations génériques, 2002.

[Cormen *et al.*, 1990] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, Massachusetts, 1990.

[Crapo, 1996] H. Crapo. *On the Generic Rigidity of Plane Frameworks Structures in the Plane*. Advances in Applied Mathematics. , 1996.

[Davis, 1987] E. Davis. Constraint propagation with interval labels. *AI*, 32(3) :281–331, 1987.

[Dechter, 1990] R. Dechter. Enhancement schemes for constraint processing : Backjumping, learning, and cutset decomposition. *AI*, 41(3) :273–312, 1990.

[Delobel, 2000] F. Delobel. *Résolution de systèmes de contraintes réelles non linéaires*. Thèse de doctorat, Université de Nice Sophia Antipolis, 2000.

[Dennis and Schnabel, 1983] J.E. Dennis and R.B. Schnabel. *Numerical Methods For Unconstrained Optimization And Nonlinear Equations*. Prentice-Hall, 1983.

[Dufourd *et al.*, 1998] J.-F. Dufourd, P. Mathis, and P. Schreck. Geometric construction by assembling solved subfigures. *Artificial Intelligence*, 99(1) :73–119, 1998.

[Dulmage and Mendelsohn, 1958] A.L. Dulmage and N.S. Mendelsohn. Covering of bipartite graphs. *Canad. J. Math.*, 10 :517–534, 1958.

[Durand, 1998] C. Durand. *Symbolic and Numerical Techniques For Constraint Solving*. PhD thesis, Purdue University, 1998.

[Elkadi and Mourrain, 1998] M. Elkadi and B. Mourrain. Some applications of bezoutians in effective algebraic geometry. Rapport de Recherche 3572, INRIA, 1998.

[Emiris and Canny, 1995] I.Z. Emiris and J.F. Canny. A general approach to removing degeneracies. *SIAM J. Computing*, 24(3) :650–664, 1995.

[Ericson and Yap, 1988] L.W. Ericson and C.K. Yap. The design of linetool, a geometric editor. In *Proceedings of The Computational Geometry Conference*, volume 333 of *LNCS*, pages 83–92, 1988.

[Essert-Villard, 2001] C. Essert-Villard. *Sélection dans l'espace des solutions engendrées par un plan de construction géométrique*. Thèse de doctorat, Université Louis Pasteur, Strasbourg, 2001.

[Faltings, 1994] B. Faltings. Arc consistency for continuous variables. In *Artificial Intelligence 65(2)*, 1994.

[Ford and Fulkerson, 1962] L.R. Ford and D.R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[Fudos and Hoffmann, 1996] I. Fudos and C.M. Hoffmann. Correctness proof of a geometric constraint solver. *International Journal of Computational Geometry and Applications*, 6 :405–420, 1996.

[Fudos and Hoffmann, 1997] I. Fudos and C.M. Hoffmann. A graph-constructive approach to solving systems of geometric constraints. *ACM Transactions on Graphics*, 16(2) :179–216, 1997.

[Gangnet and Rosenberg, 1992] M. Gangnet and B. Rosenberg. Constraint programming and graph algorithms. In *Second International Symposium on Artificial Intelligence and Mathematics*, 1992.

[Gelfand, 1994] I.M. Gelfand. *Discriminants, Resultants and Multidimensional Determinants*. Birkhauser, 1994.

[Gluck, 1975] H. Gluck. *Almost all simply connected closed surfaces are rigid, in Geometric Topology*, volume 438 of *Lecture Notes In Mathematics*, pages 225–239. , 1975.

[Gondran and Minoux, 1984] M. Gondran and M. Minoux. *Graphs And Algorithms*. Wiley Interscience Publication, 1984.

[Graver, 2002] J. Graver. *Counting on Frameworks : Mathematics to Aid the Design of Rigid Structures*. Number 25 in Dolciani Mathematical Expositions. Mathematical Association of America, 2002.

[Grübler, 1917] Grübler. *Getriebelehre*. Springer, Berlin, 1917.

[Hansen, 1992] E. Hansen. *Global Optimization Using Interval Analysis*. Marcel Dekker, 1992.

[Hendrickson, 1992] B. Hendrickson. Conditions for unique realizations. *SIAM j Computing*, 21(1) :65–84, 1992.

[Henneberg, 1911] L. Henneberg. *Die graphische Statik der starren Systeme*. , Leipzig, 1911.

[Heydon and Nelson, 1994] A. Heydon and G. Nelson. The juno-2 constraint-based drawing editor. Technical Report 131a, Systems Research Center, Digital, 1994.

[Hoffmann and Chiang, 2001] C.M. Hoffmann and C.-S. Chiang. Variable radius circles of cluster merging in geometric constraints, part i&ii. *Journal of Computer Aided Design*, 33, 2001.

[Hoffmann and Joan-Arinyo, 1997] C.M. Hoffmann and R. Joan-Arinyo. Symbolic constraints in constructive geometric constraint solving. *JOURNAL of Symbolic Computation*, 23 :287–299, 1997.

[Hoffmann *et al.*, 1997] C.M. Hoffmann, A. Lomonosov, and M. Sitharam. Finding solvable subsets of constraint graphs. In *Principles and Practice of Constraint Programming CP'97*, pages 463–477, 1997.

[Hoffmann *et al.*, 1998] C.M. Hoffmann, A. Lomonosov, and M. Sitharam. Geometric constraint decomposition. In Beat Bruderlin and Dieter Roller, editors, *Geometric Constraint Solving and Applications*, pages 170–195. Springer, 1998.

[Hoffmann *et al.*, 2000] C.M. Hoffmann, A. Lomonosov, and M. Sitharam. Decomposition plans for geometric constraint systems. In *Proc. J. Symbolic Computation 2000*, 2000.

[Hsu and Brüderlin, 1997] C.-Y. Hsu and B. Brüderlin. A degree-of-freedom graph approach. In *Geometric Modeling : Theory And Practice*, pages 132–155. , 1997.

[ILO, 1997] ILOG, 2 Av. Galliéni, F-94253 Gentilly. ILOG SOLVER *Version 4.0, Users' Reference Manual*, 1997.

[Imai, 1986] M. Imai. A double-tree structured multicomputer system and its application to combinatorial problems. In *Trans. IEICE*, volume E69,9, pages 1002–1010, 1986.

[Jermann and Trombettoni, 2002] C. Jermann and G. Trombettoni. Complexité de la détection de rigidité dans des systèmes de contraintes géométriques. In *8ièmes Journées Nationales de la Résolution Pratique de Problèmes NP-Complets, JNPC'02*, pages 143–154, 2002.

[Jermann *et al.*, 2000a] C. Jermann, G. Trombettoni, B. Neveu, and M. Rueher. A constraint programming approach for solving rigid geometric systems. In *Principles and Practice of Constraint Programming, CP 2000*, volume 1894 of *LNCS*, pages 233–248, 2000.

[Jermann *et al.*, 2000b] C. Jermann, G. Trombettoni, B. Neveu, and M. Rueher. Résolution par contraintes de systèmes géométriques rigides. In *6ièmes Journées Nationales de la Résolution Pratique de Problèmes NP-Complets, JNPC'00*, pages 119–133, 2000.

[Jermann *et al.*, 2002a] C. Jermann, B. Neveu, and G. Trombettoni. De la rigidité structurelle. In *Groupe de Travail en Modélisation Géométrique, GTMG 2000*, pages 106–115, 2002.

[Jermann *et al.*, 2002b] C. Jermann, B. Neveu, and G. Trombettoni. On the structural rigidity for gcsps. In *Proceedings of the 4^th International Workshop on Automated Deduction in Geometry*, 2002.

[Joan-Arinyo and Soto-Riera, 1999] R. Joan-Arinyo and A. Soto-Riera. Combining constructive and equational constraint solving techniques. *ACM Transactions on Graphics*, 18(3) :35–55, 1999.

[Joan-Arinyo *et al.*, 2002] R. Joan-Arinyo, A. Soto-Riera, S. Vila-Marta, and J. Vilaplana-Pasto. Revisiting decomposition analysis of geometric constraint graphs. In *Proceedings of the 7th ACM Symposium on Solid Modeling and Applications*, pages 105–115, 2002.

[Kearfott, 1987] R.B. Kearfott. Some test of generalized bisection. *ACM Trans. on Mathematical Software*, 13(3) :197–220, 1987.

[König, 1916] D. König. Über graphen und ihre anwendung auf determinantentheorie und mengenlehre. In *Math Ann 77*, pages 453–465, 1916.

[Kondo, 1992] K. Kondo. Algebraic methods for manipulation of dimensional relationships in geometric models. *Computer Aided Design*, 24(3) :141–147, 1992.

[Kramer, 1992] G. Kramer. *Solving Geometric Constraint Systems*. MIT Press, 1992.

[Lahaye, 1934] E. Lahaye. Une méthode de résolution d'une catégorie d'équations transcendantes. *Compte-rendu des Séances de L'Académie des Sciences*, 198 :1840–1842, 1934.

[Laman, 1970] G. Laman. On graphs and rigidity of plane skeletal structures. *J. Eng. Math.*, 4 :331–340, 1970.

[Lamure and Michelucci, 1998] H. Lamure and D. Michelucci. Qualitative study of geometric constraints. In B. Bruderlin and D. Roller, editors, *Geometric Constraint Solving and Applications*, pages 234–258. Springer, 1998.

[Latham and Middleditch, 1996] R.S. Latham and A.E. Middleditch. Connectivity analysis : A tool for processing geometric constraints. *Computer Aided Design*, 28(11) :917–928, 1996.

[Lazard, 1981] D. Lazard. Résolution des systèmes d'équations algébriques. *Theoretical Computer Science*, 15 :77–110, 1981.

[Lesage *et al.*, 2000] D. Lesage, J.-C. Léon, and P. Serré. A declarative approach to a 2d variational modeler. In *proceedings of the 3rd International Conference on Integrated Design and Manufacturing in Mechanical Engineering (IDMME)*, 2000.

[Lhomme *et al.*, 1998] O. Lhomme, O. Gotlieb, and M. Rueher. Dynamic optimization of interval narrowing algorithms. *Journal Of Logic Programming*, 37(1-3) :165–183, 1998.

[Lhomme, 1993] O. Lhomme. Consistency techniques for numeric CSPs. In *IJCAI'93 : Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 232–238, 1993.

[Light *et al.*, 1981] R. Light, V. Lin, and D.C. Gossard. Variational geometry in cad. *Computer Graphics*, 15(3) :171–177, 1981.

[Lovasz and Plummer, 1986] L. Lovasz and M.D. Plummer. *Matching Theory*. Number 29 in Annals of discrete mathematics. North-Holland, 1986.

[Lovasz and Yemini, 1982] L. Lovasz and Y. Yemini. On generic rigidity in the plane. *SIAM J. Alg. Discrete Methods*, 3 :91–98, 1982.

[Macaulay, 1902] F.S. Macaulay. Some formulae in elimination. In *Proc. of London Mathematical Society*, volume 33, pages 1–27, 1902.

[Mackworth, 1977] A. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1) :99–118, 1977.

[Mathis, 1997] P. Mathis. *Constructions géométriques sous contraintes en modélisation à base topologique*. Thèse de doctorat, Université Louis Pasteur, 1997.

[Michelucci and Lamure, 1994] D. Michelucci and H. Lamure. Résolution de contraintes géométriques par homotopie. In *Actes de AFIG 1994*, 1994.

[Montanari, 1974] U. Montanari. Networks of constraints : Fundamentals properties and applications to picture processing. *Information Science*, 7(2) :95–132, 1974.

[Moore and Qi, 1982] R.E. Moore and L. Qi. A successive interval test for nonlinear systems. *SIAM J. on Numerical Analysis*, 19(4) :845–850, 1982.

[Moore, 1966] R.E. Moore. *Interval Analysis*. Prentice-Hall, 1966.

[Morgan, 1983] A.P. Morgan. *Solving Polynomial Systems Using Continuation For Scientific And Engineering Problems*. Prentice-Hall, 1983.

[Mourrain, 1999] B. Mourrain. An introduction to linear algebra methods for solving polynomial equations. In *Proceedings of HERCMA'98*, pages 179–200, 1999.

[Neumaier, 1990] A. Neumaier. *Interval Methods For Systems Of Equations*. Cambridge University Press, 1990.

[Oung *et al.*, 2001] J.J. Oung, M. Sitharam, B. Moro, and A. Arbree. Frontier : fully enabling geometric constraints for feature based modeling and assembly. In *Proceedings of ACM Solid Modeling symposium, 2001*, 2001.

[Owen, 1991] J. Owen. Algebraic solution for geometry from dimensional constraints. In *Proceedings of the* 1^st *ACM Symposium on Solid Modeling and CAD/CAM Applications*, pages 397–407. ACM Press, 1991.

[Papadimitriou, 1994] C.H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[Peitgen and Richter, 1986] H.O. Peitgen and P.H. Richter. *The Beauty Of Fractals, Images Of Complex Dynamical Systems.* Springer Verlag, 1986.

[Ritt, 1950] J.F. Ritt. *Differential Algebra.* American Mathematical Society, 1950.

[Rouillier *et al.*, 2000] F. Rouillier, M.F. Roy, and M. Safey. Finding at least one point on each connected component of a real algebraic set defined by a single equation. *Journal Of Complexity*, 16 :716–750, 2000.

[Rouillier, 1999] F. Rouillier. Solving zero-dimensional systems through the rational univariate representation. *Journal of Applicable Algebra in Engineering, Communication and Computing*, 9(5) :433–461, 1999.

[Ruiz and M, 1996] O. E. Ruiz and Ferreira P. M. Algebraic geometry and group theory in geometric constraint satisfaction for computer-aided design and assembly planning. *IIE Transactions on Design and Manufacturing*, 28 :281–294, 1996.

[Sanella, 1994] M. Sanella. *Constraint Satisfaction and Debugging for Interactive User Interfaces.* PhD thesis, University of Washington, Seattle, 1994.

[Serrano, 1987] D. Serrano. *Constraint Management in Conceptual Design.* PhD thesis, MIT, 1987.

[Sitharam, 2000] M. Sitharam. Personal communication on the minimal dense algorithm. , 2000.

[Sosnov and Macé, 2002] A. Sosnov and P. Macé. Rapid algebraic resolution of 3d geometric constraints and control of their consistency. In *Proceedings of the* 4^th *International Workshop on Automated Deduction in Geometry*, 2002.

[Sugihara, 1986] K. Sugihara. *Machine Interpretation of Line Drawing.* MIT Press, 1986.

[Sunde, 1986] G. Sunde. Specification of shapes by dimensions and other geometric constraints. In *IFIP WG 5.2 Geometric Modeling*, 1986.

[Sutherland, 1963] I. Sutherland. *Sketchpad : A Man-Machine Graphical Communication System.* PhD thesis, Department of Electrical Engineering, MIT, 1963.

[Trombettoni, 1997] G. Trombettoni. *Algorithmes de maintien de solution par propagation locales pour les systèmes de contraintes.* Thèse de doctorat, UNSA, NICE, 1997.

[Trombettoni, 1998] G. Trombettoni. A polynomial time local propagation algorithm for general dataflow constraint problems. In *cp98 : proceedings of the Fourth International Conference on Principles and Practice of Constraint Programming*, volume 1520 of *LNCS*, pages 432–446, 1998.

[Vander Zanden, 1996] B. Vander Zanden. An incremental algorithm for satisfying hierarchies of multi-way, dataflow constraints. *ACM Transactions on Programming Languages and Systems*, 18(1) :30–72, 1996.

[Verroust *et al.*, 1992] A. Verroust, F. Schonek, and D. Roller. Rule oriented method for parametrized computer aided design. *Computer Aided Design*, 24(6) :531–540, 1992.

[Waltz, 1972] D.L. Waltz. Generating semantic descriptions from drawings of scences with shadows. Technical Report ai-tr-271, Cambridge, 1972.

[Wampler *et al.*, 1990] C.W. Wampler, A.P. Morgan, and A.J. Sommese. Numerical continuation methods for solving polynomial systems arising in kinematics. *ASME Journal On Design*, 112 :59–68, 1990.

[Wang, 2002] D. Wang. Geother 1.1 : Handling and proving geometric theorems automatically. In *Proceedings of the* 4^th *International Workshop on Automated Deduction in Geometry*, 2002.

[Whiteley, 1987] W. Whiteley. Applications of the geometry of rigid structures. In Henry Crapo, editor, *Computer Aided Geometric Reasoning*, pages 219–254. INRIA, 1987.

[Whiteley, 1998] W. Whiteley. Rigidity and scene analysis. In J. E. Goodman and J. O'Rourke, editors, *Handbook for discrete and computational geometry*, pages 893–916. CRC Press LLC, 1998.

[Whiteley, 2002] W. Whiteley. Communication personnelle au sujet de la théorie de la rigidité, 2002.

[Wu, 1986] W. Wu. Basic principles of mechanical theorem proving in elementary geometries. *J. Automated Reasoning*, 2 :221–254, 1986.