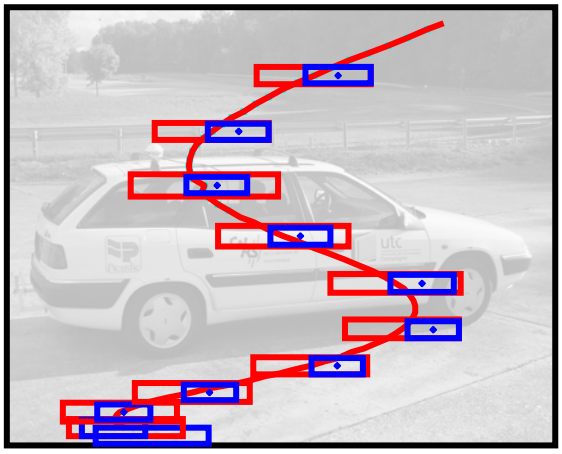
Localisation garantie d'automobiles. Contribution aux techniques de satisfaction de contraintes sur les intervalles

El Hadji Amadou Gning

Localisation garantie d'automobiles. Contribution aux techniques de satisfaction de contraintes sur les intervalles



Thèse

présentée et soutenue publiquement le 17 Mars 2006 pour l'obtention du grade de

Docteur de l'Université de Technologie de Compiègne

Spécialité : Technologies de l'information et des systèmes

Par ***GNING El Hadji Amadou***

Thèse soutenue devant le jury composé de

- Mr Jaulin L.	Professeur, ENSIETA,	*Rapporteur*
- Mme. Lesecq S.	Maître de conférences, UJF,	*Rapporteur*
- Mme. Cherfaoui V.	Maître de conférences, UTC,	*Examinateur*
- Mr. Peyret F.	Directeur de recherche, LCPC Nantes,	*Examinateur*
- Mr. Bonnifait P.	Maître de conférences, UTC,	*Directeur de thèse*
- Mr Meizel D.	Professeur, ENSIL,	*Directeur de thèse*

A mes très chers parents pour m’avoir inculqué le goût du savoir.

A ma famille proche en France pour avoir égayé mon quotidien.

A ma très tendre femme…la recherche n’a pas débouché que sur les trouvailles rapportées dans ce manuscrit…

Remerciements

Le présent travail a été réalisé au sein du laboratoire "**Heu**ristique et **dia**gnostic des **sy**stèmes **c**omplexes" dirigé par Monsieur Rogélio Lozano. En son nom, je remercie tout le personnel du laboratoire pour l'accueil chaleureux et pédagogique auquel j'ai eu droit.

Le travail s'est inscrit dans le thème **S**ystème de **P**erception et de **C**ommande du laboratoire sous la direction de Messieurs Philippe BONNIFAIT et Dominique MEIZEL. Je remercie Mr BONNIFAIT pour sa convivialité, son énergie pour la recherche et surtout les relations plus que chaleureuses que j'ai eu l'honneur d'entretenir avec lui. Je remercie également Mr MEIZEL pour son soutien et ses remarques constructives tout au long de cette thèse.

Mes remerciements vont à Mr JAULIN Luc, Professeur à l'ENSIETA de Brest, ainsi qu'à Mme LESECQ Suzanne Maître de conférences à l'UJF Grenoble pour avoir accepté de juger ce travail en tant que rapporteurs de mon jury.

Je remercie fortement mes amis et collègues du laboratoire et tout particulièrement ceux du groupe véhicule avancé. Au cours de cette thèse, le travail d'équipe mené pour divers projets m'a permis d'engranger beaucoup de connaissances et m'a apporté des affinités sincères et égayantes avec les membres du laboratoire.

Pour finir, je remercie toute ma famille vers qui j'ai toujours eu réponse à mes sollicitations de conseils et prières.

PUBLICATIONS DE L'AUTEUR

Publications dans des revues nationales et internationales avec comité de lecture :

[1] Gning A., Bonnifait P. "Fusion de données redondantes avec une technique ensembliste atteignant la consistance globale", Revue e-sta : science et technologie de l'automatique *SEE*. 2004.

[2] Gning A., Bonnifait P. "Constraints Propagation Techniques on Intervals for a Guaranteed Localization using Redundant Data", revue Automatica. A paraître.

Publications dans des conférences internationales avec actes et comité de lecture :

[3] Gning A., Bonnifait P. "Guaranteed Dynamic Localisation using Constraints Propagation Techniques on Real Intervals". IEEE International Conference on Robotics and Automation, ICRA04, New Orleans, Avril 2004.

[4] Gning A. Bonnifait P. "Fusion de données redondantes avec une technique ensembliste atteignant la consistance globale". Conférence internationale francophone d'automatique, CIFA2004, Tunisie, Novembre 2004.

[5] Gning A. Bonnifait P. "Guaranteed Dynamic Localization using Constraints Propagation Techniques on Real Intervals". IEEE International Conference on Industrial Technology (ICIT 04), Hammamet, Tunisie, Décembre 2004.

[6] Gning A., Bonnifait P. " Dynamic Vehicle Localization using Constraints Propagation Techniques on Intervals, A comparison with Kalman Filtering". IEEE International Conference on Robotics and Automation, ICRA05, Barcelone, Avril 2005.

ATELIERS INTERNATIONAUX

[7] F. Abdallah, Ph. Bonnifait, A. Gning. “A Particle Filter on Interval Data for Mobile Localization”. Worhshop IntCP 2005 *Interval Analysis and Constraint Propagation for Applications* de la conférence Eleventh International Conference on Principles and Practice of Constraint Programming (CP 2005), Sitges (Barcelona) Spain, October 1 - 5, 2005.

BREVETS

Brevet CNRS/UTC intitulé “Procédé de synchronisation des données notamment distribuées, prenant en compte les imprécisions et les dérives des horloges”. Inventeurs : Ph. Bonnifait, P. Crubillé, O. Bezet, V. Cherfaoui, G. Brissot, A. Gning et M. Shawky. Numéro de dossier CNRS 63487, Mars 04.

Index des notations

- x : un vecteur de variable de IR^n,
- $\tilde{x}$: une instanciation de x,
- $[x]$: la notation d'un pavé de IR^n,
- $(f_i)_{i=1\cdots p}$: des fonctions réelles permettant d écrire les contraintes sur x $(f_i(x)=0)_{i=1\cdots p}$,
- V_{f_i} : un ensemble des variables concernant une contrainte f_i,
- R_{f_i} : résolveur d'une contrainte f_i,
- $F=(f_1,\cdots f_p)$: système de contraintes constitué à partir des f_i,
- $\mathcal{H}$ = (F(x)=*0*, $x \in [x]$) : CSP formulé à partir de x, F et $[x]$
- $I=(1,\cdots,n)$: ensemble des indices de 1à n,
- $J=(j_1,\cdots,j_m)$: un sous ensemble de I de cardinal m,
- $K = I \setminus J$: le complémentaire de J dans I,
- x_J : le sous-vecteur de x associé à J,
- $[x_J]$: le sous-pavé de $[x]$ associé à J,
- $\prod_{f_i,j}([x]) \triangleq \prod_{f_i,x_J}([x])$: projection des solutions de la contrainte f_i sur $[x_j]$ (la dimension j du pavé $[x_J]$),
- $\prod_{G,j}([x]) \triangleq \prod_{G,x_J}([x])$: projection des solutions du système de contrainte G sur $[x_j]$,
- $\prod_J([x]) \triangleq \prod_{F,J}([x]) \triangleq \prod_{F,x_J}([x]) \triangleq \prod_{x_J}([x])$: projection des solutions du système de contrainte F sur $[x_j]$,

- $D_F(\tilde{x}_J)$: Etant donnés un sous-vecteur $\tilde{x}_J$ et le CSP $\mathcal{H}$, $D_F(\tilde{x}_J)$ est le domaine consistance qui lui est associé,
- $D_F(A_J)$: Etant donnés un ensemble de sous-vecteur A_J et le CSP $\mathcal{H}$, $D_F(A_J)$ est le domaine de consistance qui lui est associé

Introduction Générale

1 PROLEMATIQUE ET CONTEXTE DU TRAVAIL

Le sujet traité dans cette thèse rentre dans le cadre des méthodes à erreurs bornées appliquées à un problème de localisation dynamique, à contraintes temps-réel, d'un véhicule sur la route moyennant des capteurs embarqués. Plus précisément, on cherche à déterminer la pose (position et attitude) d'un véhicule dans un référentiel absolu terrestre, de façon garantie et en temps-réel, en utilisant les techniques de satisfaction de contraintes sur les intervalles.

La particularité des approches à erreurs bornées est qu'à la sortie de leurs algorithmes, elles restituent un ensemble (ellipses, intervalles, zonotopes…) contenant toutes les solutions possibles des problèmes traités. De telles approches soulèvent de nombreuses questions du fait notamment que l'homme a plutôt l'habitude de raisonner avec des points qu'avec des intervalles ou autres ensembles. Une des questions primordiales est de savoir quels intérêts effectifs peut-on tirer de l'information sur l'ensemble estimé ?

Une première réponse à cette question est qu'un intervalle peut être regardé, d'une certaine façon, comme un point si sa taille est petite vis à vis du problème à résoudre. C'est le cas, par exemple, lorsque l'on considère la distance d'un satellite à un récepteur avec les erreurs commises (distance intervalle) en lieu et place d'une valeur ponctuelle estimée de cette distance. Dans ce cas précis, comparée à la grandeur de la distance effectivement mesurée, l'erreur commise est négligeable et un intervalle peut être vu comme ponctuel. A l'opposé, si la taille des intervalles n'est pas négligeable, l'ambiguïté sur la vraie valeur solution est très importante et cette information peut s'avérer intéressante pour traduire un comportement inattendu du système étudié. Ce dernier point est très intéressant vis-à-vis d'applications visant un degré élevé d'intégrité.

Une seconde réponse est que, pour certaines applications, des approches à erreurs bornées peuvent sembler être mieux adaptées. On peut donner l'exemple de robots secouristes ou de robots collaborant (Farinelli *et al.*, 2004). En effet, il peut être plus pertinent qu'un robot en mission transmette l'information « ma position se trouve de façon garantie dans le pavé P

suivant » plutôt que de transmettre l'information « ma position calculée est le point M avec une erreur de $\pm\eta$ sans certitude… ».

Etre capable de développer des algorithmes fournissant des informations garanties sur les solutions possibles est donc un champ de recherche crucial pour de nombreuses applications. Dans cet optique, les approches à erreurs bornées commencent à faire leur apparition dans l'automatique et la robotique et à être appliquées à de nombreux problèmes. Cette thèse s'inscrit dans cette logique d'appliquer ces outils ensemblistes à un problème concret de localisation dynamique.

Ce travail s'est effectué au sein du Laboratoire mixte Heudiasyc[1] du CNRS et de l'UTC, plus précisément au sein de l'équipe « véhicule avancé » du thème SPC[2]. Il fait suite à un premier travail effectué dans le cadre d'un stage de DEA [Gning 2002] où une étude de la méthode de propagation de contraintes a été effectuée sur plusieurs simulations. Cette thèse est donc la suite logique d'un travail appliqué à des données réelles.

Sur le plan expérimental, la collecte de données a été rendue possible grâce au projet national ARCOS[3] s'inscrivant dans le cadre des actions fédératives du PREDIT[4]. La thèse a été elle-même financée sur le volet localisation de ce projet matérialisé par le thème 3. Ce projet s'est étalé sur 4 ans, de 2001 à 2004, pour un investissement recherche et développement d'environ 13M d'euros, en regroupant une soixantaine de laboratoires publics et entreprises privées parmi lesquels les principaux constructeurs et équipementiers français. Le maître mot de ce projet était l'amélioration de la sécurité routière en adoptant une approche coopérative des véhicules et infrastructures afin d'assister, d'informer ou encore d'alerter le conducteur. Ainsi, le projet s'est articulé autour de la réalisation de quatre "*fonctions*" jugées très pertinentes avec des cibles, pour chacune d'entre elles, allant du réalisable en l'état actuelle des technologies, à des concepts totalement novateurs mais déjà très envisageables. Ces fonctions étaient respectivement de « gérer les interdistances entre véhicules », « prévenir les collisions sur obstacles », « prévenir les sorties de routes », et enfin, « alerter les véhicules en amont ». Pour réaliser ces fonctions, les recherches se sont

1 Heudiasyc : **Heu**ristique et **dia**gnostic des **sy**stèmes **c**omplexes

2 SPC : Systèmes de Perception et de Commande

3 ARCOS : Action de recherche pour une conduite sécurisée

4 PREDIT : Programme national de recherche et d'innovation dans les transports terrestres

réparties sur onze thèmes intégrant des domaines aussi variés que les sciences de l'ingénieur, les sciences humaines et les sciences sociales. Le thème 3, en particulier, avait pour objectifs, d'une part, de localiser le véhicule de façon précise, fiable et intègre dans un référentiel global terrestre, et d'autre part, de se projeter dans une base de données cartographiques afin de pouvoir tirer profit de l'environnement du véhicule.

2 Plan du mémoire

Ce mémoire est composé de cinq chapitres et d'une annexe. Pour résumer, les deux premiers chapitres présentent l'intérêt de la localisation, les problèmes rencontrés dans la recherche et la réalisation de processus de localisation ainsi que les solutions existantes. Le troisième chapitre présente la méthode garantie de localisation, basée sur les techniques de satisfaction de contraintes sur les intervalles, étudiée dans cette thèse ainsi que les résultats obtenus sur des données réelles. Les deux derniers chapitres introduisent de nouveaux outils en vue d'une amélioration des techniques existantes de satisfaction de contraintes. Plus dans les détails, on peut décrire ces différents chapitres de la façon suivante :

- le chapitre 1 présente l'intérêt des systèmes de localisation dans l'évolution actuelle des progrès techniques ainsi que les différents concepts de localisation. On y présente le cheminement qui nous a mené au choix des capteurs embarqués. On y présente également le principe des capteurs qui seront utilisés pour la suite de la thèse.
- le chapitre 2 traite de la fusion de données. On y présente l'approche de résolution d'un problème de fusion de données avec un modèle d'évolution et les erreurs qui découlent de cette représentation. Puis, on présente les approches les plus classiquement utilisées pour résoudre les problèmes de fusion de données en faisant la différence entre les approches « ponctuelles » et les approches « ensemblistes ».
- dans le chapitre 3, on présente l'analyse par intervalles avec ses définitions classiques ainsi que la formulation sous forme de problème de satisfaction de contraintes. Puis, la méthode de propagation de contraintes qui est utilisée pour la fusion de donnée garantie est présentée. Ensuite, une méthodologie générale pour exprimer un problème de fusion de donnée sous forme d'un problème de satisfaction de contraintes est présentée. Enfin, cette méthodologie est appliquée au problème de localisation sur des données réelles.

- le chapitre 4 fait suite au chapitre précédent et est une perspective qui répond à des insuffisances et limitations constatées lors de l'étude faite sur le cas réel. Ainsi, on y présente une nouvelle notion que nous avons appelé « domaine de consistance » autour de laquelle toute une théorie est développée. Cette notion répond à un besoin de recherche d'algorithmes nouveaux de satisfaction de contraintes plus efficaces en termes de temps de calcul et d'optimalité des ensembles retournés (optimalité dans le sens de la précision des ensembles à estimer). On présente dans ce chapitre des exemples prometteurs où on dégage un cheminement permettant de valider la théorie.
- le chapitre 5, toujours pour améliorer les techniques de satisfaction de contraintes, présente une nouvelle méthode de vectorisation qui répond à un besoin pertinent d'organiser les contraintes afin, soit d'accélérer les temps de calcul, soit simplement de pouvoir les connaître à l'avance. Deux exemples sont présentés, un académique et un sur un problème réel d'estimation de la cinématique d'un véhicule, en se basant sur la technique de vectorisation présentée.

CHAPITRE 1. LOCALISATION D'UN VEHICULE ROUTIER

Introduction

Le développement continu et à grande vitesse des technologies de l'information et de la communication ainsi que la grande diversité des champs d'applications constituent des motivations grandissantes pour les systèmes de localisation terrestre. Maîtriser les déplacements des personnes et de leurs biens, apporter diverses fonctionnalités aux systèmes d'aide à la conduite, localiser pour mieux secourir etc., telles sont des raisons de pousser les progrès des systèmes de localisation encore plus loin.

Le terme "localisation" signifie, dans le domaine de la robotique mobile, la connaissance de la pose du mobile dans un système de référence explicite. La pose est définie par le couple position et attitude de l'objet. La position est souvent un ensemble de coordonnées géographiques ou cartésiennes. L'attitude quant à elle, désigne les différents angles par rapport à un référentiel ou par rapport à des amers caractérisant le système.

Dans cette thèse, on s'intéresse exclusivement à la localisation d'un véhicule se déplaçant sur route, dans un référentiel global terrestre, et avec une contrainte forte de temps réel. Dans ce chapitre, on s'évertue d'abord à montrer toute l'importance des systèmes de localisation dans la société d'aujourd'hui et de demain. Ensuite, on présente les concepts élémentaires de localisation à l'estime et localisation absolue, et enfin on décrit les capteurs utilisés dans le cadre de cette thèse, et qui sont représentatifs des capteurs que l'on trouve couramment dans les voitures actuelles

1 IMPORTANCE DES SYSTEMES DE LOCALISATION

1.1 Systèmes de localisation couplés à la cartographie

L'idée de se localiser est très cohérente avec l'idée de connaître des informations pertinentes sur son environnement. Pour pouvoir se localiser et se servir de cette localisation pour connaître son environnement, on a recours à une base de données appelée dans ce cas "cartographie".

La construction d'une cartographie de la route s'articule en deux principales phases : d'abord choisir une représentation géométrique de la route et la sauvegarder, puis choisir des informations pertinentes autour de la route et les sauvegarder. Ces informations pertinentes sont appelées attributs et elles peuvent être, par exemple, la vitesse limite autorisée, la courbure de la route, le nombre de voies ou encore la largeur de la route. La plupart des cartographies existantes représentent la route en deux dimensions. La route est alors subdivisée en un ensemble de tronçons qui sont assimilés à des segments de droites. Pour stocker ces droites, seules les extrémités suffisent. La route est alors matérialisée par 2 points et d'autres informations telles que la pente, le dévers ou la courbure peuvent être conservées comme attributs.

Les systèmes de localisation couplés à la cartographie ouvrent un domaine d'application gigantesque, dans la mesure où les capacités de stockage d'attributs dans les cartes vont crescendo au cours du temps. On verra dans les prochains paragraphes des exemples d'application de la localisation déjà existants et toute une panoplie d'applications envisagées. Les exemples d'apport de la localisation sont classés selon les secteurs de sécurité, de confort et de gestion des infrastructures.

1.2 Apport en matière de sécurité

Au terme général de "sécurité" qui peut renvoyer à la santé des individus ou encore à la pérennité de leurs biens, le terme "localisation" est presque toujours associé. En effet, pour assurer la sécurité d'un individu ou d'un objet, un début logique et très intuitif de solution consiste à localiser l'individu ou l'objet, ceci pour pouvoir surveiller, avertir ou encore

intervenir dans des cas limites. Pour bien aborder l'apport de la localisation, on distinguera deux besoins en sécurité : la sécurité physique des individus et la sécurité de leurs biens.

1.2.1 Sécurité des individus

Le domaine des dispositifs d'assistance à la conduite constitue un enjeu économique de vente, mais aussi une nécessaire adaptation par rapport aux moyens technologiques en constante évolution. Cette adaptation est cependant constamment remise en question de par sa propension à grignoter toujours plus de tâches au conducteur. Cette propension fait craindre à long terme une diminution des capacités ou réflexes du conducteur et de son "plaisir de conduire". Ceci d'autant plus que d'un pays à l'autre, les mêmes systèmes d'aide à la conduite ne donnent ni les mêmes résultats ni les mêmes impressions et appréciations des utilisateurs. S'y ajoutant, les statistiques prouvent que les politiques de répressions et de conscientisation des habitués de la route s'avèrent bien plus efficaces. Sur un autre volet, les aspects juridiques face à des systèmes qui exerceraient une emprise sur certaines actions des véhicules constituent un facteur retardant la mise en place de systèmes innovants ou parfois même bloquant toute possibilité de commercialisation. Cependant, la présence actuelle d'un certain nombre de systèmes d'aide à la conduite qui sont bien acceptés et bien ancrés dans les habitudes des conducteurs constitue en soi, un motif suffisant pour poursuivre la recherche de systèmes innovants.

Les systèmes d'aide à la conduite utilisant un système de localisation, sont à classer dans la famille des systèmes ADAS (Advanced Driver Assistance Systems). On peut citer par exemple des systèmes de type "ISA" (adaptation intelligente de vitesse). Ces systèmes sont très utiles pour la maîtrise de la vitesse (une combinaison d'un attribut de la cartographie "vitesse limite" et de la localisation permet de comparer la vitesse du véhicule avec la vitesse autorisée dans le tronçon où on se situe le véhicule). On peut aussi citer les systèmes de type "*Lane Keeping*" (prévention de sortie de voie) : grâce à une reconnaissance de la limite entre les voies (par des capteurs optiques), le véhicule ajuste automatiquement sa position latérale pour ne pas sortir de sa voie.

Pour avoir une idée des perspectives des systèmes ADAS, on peut se référer au projet national français ARCOS (PREDIT) où plusieurs applications ADAS ont été développées ou étudiées. Dans ce projet à spectre large et ambitieux, il s'agissait au final de développer 4 fonctions : gérer les inter-distances, prévenir les collisions, prévenir les sorties de route,

alerter les véhicules en amont. La localisation est apparue comme un point central pouvant apporter un plus considérable à chacune de ces fonctions. Parmi les idées pertinentes et indissociables d'un système de localisation, on peut citer, le système d'Alerte de Vitesse excessive en approche de Virage (SAVV). Le principe est de stocker dans une base de données routières des carrefours dangereux ou des virages serrés. Lorsque le véhicule, localisé avec une précision suffisante, les aborde à une vitesse inadaptée, un système d'avertissement se déclenche. Une autre perspective intéressante est l'éclairage intelligent. Le faisceau lumineux pourra s'adapter à la vitesse du véhicule et à la géométrie de la route. Ainsi, en ligne droite, le faisceau sera étroit et long pour porter loin sans éblouir les conducteurs venant en sens inverse. Dans un virage ou un carrefour, par contre, le faisceau sera large et devra suivre la courbure de la route, permettant ainsi au conducteur d'apercevoir plus rapidement d'éventuels obstacles ou dangers.

Dans un autre volet très prometteur, les systèmes de communication inter-véhicules couplés à la localisation ouvrent d'intéressantes perspectives dans la mesure où l'information recueillie par un véhicule peut être échangée avec d'autres véhicules. Les véhicules pourraient par exemple avertir du trafic au-delà d'un virage, ou bien, deux voitures approchant d'un carrefour sans visibilité pourraient s'avertir mutuellement, ou encore les véhicules pourraient émettre des alertes radio en cas de carambolage.

Une autre classe d'application très concernée par les systèmes de localisation est celle des "*robots exploratoires*" et des "*robots de secours*". Le besoin est de pouvoir envoyer des robots à la place des hommes dans des endroits qui sont à risques ou qui sont inaccessibles. Les applications peuvent être de rechercher des survivants dans des décombres ou de fournir les premiers soins ou secours (de l'eau, de la nourriture, etc.)

On voit donc que les applications de la localisation en matière de sécurité sont nombreuses et certaines avancées dans leur réalisation. Dans la section suivante, on s'intéresse à un autre volet de la sécurité, notamment celle des biens "physiques" et "matériels" des individus.

1.2.2 Sécurité des biens des individus

Au cours de ces dernières années, l'idée de systèmes "espions" a fait son apparition. Malgré des obstacles juridiques, ces systèmes sont de mieux en mieux acceptés et demandés

par les populations pour sécuriser leurs biens. Ces systèmes sont majoritairement basés sur la transmission d'information de localisation. Leurs objectifs sont en grande partie de surveiller, et le cas échéant d'intervenir. Un exemple d'un système qui connaît un succès grandissant et très demandé, est le bracelet électronique équipé d'un système de localisation par GPS, dont sont munis des prisonniers en liberté conditionnelle, ou encore des enfants mineurs qui pourront être de cette façon repérés par leurs parents, mais aussi des animaux domestiques, etc. On peut citer dans le même ordre d'idée les systèmes comme le TrimTrac commercialisé par Trimble. L'idée est de dissimuler dans son véhicule un système de localisation pour pouvoir situer rapidement ce dernier en cas de vol. On retrouve à travers ces concepts, une philosophie bien ancrée dans la mentalité de l'homme, selon laquelle, savoir où se trouve à tout moment notre bien ou l'être à protéger et/ou à surveiller, apporte un sentiment de sécurité.

Un autre exemple plus frappant, dans un pays comme les Etats-Unis où les lois le permettent, est le cas des véhicules "appâts". Le besoin est de diminuer les vols de véhicules, et le principe est de placer un véhicule appât dans des zones sensibles avec un système de localisation espion. Dans cet exemple, on se situe en amont de la sécurité, dans le sens où l'objectif est soit de dissuader, soit d'éradiquer le danger.

1.3 Apport en matière de confort et d'économie

Les applications en matière de confort étant justifiées essentiellement par des enjeux économiques et concurrentiels, l'évolution et la commercialisation des systèmes dotés de localisation sont très rapides. Les applications sont multiples et touchent à des domaines variés. Des systèmes existants couplés avec la cartographie permettent par exemple, de se faire guider vers une destination ou prendre l'itinéraire le plus rapide ou le plus économique. Ils permettent aussi déjà à des agriculteurs de gérer leurs terres agricoles en repérant les terres en jachère, ou en répartissant de manière réfléchie leur stock d'engrais. Enfin ils permettent à l'automobiliste de trouver des lieux utiles (restaurants, magasins, hôpitaux, etc.) ou aident le touriste à repérer les lieux répertoriés.

1.4 Apport en matière de gestion des infrastructures

Les applications des systèmes de localisation pour la gestion des infrastructures sont très prometteuses parce qu'elles n'impliquent pas directement l'avis de l'utilisateur grand public mais plutôt celui des organismes dédiés. Les applications existantes sont nombreuses : pour une société gérant une flotte de taxis par exemple, une cartographie avec les positions de ses véhicules permet d'optimiser le temps d'attente des clients et de minimiser pour ces derniers, les prix à payer consécutifs à la mobilisation d'un taxi. Dans le même ordre d'idée, les voitures de pompiers, les voitures de police, les ambulances, les camions des transporteurs routiers peuvent être gérés plus efficacement en terme de rapidité, nombre, efficacité, sécurité et coût.

La navigation des trains est également utilisatrice de systèmes de localisation. Les systèmes avancés de voies ferrées sont très développés. Les trains peuvent être précisément positionnés en temps réel grâce à des systèmes de localisation. Leurs positions, par l'intermédiaire d'ondes radio sont, par la suite, relayées au centre de contrôle. Les intérêts sont nombreux. La centrale de contrôle peut vérifier le cheminement des trains sur un écran et informer les voyageurs de tout retard. Elle peut également effectuer plus rapidement les interventions de maintenance ou contrôler les correspondances entre cars et trains, ou encore gérer des secours en cas d'immobilisation ou d'accident d'un train.

Les applications de la localisation sont donc primordiales aussi bien pour des raisons sécuritaires que pour gérer des infrastructures. Dans la suite de cette section, on propose de décliner les différents concepts de localisation liés à un système de référence donné.

2 DIFFERENTS CONCEPTS DE LOCALISATION

Le terme de localisation pour ce qui concerne un véhicule, peut prendre trois sens différents qui se déclinent en *localisation absolue, localisation à l'estime* et *localisation hybride* [Fargeon et al, 93]. Cette distinction est liée à la nature des capteurs et méthodes utilisés. On distingue deux familles de capteurs embarqués à bord d'un véhicule :

- les capteurs extéroceptifs : ils fournissent des mesures qui se rapportent à l'environnement du véhicule (distance par rapport à des amers, position par rapport à un référentiel terrestre, angles),

- les capteurs proprioceptifs : ils fournissent des mesures qui renseignent sur le comportement interne du véhicule (distance parcourue par les roues, angle au volant, variation de l'angle de roulis).

Comme on le précisera par la suite, les capteurs extéroceptifs se rapportent à la localisation absolue, les capteurs proprioceptifs à la localisation à l'estime et la localisation hybride à l'utilisation conjointe des deux types de capteurs.

2.1 Localisation absolue

La localisation absolue nécessite l'utilisation d'au moins un capteur extéroceptif dont les mesures télémétriques et/ou goniométriques sur l'environnement du véhicule, permettent de déterminer la pose de l'engin dans un repère global lié à l'environnement. La principale qualité de ce type de localisation tient à sa précision qui ne dérive pas au cours du temps. Celle-ci dépend des capteurs, de la configuration et du nombre d'amers présents dans l'environnement du véhicule. Cependant et en règle générale, les coûts de calcul liés au temps de traitement des informations des capteurs extéroceptifs posent des problèmes de latence des mesures (décalage entre l'instant précis où la mesure a été effectuée et l'instant où la mesure est effectivement fournie par le capteur) ou encore de cadence (fréquence des données reçus pas assez élevées) et de disponibilité (exemple du GPS qui souffre d'absence de signaux satellitaires dans certains milieux).

Comme exemple de capteur extéroceptif, on peut citer les compas et les inclinomètres qui fournissent des angles par rapport au nord magnétique et à la gravité terrestre. On peut aussi citer les capteurs à ultrasons qui sont très utilisés pour des problèmes de robotique d'intérieur (milieu bien adapté à la faible portée des ultrasons en plus de l'absence de perturbations liées au vent). On leur associe souvent une carte puisqu'en se positionnant sur la carte, on peut en déduire la pose de l'objet à localiser.

On peut citer également les systèmes utilisant des balises dont les positions sont parfaitement connues. On y trouve de nombreux systèmes optiques (caméras, lasers, systèmes infrarouges.) ou hyperfréquence (radars, GPS, LORAN[1] …). Signalons que le capteur le plus utilisé pour se localiser sur le globe terrestre est sans aucun doute le système GPS, qui sera le

[1] LORAN : LOng RAnge Navigation : c'est un système de radio-navigation avec des amères terrestre fixes. Il est antérieur aux systèmes de positionnement par satellites.

capteur extéroceptif considéré pour nos expérimentions dans la suite de ce manuscrit. Le principe de ce dernier sera détaillé dans la section 3.3.

2.2 Localisation à l'estime

La localisation à l'estime, encore appelée localisation relative, se base uniquement sur des capteurs proprioceptifs. Sous condition de la connaissance de la configuration de départ, le principe de fonctionnement repose sur une intégration des mesures des capteurs proprioceptifs pour estimer, au cours du temps, la pose du véhicule. Il s'en suit également une intégration des erreurs de mesure et de modèle, ce qui entraîne un accroissement de ces dernières de façon continue au cours du temps et du déplacement. Le principal apport de ces capteurs est la possibilité d'obtenir des positions à une fréquence élevée (jusqu'à 100Hz) contrairement à des capteurs extéroceptifs (qui dépassent rarement des fréquences de plus de 10Hz). Cette caractéristique intéressante explique en partie l'utilisation systématique de ces capteurs en robotique mobile, ainsi que dans de nombreux secteurs tels que les domaines aéronautique, spatial, maritime et militaire (avions, fusées, missiles, sous-marins, etc.).

Parmi les capteurs les plus utilisés, on peut citer les codeurs optiques d'odométrie disposés sur les roues du véhicule, les radars (ou sonar) à effet Doppler, les accéléromètres et les gyromètres. Dans les sections 3.2.1 et 3.2.2, il sera donné une description plus détaillée des capteurs utilisés dans les expérimentations. Il s'agit respectivement des odomètres et du gyromètre.

2.3 Localisation hybride

L'idée de la localisation hybride est d'utiliser conjointement la localisation à l'estime et la localisation absolue. On parle alors de recalage dynamique. Cette idée se justifie par la complémentarité évidente des capteurs proprioceptifs et des capteurs extéroceptifs. En effet, la localisation à l'estime est presque toujours capable de fournir des mesures à une fréquence très élevée avec cependant une dérive des estimations et une nécessaire initialisation de son point de départ. A contrario, la localisation absolue ne présente pas de problèmes de dérive et d'initialisation, mais souffre de problème de disponibilité, de latence et de fréquence souvent insuffisante pour certaines applications. L'hybridation peut se faire par le biais de nombreuses

techniques de fusion de données[2] pour pouvoir obtenir de meilleures estimations en termes de précision, disponibilité ou intégrité.

Dans la section suivante, on propose de justifier le choix des capteurs utilisés au cours de cette thèse et ensuite de présenter succinctement ces différents capteurs.

3 DESCRIPTION DES CAPTEURS UTILISES

3.1 Capteurs présents dans les véhicules modernes

Pour étudier des systèmes de localisation réalistes, crédibles et pertinents, il faut prendre en compte la réalité des capteurs présents dans les véhicules. En effet, pour des raisons de coût économique, ces systèmes ne peuvent se baser sur toute la panoplie de capteurs existants. Une stratégie cohérente avec la réalité du marché pour réfléchir à de nouveaux systèmes consiste à se baser le plus possible sur les capteurs déjà présents sur les véhicules.

A ce jour, divers capteurs sont souvent présents dans les véhicules. On peut citer parmi ceux ci : des odomètres, des gyromètres, des accéléromètres, des télémètres radar et des récepteurs GPS. Ils sont utiles aux fonctions :

- ABS (Anti-lock Braking System) qui dispose des informations d'odométrie (déplacement élémentaire et rotation élémentaire). L'objectif de l'ABS est d'empêcher les roues de glisser exagérément lors du freinage. Il permet ainsi au véhicule de rester stable, même lors de freinages brusques et à pleine puissance : la direction du véhicule reste en outre pleinement opérationnelle. Il utilise des capteurs sur les quatre roues pour mesurer en permanence la vitesse de rotation de celles-ci. Grâce à cette information, l'électronique de commande agit sur le dispositif de freinage dès que la vitesse de rotation d'une roue diminue brusquement.
- ESP (Electronic Stability Program) qui dispose d'un gyromètre fournissant la vitesse de lacet (rotation autour d'un axe vertical), d'un capteur d'accélération transversale et d'un capteur de l'angle au volant. L'objectif de l'ESP est de mieux contrôler la trajectoire du véhicule : il détecte le moindre début de dérapage du véhicule et agit sur les freins ou sur le moteur pour le stabiliser.

[2] Les techniques de fusion de données seront détaillées dans le chapitre 2

- Assistance au parking : les voitures utilisent les mesures de distance de plusieurs télémètres fixes (souvent à ultra sons) pour détecter des obstacles en arrière du véhicule. Ainsi le conducteur est assisté lors de manœuvre de recul ou pour se garer.
- Systèmes de navigation : les véhicules disposent d'un système de cartographie, d'un récepteur GPS éventuellement hybridé avec d'autres capteurs proprioceptifs. On peut ainsi se localiser sur la carte routière et en utiliser les informations. Les applications sont essentiellement orientées vers les fonctions de guidage et de détermination de chemins optimums.

Un autre atout peut être rajouté à la présence déjà acquise de ces capteurs dans certains véhicules modernes : il est possible d'obtenir les informations fournies par ces capteurs grâce à un bus CAN.

Suite à cette analyse sur les capteurs dans les véhicules d'aujourd'hui, l'ensemble de capteurs {gyromètre, odomètre, GPS} a été considéré comme un bon candidat pour aborder le problème de localisation. Les méthodologies développées dans cette thèse s'appliqueront donc principalement sur la fusion de ces 3 capteurs. Nous proposons, dans la partie qui suit, de les décrire.

3.2 Capteurs proprioceptifs

3.2.1 Les capteurs du système ABS en tant qu'odomètre

Un odomètre est un capteur de distance parcourue. Ce capteur est très utilisé en robotique mobile grâce à sa mise en œuvre facile et surtout grâce à son coût raisonnable. Les odomètres utilisant les capteurs du système ABS permettent d'obtenir des signaux dès que la voiture se déplace à une vitesse suffisamment élevée. Ces signaux permettent de quantifier les déplacements curvilignes du véhicule à partir des rotations de ses roues.

Le système ABS peut être composé de roues dentées et de capteurs *proxymètres* ou *solénoïdes* (voir Figure 1.1). Le principe est le suivant : la voiture est équipée de roues dentées, fixées derrière les disques de freins. Le capteur placé à proximité de chaque roue dentée délivre un signal numérique avec une fréquence proportionnelle à la rotation de la roue. En effet, ce capteur fonctionne par induction électromagnétique et un enchaînement métal / non-métal entraîne l'apparition d'un signal de forme carrée.

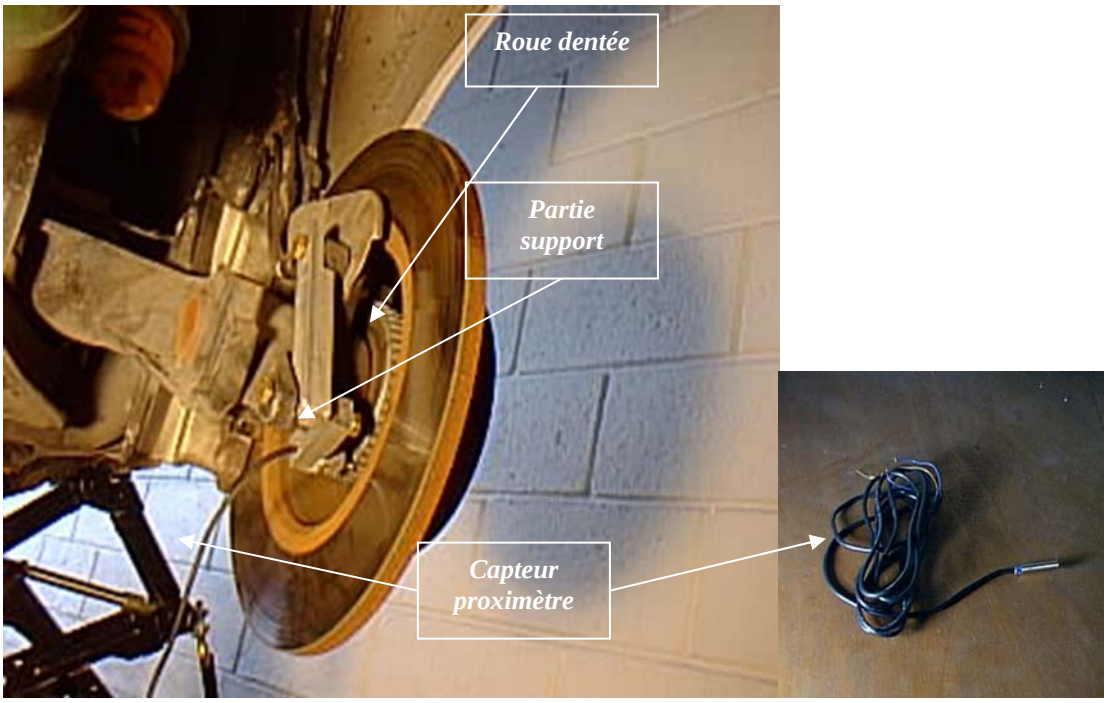


Figure 1.1 : Composantes d'un capteur odométrique (Citroën Xantia)

Pour exploiter chaque capteur ABS, on lui associe un compteur. Un compteur de "n" bits est un circuit logique qui compte de 0 jusqu'à 2^n-1. Il possède au moins deux entrées : une entrée de remise à zéro et une entrée pour le signal numérique. En sortie, il fournit un nombre de tops du signal compté depuis sa dernière remise à zéro. Ainsi, un compteur associé à un capteur ABS fournit des données correspondant à une distance parcourue par la roue. La relation entre le nombre de tops récupéré et la distance parcourue par la roue est, en première approche, obtenue grâce à la connaissance du périmètre de la roue et de sa résolution (nombre de dents de la roue). Un étalonnage précis est souvent nécessaire pour une meilleure précision.

3.2.2 Le gyromètre à fibre optique

Un gyromètre permet de mesurer des vitesses de rotation angulaire. En robotique mobile, ces types de capteur constituent le complément idéal des capteurs odométriques. En effet, ces derniers utilisés seuls induisent souvent une dérive angulaire importante qui affecte de manière beaucoup plus visible l'erreur sur la position réelle que la dérive sur les distances parcourues seules (Boreinstein et al 96).

Les gyromètres peuvent être fabriqués sur la base de plusieurs principes technologiques. Tout d'abord sont apparus des gyromètres mécaniques vibrants. Les précisions ont ensuite été nettement améliorées grâce à des technologies basées sur le laser ou l'optique.

Les capteurs à fibre optique sont issus de la rencontre de deux technologies : les fibres optiques utilisées en télécommunication et l'optoélectronique. L'une et l'autre ont connues un développement spectaculaire ces dernières décennies : amélioration des performances, diminution des coûts et miniaturisation. Par définition, un capteur à fibre optique est un dispositif dans lequel l'information est créée dans le chemin optique par réaction de la lumière à la grandeur à mesurer avant d'être acheminée vers le récepteur optique par 1'intermédiaire d'une ou plusieurs fibres optiques. Pour le gyromètre à fibre optique, le principe physique utilisé est l'effet Sagnac [Sagnac 13]. La lumière émise par une diode laser est divisée en deux et introduite dans deux fibres optiques enroulées sur elles-mêmes. Lorsque le gyromètre est en rotation, d'après l'effet Sagnac, il va exister une différence entre les deux temps de parcours. Cette différence est alors observée par interférométrie et on peut en déduire une valeur de la rotation observée.

Malheureusement, les gyromètres très précis n'ont généralement pas un prix abordable pour une utilisation "grand public". Cependant, ceux à fibre optique réputés avoir une grande précision, se présentent, au vu de la chute progressive de leur prix, comme une solution fort plausible et acceptable pour les systèmes de localisation.

Le gyromètre utilisé au cours de cette thèse est un gyromètre à fibre optique KVH RD 2030. Il délivre un signal numérique à une fréquence de 100 Hz sur une liaison série RS232 avec une bande passante de 50Hz et une pleine echelle de 30°/s.

3.3 Capteurs extéroceptifs : le GPS

3.3.1 Généralités

De nos jours le système GPS s'est placé au centre des systèmes de localisation absolue. Les raisons sont multiples : d'abord un coût abordable qui a permis d'en faire une production à grande échelle, ensuite une couverture mondiale qui a permis d'étendre le potentiel d'utilisateurs. Nous proposons dans cette partie de faire un bref historique du système GPS, puis, de faire une description de ses constituants.

3.3.1.1 Origine du GPS et description du système

L'idée d'un système GPS vient du besoin de l'homme de connaître sa position sur le globe terrestre. Ainsi, l'homme a dû de façon naturelle lever ses yeux vers le ciel, ayant tout de suite eu l'intuition de devoir chercher un référentiel à l'aide duquel il va pouvoir s'orienter. C'est ainsi que pendant des décennies l'homme a pu s'orienter et parcourir la terre en se référant aux étoiles mais en étant tributaire des conditions climatiques et de la présence du soleil. Il a fallu attendre le XX^{ième} siècle pour que l'homme puisse enfin avoir l'ambition et les moyens techniques de créer un système de référentiel sous son contrôle et lui permettant de s'affranchir des moyens empiriques à sa disposition. Ainsi, sont apparus les systèmes dits GNSS (Global Navigation Satellite System) dont le principe est de se localiser moyennant un système de satellites dans un référentiel terrestre. Les systèmes GNSS remplissent également un rôle important de datation précise, qui profite à tout utilisateur muni d'un récepteur.

Il existe actuellement deux systèmes candidats pour remplir les fonctions de positionnement et de datation du GNSS : le système militaire américain NAV.S.TAR-G.P.S (NAVigation System Time And Ranging - Global Positioning System) et le système russe GLO.NA.S.S (GLObale NAvigation Satellite System). Ces systèmes ont un potentiel très important car ils allient pour la première fois des qualités exceptionnelles : couverture quasi mondiale et quasi permanente, précision de localisation, nombre d'utilisateurs illimité et coût très abordable du service. Le GPS (Global Positioning System) s'est cependant imposé face à son rival compte tenu des difficultés économiques du système GLO.NA.S.S russe.

Le GPS a été imaginé par le département de la défense des Etats-Unis qui contrôle et finance entièrement le projet à ce jour. Le système actuel est composé de la seconde génération de satellites, appelés Block II. Il était initialement destiné exclusivement à l'armée des Etats-Unis. Cependant, le problème de la vulgarisation du service s'est très rapidement posé. En effet, de nombreuses applications civiles pouvaient vouloir recourir à ce système tout en sachant que le GPS est un outil militaire décisif. On s'est donc vite orienté vers le compromis suivant:

- un premier service aux possibilités dégradées décliné sous l'acronyme de SPS (Standard Position Service). Ce service a été mis à disposition des utilisateurs civils. Une dégradation volontaire avait été mise en place par les gérants du système GPS et limitait la précision de positionnement à 100 m (jusqu'en mai 2000).

- un service de grande précision décliné sous l'acronyme de PPS (Precise Position Service). Ce service est réservé aux militaires américains et à certains utilisateurs autorisés par le département de la défense. Les récepteurs sont équipés d'algorithmes de décryptage spécifique.

A ce jour, le gouvernement américain a mis fin à la dégradation volontaire du signal GPS. Cependant, le succès scientifique et opérationnel du GPS et l'utilisation irréversible des satellites pour les opérations de positionnement et de navigation rend désormais la dépendance vis à vis du gouvernement des États-Unis imprudente et inacceptable pour les autres pôles militaro-économiques de la planète. Pour échapper à cette dépendance, des instances internationales ainsi que certains états évoquent l'idée d'un nouveau GNSS (Global Navigation Satellite Système), à l'exemple de l'union Européenne qui prévoit de déployer à l'horizon de 2010 un système indépendant du GPS à travers Galiléo.

3.3.1.2 Constitution du système GPS

1. Partie spatiale

   La partie spatiale est constituée d'un ensemble de 24 satellites répartis sur 6 plans orbitaux (voir Figure 1.2). Ces satellites évoluent à une altitude d'environ 20000 km et mettent 12 heures pour effectuer une rotation autour de la terre. Chaque satellite possède un oscillateur qui fournit une fréquence fondamentale de 10,23 MHz calibrée sur des horloges atomiques. L'émetteur génère deux ondes (L1 et L2) de fréquences respectives 1575,42 MHz et 1227,60 MHz. Il transmet régulièrement des signaux horaires, la description de l'orbite suivie (éphéméride) et diverses autres informations.

Figure 1.2 : Constellation des satellites du système GPS.

2. Partie de contrôle

   La partie de contrôle qui permet de piloter et surveiller le système, est composée de cinq stations au sol situées à travers le monde (voir Figure 1.3). Elles enregistrent tous les signaux émis par les satellites, calculent leurs éphémérides (orbites, paramètres d'horloge, codes, etc...), et transmettent des données aux satellites. La station principale se trouve à la base aérienne à Schriever.

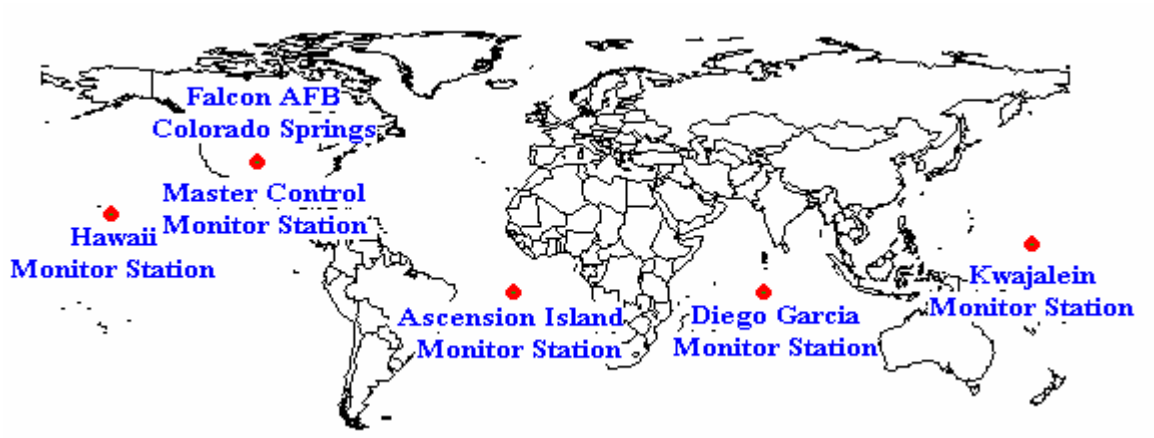


Figure 1.3 : stations de contrôle au sol.

3. Partie utilisateur

   La partie utilisateur comprend l'ensemble des récepteurs civils et militaires qui ne font que recevoir les informations des satellites. Pour déterminer la position, la vitesse et le temps, au moins quatre signaux satellites de la constellation GPS sont nécessaires.

3.3.2 Fonctionnement

3.3.2.1 Structure du signal GPS et observations faites par un récepteur

Une bonne compréhension des notions "d'onde porteuse" et de "modulation" est nécessaire pour décrire le signal GPS. Moduler une onde signifie faire varier une de ses grandeurs caractéristiques (amplitude, fréquence ou phase) en fonction des variations d'un signal. Pour retrouver les signaux après transmission, il faut procéder à une démodulation. Une onde est dite porteuse si une de ses grandeurs caractéristiques est destinée à suivre les variations d'un signal dans une modulation. L'intérêt de moduler un signal se trouve dans le codage d'informations et aussi dans la possibilité de manipuler des signaux hautes fréquences (ce qui permet d'avoir des antennes de récepteur de dimensions raisonnables).

Comme énoncé précédemment, chaque satellite GPS transmet donc ses informations dans la bande L sur deux fréquences L1 et L2. La fréquence d'émission des signaux des satellites GPS est très stable grâce à des horloges atomiques embarquées. L'onde porteuse L1 est *modulée* par 2 codes C/A (Coarse/Acquisition code) et P (Precise code) et par un signal contenant les messages de navigation. Ces messages sont constitués des données d'éphémérides des satellites, des données de correction de propagation ainsi que du temps local des satellites. Ces messages permettent au récepteur d'identifier le satellite et de connaître précisément sa position dans le référentiel de temps absolu GPS. La seconde onde porteuse L2, quant à elle, est modulée uniquement par le code P ainsi que par les mêmes messages de navigation dans L1. L2 est utile lorsque la fréquence L1 est perturbée. Elle permet aussi de corriger les retards de transmission liés à l'ionosphère.

Trois types d'observations GPS peuvent être faites par les récepteurs : des mesures de pseudo distance, des mesures de phase des ondes porteuses et des mesures de Doppler.

- ***Les mesures de pseudo-distance***

  La mesure de pseudo-distance à un facteur près (en l'occurrence de la célérité de la lumière) est une mesure de temps de propagation d'un signal. Au préalable, dans une étape d'initialisation, le récepteur procède à une phase dite "d'accrochage". Cette étape consiste à rechercher les satellites. Pour cela, le récepteur génère localement une réplique du code des satellites puis effectue une inter-corrélation avec les codes reçus. Ainsi le récepteur est capable d'identifier les satellites visibles. Une fois cette étape d'accrochage faite, le récepteur poursuit le signal de chaque GPS visible grâce à des boucles à verrouillage de code et de phase.

  Pour mesurer le temps $t_{S_i,R}$ de propagation d'un satellite S_i au récepteur R, le récepteur se sert du code répliqué et du code reçu correspondant. En effet, le décalage temporel à appliquer sur le code généré afin de le faire coïncider avec le code reçu correspond au temps de propagation du signal $t_{S_i,R}$. Ce temps de propagation est perturbé par plusieurs erreurs. L'erreur la plus importante est due à la désynchronisation des horloges des satellites et du récepteur. En effet, les codes répliqués par le récepteur sont issus de l'horloge du récepteur tandis que le code généré par un satellite est cadencé par une horloge atomique embarquée très précise (de l'ordre de la nanoseconde). Le décalage

entre le code généré et celui reçu est donc le temps de propagation du signal plus une variable représentant le décalage entre les horloges.

En outre, la distance calculée est faussée à cause de la vitesse de propagation changeante du signal essentiellement à cause de la traversée de l'ionosphère (partie haute de l'atmosphère terrestre). La distance ainsi obtenue avec les erreurs temporelles et les erreurs de décalage est appelée mesure de pseudo-distance entre le satellite et le récepteur.

- ***Les mesures de phase***

  Ces mesures sont plus compliquées à mettre en œuvre que celles faites sur le code et elles ne peuvent pas être exploitées par les récepteurs "bas de gamme". Ces mesures permettent d'obtenir une meilleure précision que celles effectuées sur le code. L'idée consiste à déterminer le déphasage entre le satellite et le récepteur sur les ondes porteuses L1 et L2. On compare ainsi la phase de l'onde reçue du satellite avec la phase de sa réplique générée par le récepteur. La justification de cette mesure vient du fait que la pseudo-distance est proportionnelle à la différence de phase entre le récepteur et le satellite. Ce calcul de pseudo-distance par les mesures de phase constitue donc un autre moyen de localiser le récepteur.

  La difficulté liée à cette mesure provient des ambiguïtés pour déterminer la différence de phase. En effet, seule la partie décimale (entre 0 et 2π) de la différence de phase est accessible par des techniques de corrélation entre les deux phases. Ainsi, une ambiguïté est due à l'impossibilité de faire la distinction entre les différents cycles de la porteuse. La partie entière de la différence de phase est appelée ambiguïté entière.

  Il existe de nombreuses méthodes pour traiter le problème posé par les ambiguïtés sur la mesure de phase. En règle général, une fois les ambiguïtés déterminées à un instant t_0 donné, le récepteur utilise une stratégie lui permettant de garder cette information pour les mesures futures. En effet, chaque boucle à verrouillage de phase dispose d'un compteur de tours qui comptabilise le nombre de cycles entiers de la phase écoulés depuis l'instant t_0. Tant qu'il n'y a pas d'interruption dans la réception du signal, le récepteur mesure à chaque instant $t_k > t_0$ la partie décimale de la différence de phase et le nombre de cycles entiers de la phase écoulée depuis la mesure à l'instant t_0. Les interruptions provoquent des "sauts de cycles" et sont principalement causées par des masquages des satellites (à cause de bâtiments, montagnes, arbres, etc) ou par des interférences. La résolution de l'ambiguïté initiale peut se faire, par exemple, en initialisant le GPS en un point dont la

position est connue avec une très grande précision. Sinon, différents algorithmes permettent cette initialisation avec des temps de calcul de l'ordre de la minute sous de bonnes conditions de configurations et de visibilité satellitaires (ceci fonctionne même en mode déplacement).

- ***Les mesures de Doppler***

  Cette mesure est une différence entre la fréquence de la porteuse du signal généré par le récepteur et la fréquence reçue. Cette dernière est affectée par le mouvement relatif du satellite par rapport au récepteur. Ce phénomène physique est connu sous le nom d'effet Doppler. Suivant ce principe, dans un référentiel de temps commun, le rapport des fréquences est fonction des positions et des vitesses relatives du satellite et de l'utilisateur. La mesure ainsi obtenue permet de déterminer la vitesse instantanée du récepteur mobile. Elle peut aussi servir à détecter et corriger les sauts de cycles de la mesure de phase.

3.3.2.2 Calcul de position

La position donnée par le GPS est celle de l'antenne du récepteur de l'utilisateur. Le référentiel GPS est le WGS84 (World Geodetic System 1984). Le principe de la détermination de cette position est la *trilatération* : on mesure la distance entre l'utilisateur et un certain nombre de satellites dont on connaît les positions puis, de façon imagée, on obtient des sphères centrées sur des satellites dont l'intersection permet de reconstituer les trois informations de position de l'utilisateur (latitude, longitude et hauteur ellipsoïdale). La distance par rapport à chaque satellite est obtenue grâce à sa relation de proportionnalité avec le temps de parcours du signal émis par chaque satellite. En effet, connaissant t_{S_i}, l'heure exacte d'émission d'un message par un satellite S_i et l'heure exacte, t_R, d'arrivée de ce message sur le récepteur R, on peut en déduire le temps de propagation du signal. En multipliant ce temps par la célérité de la lumière, on obtient la distance $\Delta_{S_i,R}$ du satellite t_{S_i} au récepteur R.

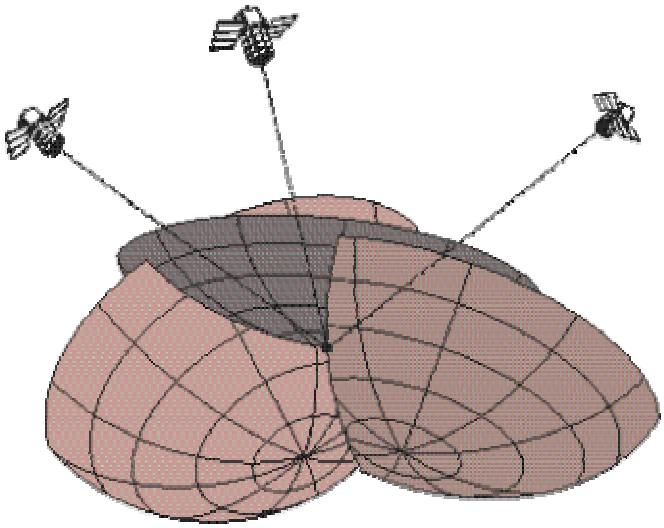


Figure 1.4 : illustration du principe de la trilatération avec un récepteur parfaitement à l'heure.

Dans le cas idéal, trois satellites suffisent pour déterminer la position du récepteur (voir Figure 1.4). En effet, l'intersection des trois sphères distinctes donne 2 points. L'un de ces deux points se situe loin de la surface du globe terrestre et n'est donc pas crédible. En réalité, quatre satellites au moins sont nécessaires, si on prend en compte les erreurs d'horloge. En effet, on peut remarquer que la distance $\Delta_{S_i,R}$ mesurée qui est la pseudo distance est faussée par le décalage temporel ΔT_i entre le temps GPS et celui du récepteur. Ce décalage qui peut prendre de grandes valeurs est considéré comme une quatrième variable à estimer.

En outre, pour une utilisation plus efficace du GPS, les satellites NAVSTAR reçoivent quotidiennement des paramètres de corrections diffusés par les stations. Il est donc possible à l'utilisateur de régler son horloge lors du calcul de positionnement qui fournit l'imprécision ΔT de l'horloge utilisateur (en négligeant l'erreur de l'horloge GPS).

La prise en compte de toutes les erreurs qui perturbent la propagation du signal GPS permet d'atteindre des précisions sub-métriques. Les sources d'erreurs sont vues en détail dans la prochaine section.

3.3.3 Sources d'erreurs

L'erreur moyenne sur la position d'un récepteur du système GPS est de l'ordre d'une quinzaine de mètres. Avant de parler des différentes sources d'erreurs ayant une influence sur la mesure GPS, il est important de préciser qu'un positionnement issu de mesures GPS dépend de la géométrie de la distribution des satellites et du nombre de ces derniers au moment de la

mesure. En effet, puisqu'il existe quatre inconnues principales dans le calcul GPS (la position en trois dimensions et un décalage d'horloge), il faut au moins la visibilité de quatre satellites pour déterminer une position GPS. De surcroît, une bonne configuration géométrique des satellites est nécessaire pour une bonne précision du positionnement. Ainsi, les récepteurs utilisent le critère de GDOP (Geométric Dilution Of Precision) pour évaluer la qualité de cette configuration satellitaire. Au dessus d'un certain seuil (généralement GDOP > 5), la mesure n'est pas jugée utilisable.

Les erreurs peuvent être subdivisées en erreurs de manipulation et en erreurs systématiques. Dans les erreurs de manipulations, on range les erreurs humaines qui peuvent être commises par les utilisateurs du GPS ou dans les stations de contrôle. Les erreurs systématiques sont, quant à elles, difficiles à modéliser et leur estimation permet d'obtenir de grandes précisions. On peut les diviser en erreurs matérielles, en erreurs d'orbites, de propagation, et en erreurs dues à l'environnement du récepteur. On peut détailler ces différentes erreurs de la façon suivante :

- ***Erreurs dues aux matériels***

  Les erreurs matérielles les plus signifiantes sont celles au niveau des antennes des récepteurs ou celles qui découlent de la résolution des mesures par les récepteurs. Les erreurs d'antenne sont dues à une variation d'un point appelé "centre de phase". C'est un point immatériel qui correspond au point où est capté le signal GPS. Les variations du centre de phase sont fonction de l'azimut et de l'élévation des satellites ainsi que de leur configuration. En pratique ce point est fixé pour un récepteur donné et les constructeurs calibrent les antennes et fournissent les modèles de calibration permettant de corriger les mesures.

- ***Erreurs d'orbites***

  Les satellites sont soumis à de nombreux phénomènes physiques tels que la gravitation (attraction lunaire ou solaire, perturbation des marées océaniques), l'accélération relativiste, la pression des radiations solaires, etc. Les trajectoires prévues des satellites sont donc influencées par ces phénomènes. Les messages de navigation émis par le satellite ne permettent donc de calculer l'orbite du satellite en temps réel qu'avec une certaine erreur (à 20 m près). En outre, un terme correctif lié à la rotation terrestre est pris en compte lors du calcul de positionnement GPS (en effet, la terre a tourné pendant le temps de vol du signal).

- ***Erreurs de propagation du signal***

  Plusieurs facteurs peuvent affecter la propagation du signal. On peut citer parmi ceux-ci, les erreurs non critiques liées à des retards de l'électronique des récepteurs et celle des satellites. Les erreurs qui influent le plus sur le temps de propagation, sont liées à la traversée de la troposphère (partie basse de l'atmosphère) et de l'ionosphère (partie haute de l'atmosphère). Ces deux couches de l'atmosphère, l'ionosphère plus particulièrement, rallongent le temps de parcours du signal. Ces erreurs sont d'autant plus difficiles à estimer qu'elles dépendent de l'élévation des satellites. Pour la troposphère, l'erreur dépend aussi de la température, de la pression atmosphérique et de l'humidité.

- ***Erreurs liées à l'environnement du récepteur***

  Ces erreurs sont essentiellement dues aux multi-trajets du signal ou à un masquage satellitaire. Elles dépendent de l'environnement du récepteur et ne sont donc pas modélisables ou quantifiables a priori. Pour limiter l'effet de ces multi-trajets, les antennes des récepteurs sont équipées de systèmes qui filtrent en partie les multi-trajets.

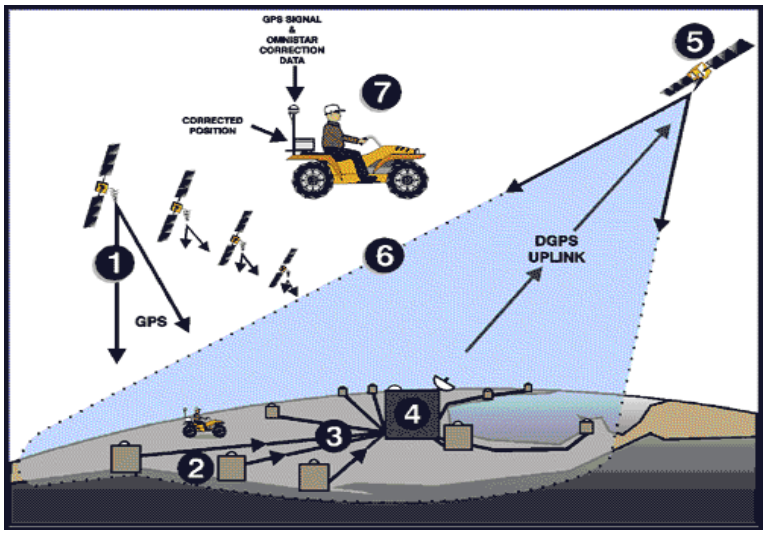
3.3.4 Le GPS différentiel

Il existe une version améliorée du système GPS appelé DGPS (GPS Différentiel) qui permet d'améliorer la précision et l'intégrité des signaux du GPS. Son principe est basé sur l'hypothèse raisonnable que deux positions mesurées à un instant donné, par deux récepteurs distants l'un de l'autre de moins d'une centaine de Km présentent la même erreur. Cette hypothèse se justifie en considérant la grande distance entre les satellites et les récepteurs qui est de l'ordre de 20000 Km. En effet, les deux récepteurs vus par un satellite sont quasiment confondus et donc les perturbations dues à la traversée de l'atmosphère sont quasi identiques. En partant de cette hypothèse, si on effectue régulièrement une mesure à un point donné (station de référence) dont on connaît parfaitement la position, il est possible de calculer toutes les erreurs et de transmettre des valeurs de correction à tous les utilisateurs dans un rayon de quelque dizaines de Km. Les récepteurs peuvent de fait sensiblement améliorer leur calcul de position. On appelle ligne de base, le vecteur constitué de deux récepteurs. Dans le positionnement différentiel, il est nécessaire de considérer au moins une ligne de base.

Deux types de fonctionnement sont possibles pour les lignes de base. Un premier fonctionnement consiste à faire des corrections en temps réel. Le second traitement consiste à traiter les informations du récepteur seulement après leur collecte, en post-traitement. Le

premier mode nécessite un matériel supplémentaire, puisqu'il faut créer une liaison entre le mobile et la station de référence, pour envoyer un message de correction vers le mobile. Cette liaison est généralement effectuée soit par radio (UHF ou VHF), soit par une liaison téléphonique (GSM ou GPRS). Ces techniques permettent d'éliminer un certain nombre d'erreurs systématiques. Cependant, la précision diminue vite en fonction de l'éloignement à la station de référence. Le positionnement obtenu n'est alors précis que dans un rayon d'une dizaine de Km autour de la station de référence. En s'éloignant de la station, on peut cependant avoir un positionnement différentiel sub-métrique grâce aux corrections émises dans un rayon plus grand (de l'ordre d'une centaine de Km).

Les modes différentiels de plus en plus répandus sont ceux de type SBAS (Satellite-Based Augmentation System). Un système d’augmentation par satellites signifie une augmentation du nombre de satellites GPS par d’autres systèmes de satellites géostationnaires améliorant ainsi la disponibilité du signal. De plus, ces satellites envoient des corrections pour améliorer considérablement la précision et l’intégrité du positionnement. Parmi ces systèmes, on peut citer le WAAS (Wide Area Augmentation System) en Amérique du Nord, le système japonais MSAS (Multifunctional Transport Satellite-based), le système chinois SNAS (Satellite Navigation Augmentation System) et le Système européen EGNOS (European Geostationary Navigation Overlay System). Il existe également des sociétés privées qui proposent des corrections par SBAS parmi lesquelles OmniSTAR que nous avons utilisé dans nos expérimentations. C’est un système de diffusion de corrections GPS différentiel en temps réel couvrant l’Europe et l’Amérique du nord. Les corrections sont issues d’un ensemble de stations de base situées à travers le monde. Il utilise ce réseau de stations de référence (ou stations de base) pour mesurer les erreurs induites dans le signal GPS. Ces données de référence sont ensuite transmises au centre de contrôle où elles sont vérifiées quant à leur intégrité et à leur fiabilité, puis envoyées par liaison ascendante à un satellite géostationnaire qui diffuse les données sur son empreinte. Le centre de contrôle utilise une solution VRS (Virtual Reference Station) pour fusionner les informations des stations de base. Le principe est de modéliser finement les différentes sources d’erreurs en s’appuyant sur les stations de base, puis d’interpoler l’influence de ces erreurs à proximité de l’utilisateur en simulant les données d’une station virtuelle.



1. *Satellites GPS*
2. *Station de référence OmniSTAR*
3. *corrections GPS expédiées au centre de commande*
4. *station centrale qui reçoit les corrections et les transmet au satellite géostationnaire*
5. *Satellite Géostationnaire*
6. *Le secteur où les corrections du satellite géostationnaire sont recommandées*
7. *Récepteur*

Figure 1.5 : fonctionnement du système OMNISTAR.

3.3.5 Points forts du GPS

La principale force du système GPS aujourd'hui vient du fait qu'il se soit imposé dans sa "guerre froide" avec le système GNSS russe (GLONASS) qui souffre essentiellement de problème d'entretien de ses satellites et donc de disponibilité. Ainsi, le GPS s'impose pour l'utilisateur comme le seul système GNSS disponible et donc le seul système qui, utilisé seul, peut fournir une position dans un repère global (WGS84), ce qui en fait un outil quasi indispensable pour des applications de localisation à la surface de la terre. Il faut ajouter à cela que sa précision s'est améliorée en permanence au cours du temps grâce à diverses techniques qui ont évolué très rapidement. Un autre avantage du GPS est le coût des récepteurs qui est constamment revu à la baisse grâce, d'une part, à sa constante évolution technologique, mais aussi, grâce à l'immensité des champs d'application et au nombre gigantesque de récepteurs.

3.3.6 Points faibles du GPS

Le principal point faible du GPS, qui est en même temps une limitation, se situe dans la disponibilité des données. En effet, la réception des signaux peut s'avérer très sporadique dans certains milieux mal adaptés (en milieu urbain, forêt, tunnel, bâtiment, etc.). Ce problème étant difficilement soluble par les récepteurs, il justifie amplement la nécessaire hybridation du GPS avec des capteurs proprioceptifs pour des applications critiques. Ainsi dans des applications critiques dans des milieux hostiles, le GPS est surtout utilisé pour recaler les dérives des estimations issues des capteurs proprioceptifs.

Dans le cadre de cette thèse, le GPS utilisé pour se localiser est un Trimble AgGPS132, qui est mono fréquence (C/A sur L1) avec des corrections différentielles satellitaires Omnistar. Sa fréquence d'utilisation standard est de 5Hz avec une précision nominale de l'ordre du mètre en mode différentiel, et une latence de l'ordre de 150 ms. Il dispose en outre du signal PPS (Pulse per second).

Conclusion

Les systèmes de localisation constituent un module important pour un très grand nombre d'applications allant de la sécurité à des besoins de confort et de gestion d'infrastructures. Pour réaliser ces tâches, de nombreuses méthodes existent et répondent déjà aux contraintes de localisation pour bon nombre d'applications. Dans ce chapitre, on a pu faire un inventaire de toute l'étendue des applications concernées par les modules de localisation. Par rapport à l'objectif de cette thèse qui est de localiser des véhicules routiers, on a pu également décrire les capteurs qui seront considérés. On a vu aussi que ces capteurs étaient déjà bien présents dans les véhicules modernes.

Les recherches sur les systèmes de localisation sont essentiellement orientées sur l'amélioration des services en terme de précision, de disponibilité et d'intégrité. Le présent manuscrit est à ranger dans le domaine de la recherche de systèmes de plus en plus intègres, adaptés à des applications critiques et temps réels. Parmi toutes les méthodologies existantes, on se place dans le cadre de celles cherchant à fournir une garantie dans les positions fournies. Dans le prochain chapitre, nous poserons le problème de localisation comme un problème d'observation de l'état d'un système et nous rappellerons les méthodes classiques les plus utilisées.

CHAPITRE 2. FUSION DE DONNEES MULTI SENSORIELLES

Introduction

L'idée de fusionner des données correspond à la volonté de regrouper simultanément différentes sources de données, ayant chacune éventuellement des caractéristiques différentes, afin d'obtenir une nouvelle information plus riche. La fusion de données se rencontre dans de nombreux domaines parmi lesquels on peut citer le traitement du signal, la fouille de données, le diagnostic, la robotique, la compréhension automatique de documents scientifiques, ou encore la représentation logique de la connaissance. Pour tous ces domaines, le problème rencontré est celui de combinaison et d'utilisation simultanée de données ou informations venant de plusieurs sources ou capteurs. Tous ces domaines ont en commun le fait de rechercher à élaborer une information plus riche à partir de données de nature et de sources variées. Cette notion de richesse de l'information qui caractérise la fusion de données dépend alors du type d'application. On peut classer la richesse de l'information en deux familles d'idées dissociables :

- dans un premier cas, il s'agit d'une grande quantité de données provenant de différentes bases de données. L'idée de les rendre plus riches se rapporte à une reconstruction d'une nouvelle structure de données qui va résumer le tout, ou à une extraction de quelques parties réellement déterminantes. Par exemple, la fusion de plusieurs documents scientifiques permettra d'obtenir un document composite résumant les relations et parties pertinentes qui existent entre les documents fusionnés. Le résultat sera une information plus facile à utiliser pour un être humain. La richesse de l'information obtenue ici est motivée par le désir de faciliter la lecture d'un grand nombre d'informations.
- dans un second cas, il s'agit de données qui ne sont pas forcément très nombreuses. Enrichir l'information s'évalue en termes plus techniques. On parle alors d'objectifs de précision, fiabilité ou intégrité. Le contexte applicatif consiste souvent à vouloir percevoir

un environnement à partir de différents capteurs, chacun de ces derniers ayant ses propres propriétés en matière de fiabilité, précision, disponibilité, intégrité et champ d'auscultation.

Dans ce chapitre nous proposons d'aborder le problème de la fusion de données sous l'angle de la fusion technique d'informations provenant de divers capteurs avec une approche par représentation d'état d'un système. Dans un premier temps, nous présenterons la formulation classique sous forme d'un système avec une représentation d'état. Puis, nous ferons un état de l'art des techniques d'observation d'état existantes.

1 FUSION DE DONNEES AVEC MODELE D'EVOLUTION

Une notion fondamentale pour étudier les problèmes physiques en général est la notion de système. Un système peut être défini comme un objet complexe, formé de composants distincts, en interactions dynamiques et organisés pour satisfaire des objectifs de fonctionnement. Les composants sont considérés comme des sous-systèmes. On s'intéresse alors à étudier l'évolution du système au cours du temps afin de pouvoir l'observer, le contrôler ou le stabiliser. L'évolution d'un système est conditionnée à la fois par les modifications internes qui peuvent affecter les composants ou encore par les interactions qui s'établissent entre le système et son environnement.

Dans sa formulation mathématique, un système peut être représenté par un ensemble de variables. Cet ensemble de variables (si elles sont connues) permet de connaître, reconstituer ou prévoir le fonctionnement du système. Ces variables sont regroupées pour former un vecteur appelé état du système. L'observation d'état d'un système a alors pour but de suivre le comportement de variables internes du système grâce à des mesures effectués sur ce dernier. En règle générale, pour étudier un système, le premier travail consiste à définir des variables qui sont représentatives, à un instant donné, du comportement physique du système et qui formeront l'état du système. Une seconde étape consiste alors à modéliser l'évolution de ces variables au cours du temps. En automatique, on définit les notions "d'entrées" et de "sorties" du système. Les entrées du système sont généralement des variables sur lesquelles on peut agir directement afin de contrôler le système selon un fonctionnement voulu. Les sorties du

système sont des variables dont on peut mesurer les valeurs prises au cours du temps. Grâce à ces sorties, on cherche à évaluer le comportement du système.

On propose de donner dans la suite les principales définitions liées à la modélisation de l'évolution d'un système.

1.1 Modèle d'évolution d'un système

On peut définir un modèle comme une approximation ou une vue partielle plus ou moins abstraite de la réalité. Il est établi pour un objectif donné, selon un point de vue, dans un domaine de validité et pour un instant donné. Une modélisation sert à appréhender le fonctionnement d'un système. Un modèle est donc subjectif puisqu'il peut être établi en fonction des objectifs, du jugement, de la nature et de la qualité des informations dont on dispose. En automatique, il est le plus souvent exprimé par des équations mathématiques reliant l'état du système et d'autres variables tels que les entrées, les sorties et des paramètres du système. Si ces équations sont algébriques, le modèle est dit *statique*. Si ces équations sont des équations différentielles ou des équations aux différences récurrentes, le modèle est dit *dynamique*, respectivement *à temps continu* ou *à temps discret*.

1.1.1 Modèle statique

On parle de modèle statique lorsque les équations du système sont indépendantes du temps dans le cas continu ou lorsque les équations ne relient que des variables de l'échantillonnage courant, dans le cas discret. Un modèle statique correspond donc à un problème dans lequel le temps n'intervient pas explicitement entre les instants d'échantillonnage. Pour la fusion de données, ce type de modèle concerne une catégorie appelée *fusion statique*.

Pour $x \in \mathrm{IR}^n$ l'état du système, $u \in \mathrm{IR}^p$ l'entrée du système, $\theta \in \mathrm{IR}^q$ l'ensemble des paramètres du système et $y \in \mathrm{IR}^m$ l'ensemble des mesures effectuées, le problème d'observation d'état du système statique peut être exprimée sous la forme générale écrite dans (2.1)

$$\begin{cases} f(x,u,\theta)=0 \\ y=h(x,\theta) \end{cases} \tag{2.1}$$

La relation $y = h(x,\theta)$ reliant les sorties *y,* les paramètres θ et l'état *x* est appelée équation d'observation du système. Souvent, *m>n* et on a une redondance de données et le problème d'observation de l'état correspond à un problème de fusion de données.

1.1.2 Modèle dynamique

Les problèmes de fusion de données ayant un modèle d'évolution dynamique correspondent à une catégorie appelée *fusion dynamique*. En général, dans les problèmes pratiques, il faut être capable de modéliser l'évolution dans le temps de toute ou d'une partie des composantes de l'état.

1.1.2.1 Modèle à temps continu

Les modèles à temps continu font intervenir des équations différentielles avec des dérivées par rapport au temps. Dans un problème de fusion de données traité par un calculateur, les mesures sont souvent obtenues à un échantillonnage discret et, par conséquence, les modèles à temps continu sont souvent discrétisés pour obtenir des modèles discrets.

A un instant donné t, le modèle d'état du problème d'observation dynamique à temps continu peut être exprimé sous la forme générale écrite dans 2.2 :

$$\begin{cases} \dot{x}(t) = f(x(t),u(t),\theta(t)) \\ y(t) = h(x(t),\theta(t)) \end{cases} \tag{2.2}$$

Où :

- $x(t) \in IR^n$ est l'état du système à l'instant *t*,
- $u(t) \in IR^p$ est l'entrée du système à l'instant *t*,
- $\theta(t) \in IR^q$ est l'ensemble des paramètres fixes du système à l'instant *t*,
- $y(t) \in IR^m$ est l'ensemble des mesures effectuées à l'instant *t*.

1.1.2.2 Modèle à temps discret

Les modèles à temps discret font intervenir une récurrence entre les variables du système échantillonné selon un pas de temps connu. Pour une période d'échantillonnage T_e et pour un instant d'échantillonnage t_k = Te**k*, la forme générale des modèles à temps discret peut alors être représentée par une des formes (2.3) ou (2.4) :

$$\begin{cases} x_{k+1} = f(x_k, u_k, \theta_k) \\ y_k = h(x_k, \theta_k) \end{cases} \quad (2.3)$$

$$\begin{cases} x_k = f(x_{k-1}, u_k, \theta_k) \\ y_k = h(x_k, \theta_k) \end{cases} \quad (2.4)$$

Où

- $x_k \in IR^n$ est l'état du système à l'instant t_k,
- $u_k \in IR^p$ est l'entrée du système à l'instant t_k (dans 2.3, u_k est connue et est donné, par exemple, par un régulateur alors que dans 2.4, u_k est mesurée.
- $\theta_k \in IR^q$ est l'ensemble des paramètres fixes du système à l'instant t_k
- $y_k \in IR^m$ l'ensemble des mesures effectuées à l'instant t_k.

1.1.3 Cas particulier : modèles linéaires

Les modèles linéaires font partie des cas les plus simples à traiter car on sait passer du continu au discret à partir de la connaissance de la forme des signaux d'entrée (souvent suivant une évolution en marches d'escalier ou signal bloqué à l'ordre 0). De surcroît, de nombreux concepts bien adaptés et des résultats fondamentaux existent autour des modèles linéaires.

Une des contributions significatives à l'automatique linéaire revient aux travaux de Rudolf Kalman[1], à qui l'on doit les principaux concepts et résultats de la théorie. Les outils développés pour les modèles linéaires sont si aboutis qu'une approche courante face à un problème présentant des non linéarités est la linéarisation des modèles afin de pouvoir utiliser

[1] Mathématicien américain né à Budapest en 1930

les concepts et résultat spécifiques. De plus, de nombreux concepts telles que l'observabilité, l'identifiabilité ou la contrôlabilité ont été initialement définis pour des problèmes linéaires avant d'être étendus aux autres classes de problèmes. Les problèmes linéaires sont donc ceux qui sont le plus facilement appréhendables et qui permettent d'introduire des notions qui s'avèrent utiles pour les autres classes de problèmes non linéaires.

1.1.4 Observabilité

L'idée d'observabilité d'un système vient d'un besoin précis qui est celui de recalculer son état à partir des sorties et des entrées, si celui-ci n'est pas entièrement mesuré. L'observabilité peut donc se définir comme l'aptitude à reconstruire l'état en un temps fini et de façon unique par observation des sorties et connaissance des entrées.

1.2 Contraintes de temps réel

Le temps réel peut se définir comme une exigence d'adéquation entre les capacités de traitement et le temps de réaction du système. En automatique, les besoins sont de contrôler un système ou d'observer son évolution avec la contrainte forte de devoir gérer le temps. Par exemple, pour contrôler la vitesse d'un véhicule, il faut être capable de modifier la vitesse dans un délai assez court garantissant que la vraie vitesse au moment où la consigne est exécutée est encore cohérente avec cette dernière. Dans cet exemple, les difficultés du temps-réel sont multiples. Tout d'abord, il faut une adéquation entre, d'une part, les fréquences des mesures effectuées et, d'autre part, les temps d'acquisition et les temps de traitement. Ensuite, il faut aussi connaître avec une bonne précision les temps de réactions du système. Une fois cette adéquation et ces informations obtenues, il faut vérifier la compatibilité mesures/consignes pour pouvoir agir sur le système.

Un autre problème transverse au problème de modélisation est l'étude des différentes erreurs intervenant dans les connaissances sur le système et sur les instruments de mesure. La connaissance de ces erreurs est indispensable pour pouvoir agir sur le système. Dans la section suivante l'étude des catégories d'erreurs présentes dans un problème d'observation d'état est faite.

2 Etude des erreurs

Autour de la notion d'erreur, existent deux termes très proches couramment utilisés que sont l'imprécision et l'incertitude, prêtant souvent à confusion. Nous proposons de les clarifier de la façon suivante :

- l'incertitude d'une valeur mesurée ou calculée pour une variable traduit la confiance que l'on a que cette valeur soit vraie. Plus l'incertitude d'une valeur est grande, plus les chances que celle-ci soit vraie sont faibles.
- l'imprécision d'une grandeur correspond à l'écart entre la valeur mesurée ou calculée de cette grandeur et sa vraie valeur. L'imprécision est une grandeur avec une unité qu'on cherche à minorer à défaut de pouvoir connaître sa valeur exacte.

Les erreurs résultant d'un processus de fusion de données peuvent être multiples. Elles peuvent être dues à des erreurs de modélisation de l'évolution du modèle ou à des erreurs sur les mesures fournies par les différents capteurs.

2.1 Erreur de modèle

Les erreurs de modélisation sont les plus difficiles à évaluer et à limiter dans le temps. En effet, les modèles sont toujours faits sur la base d'hypothèses qui ont leur domaine de validité dans le temps et dans l'espace. On peut donner l'exemple d'un véhicule en déplacement pour lequel on prend l'hypothèse qu'il roule sans glissement. C'est une hypothèse qui est vraie, la plupart du temps, sur des portions de route rectilignes avec une bonne texture, sans perturbations comme la pluie ou le verglas. Différents paramètres totalement imprévisibles et non modélisables peuvent donc fausser un modèle à tout moment.

En plus de ces erreurs dues aux hypothèses, différentes sortes d'approximation peuvent être sources d'erreur. On peut résumer cet ensemble d'approximations comme suit :

- dans le cas des modèles dynamiques continus, les erreurs dues à la discrétisation (on discrétise pour être cohérent avec la nature discrète des mesures),
- le mauvais étalonnage des paramètres physiques du système,

- les simplifications faites sur les équations afin de faciliter les algorithmes et d'accélérer les temps de calcul,
- la latence de certains capteurs et la désynchronisation des capteurs.

2.2 Erreur de mesures

Une erreur de mesure est définie comme la différence entre la valeur mesurée et la valeur vraie de la grandeur. Les erreurs de mesures peuvent être subdivisées en deux catégories que sont les erreurs systématiques et les erreurs accidentelles [Ragot et al., 90]. Les erreurs dites systématiques sont généralement inhérentes aux principes de fabrication des capteurs. L'hypothèse la plus répandue que l'on fait sur un capteur est qu'il existe une relation linéaire entre l'entrée du capteur et sa sortie. Il peut alors apparaître des erreurs constantes appelées *biais* et d'autres erreurs plus lentes qui agissent à long terme avec la durée de mesure, appelées *dérives*. Les erreurs de mesures dites accidentelles sont souvent aléatoires et sont donc moins prévisibles. Elles peuvent être d'origine diverses comme la présence de signaux parasites (à l'exemple d'un récepteur GPS) ou la présence d'autres appareils de mesure (à l'exemple d'un accéléromètre qui peut être perturbé par les champs magnétiques d'appareils électriques dans son entourage).

2.3 Modélisation des erreurs

Les erreurs de modèle et de mesure sont donc difficiles à déterminer et à modéliser. Les erreurs de mesure en particulier sont spécifiques à la nature des capteurs et peuvent être aléatoires. Les erreurs de modèle sont quant à elles tributaires des connaissances limitées de l'utilisateur sur le système. Il existe différentes approches pour modéliser ces erreurs. De cette modélisation des erreurs dépend en grande partie la méthode utilisée pour fusionner les données en jeu. On peut distinguer deux grandes approches pour modéliser ces erreurs :

- ***Les méthodes probabilistes*** sont de loin les plus utilisées et les mieux connues. Une variable est représentée par une densité de probabilité. Les avantages de cette modélisation sont nombreux. Outre la richesse des outils de la théorie des probabilités, les algorithmes sont capables de fournir une estimation ponctuelle qui représente le point le plus probable.

- ***Les approches à erreurs bornées*** sur lesquelles sont à rattacher les techniques utilisées dans cette thèse. Elles sont bien adaptées à la nature de la majorité des erreurs que l'on rencontre dans la réalité. En effet, la seule hypothèse à vérifier est que toutes les erreurs sont bornées. De plus pour une mesure effectuée, une stratégie simple pour déterminer la borne de l'erreur commise peut être une sommation des amplitudes maximales des différentes sources d'erreurs répertoriées.

Dans la section suivante, on s'intéresse à donner brièvement des éléments sur différentes approches connues qui ont été développées pour reconstruire l'état d'un système au cours du temps.

3 Fusion dynamique par observateurs

Un observateur peut être décrit comme un processus au cours du temps dont l'objectif principal est de reconstituer, dans une fenêtre de temps précise, l'état du système grâce à la connaissance des paramètres du système et des mesures effectuées. Pour la fusion de données, il existe de nombreuses techniques pour observer l'état du système. Nous proposons, dans cette partie, de faire un état de l'art des techniques existantes. Pour cela nous avons distingué les observateurs "*ponctuels*" qui sont plus classiques des observateurs "*ensemblistes*" qui sont plus récents dans le domaine de l'automatique.

3.1 Approches ponctuelles

Par approches ponctuelles, on désigne les observateurs d'état fournissant comme sortie une estimation sous la forme d'un point (ou d'un vecteur). Dans ce type d'approche, l'état est souvent traité comme une variable aléatoire avec une densité de probabilité.

3.1.1 Filtrage de Kalman

Le filtre de Kalman est à ce jour, à plusieurs variantes près, l'observateur le plus utilisé pour résoudre des problèmes d'estimation d'état. Selon un critère précis, il a pour objectif de reconstruire une estimation optimale de l'évolution d'un système moyennant des mesures bruitées. Il a été introduit en 1960 dans [Kalman 60] pour des systèmes contrôlés. C'est un prédicteur/estimateur séquentiel qui a pour cadre d'application les systèmes avec un modèle

linéaire ayant une représentation d'état dite "*stochastique*". La forme générale d'un modèle stochastique, linéaire, discret peut être représenté par le système (2.5)

$$\begin{cases} x_k = Ax_{k-1} + Bu_k + \alpha_k \\ y_k = Cx_k + \beta_k \end{cases} \tag{2.5}$$

où :

- $x_k \in \mathrm{IR}^n$ est l'état du système, $u_k \in \mathrm{IR}^p$ est l'entrée du système et $y_k \in \mathrm{IR}^m$ la sortie du système,
- $\alpha_k \in IR^n$ représente les erreurs de modélisation du système,
- $\beta_k \in IR^m$ représente les bruits de mesure perturbant la sortie y_k.

Plusieurs hypothèses sont faites sur les bruits α_k et β_k. Ils sont supposés être des variables aléatoires dont les distributions sont gaussiennes. Les bruits sont supposés en outre blancs, centrés et indépendants de l'état initial du système. Leurs matrices de covariance respectives Q_α et Q_β sont souvent supposées connues et constantes pour alléger les équations. Le filtre de Kalman est constitué des deux étapes de prédiction /estimation suivantes :

- ***la prédiction*** : connaissant l'état x_{k-1} à l'instant $k-1$, la prédiction $x_{k|k-1}$ de l'état x_k sachant $k-1$ se fait en utilisant l'équation du modèle d'évolution : $x_k = Ax_{k-1} + Bu_k + \alpha_k$
- ***l'estimation ou correction*** : Pour y_k le vecteur de la mesure à l'instant k et $y_{k|k-1}$ la prédiction de la mesure connaissant l'état à l'instant $k-1$, l'erreur entre la mesure et la prédiction de la mesure est $ey_{k|k-1} = y_k - y_{k|k-1}$. L'idée originale du filtre de Kalman est d'utiliser cette erreur $ey_{k|k-1}$, de façon optimale selon un critère, pour corriger l'état prédit : ce qui correspond à la deuxième étape d'estimation (étape encore appelée correction).

Le filtre de Kalman est optimal dans le sens où il fournit une estimation non biaisée et une erreur de variance minimum à chaque étape. Cette optimalité est obtenue grâce à un critère statistique (*l'orthogonalité* notamment) qui garantit une erreur statistiquement centrée et orthogonale aux observations et aux prédictions. Malheureusement, le filtre ne fournit une estimation théorique optimale que si les conditions sur les bruits sont vérifiées et ces conditions sur les bruits sont rarement vérifiées en pratique.

En outre, plusieurs extensions du filtre de Kalman existent pour répondre à des contraintes fortes telles que la non linéarité des modèles. En ce qui concerne la non linéarité, on utilise souvent la version étendue du filtre de Kalman dite *Extended Kalman Filter* (EKF). L'idée simple de cette extension est de linéariser le modèle dynamique autour d'une solution approchée qui est généralement la prédiction de l'état. Dans les années 90, Julier et Uhlmann [Julier et al., 1996] introduisent le *Unscented Kalman Filter* (UKF) basé sur une méthode innovatrice pour calculer la statistique d'une variable aléatoire soumise à une fonction non-linéaire. Cette méthode repose sur l'idée qu'il est plus simple d'approximer une distribution discrète gaussienne que d'approximer une fonction ou une transformation non-linéaire appliquée à une gaussienne (comme on le fait avec l'EKF). Pour cela, on choisit un échantillon précis de points pondérés à partir de la moyenne et de la variance. Puis, à chacun de ces points est appliqué la fonction non-linéaire pour obtenir un nouveau nuage de points dont on pourra approximer moyennes et variances. Il existe également différentes approches proches de l'UKF telles que les DD[2] [Norgaard et al., 2000] et le CDKF[3] [Ito et al., 2000]. Ces méthodes utilisent la même approche d'échantillonnage déterministe supprimant ainsi le recours à la linéarisation de fonctions.

Malheureusement, la non-linéarité du modèle peut entraîner une multi-modalité de la loi conditionnelle de l'état alors que le filtre de Kalman approche la densité de l'état sachant l'observation par une densité gaussienne, donc uni-modale. Ainsi, lorsque les équations du système sont fortement non-linéaires et/ou non Gaussiennes, les extensions du filtre de Kalman peuvent être inadaptées et diverger. Dans ces cas, les approches dites particulaires qui seront introduites dans la section suivante semblent être les filtres mieux adaptés.

3.1.2 Approches particulaires

Les capacités de calcul étant de plus en plus importantes, il est possible de développer des filtres non-linéaires basés sur des approximations Monte-Carlo appelés filtres particulaires. C'est un domaine en pleine expansion aussi bien au niveau théorique qu'au niveau des applications. Elles ont été introduites dans [Gordon et al., 1993] et [Moral et al., 92]. Le filtrage particulaire est, par exemple, utilisé en suivi de formes en traitement d'images ou en localisation autonome de robots. L'idée principale de ces méthodes est de représenter la loi

[2] DD :différences divisées

[3] CDKF : Central Difference Kalman Filter

conditionnelle de l'état par un nombre fini de Dirac pondérés. Ainsi, un ensemble de points appelés particules est généré. Chacune de ces particules représente un état probable du système auquel on affecte un coefficient de pondération (ou poids). Ces poids sur chaque particule sont une mesure de la confiance que l'on a pour que les particules représentent effectivement l'état. La méthode fonctionne, comme pour le filtre de Kalman, de façon récursive et selon deux étapes de prédiction et d'estimation. Les particules évoluent suivant l'équation d'état du système (étape de prédiction) et leurs poids sont remis à jour en fonction des observations (étape de correction).

Les avantages principaux de cette méthode sont la possibilité de gérer la multi-modalité et des cas de forte non-linéarité. Malheureusement, en plus de son coût en termes de ressources et de temps de calcul, le nombre de particules à générer reste un problème ouvert. Le filtre particulaire présente également, intrinsèquement, un défaut majeur. En effet, les poids des particules ont tendance à dégénérer après un certain nombre de mesures. On peut se retrouver alors dans une situation où la plupart des particules ont un poids négligeable. Le nuage de particules s'appauvri et par conséquent la densité conditionnelle peut être mal représentée et le filtre diverger. Cette situation est connue sous le nom de "*dégénérescence des poids*". Pour éviter cette situation, une nouvelle étape appelé ré-échantillonnage a été introduite. Le principe est de dupliquer les particules de poids fort et d'éliminer les particules de poids faible. Les méthodes de ré-échantillonnage proposées sont souvent des heuristiques, ce qui a donné lieu a de nombreuses versions de filtrage particulaire.

A l'opposé de ces approches dites ponctuelles, sont apparues des approches dites ensemblistes dont l'objectif est d'englober toutes les solutions possibles. Dans la section suivante, on donne un aperçu de différentes classes d'ensembles particuliers ayant fait l'objet de recherches probantes.

3.2 Approches ensemblistes

Par approches ensemblistes, on désigne les observateurs d'état fournissant comme sortie une estimation garantie sous la forme d'un ensemble particulier de IR^n (ellipse, intervalle, zonotope). La notion d'estimation garantie a été considérée pour la première fois dans la fin des années 60 notamment dans [Bertsekas et al., 1968 ; Schweppe, 1968 et Witsenhausen, 1968]. Dans ce type d'approche, l'état est traité comme un ensemble garanti de valeurs possibles englobées dans un domaine initial que l'on cherchera à réduire le plus possible.

3.2.1 Approches ellipsoïdales

L'approche ellipsoïdale connut à ses débuts surtout des adeptes russes [Kurzhanskii, 1977 ; Chernousko, 1981]. De nos jours, la communauté s'y intéressant s'est fortement agrandie et la théorie avec. Le domaine d'application regroupe les systèmes linéaires avec une unique hypothèse sur les erreurs notamment leur nature bornée. Les méthodes sont séquentielles et reposent essentiellement sur deux étapes de prédiction et d'estimation. La prédiction vise le calcul d'un ellipsoïde optimal satisfaisant les équations d'état du système. L'estimation est constituée de deux étapes (voir Figure 2.1). La première étape permet de calculer un ellipsoïde, dans l'espace d'état, réciproque de l'ellipsoïde correspondant aux mesures. La seconde étape correspond au calcul d'un ellipsoïde qui englobe l'intersection de l'ellipsoïde prédit et de l'ellipsoïde provenant des observations. L'optimalité est alors définie par la minimisation d'un critère qui peut être fonction du volume des ellipsoïdes englobant, sur la trace des matrices caractérisant ces ellipsoïdes ou encore la projection des ellipsoïdes selon un vecteur choisi (voir [Chernousko 1994 ; Chernousko 1999]). Le lecteur intéressé pourra se documenter plus amplement dans [Fogel et al., 1982 ; Milanese et al. 1996 ; Durieu et al., 2001, Lesecq et al. 2002].

Les principaux atouts de ces méthodes sont sa nature séquentielle et des algorithmes qui suivent les étapes classiques des filtres ponctuels. On peut également évoquer la richesse des techniques d'optimisation et de la théorie autour des ellipsoïdes. Sans compter la maniabilité et les possibilités de rotations des ellipses qui diminuent grandement le pessimisme des phases d'englobement.

La principale limitation de la méthode apparaît suite aux difficultés rencontrées pour manier des ellipses lorsque les problèmes sont non-linéaires. S'y ajoute la difficulté sur le choix d'un critère judicieux pour l'optimalité qui est lié à la nature des applications.

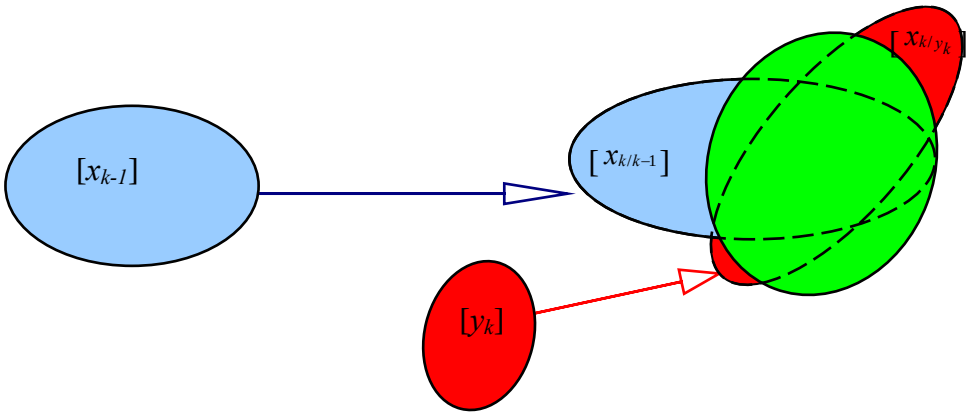


Figure 2.1 : Etape de prédiction/estimation pour les méthodes ellipsoïdales

3.2.2 Approches par des zonotopes

La recherche sur des méthodes d'estimations d'états ensemblistes par les zonotopes est récente [puig et al., 2001 ; Combastel, 2003 ; Alamo et al., 2005]. L'approche par les zonotopes est très proche de celle par les ellipsoïdes. L'objectif est aussi d'englober, de façon optimale (selon un critère visant à diminuer le pessimisme), l'ensemble des solutions admissibles par un zonotope. On considère également les deux étapes de prédiction et d'estimation avec des outils et techniques d'optimisation spécifiques aux zonotopes. Un zonotope Z peut être représenté par un point M et un ensemble de p vecteur (u_1,…, u_p). Z s'écrit mathématiquement :

$$Z = M+[S_1]\dots +[S_p], \text{ où } [S_i] = [-1, 1]^* u_i \text{ (voir Figure 2.2).}$$

Le principal avantage des zonotopes est la possibilité d'approximer très finement les ensembles solutions [Kühn, 1998]. En effet, en augmentant le nombre de vecteurs, on obtient des domaines convexes très complexes qui offrent plus de richesse que de simples ellipsoïdes ou intervalles.

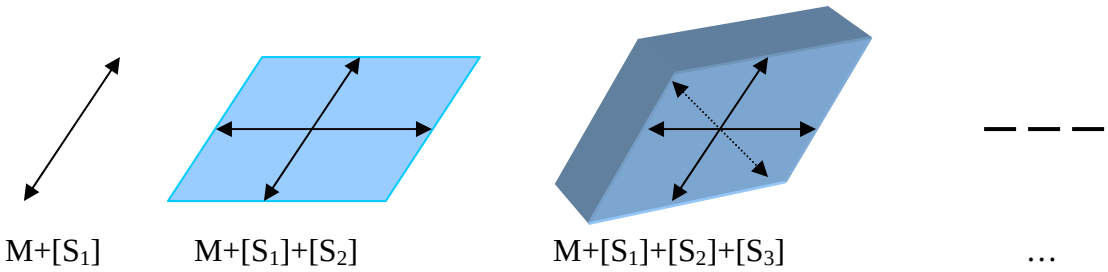


Figure 2.2 : Etape de prédiction/estimation pour les méthodes ellipsoïdales

3.2.3 Analyse par intervalles

Cette approche ensembliste a été celle utilisée au cours de cette thèse. Le lecteur est renvoyé au chapitre 3 pour tous les détails précis sur les techniques d'analyse par intervalles. Néanmoins, on parlera ici des méthodes existantes d'observation d'état à erreurs bornées basées sur le mécanisme prédicteur/estimateur.

Pour l'estimation d'état, les applications de l'analyse par intervalles à la robotique sont récentes et généralement basées sur l'algorithme SIVIA[4] (Bouron et al. 2001, Meizel et al., 2002, Kieffer et al., 2002). L'idée de l'algorithme SIVIA (Jaulin et al., 1997), est de calculer l'image réciproque par une fonction *f,* linéaire ou pas, d'un pavé dans IR^n. Pour cela, on procède à un sous-pavage du domaine de recherche initial [*x*] (un sous-pavage est une union finie de pavé non sécants). Le principe est alors de tester pour tout pavé élément du sous-pavage initial, si son image par une fonction d'inclusion, $[f]$, de *f* est contenu dans le pavé [*y*] de l'espace d'observation. Un pavé vérifiant cette propriété est mémorisé. A contrario, un pavé dont l'image par $[f]$ n'a pas d'intersection avec le pavé [*y*] de l'espace d'observation, est exclu. Il reste alors les pavés ayant juste une intersection avec [*y*]. Le processus est alors réitéré en bissectant les éléments restants tant qu'un seuil pré-spécifié, sur la taille des pavés, n'est pas atteint.

L'inversion ensembliste par SIVIA permet de retrouver les 2 étapes de prédiction/estimation de la façon suivante :

- la prédiction : cette étape consiste à un simple calcul de l'image de l'état $[x_{k+1}]$ à l'instant $k+1$ grâce au modèle d'évolution appliqué aux pavés de l'instant précédent *k* (les pavés comme $[x_k]$ et $[u_k]$),
- la correction: cette étape consiste à faire l'intersection entre le pavé état estimé $[x_{k+1}]$ et le domaine consistant avec le pavé d'observation $[y_{k+1}]$ obtenu par SIVIA.

Cette méthode présente de nombreux avantages telles que la précision atteinte pour les pavés estimés. Sans compter que cette méthode utilise le mécanisme récursif habituel de prédicteur/estimateur.

---

[4] SIVIA : **S**et **I**nversion **V**ia **I**nterval **A**nalysis

Malheureusement, les algorithmes de découpage ont une complexité exponentielle incompatible avec le temps-réel. Par ailleurs, dans des problèmes à plusieurs dimensions se pose la question, non encore élucidée, du choix des dimensions à bissecter.

Cette thèse s'est, de façon délibéré, privé des possibilités de découpage. La motivation de ce choix est exclusivement d'étudier d'autres techniques supposées compatible avec le temps-réel comme on le verra dans le prochain chapitre.

Conclusion

On a vu dans le chapitre 1 précédent que le problème de localisation dynamique d'une automobile avec des capteurs proprioceptifs et extéroceptifs est un problème de fusion de données. Dans ce chapitre, un tour d'horizon a été fait, d'une part, sur la formulation classique permettant d'aborder les problèmes de fusion de données et, d'autre part, sur une grande variété d'approches et techniques permettant de les résoudre. Parmi les techniques citées, on en a distingué deux classes importantes. Les plus classiques étant celles qu'on a rangées dans les approches ponctuelles. Les approches ensemblistes, quant à elles, ont été introduites pour répondre à une problématique de résultats garantis. Parmi ces approches ensemblistes, cette thèse est à ranger dans celles voulant explorer spécifiquement l'apport des nombreux outils d'analyse par intervalle.

Dans le prochain chapitre, après une présentation d'outils de l'analyse par intervalle, on présente les fondements de la méthode choisie, à savoir la propagation de contraintes sur les intervalles, pour implémenter une technique d'estimation garantie. Puis, on montre les résultats de cette méthode appliquée sur des données réelles.

CHAPITRE 3 : APPLICATION DES TECHNIQUES DE SATISFACTION DE CONTRAINTES SUR LES INTERVALLES AU PROBLEME DE LOCALISATION

Introduction

L'introduction des méthodes basées sur l'analyse par intervalles dans le domaine de l'automatique est très récente. L'analyse par intervalles constitue une approche permettant d'explorer avec des outils puissants, l'aspect garanti des problématiques classiques de l'automatique. Pour notre part, nous nous intéressons aux problèmes d'observation d'état de systèmes statiques ou dynamiques, linéaires ou non linéaires.

L'analyse par intervalles a été introduite pour la première fois par Moore en 1966 dans son livre *Interval Analysis* [Moore, 1966] qui est devenu une référence à ce jour. C'est un ensemble d'outils basés sur l'idée simple de substituer les nombres réels par des intervalles (ou plus généralement les vecteurs réels par des pavés) et de pouvoir parfaitement manipuler les intervalles de la même façon qu'on sait le faire avec les nombres réels. Pour cela, les opérations arithmétiques et ensemblistes, classiquement définies pour les nombres réels, ont été étendues aux intervalles. Grâce à ces outils on peut envisager d'obtenir des domaines garantis, par une simple transposition aux intervalles des algorithmes numériques classiques opérant avec des nombres réels.

Il faut remarquer qu'en règle générale, l'homme est habitué dans toutes ses conceptions à travailler avec des nombres réels, quoique incertains, faciles à appréhender, plutôt que des ensembles décrivant un ensemble de solutions possibles. C'est ainsi que le tableau de bord d'une voiture, par exemple, donne toujours au conducteur une vitesse ponctuelle plutôt qu'un intervalle de vitesses possibles. Autre exemple, les organismes de sondage s'affairent à

publier des chiffres bruts qu'ils fournissent aux consommateurs plutôt que des intervalles modélisant les marges d'erreurs et plus représentatifs de la réalité.

Le raisonnement par intervalle est inhabituel pour l'esprit humain plutôt porté vers l'obtention d'approximations ponctuelles. Ce raisonnement par des ensembles a cependant l'avantage de répondre à des questions qui ne sont pas résolubles par les outils classiques. En effet, il devient naturel d'apprendre à manipuler des ensembles lorsque l'on cherche à déterminer un ensemble de solutions possibles.

Un autre avantage de ces méthodes est la possibilité de répondre aux problèmes numériques dus à la représentation, nécessairement fini, des réels sur les ordinateurs. De nombreuses études basées sur l'analyse par intervalles ont fini de convaincre sur l'intérêt d'estimer ces erreurs d'arrondi dans des problèmes nécessitant un haut niveau d'intégrité.

L'analyse par intervalles est donc un outil qui éveille notre curiosité du fait de son coté rénovateur par rapport à la façon de raisonner de l'homme et qui a d'emblée soulevé de nombreuses questions par rapport à son utilité devant tous les problèmes physiques que l'on ne sait résoudre qu'approximativement avec les outils classiques.

Dans un premier temps nous rappellerons quelques notions élémentaires et théoriques de l'analyse par intervalles avec tout ce qu'elle comporte comme définitions et extensions. L'objectif est de bien expliquer les outils qui seront utiles pour la suite et de donner un aperçu sur d'autres travaux qui ont été réalisés dans ce domaine sans être trop explicite, dans un souci de synthèse. Le lecteur pourra se référer au livre [Jaulin et al, 2001a] pour plus de détails. Puis, nous présenterons une autre théorie étendue aux intervalles que sont les "problèmes de satisfaction de contraintes". Cette formulation permettra de traiter les problèmes d'estimation d'état avec une logique d'approximation de l'ensemble des solutions possibles avec le moins de pessimisme possible. Nous pourrons alors appliquer cette approche au problème de la localisation sur des données réelles enregistrées avec le véhicule expérimental du laboratoire.

1 NOTIONS D'ANALYSE PAR INTERVALLES

Cette partie se veut une synthèse des définitions classiques de l'analyse par intervalle. Nous nous limiterons ainsi aux définitions juste utiles pour bien appréhender les techniques de satisfaction de contraintes détaillées dans la section 2.

1.1 Définitions et notations

- Un intervalle réel est un sous-ensemble fermé borné connexe de I. On le note :

$$[x] = \left[\underline{x}, \overline{x}\right] = \{x \in \mathrm{IR} \mid \underline{x} \leq x \leq \overline{x}\} \tag{3.1}$$

où $\underline{x}$ et $\overline{x}$ sont respectivement ses bornes inférieures et supérieures.

Pour caractériser un intervalle on utilise aussi :

- Son centre (ou milieu) noté $m([x]) = \frac{\underline{x}+\overline{x}}{2}$
- Sa longueur noté $w([x]) = \underline{x}-\overline{x}$ (ou son rayon noté $r([x]) = \frac{\underline{x}-\overline{x}}{2}$)

- Un intervalle est dit dégénéré si $\underline{x}=\overline{x}$ (cette dernière définition permet de caractériser les réels dans l'ensemble des intervalles de IR)
- On définit pour les intervalles toutes les opérations ensemblistes habituelles : les notions d'égalité (=), d'inclusion ($\subset$, $\supset$, $\subseteq$, $\supseteq$) ainsi que d'intersection ($\cap$). Cependant la réunion de deux intervalles n'étant pas un général un intervalle, on définit l'union convexe de deux intervalles comme le plus petit intervalle contenant l'union de ces deux intervalles.

$$[x] \cup [y] = [\min(\underline{x}, \underline{y}), \max(\overline{x}, \overline{y})] \tag{3.2}$$

- L'ensemble de tous les intervalles dans IR sera noté IIR. Pour un ensemble $D \subset \mathrm{IR}$, l'intervalle noté [D] est défini comme le plus petit intervalle contenant D :

$$[D] = \cap \{[x] \in \mathrm{IIR} \mid D \subset [x]\} \tag{3.3}$$

- On peut étendre de façon naturelle toutes les opérations arithmétiques de base {+, −, * , /} pour pouvoir évaluer l'ensemble que décrit un réel qui est le résultat d'une opération

binaire sur deux réels incertains inclus dans deux intervalles [Hanson, 1968 ; Kahan, 1968] . Pour ◊ représentant un des signes dans {+, −, *, /}, on définit son extension dans IR×IR de la manière suivante :

$$[x] \Diamond [y] = [\{x \Diamond y \mid x \in [x] \text{ et } y \in [y]\}] \tag{3.4}$$

Remarquons qu'il résulte immédiatement de cette définition que les propriétés de commutativité et d'associativité restent valables. Mais en général les propriétés algébriques sont perdues. L'exemple le plus expressif est que $[x]-[x]$ n'est généralement pas égal à [0, 0] ; ceci parce que $[x]-[x] = \{x-y \mid x \in [x], y \in [x]\}$ est différent de $\{x-x \mid x \in [x]\}$.

Pour caractériser le résultat de l'opération $[x] \Diamond [y]$ pour l'addition, la soustraction et la multiplication, on peut se servir des formules suivantes :

$$\begin{cases} [x]+[y] = [\underline{x}+\underline{y}, \overline{x}+\overline{y}] \\ [x]-[y] = [\underline{x}-\overline{y}, \overline{x}-\underline{y}] \\ [x]*[y] = [\min(\underline{x}*\underline{y}, \underline{x}*\overline{y}, \overline{x}*\underline{y}, \overline{x}*\overline{y}), \max(\underline{x}*\underline{y}, \underline{x}*\overline{y}, \overline{x}*\underline{y}, \overline{x}*\overline{y})] \end{cases} \tag{3.5}$$

Il reste à caractériser la division pour les intervalles. Pour cela, on a recourt à des intervalles généralisés présentés dans la partie 1.2 qui suit.

1.2 Intervalles généralisés

La division telle qu'elle a été définie précédemment dans (3.4) ne permet pas de considérer le cas d'un intervalle diviseur contenant le réel 0 puisqu'on risque de faire des calculs indéterminés. Pour parer à ces éventualités, il convient d'introduire la notion d'intervalle généralisé ou étendu. Pour cela, IR est complété en lui ajoutant $+\infty$ et $-\infty$. En plus des intervalles de IIR**,** les nouveaux intervalles ont la forme :

$$\{[-\infty, r] \mid r \in \text{IR}\}, \{[r, +\infty] \mid r \in \text{IR}\}, \{[-\infty, +\infty]\}$$

On peut résumer dans le Tableau 3.1, l'arithmétique supplémentaire dans IR induite par le rajout de $\pm\infty$. Les opérations aboutissant à un résultat indéterminé ne posent pas de problème pour les intervalles généralisés. En effet, une opération impliquant une indétermination dans

le calcul des bornes résultantes donne toujours comme résultat $[-\infty, +\infty]$ (comme par exemple $[-\infty, r]+[r, +\infty]$).

	$\Diamond = +$	$\Diamond = -$	$\Diamond = *$
$+\infty\Diamond x$	$+\infty$	$+\infty$	$\text{Sign}(x)*\infty$
$+\infty\Diamond 0$			0
$-\infty\Diamond x$	$-\infty$	$-\infty$	$-\text{Sign}(x)*\infty$
$+\infty\Diamond 0$			0
$-\infty\Diamond-\infty$	$-\infty$	indéterminé	$+\infty$
$+\infty\Diamond+\infty$	$+\infty$	indéterminé	$+\infty$
$+\infty\Diamond-\infty$	indéterminé	indéterminé	$-\infty$

Tableau 3.1 : Arithmétique supplémentaire de $\text{IR} \cup \{+\infty, -\infty\}$ (avec $x \in \text{IR}^*$).

Ces nouveaux intervalles permettent de compléter la définition de la division notamment pour les cas où le diviseur contient 0. On s'intéresse d'abord à l'inverse d'un intervalle contenant 0. Le résultat de cette opération selon les cas envisageables peut être résumé par le formulaire suivant :

$$
\begin{aligned}
1/[y] &= \varnothing && \text{si } [y]=[0, 0] \\
&= [1/\overline{y}\,, 1/\underline{y}\,] && \text{si } 0 \notin [y] \\
&= [1/\overline{y}\,, \infty] && \text{si } \underline{y} = 0 \text{ et } \overline{y} > 0 \\
&= [-\infty, 1/\underline{y}\,] && \text{si } \underline{y} < 0 \text{ et } \overline{y} = 0 \\
&= [-\infty, \infty] && \text{si } \underline{y} < 0 \text{ et } \overline{y} > 0
\end{aligned}
$$

Par suite, la division de deux intervalles se fait par le calcul : $[x]/[y] = [x]*(1/[y])$, en utilisant l'extension de l'opération * à $\text{IR} \cup \{+\infty, -\infty\}$.

1.3 Vecteurs et matrices d'intervalles

Le terme pavé désigne un vecteur d'intervalles et l'ensemble de tous les pavés réels est noté $\mathbb{IR}^n$. Un pavé $[x]$ sera écrit : $[x] = [x_1] \times \ldots \times [x_n]$. Les extensions des opérations ensemblistes et binaires précédemment définies sont obtenues par une simple application des définitions précédentes composante par composante. Ainsi, pour un $\Diamond$ représentant un des signes ensemblistes $\{\cap, \cup\}$ ou un des signes binaires $\{+, -, *, /\}$, on peut écrire :

$$[z] = [x] \Diamond [y] = \begin{pmatrix} [x_1] \Diamond [y_1] \\ \ldots \\ [x_n] \Diamond [y_n] \end{pmatrix}$$

Après avoir défini les opérations binaires et ensemblistes sur les intervalles on s'intéresse, dans la suite, aux fonctions d'inclusion qui permettent de calculer, avec le moins de pessimisme possible, l'image intervalle d'un intervalle par une fonction.

1.4 Fonctions d'inclusion

1.4.1 Définition

Soit f une fonction réelle définie dans un domaine B de $\mathbb{IR}^n$ à valeurs dans $\mathbb{IR}^m$. L'image par f d'une variable pavé $[x] \subseteq$ B est par définition l'ensemble : $\{f(x) | x \in [x]\}$. Il apparaît que l'image par f d'un intervalle n'est pas toujours un intervalle. Ceci conduit à introduire les "*fonctions d'inclusion*" définies comme suit : une fonction d'inclusion pour f notée $[f]$ est une fonction qui vérifie pour tout pavé $[x]$ de $\mathbb{IR}^n$ les deux assertions suivantes :

- $[f]([x])$ est un intervalle
- $f([x]) \subseteq [f]([x])$

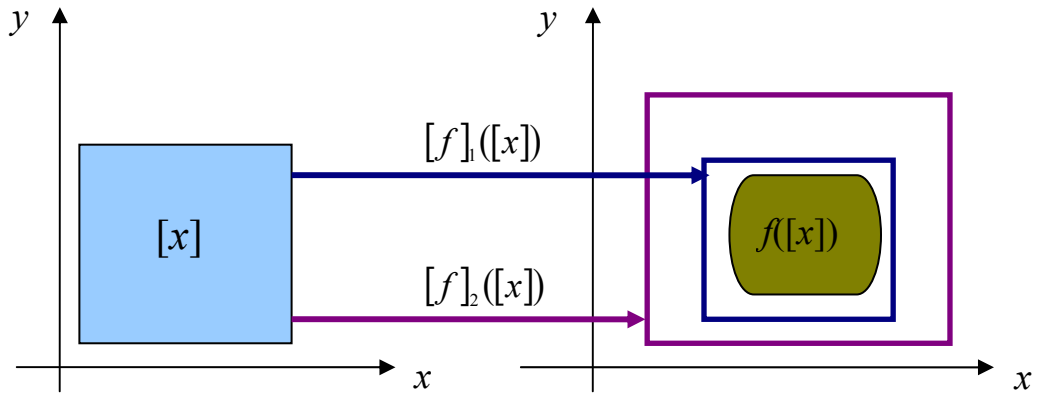


Figure 3.1 : exemple de deux fonctions d'inclusion $[f]_1$ et $[f]_2$ pour f.

Comme exemple de fonction d'inclusion toujours vérifiée pour toute fonction f de IIR^n dans IIR^m, on peut citer la fonction intervalle : $[f]([x])=[y]$ où $[y]=\underset{1\cdots m}{\times}[-\infty,+\infty]$, pour tout $[x]$ dans IIR^n. Pour une fonction f, on peut donc logiquement trouver une multitude de fonctions d'inclusion d'où un travail nécessaire d'optimisation adapté à la nature de la fonction. Plusieurs travaux se sont très tôt orientés vers la recherche de fonctions d'inclusions adaptées à chaque classe de fonctions étudiée. Ces fonctions peuvent être polynomiales [Malan et al., 1992] ou données par un algorithme [Moore, 1979], ou encore, solutions d'équations différentielles [Lohner, 1987].

Englober les ensembles images par des intervalles entraîne forcément un pessimisme comme l'illustre la Figure 3.2. La conséquence directe en est que, dans certains cas, une fonction d'inclusion mal choisie peut engendrer un grand pessimisme. Cependant il faut remarquer que certaines propriétés remarquables caractérisant f peuvent raisonnablement être conservées par de bonnes approximations $[f]$ (des propriétés telles que la positivité ou la non contenance de 0 de l'ensemble image).

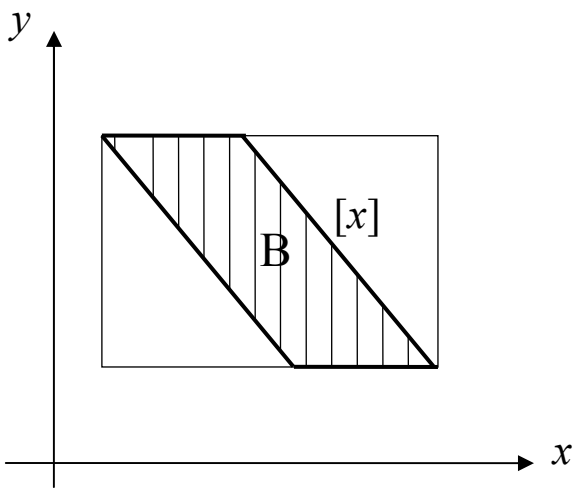


Figure 3.2 : pessimisme dû à l'approximation intervalle des ensembles : [x] est le plus petit intervalle contenant B.

Remarque 3.1

■Considérons une fonction f de IR^n dans IR^m, et $[f_j]$ (j = *1, ..., m*), m fonctions d'inclusion de IR^n dans IR associées aux différentes fonctions composantes f_j de f. Une fonction d'inclusion pour f est alors donnée par :

$$[f]([x]) = [f_1]([x]) \times \cdots \times [f_m]([x]) \tag{3.6}$$

La construction d'une fonction d'inclusion pour f est donc généralement ramenée à celle de chacune de ses fonctions coordonnées. Par suite, les propriétés et résultats énoncés ne concerneront que les fonctions d'inclusion réelles.■

1.4.2 Propriétés

- Une fonction d'inclusion $[f]$ de f sera dite *monotone* lorsque pour tout couple d'intervalles ([x], [y]) de IIR^n tels que : [x] $\subseteq$ [y] on a $[f]([x]) \subseteq [f]([y])$.
- $[f]$ est dite *minimale* si pour tout pavé [x], $[f]([x])$ est le plus petit pavé contenant f([x]). Cette fonction d'inclusion minimale, pour toute fonction f, est unique par définition et sera notée $[f]^*$.
- Pour toutes les fonctions continues (parmi lesquelles les fonctions élémentaires classiques exp, tan, sin, cos, …), on a comme conséquence directe du théorème des valeurs

intermédiaires, que l'image d'un intervalle reste un intervalle et donc par suite : $[f]^*([x]) = f([x])$.

- Si de plus f est monotone dans un domaine contenant l'intervalle $[x]$, $[f]^*([x])$ est alors obtenue par un simple calcul sur les bornes de $[x]$:

$$[f]^*([x]) = \left[min\left(f(\underline{x}), f(\overline{x})\right), max\left(f(\underline{x}), f(\overline{x})\right)\right] \tag{3.7}$$

Exemple 3.1

■ [arctan]([0, ∞]) = [arctan(0), arctan(∞)] = [0, π/2]

[exp]([0, 1]) = [exp(0), exp(1)] = [1, e]

[Λ²]([2, 3]) = [Λ²(2), Λ²(3)] = [4, 9] (Λ² est la fonction carrée)

[Λ²]([−3, −2]) = [Λ²(−2), Λ²(−3)] = [4, 9] ■

1.4.3 Fonctions d'inclusion naturelles

La première idée qui vient à l'esprit pour construire une fonction d'inclusion pour une fonction f: $IR^n \rightarrow IR$ est de faire appel à des méthodes d'optimisation pour déterminer le plus petit élément et le plus grand élément dans $f([x])$, lorsque chaque x_i varie dans $[x_i]$. Il serait alors possible d'approcher la fonction d'inclusion minimale de f. Cependant, ces méthodes d'optimisation ne sont pas évidentes à faire tourner en général et peuvent s'avérer être coûteuses selon la nature de la fonction à étudier. Lorsque f est composée d'opérateur arithmétique (+, −, *, /, …) et de fonctions élémentaires classiques (sin, cos, exp, …) dont on connaît au préalable des fonctions d'inclusion minimales, une fonction d'inclusion pour f s'obtient en remplaçant chaque sous variable x_i, chaque opérateur ou chaque fonction par son représentant intervalle. La fonction d'inclusion ainsi obtenue est appelée la fonction d'inclusion naturelle de f. Cette fonction naturelle est minimale à condition que tous les opérateurs et fonctions élémentaires de f soient continus, et à condition que toutes les sous-variables x_i n'apparaissent qu'une fois dans l'expression de f. Ce résultat vient d'un théorème connu, démontré dans [Moore, 1979].

Exemple 3.2

- Considérons la fonction f réelle définie dans IR^2 par : $f(x_1,x_2) = \ln(\exp(x_1)+\cos(x_2))$. Sa fonction naturelle est donnée par : $[f]([x_1],[x_2]) = [\ln]([\exp]([x_1])+[\cos]([x_2]))$.

  Pour $[x_1] = [0, 1]$, $[x_2] = [0, \pi/6]$: $f([x_1],[x_2]) = \ln(\exp([0, 1])+\cos([0, \pi/6])$
  $= \ln([1, e]+[1/2, 1]) = \ln([3/2, 1+e])$
  $= [\ln(3/2), \ln(1+e)]$; on obtient l'image exacte de $[x]$

Exemple 3.3

- Considérons les expressions suivantes toutes représentatives d'une même fonction f :

  $f_1(x) = x^2+4x$; $f_2(x) = xx+4x$; $f_3(x) = (x+2)^2 -4$.

  Les évaluations des fonctions naturelles pour ces différentes écritures de f et pour $[x_0] = [-1, 1]$ donnent :

  $[f_1]([x_0]) = [x_0]^2+4[x_0] = [0, 1]+[- 4, 4] = [-4, 5]$

  $[f_2]([x_0]) = [x_0][x_0]+4[x_0] = [-1, 1]+[-4, 4] = [-5, 5]$

  $[f_3]([x_0]) = ([x_0]+2)^2 -4 = [1, 3]^2 - 4 = [-3, 5]$

  Dans cet exemple l'occurrence de la variable $[x_0]$ dépend de l'expression de f. Puisque f_3 est continue et que x n'apparaît qu'une seule fois dans f_3, $[f_3]$ est alors minimale. ■

L'utilisation des fonctions d'inclusion naturelles n'est cependant pas toujours recommandée. Leur optimalisation est fortement liée à l'occurrence des variables en jeu, ce qui est, en pratique, difficile à réduire. Un important champ d'investigation dans le domaine de l'analyse par intervalle a justement été la construction d'autres types de fonctions d'inclusion moins pessimistes dans certains cas précis [Ratschek et al, 1984]. En ce qui concerne ce manuscrit, le choix d'utiliser des méthodes de propagation de contraintes (voir section 2) fait que l'utilisation des fonctions d'inclusion naturelles uniquement suffit pour élaborer les algorithmes.

Dans la deuxième partie de ce chapitre, on propose de rappeler la formulation générale qui permet d'aborder un problème de recherche de variables appartenant à des pavés initiaux et vérifiant un ensemble de contraintes. Puis, nous ferons des rappels sur la méthode de propagation de contraintes que nous avons choisie pour traiter le problème de localisation.

2 PROBLEME DE SATISFACTION DE CONTRAINTES

2.1 Introduction

A l'origine, les problèmes de satisfaction de contraintes ont été définis pour des domaines discrets, i.e. la valeur prise par chaque x_i appartient à un ensemble fini [Clowes, 1971 ; Mackworth, 1977 ; Freuder, 1978). Récemment, des techniques utilisées dans cadre de CSP discrets ont été étendues à des ensembles continus de IR dont les intervalles (Cleary, 1987 ; Davis, 1987 ; Hyvönen, 1992 ; Lhomme, 1993 ; Benhamou et al., 1994 ; Sam-Haroud, 1995 ; Jaulin et al., 2001a).

Classiquement, un problème de satisfaction de contraintes (qui est noté habituellement CSP pour '**C**onstraint **S**atisfaction **P**roblem') est défini par un triplet (x, D, C) : on recherche l'ensemble des vecteurs de variables x, dans un domaine initial D, vérifiant un ensemble de contraintes C. En ce qui concerne les CSP "intervalle", le triplet (x, D, C) est habituellement écrit sous la forme suivante :

- $x = (x_1, x_2, ..., x_n)$ est un vecteur de n variables $x_i \in$ IR,
- D est un pavé noté $[x]$,
- et le système de contraintes C est noté F et a la forme : $f_j(x_1, x_2, ..., x_n) = 0, j \in \{1, \ldots, m\}$

En définitive, un problème de satisfaction de contrainte $\mathcal{H}$ s'écrit sous la forme :

$$\mathcal{H} : (\mathrm{F}(x)=0 \mid x \in [x]) \tag{3.8}$$

Pour la suite, on présente plusieurs notions qui sont habituellement associées à la formulation des CSP telles que la *consistance*, la *contraction*, les *contracteurs* et les *résolveurs*.

2.2 Définition de la contraction

Résoudre un CSP continu $\mathcal{H}$ revient à englober avec le moins de pessimisme possible les solutions du CSP. Pour atteindre cet objectif, des *contractions* successives du domaine initial permettent d'englober avec le moins de pessimisme l'ensemble des solutions de $\mathcal{H}$. On définit l'ensemble solution de $\mathcal{H}$ comme

$$S = \left\{ \tilde{x} \in [x] \middle| \, \mathrm{F}(\tilde{x}) = 0 \right\} \tag{3.9}$$

Contracter $\mathcal{H}$ signifie remplacer [x] par un domaine intervalle plus petit [x'] tel que le domaine solution reste toujours englobé, i.e. S $\subset$ [x]' $\subset$ [x]. Il existe une contraction optimale de $\mathcal{H}$ atteinte pour un [x]*. Cette contraction représente le plus petit intervalle englobant l'ensemble solution S.

2.3 consistance

2.3.1 Consistance d'un point

Un scalaire $\tilde{x}_i$ (appartenant à la $i^{ème}$ composante de [x]) est *globalement consistant* avec $\mathcal{H}$ s'il est possible de trouver un vecteur $\tilde{x}$ dans S l'ayant comme $i^{ème}$ coordonnée i.e.

$$\forall k \in \{1,\cdots,n\} - i, \exists \tilde{x}_k \in [x_k] \, f(\tilde{x}_1,\cdots,\tilde{x}_n) = 0 \tag{3.10}$$

En pratique, il n'est pas toujours possible de trouver les points globalement consistants avec toutes les contraintes prises à la fois dans $\mathcal{H}$. Il n'est alors possible d'atteindre que la *consistance locale*. Un scalaire x_i appartenant à la $i^{ème}$ composante de [x] est localement consistant avec $\mathcal{H}$ si pour chaque contrainte f_i (prise séparément), il est possible de trouver un vecteur $\tilde{x}$ consistant avec f_i l'ayant comme $i^{\text{ème}}$ coordonnée i.e.

$$\forall f_i, \forall k \in \{1,\cdots,n\} - i, \exists \tilde{x}_k \in [x_k] \, f(\tilde{x}_1,\cdots,\tilde{x}_n) = 0 \tag{3.11}$$

Il découle immédiatement de la définition de la consistance locale, que la consistance globale entraîne celle locale. L'exemple de la Figure 3.3 représente, en dimension 2, deux contraintes sous forme d'ellipse. Les deux points représentés sont localement consistants mais seul le point le plus à droite est globalement consistant.

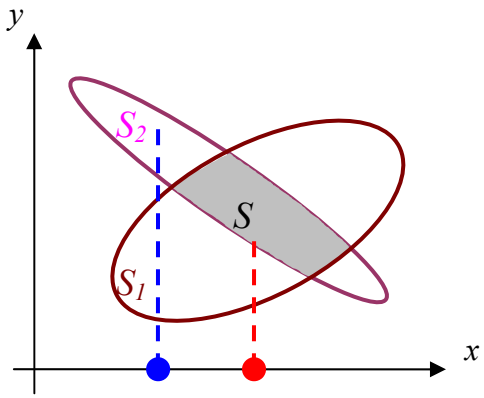


Figure 3.3 : illustration de la consistance globale (point à droite) et de la consistance locale (point à gauche).

2.3.2 Consistance d'un intervalle

Un intervalle est dit globalement (resp. localement) consistant, s'il est le plus grand intervalle englobant tous les points dans cet intervalle qui sont globalement (resp. localement) consistants. La consistance globale d'un intervalle entraîne la consistance locale de cet intervalle. Cette dernière est celle qui est généralement atteinte par les algorithmes du fait de la difficulté de considérer toutes les contraintes à la fois pour des systèmes complexes. Dans l'exemple de la Figure 3.4, avec un intervalle globalement consistant (à gauche) et un intervalle localement consistant (à droite), on met en évidence le pessimisme qui apparaît lorsque l'on considère des contraintes une par une.

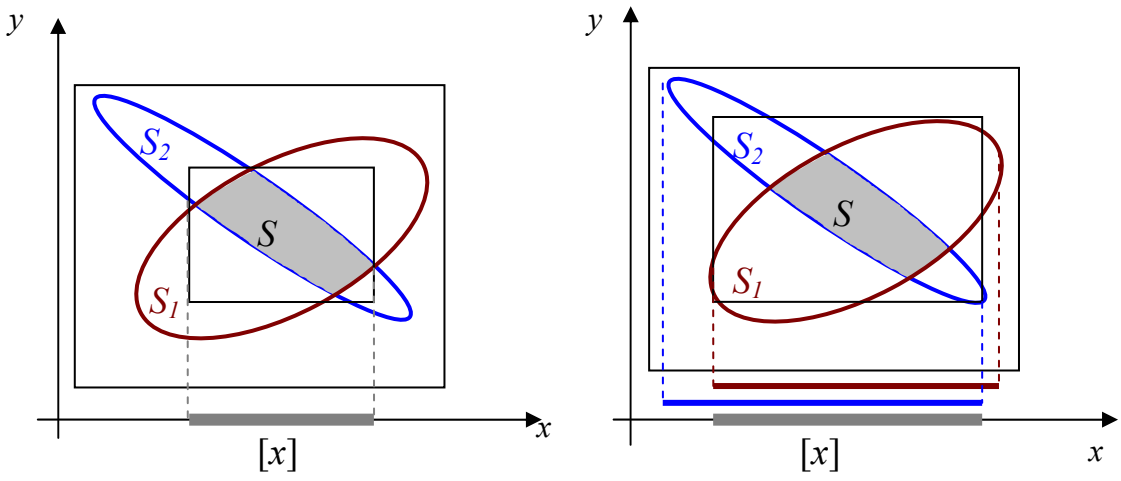


Figure 3.4 : consistance globale d'un intervalle (à gauche) et consistance locale d'un intervalle (à droite)

2.4 Contracteurs et résolveurs

Un contracteur pour $\mathcal{H}$ est par définition toute opération qui peut être utilisée pour le contracter. Un résolveur pour un CSP est un contracteur qui permet d'obtenir le pavé solution.

Diverses techniques ont été utilisées pour élaborer des contracteurs [Neumaier, 1990 ; Sam-Haroud, 1995 ; Jaulin et al, 2001a]. Dans la suite de ce chapitre, on parlera exclusivement de la propagation de contraintes qui est la méthode utilisée au cours de cette thèse. Le lecteur pourra se rapporter à l'annexe (partie A) pour des éléments supplémentaires sur les contracteurs existants. Les caractéristiques des méthodes de propagation de contraintes qui ont guidé notre choix sont :

- un temps de calcul raisonnable et une facilité de mise en oeuvre
- l'indépendance de la méthode vis-à-vis de la longueur et/ou de la taille des intervalles,
- l'indépendance de la méthode vis-à-vis de la linéarité ou non des équations.

2.5 Propagation de contraintes ou algorithme de Waltz

Les méthodes de propagation de contraintes ont été introduites pour la première fois par Waltz dans [Waltz, 1975]. Plusieurs études sur la convergence de l'algorithme ont été faites notamment dans [Cleary, 1987; Davis1987].

2.5.1 Principe de l'algorithme de Waltz

La particularité de ce contracteur est de prendre en compte une à une les *m* contraintes $f_j(x)=0$. En supposant que chaque contrainte f_j peut être décomposée en une séquence d'opérateurs et de fonctions élémentaires (telles que : +, –, /, ∗, sin, cos…), il est alors possible de décomposer chaque contrainte en un ensemble de *contraintes primitives* (Lhomme, 1993). Une contrainte primitive étant une contrainte composée au plus d'une fonction élémentaire ou d'un opérateur élémentaire.

Le principe de l'algorithme de Waltz est de contracter $\mathcal{H}$ successivement avec chacune des contraintes f_j prises une à une, sans ordre a priori jusqu'à ce qu'aucune ne contracte plus ou plus assez selon un seuil. La mise en œuvre d'un tel algorithme a l'avantage d'être simple et facilement automatisable. Cette idée de propager des contractions est à l'origine de

l'appellation "propagation de contraintes". Un algorithme de propagation de contraintes original a été défini dans [Jaulin et al., 2001b]. Pour un CSP dont le graphe de représentation est un arbre, il existe un agencement des contraintes qui permet de se passer d'une boucle "tant que". En effet, les contractions successivement suivant l'ordre des feuilles vers les racines puis des racines vers les feuilles est optimal. Cet agencement de contraintes est appelé dans la littérature *FALL-CLIMB*.

Exemple 3.4 : décomposition en contraintes primitives

■Considérons la contrainte $x_1\log(x_2)+\cos(x_3)=0$. On peut la décomposer en contraintes primitives de la manière suivante :

$$\begin{cases} a_1 = \log(x_2) \\ a_2 = x_1.a_1 \\ a_3 = \cos(x_3) \\ a_2 + a_3 = 0 \end{cases} \tag{3.12}$$

Les nouveaux domaines associés aux nouvelles variables (ici a_1, a_2 et a_3) sont initialisés à $[-\infty, \infty]$.■

Exemple 3.5 : Algorithme de Waltz

Reconsidérons la contrainte $x_1\ln(x_2)+\cos(x_3)=0$ décomposée comme indiqué dans 3.12, et la contrainte $x_1+3x_2=0$. On peut écrire un algorithme de Waltz selon la succession de contraintes suivante :

Tant qu'une variable contracte de plus de ε

Faire

1	$[a_1]=[a_1]\cap\log([x_2])$	2	$[a_2]=[a_2]\cap([x_1]*[a_1])$	3	$[a_3]=[a_3]\cap\cos([x_3])$
4	$[a_3]=[a_3]\cap-[a_2]$	5	$[x_1]=[x_1]\cap-3[x_2]$	6	$[a_2]=[a_2]\cap-[a_3]$
7	$[x_3]=[x_3]\cap\cos^{-1}([a_3])$	8	$[a_1]=[a_1]\cap([a_2]/[x_1])$	9	$[x_1]=[x_1]\cap([a_2]/[a_1])$
10	$[x_2]=[x_2]\cap\exp([a_1])$	11	$[x_2]=[x_2]\cap-[x_3]/3.$		

Fin

2.5.2 Interprétation géométrique

Considérons les deux contraintes : $y=x+a$ appelée 1^{ière} contrainte et $y=b.x$ appelé 2^{ième} contrainte où $[x] = [-8, 8]$, $[y] = [-8, 8]$, $[a] = [1, 1]$, $[b] = [4, 4]$. Une interprétation géométrique de la contraction selon l'algorithme de Waltz est illustrée par les figures de 3.5 à 3.10.

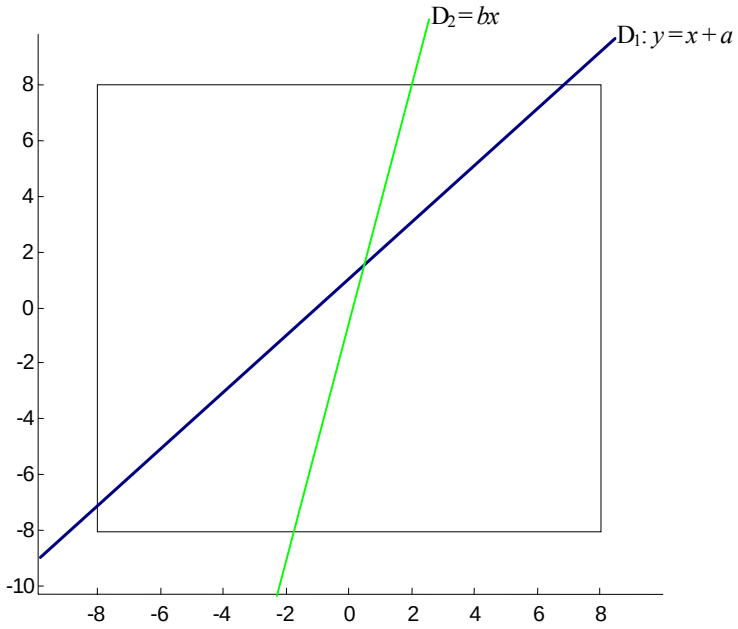


Figure 3.5 : le domaine $[x] \times [y]$ est représenté par un carré et les contraintes sont représentées par les deux droites D_1 et D_2.

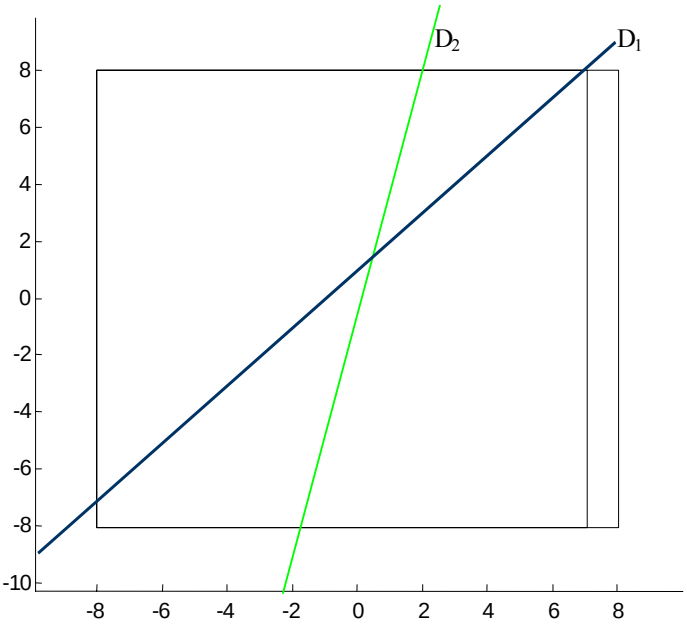


Figure 3.6 : on contracte $[x]$ selon la 1^{ière} contrainte : on projette D_1 selon x.

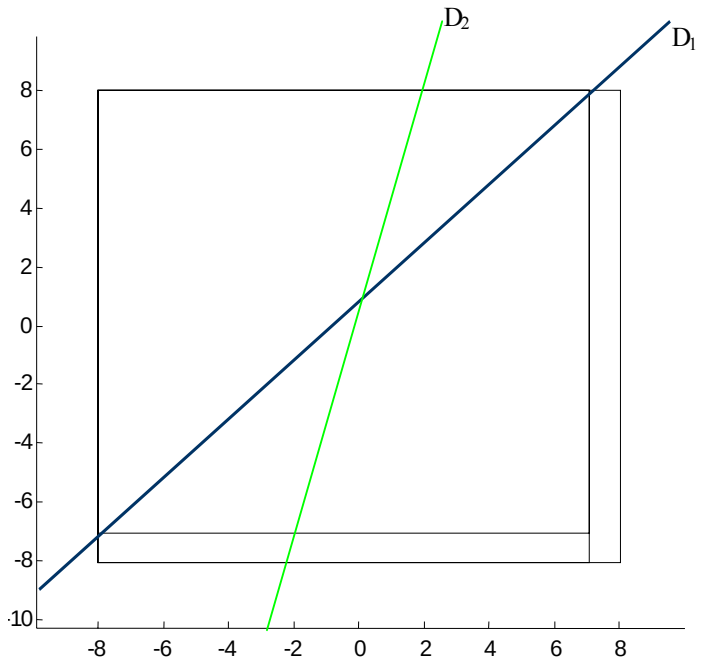


Figure 3.7 : on contracte [y] selon la 1^{ière} contrainte : on projette D_1 selon y.

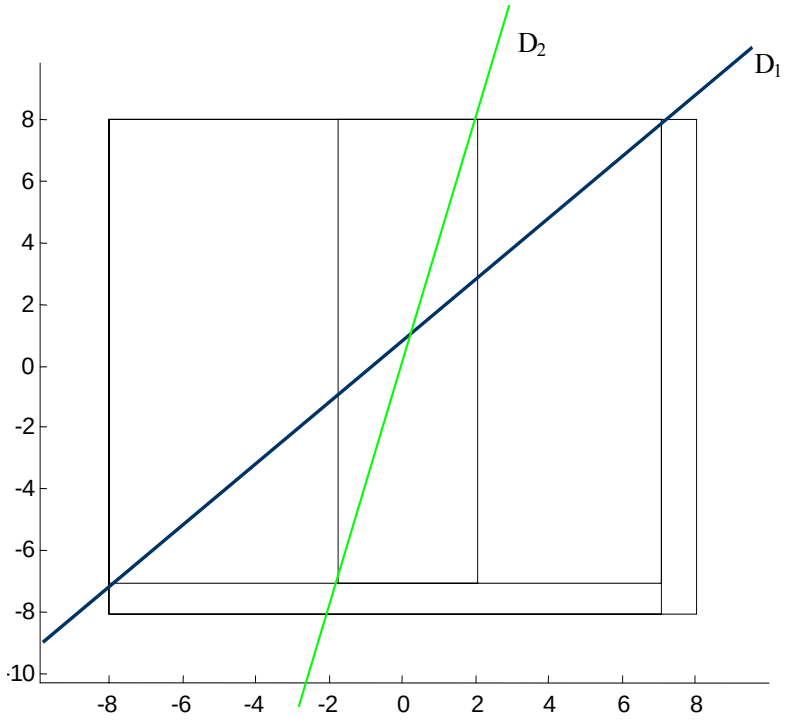


Figure 3.8 : on contracte [x] selon la 2^{ième} contrainte : on projette D_2 selon x.

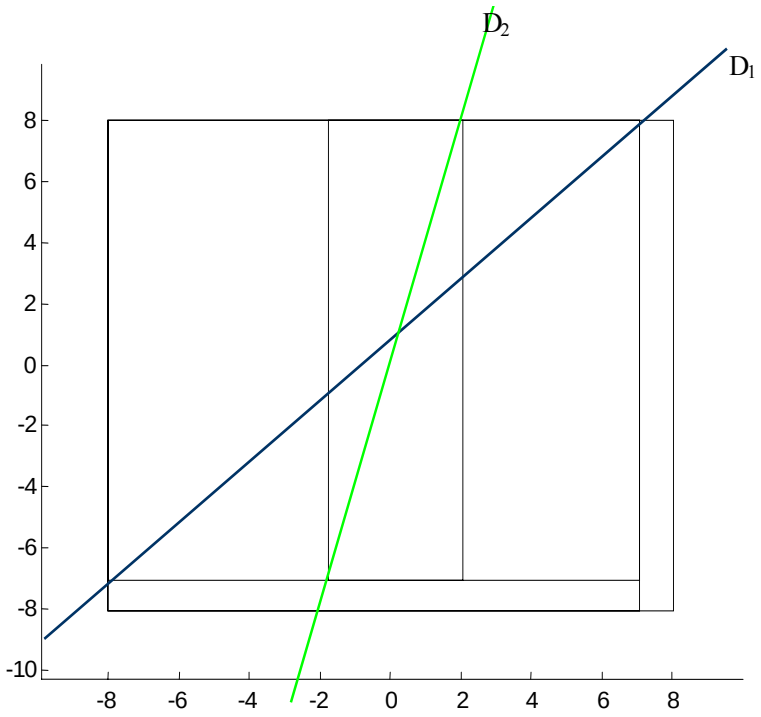


Figure 3.9 : on contracte [y] selon la 2^{ième} contrainte : on projète D_2 selon y (pas d'évolution).

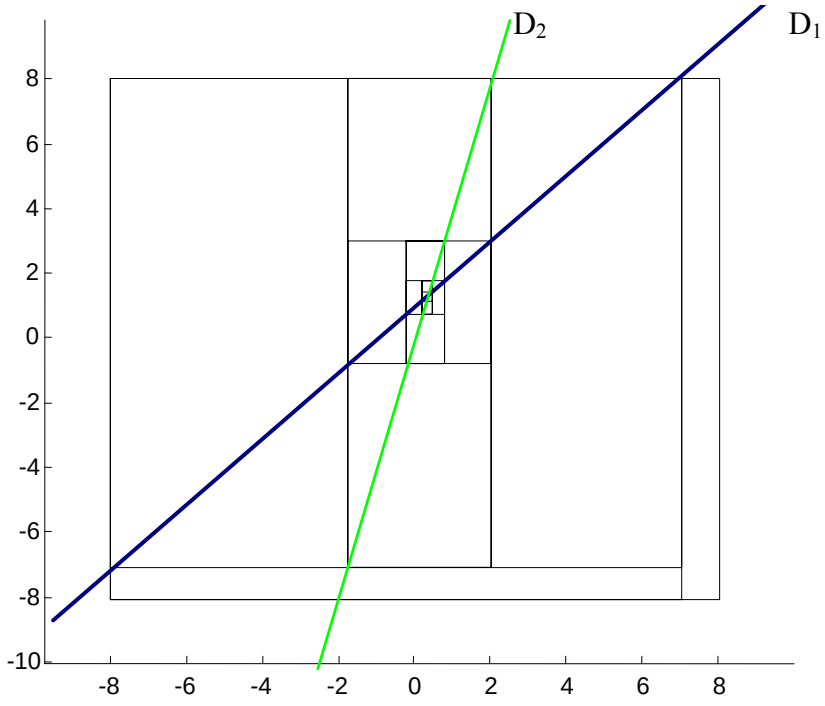


Figure 3.10 : résultat final de la boucle de l'algorithme de Waltz sans seuil de contraction imposé (10 tours).

2.5.3 Points forts de l'algorithme de Waltz

Les points forts de cette méthode sont la simplicité de sa mise en œuvre et son efficacité devant des problèmes présentant beaucoup de redondance donc beaucoup de contraintes. L'algorithme de Waltz est donc bien adapté aux problèmes de fusion de données souvent caractérisés par leur redondance de données. De surcroît, contrairement aux autres méthodes de contraction, les méthodes de propagation de contraintes sont indépendantes de la non linéarité et de la taille des intervalles.

2.5.4 Points faibles de l'algorithme de Waltz

De par son principe, l'algorithme de Waltz a la mauvaise propriété d'avoir un temps de calcul non déterminable a priori. En effet, la boucle intervenant dans l'algorithme lui confère l'impossibilité de majorer son temps de calcul notamment pour des cas présentant des cycles[1] au niveau des contraintes. Il faut y ajouter que la consistance globale n'est atteinte de façon garantie que dans des cas particuliers où il n'y a pas de cycle. La consistance locale présente deux grands désavantages. Tout d'abord, elle peut engendrer un grand pessimisme qu'il est impossible de détecter et d'évaluer avec un algorithme de Waltz seul. Ensuite, la consistance locale ne signifie pas nécessairement la présence de solutions ! (voir la Figure 3.11). Ce dernier point est particulièrement handicapant pour des problèmes où on cherche à détecter des mesures incohérentes puisque des mesures, bien qu'incohérentes entre elles, peuvent être localement consistantes.

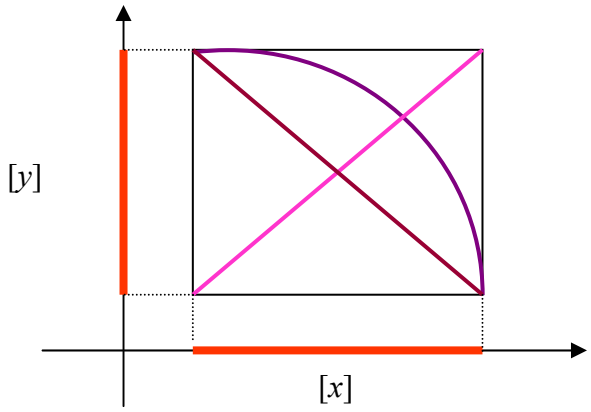


Figure 3.11 : consistance locale avec 3 contraintes pour un problème sans solution !

[1] Une étude complète des cycles sera faite dans le chapitre 5.

3 FUSION DE DONNEES DYNAMIQUE GARANTIE AVEC L'ALGORITHME DE WALTZ

Dans cette partie, une méthodologie globale basée sur la propagation de contraintes est donnée pour résoudre un problème de fusion de données caractérisé par une représentation d'état.

3.1 Modèle global pour la fusion de données dynamique

La fusion de données est résolue comme un problème d'observation d'état avec des mesures redondantes : certaines de ces dernières apparaissent dans le modèle d'évolution à l'intérieur du vecteur d'entrée u_k et les autres apparaissent dans les équations d'observation à l'intérieur du vecteur y_k. Ainsi, pour chaque instant *k*, on essaye de reconstituer x_k sachant x_{k-1} et les mesures courantes donnant u_k et y_k. On suppose que toutes les mesures sont bornées et qu'aucun capteur n'est défaillant. Le système stochastique qui décrit ce processus est donné par les équations suivantes (cas d'un système à entrée mesurée éq. (2.4) du chapitre2) :

$$\begin{cases} x_k = f(x_{k-1}, u_k, p_k, \alpha_k, \gamma_k) \\ y_k = g(x_k, p_k, \beta_k) \end{cases} \tag{3.1}$$

où :

- *k* est un entier représentant la discrétisation du temps (le temps continu 't_k' est relié à '*k*' par la relation $t_k = k\delta + t_0$ où "δ" représente la période d'échantillonnage et "t_0" l'instant initial
- $u_k \in IR^q$ représente le vecteur d'entrée du système à l'instant *k,*
- $y_k \in IR^m$ représente le vecteur des sorties du système à l'instant *k,*
- $x_k \in IR^n$ représente le vecteur d'état du système à l'instant *k,*
- $p_k \in IR^p$ est le vecteur des paramètres constants du modèle,
- $\alpha_k \in IR^n$ représente les bruits de modélisation du système, on suppose souvent qu'il affecte les paramètres p_k,

- $\gamma_k \in IR^q$ représente les bruits perturbant l'entrée u_k,
- $\beta_k \in IR^m$ représente les bruits de mesure perturbant la sortie y_k.

3.2 Formulation du problème sous forme d'un CSP

Le système (3.1) peut être vu à chaque instant comme un CSP $\mathcal{H}_k$: ($F(x) = 0 \mid x \in [x]$) où :

$$x = (x_k, \cdots, x_0, u_k, \cdots, u_0, y_k, \cdots, y_0, p_k, \cdots, p_0) \text{ et}$$

$$\mathrm{F} : x \to \mathrm{F}(x) = (x_k - f(x_{k-1}, u_k, p_k), \cdots, x_1 - f(x_0, u_1, p_1), y_k - g(x_k, p_k), \cdots, y_1 - g(x_1, p_1))$$

Les bruits α_i, γ_i et β_i n'apparaissent pas explicitement dans $\mathcal{H}_k$ puisqu'ils sont intégrés dans le choix des bornes des intervalles $[x_i]$, $[u_i]$ et $[y_i]$, sous l'hypothèse qu'ils sont additifs. Par exemple, si les bornes de β_k sont $\underline{b}$ et $\overline{b} \in IR^m$, et pour $y_{k,m}$ le vecteur de mesure à l'instant k on peut écrire:

$$y_k \in [y_{k,m} - \underline{b}, y_{k,m} + \overline{b}] \tag{3.2}$$

Il faut aussi remarquer que les paramètres p_k ne sont généralement connus qu'approximativement et sont donc traités sous forme d'intervalles pouvant donc être contractés au cours du temps.

Pour le CSP $\mathcal{H}_k$ à résoudre, la variable x inclut toutes les variables (états, mesures et paramètres) du système (3.1) de l'instant *0* à k. Cependant, en vue d'une implémentation temps réel, il est irréaliste de considérer toutes les équations de l'instant 0 à k. Ainsi, on considère plutôt une fenêtre limitée de temps (voir Figure 3.12), noté "h" pour horizon, représentant un nombre d'indices (de k à k–h+1). L'horizon h traduit le choix de n'utiliser qu'un certain nombre de relations récentes dans le temps. Considérer un horizon égal à h signifie que $\mathcal{H}_k$ a la nouvelle expression suivante :

$(F(x) = 0 | x \in [x])$ où :

- $x = (x_k, \cdots, x_{k-h+1}, u_k, \cdots, u_{k-h+1}, y_k, \cdots, y_{k-h+1}, p_k, \cdots, p_{k-h+1})$,
- $F : x \rightarrow F(x) = (x_k - f(x_{k-1}, u_k, p_k), \cdots, x_{k-h+2} - f(x_{k-h+1}, u_{k-h+2}, p_{k-h+2}), y_k - g(y_{k-1}, p_k), \cdots, y_{k-h+2} - g(y_{k-h+1}, p_{k-h+2}))$

On peut citer deux propriétés évidentes sur la notion d'horizon ainsi définie :

- h est toujours ≥2 puisque h =1 revient à ne pas considérer d'équation.
- L'ensemble des solutions du CSP correspondant à un horizon de h+1 est inclus dans l'ensemble des solutions du CSP correspondant à un horizon de h. La justification en est que les équations présentes dans le CSP d'horizon h sont incluses dans celles présentes dans le CSP d'horizon h+1. On peut déjà en tirer la première conclusion que plus l'horizon est grand, meilleures seront les contractions.

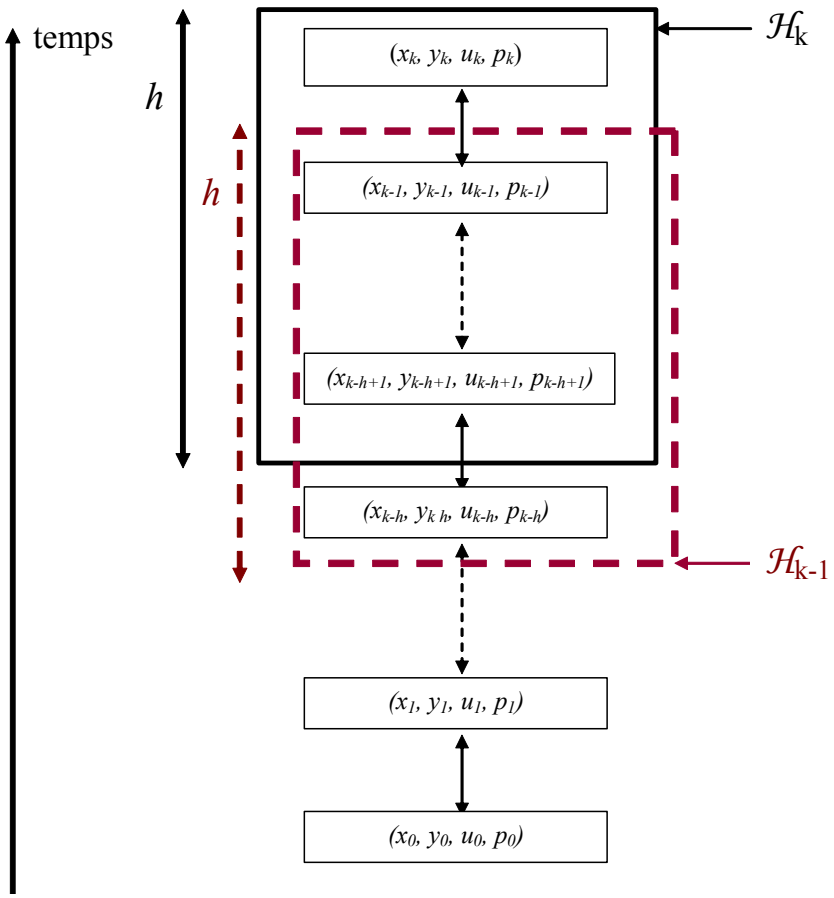


Figure 3.12 : illustration graphique d'un horizon limité pour chaque CSP $\mathcal{H}_k$.

Le problème de localisation dynamique revient donc à résoudre, à chaque instant k, le CSP $\mathcal{H}_k$. Une première limitation pourrait apparaître du fait de la présence de cycles[2]. Dans ce cas, l'algorithme de Waltz auquel on a recours ne serait pas en mesure de résoudre de façon globale le système de contraintes. A ce niveau intervient alors la redondance de capteurs (pour certaines classes de fusion de données) qui est susceptible de diminuer le pessimisme engendré par la consistance locale.

Dans le Tableau 3.2, on présente une forme générale de l'algorithme de Waltz utilisé pour résoudre le CSP $\mathcal{H}_k$ à chaque instant k.

Algorithme Waltz (Input /Output : $x = (x_k, \cdots, x_{k-h+1}, u_k, \cdots, u_{k-h+1}, y_k, \cdots, y_{k-h+1}, p_k, \cdots, p_{k-h+1})$

% **étape d'initialisation**

Choix de h (horizon) et η (seuil de contraction)

Lire l'entrée u_k et son erreur, en déduire $[u_k]$

Lire la sortie y_k et son erreur, en déduire $[y_k]$

$x_k = ([-\infty, \infty])_{i=1\ldots q}$ % le vecteur x_k n'est pas mesuré et est initialisé à IR^q

$[p_k] = [p_{k-1}]$ % les paramètres sont supposés constants.

%contraction

Tant qu'une des variables au moins "est contractée de plus de η"

pour $i=k-h+1$ à k

contracter $\mathcal{H}_k$ avec $\{x_i - f(x_{i-1}, u_i, p_i)=0\}$ et $\{y_i - g(x_i, p_i)= 0\}$

Fin Pour

Fin Tant que

Tableau 3.2 : algorithme de Waltz pour la résolution d'un problème de fusion de données dynamique.

Dans la partie suivante, cette formulation générale d'un problème de fusion de donnée est appliquée au problème de localisation introduit dans le chapitre 1.

[2] Une étude détaillée de la notion de cycle est faite au chapitre 5.

4 APPLICATION A LA LOCALISATION GARANTIE D'UNE AUTOMOBILE

Dans cette section, on présentera d'abord l'architecture du module de localisation puis on montrera les modèles d'évolution utilisés. Ensuite, on utilisera la formulation développée précédemment dans la section 3.2. Et enfin, on montrera les résultats obtenus sur des données réelles.

4.1 Architecture globale du problème de localisation

La finalité pratique du module de localisation est de fournir à tout instant une estimation de la pose du véhicule qui est ici le vecteur état $X_k = (x_k, y_k, \theta_k)$. Le vecteur (x_k, y_k) représente la position du véhicule à l'instant t_k dans le référentiel terrestre global. θ_k représente le cap du véhicule. L'origine du repère du véhicule est choisie au milieu des roues arrières (voire Figure 3.14). Pour plus de simplicité et de clarté, mais aussi par souci de modularité, nous avons considéré deux niveaux de fusion comme il est montré sur la Figure 3.13. Le déplacement élémentaire 'δ_s'et la rotation élémentaire 'δ_θ' sont obtenus au niveau d'une fusion statique qui utilise les mesures odométriques des capteurs ABS des deux roues arrières du véhicule ainsi que le déplacement d'angle de lacet fourni par un gyromètre à fibre optique. La redondance d'informations fournies par ces capteurs est susceptible de réduire sensiblement le pessimisme du pavé initial et de donner ainsi des pavés finaux $\delta_{\theta,k}$ et $\delta_{S,k}$ avec une bonne précision. Le résultat de cette fusion constitue l'entrée du module de fusion dynamique qui permet de localiser le véhicule et qui a pour sortie la pose (x_k, y_k, θ_k).

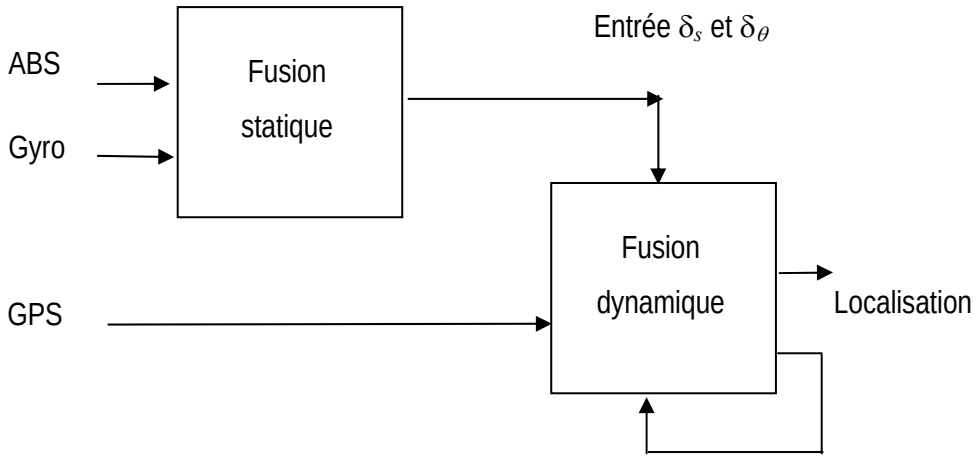


Figure 3.13 : architecture modulaire du système de localisation.

4.2 Modèles utilisés

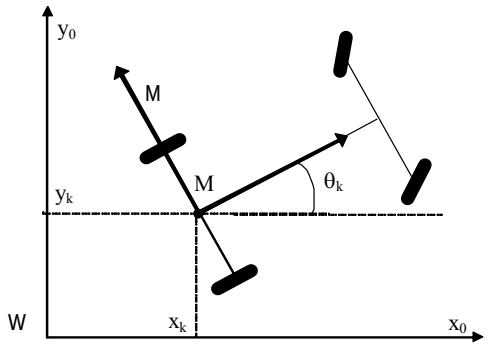


Figure 3.14 : définition du repère du véhicule.

Entre deux instants d'échantillonnage, les rotations élémentaires des deux roues arrières sont intégrées par les compteurs. Les valeurs obtenues permettent de calculer les distances parcourues entre deux échantillons par les roues arrières. Le déplacement élémentaire calculé au point M noté $\delta_{s,k}$ et la rotation élémentaire notée $\delta_{\theta,k}$ à l'instant k sont donnés par les équations suivantes :

$$\begin{cases} \delta_{s,k} = \dfrac{\delta_{RR} + \delta_{RL}}{2} \\ \delta_{\theta,k} = \dfrac{\delta_{RR} - \delta_{RL}}{e} \end{cases} \tag{3.3}$$

où :

- $\delta_{RL,k}$, $\delta_{RR,k}$ sont les déplacements élémentaires respectivement des roues arrières gauche et droite,
- $\delta_{s,k}$, $\delta_{\theta,k}$ sont les variables estimées,
- e est la distance entre les axes appelé voie.

De plus le gyromètre fournit la mesure $\delta_{\theta,\text{gyro}}$ qui est la rotation entre deux instants d'échantillonnage selon l'équation suivante :

$$\delta_{\theta,k} = \delta_{\theta,\text{gyro}} \tag{3.4}$$

Dans le module de fusion dynamique, la pose du véhicule X_k est calculée en fonction de X_{k-1} et des sorties $(\delta_{S,k}, \delta_{\theta,k})$ de la fusion statique. Le modèle d'évolution est le suivant :

$$\begin{cases} x_k = x_{k-1} + \delta_{s,k}.cos\left(\theta_{k-1} + \dfrac{\delta_{\theta,k}}{2}\right) \\ y_k = y_{k-1} + \delta_{s,k}.sin\left(\theta_{k-1} + \dfrac{\delta_{\theta,k}}{2}\right) \\ \theta_k = \theta_{k-1} + \delta_{\theta,k} \end{cases} \tag{3.5}$$

L'antenne du GPS a été installée au point M. Ainsi le modèle d'observateur est linéaire et les équations sont les suivantes :

$$\begin{cases} x_{gps} = x_k \\ y_{gps} = y_k \end{cases} \tag{3.6}$$

Où: x_{gps} et y_{gps} sont les mesures GPS.

4.3 Formulation des CSP à résoudre

4.3.1 Fusion statique

Pour le niveau statique, à chaque instant t_k, on doit résoudre le CSP $\mathcal{H}_k$: ($F(X) = 0 | X \in [X]$), où :

- $X = (\delta s, \delta_{RL}, \delta_{RR}, \delta_\theta, e)$
- F représente le système d'équations (3.3)
- les 2 roues arrières fournissent δ_{RL}, δ_{RR}. Pour les bornes des intervalles, on suppose que l'erreur de mesure sur la distance parcourue entre les deux instants t_{k-1} et t_k est inférieure à la distance correspondant à un top du compteur ABS. L'hypothèse prise est que le véhicule roule sans glisser :

$$[\delta_{roue}] = [\delta_{mes} - \delta_{ABS}, \delta_{mes} + \delta_{ABS}] \tag{3.7}$$

- $[\delta_\theta]$ est obtenu avec les mesures du gyromètre. Grâce à des tests spécifiques en statique, l'erreur maximale de cette erreur est $\delta_{\theta,gyr} \approx 1.10^{-4}$ radian (voir la Figure 3.15). En pratique la valeur utilisée est $\delta_{\theta,gyr} \approx 0.5.10^{-4}$ radian.

$$[\delta_\theta] = [\delta_{\theta,mes} - \delta_{\theta,gyr}, \delta_{\theta,mes} + \delta_{\theta,gyr}] \tag{3.8}$$

- $[\delta_S]$ n'est pas mesuré et est donc initialisé à une valeur inconnue : $[-\infty, +\infty]$
- $[e]$ est la voie du véhicule. Il est connu approximativement et peut être contracté au cours du temps.

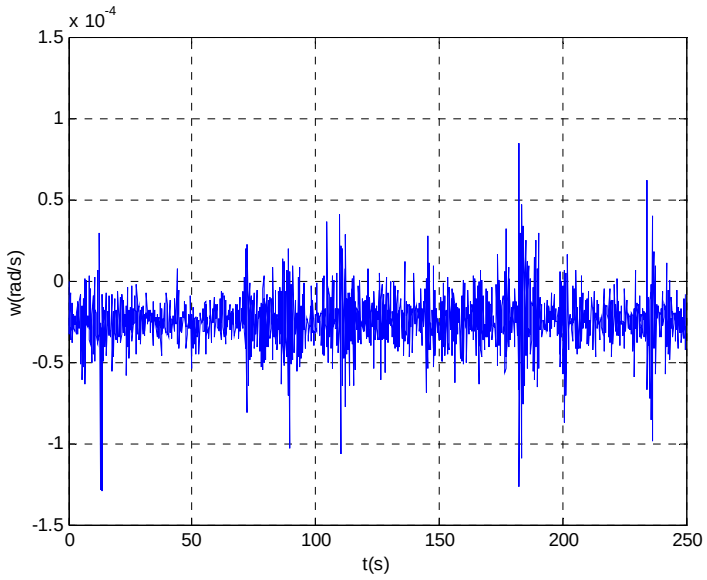


Figure 3.15 : essai en statique pour déterminer l'erreur du gyromètre.

Remarque :

■On peut noter que considérer l'équation (3.4) qui est une égalité entre $\delta_{\theta,k}$ et $\delta_{\theta,gyro}$ comme une contrainte est équivalent à initialiser l'intervalle $[\delta_{\theta,k}]$ par l'intervalle

correspondant à la mesure du gyromètre. Nous avons opté pour la deuxième façon de faire pour plus de simplicité. ■

Les estimations de $[\delta_{\theta,k}]$ et $[\delta_{s,k}]$ sont obtenues en résolvant à chaque instant le CSP $\mathcal{H}_k$. On peut remarquer qu'une originalité de la méthode est que toutes les variables du CSP peuvent être contractées. En particulier les mesures ainsi que les paramètres utilisés dans le modèle peuvent être contractés.

4.3.2 Fusion dynamique

Pour résoudre le problème de fusion dynamique, on applique l'algorithme donné dans la section 3.2. Pour chaque instant k, le CSP à résoudre avec un horizon égal à 'h' est $\mathcal{H}_k$: $(F(x)=0|x\in[x])$, où :

$$x= (x_k, \ldots, x_{k-h+1}, y_k, \ldots, y_{k-h+1}, \theta_k, \ldots, \theta_{k-h+1}, \delta_{S,k}, \ldots, \delta_{S,k-h+1}, \delta_{\theta,k}, \ldots, \delta_{\theta,k-h+1})$$

- $[\delta_{S,i}]$ et $[\delta_{\theta,i}]$ sont obtenus grâce à la fusion statique (voir la section 4.3.1 précédente)
- les mesures du GPS sont utilisées pour initialiser les intervalles $[x_k]$ et $[y_k]$. Les points longitude/latitude estimés à chaque instant sont convertis dans un repère cartésien local et les bornes des erreurs GPS sont obtenues grâce à la trame GST NMEA. En effet, les bornes des erreurs sont supposées égales à 3 fois l'estimation de l'écart type des positions $\hat{\sigma}$ fourni en temps réel par le récepteur GPS.

$$\begin{aligned}[x_{gps}] &= [x_{gps,mes} - 3\hat{\sigma}_x, x_{gps,mes} + 3\hat{\sigma}_x] \\ [y_{gps}] &= [y_{gps,mes} - 3\hat{\sigma}_y, y_{gps,mes} + 3\hat{\sigma}_y]\end{aligned} \tag{3.9}$$

- le cap $[\theta_i]$ est initialisé à $[-\infty, +\infty]$ puisqu'il n'est pas mesuré,
- F représente le modèle odométrique (3.5).

Remarques

- ■ Pour plus de simplicité et comme il a été fait pour la fusion statique, les contraintes correspondants à des égalités entre variables comme celles données par les mesures GPS ($x = x_{gps}$ et $y = y_{gps}$) sont traitées comme une simple initialisation.

- Le choix a été fait de négliger les problèmes de garanties numériques dans les choix des intervalles du fait de la tailles des erreurs effectives. ■

La résolution de ce CSP à chaque instant t_k donne une solution générale au problème de localisation en fusionnant GPS, gyromètre et odométrie.

4.4 Résultats Expérimentaux sur des données réelles

Figure 3.16 : le véhicule expérimental "STRADA".

4.4.1 Méthode de prise en compte des problèmes de synchronisation

Les travaux effectués au cours de cette thèse se sont orientés vers l'étude de la faisabilité des techniques de propagation de contraintes sur les intervalles sur des données réelles et en temps réel. Pour prototyper ces méthodes, il est nécessaire d'enregistrer des données pour pouvoir évaluer les algorithmes, ce qui correspond à faire du "post-traitement". Nous nous sommes alors orientés vers le post-traitement de données réelles, mono-machine, acquises grâce au véhicule expérimental STRADA de l'Heudiasyc (voir Figure 3.16). Comme annoncé en préambule dans le chapitre 1, les données ABS des roues arrières, les données d'un

gyromètre optique (KVH RD100) ainsi que les données d'un GPS différentiel (Trimble AG132 avec des corrections Omnistar) sont fusionnées pour résoudre le problème de localisation.

Les mesures des ABS et du gyromètre sont obtenues à une fréquence de 100 Hz. Le GPS AG132 peut fournir des fréquences respectivement de 1Hz, 5Hz, et 10Hz. Le choix a été fait de l'utiliser à une fréquence de 5hz pour avoir d'une part une fréquence assez élevée et, d'autre part, parce que la fréquence la plus élevée (celle à 10Hz) présente beaucoup d'irrégularités (on reçoit rarement 10 mesures par seconde). Le premier travail à effectuer consiste en une synchronisation des mesures acquises. On dispose de mesures du gyromètre et des ABS à 100Hz ainsi que celles du GPS à 5Hz et l'objectif que l'on se fixe est de créer une structure de données à 5Hz contenant toutes ces différentes mesures. La stratégie que l'on a utilisée consiste à dater[3] toutes le mesures pour pouvoir par la suite procéder à un rééchantillonnage par rapport à un temps de référence choisi. Ainsi, lors de l'acquisition, on adjoint à chaque mesure effectuée son temps local correspondant. La solution adoptée est alors de prendre comme temps de référence le signal PPS (Pulse Per Second). Ce dernier est émis toutes les secondes GPS avec une précision proche de quelques nanosecondes. Pour obtenir un temps PPS à 5 Hz, il suffit de diviser la période entre deux signaux PPS en 4. La principale motivation de l'utilisation du PPS est de supprimer la latence du GPS. En effet, les codes reçus pour une position donnée à une seconde exacte GPS servent d'entrée aux algorithmes du récepteur permettant de calculer les solutions de navigation. Par exemple pour l'AG132, le temps de latence est généralement inférieur à 200ms, ce qui correspond pour une vitesse moyenne de 60Km/h à une distance parcourue d'environ 3m ! En considérant le temps PPS qui représente l'instant exact correspondant à la mesure, on s'affranchit donc de la latence du GPS. Par la suite, pour les mesures du gyromètre et des ABS, on se limite à une simple recherche des temps les plus proches. La justification simple de ce choix est la haute fréquence de ces signaux qui, comparé à la fréquence de 5Hz, peuvent être assimilés à des signaux continus. De plus, pour plus de garanti, il nous est paru nécessaire de prendre en compte cette erreur de synchronisation en nous intéressant à la dynamique des capteurs. Ainsi les erreurs faites découlant de la synchronisation ont été majorées de la façon suivante :

[3] Le système d'acquisition date par des intervalles contenant toujours les dates exactes. Cependant, cette imprécision a été négligée dans cette thèse compte tenu de la taille des autres imprécisions.

- pour les ABS, on suppose qu'en considérant une vitesse moyenne de 100km/h et une erreur majorée de 10 ms (fréquence de 100Hz), l'erreur sur la distance parcourue mesurée est inférieure à 0.3m.
- pour le gyromètre, on suppose également qu'on tourne avec une vitesse angulaire n'excédant pas les 30°/s, ce qui correspond à une erreur sur l'angle mesurée inférieure à 0.3°.

En outre, les erreurs numériques ne sont délibérément pas prises en compte et sont négligées au regard des imprécisions dues à la datation et aux capteurs.

4.4.2 Résultats obtenus

Plusieurs acquisitions respectant le protocole défini dans la section 4.4.1 précédente ont été faites grâce au véhicule expérimental STRADA (voir Figure 3.17). La piste routière ayant servie pour les expérimentations est la piste du GIAT dans le cadre du projet ARCOS. Les données présentées ici correspondent à celles acquises le 28 avril 04. La Figure 3.18 montre la trajectoire du véhicule dans un repère local (dont le centre est le point de départ de l'acquisition) qui est une translation du repère Lambert. Cette trajectoire correspond à deux tours de pistes de durée approximative 10mn.

Figure 3.17 : vue sur le véhicule expérimental avec l'Ag132.

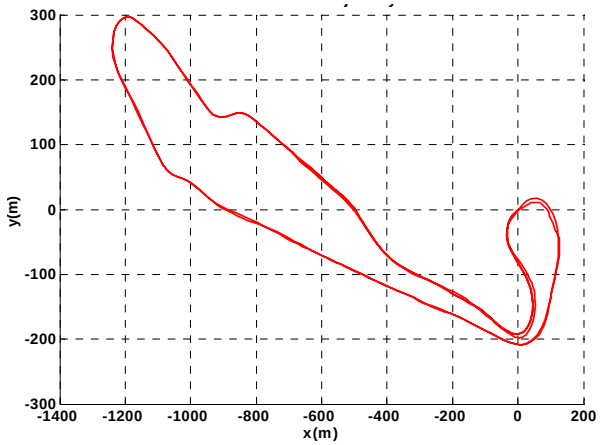


Figure 3.18 : vue de la trajectoire du véhicule dans un repère local.

Les conditions dans lesquelles l'essai s'est effectué peuvent être analysés en considérant la Figure 3.19 où sont représentées la vitesse et l'indice de qualité du GPS (trame GST). On peut observer d'une part que la vitesse moyenne de l'essai tourne autour 50 Km/h et qu'elle est limitée à 80 Km/h. De plus, la courbe de l'écart type de la latitude (trame GST) indique que les corrections différentielles du satellite géostationnaire Omnistar ont été perdues à trois reprises principalement à cause de la végétation. Dans ces cas de figure, l'AG132 propage une estimation de la correction durant 30s. Dépassé ce délai, il fonctionne en mode naturel et la précision peut significativement décroître. Il faut remarquer que ces pertes de DGPS peuvent aussi être détectée grâce à l'information de l'indice de qualité contenu dans la trame GST NMEA.

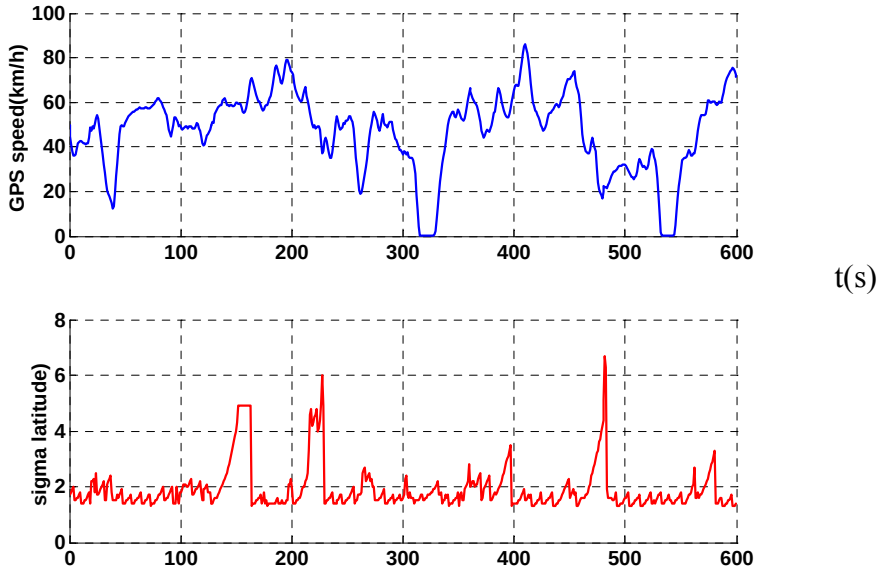


Figure 3.19 : vitesse et écart type de la latitude lors de l'essai

L'algorithme décrit sur le Tableau 3.2 a été appliqué pour différents horizons *h* variant de 0 à 20 en utilisant l'outil logiciel MATLAB. La mise en œuvre de l'algorithme de Waltz pour ce problème est simple mais nécessite la prise en compte de cas présentant des inconsistances. Bien que dans l'essai présenté dans la Figure 3.18 des cas d'inconsistances (intervalles vides) ne soient pas apparus, il nous est cependant paru nécessaire d'implémenter dans l'algorithme de Waltz de nombreux tests permettant de détecter les intervalles vides. La difficulté apparaît alors lorsqu'on se pose la question de gérer les inconsistances. En effet, du fait de la volonté de la méthode de toujours garantir les solutions, la présence d'inconsistances forcément due à des hypothèses fausses ou des capteurs défectueux remet en question toute la méthode. Cependant, les réflexions sur les stratégies possibles à adopter dans de tels cas de figure on menées aux stratégies suivantes :

- réinitialisation de l'algorithme : cette stratégie donne confiance aux informations capteurs immédiatement disponibles en pénalisant des informations issues d'étapes précédentes.
- augmentation des largeurs d'intervalles correspondant aux dernières mesures (stratégie par ailleurs adoptés dans [Jaulin, 2002]) : pour cette stratégie, on donne une grande confiance au pavé de l'instant précédent et on joue sur la largeur des intervalles provenant des mesures pour chercher une contraction optimale. Dans ce cas, soit on part d'une largeur surestimée des intervalles mesures que l'on fait décroître raisonnablement tant que la consistance est vérifiée, soit on agit, lorsqu'on rencontre des inconsistances, en gonflant les intervalles mesurés. Il faut signaler que, dans tous les cas, cette stratégie soulève de nouveaux problèmes de temps de calcul.
- détection des capteurs défectueux ou d'hypothèses de modèles non valables à l'instant courant : cette stratégie semble être la plus juste et logique par rapport au besoin de garantie. Cependant, elle est aussi la plus difficilement réalisable. L'une des difficultés majeure apparaît du fait de la consistance locale. En effet, la consistance locale ne révèle pas forcément d'emblée la présence d'inconsistances (voir Figure 3.11). Ce qui entraîne qu'une inconsistance peut théoriquement se propager un nombre de fois indéterminé avant d'être détectée sous forme d'intervalles vides. Ainsi, en cas d'apparition d'inconsistance, on n'est pas en mesure de déterminer l'instant jusqu'auquel il faudrait reculer pour chercher des mesures défectueuses.

Par rapport à la philosophie de la méthode, qui se pose comme une recherche garantie de toutes les solutions possibles, la dernière stratégie est apparue comme la plus logique. Ceci explique les orientations de la suite de ces expérimentations vers la détermination de nouveaux algorithmes qui pourraient permettre la détection des incohérences. Dans cette optique, dans le chapitre 5, nous proposerons une nouvelle approche vers des algorithmes temps réels permettant d'obtenir la consistance globale.

	Temps d'exécution pour un essai de10 mn	Pourcentage de contraction pour x	Pourcentage de contraction pour y
h=2	85.1s	6.1%	7.08%
h=3	173.4s	6.93%	8%
h=4	273.3s	7.73%	8.96%
h=5	377s	8.4%	9.8%
h=6	469.9s	8.86%	10.42%
h=7	589.1s	9.2%	10.9%
h=8	696s	9.5%	11.4%
h=9	724.8s	9.62%	11.85%
h=10	749.9s	9.74%	12.2%
h=11	750.1s	9.83%	12.4%
h=12	823.3s	9.88%	12.5%
h=13	887.7s	9.9%	12.52%
h=14	955.9s	9.9%	12.52%
h=15	1033s	9.9%	12.52%
h=16	1096.6s	9.9%	12.52%
h=17	1172.9s	9.9%	12.52%
h=18	1254.5s	9.9%	12.52%
h=19	1318.2s	9.9%	12.52%
h=20	1397.6s	9.9%	12.52%

Tableau 3.3 : résultats de l'algorithme de Waltz appliqué à différents horizons.

Les résultats rapportés sur le Tableau 3.2 ont été obtenus sur un Pentium 4 avec une vitesse de 1.8 GHz. Dans le Tableau 3.3, on représente les statistiques qui résument l'évolution du processus de contraction en fonction de l'horizon choisi. Pour un horizon

donné, on estime un temps de calcul correspondant aux 10mn d'essai. Ce temps est calculé sans qu'il soit fait l'effort d'optimiser l'algorithme. Sans compter que l'algorithme pour lequel ces temps ont été calculés contient également des tests supplémentaires permettant de détecter des inconsistances. Les temps de calcul donnés permettent cependant de faire deux constats :

- sans optimisation, le temps de calcul pour 10mn d'essais est inférieur à 10mn jusqu'à un horizon $h = 7$. Sans compter que pour le plus grand horizon choisi, qui est ici 20, le temps de calcul se situe autour de 20mn et reste encore raisonnable. Ceci nous conforte dans la faisabilité de la méthode en temps réel
- une analyse de l'évolution du temps de calcul en fonction de l'horizon révèle une certaine linéarité en fonction de l'horizon avec un facteur autour de 70.

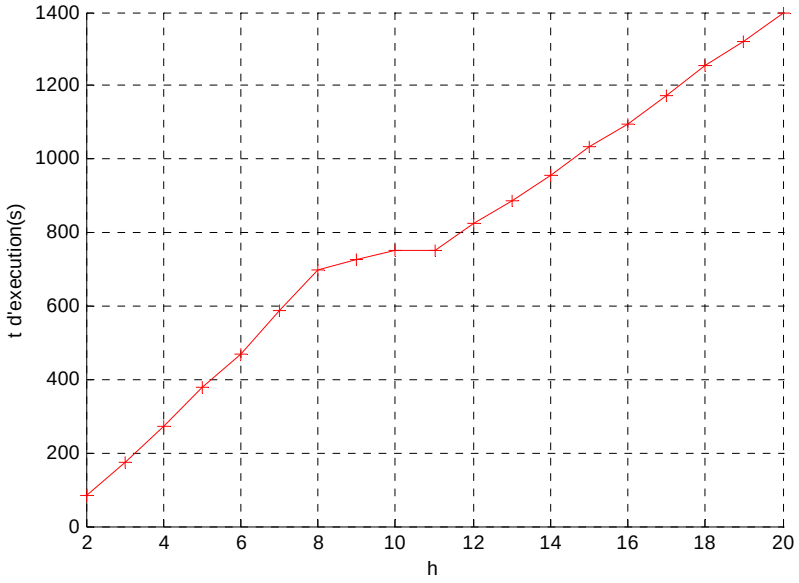


Figure 3.20 : Temps d'exécution en fonction de l'horizon *h* choisi

Par ailleurs, on peut constater sur le Tableau 3.3 qu'à partir d'un horizon supérieur à 12, l'augmentation de l'horizon n'influe plus sur la contraction des positions GPS. De plus, on peut aussi constater que les contractions en x et y ne se sont pas régulières au cours du temps. Une explication en est que l'algorithme agit comme un filtre sur les positions GPS en ne contractant de manière significative que durant les instants où la qualité du GPS est réduite (voir sur la Figure 3.21 pour x et sur la Figure 3.22 pour y). Comme conséquence de ce constat, on peut mieux comprendre la qualité de la contraction qui stagne à partir de h =13. En effet, l'algorithme tend à lisser les pavés GPS uniquement dans les phases où les écarts types sont mauvais et ces contractions ramènent les pavés à des tailles correspondants aux phases

où le GPS donnent des positions avec des bons écarts types, d'où une tendance à une limite de la contraction atteinte ici à partir des horizons supérieurs à 13.

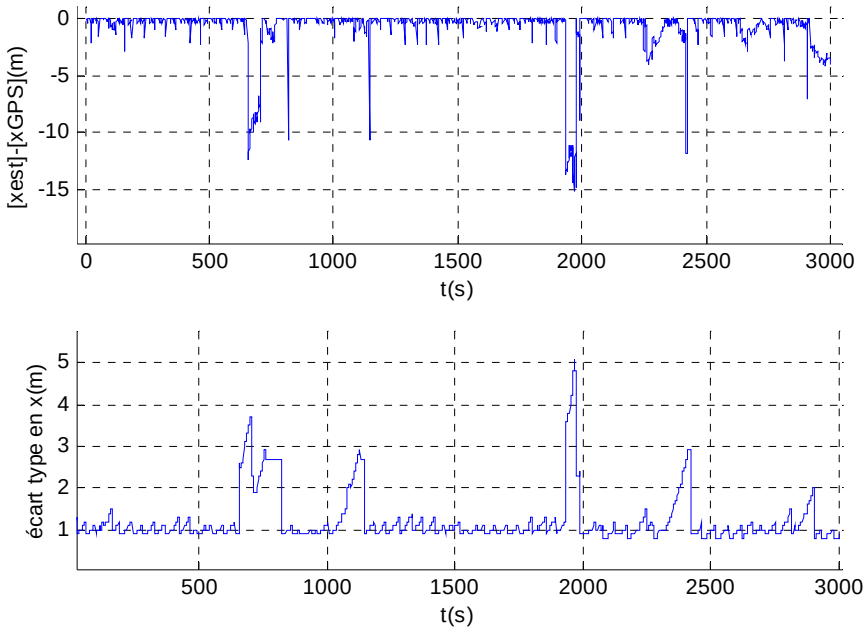


Figure 3.21 : différence entre les largeurs des pavés estimés (pour h=3) et ceux donnés par le GPS en x – évolution de l'écart type en x (donné par le GPS) en fonction du temps.

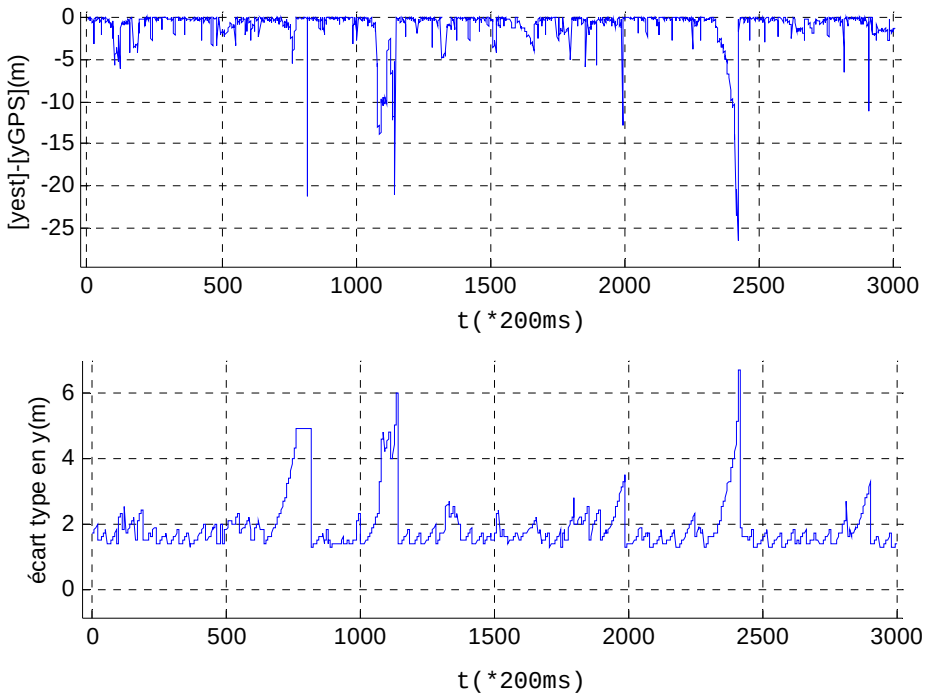


Figure 3.22 : différence entre les largeurs des pavés estimés (pour h=3) et ceux donnés par le GPS en y – évolution de l'écart type en y (donné par le GPS) en fonction du temps.

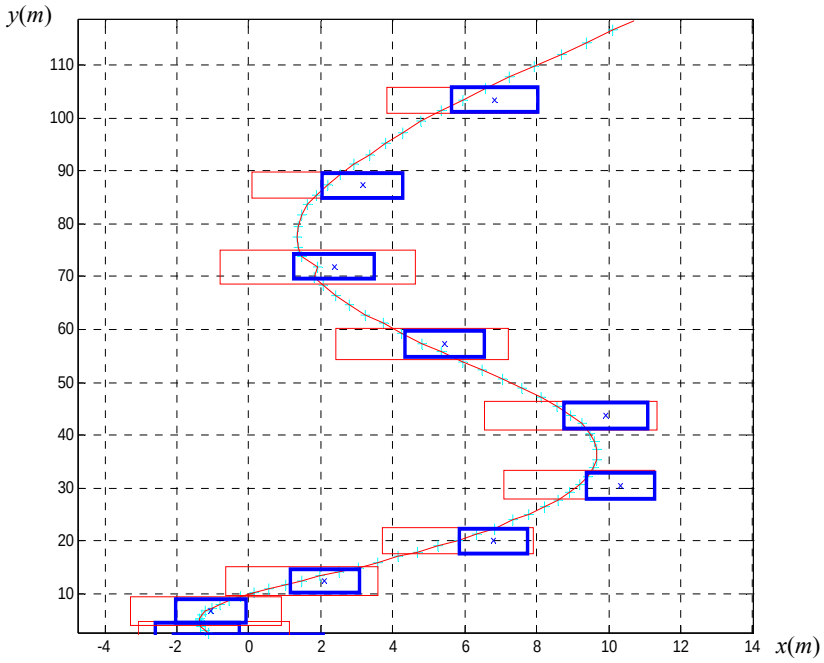


Figure 3.23 : pavés des positions possibles estimées (DGPS pavés traits fins, l'algorithme de Waltz pavés en traits foncés)

Dans la Figure 3.23, dans une portion de la trajectoire (correspondant à la période t ≈ 530s-550s sur les Figure 3.22 ou Figure 3.21), on peut observer que la fusion par l'algorithme de Waltz (pour h=3) de tous les capteurs réduit significativement l'incertitude en réduisant la taille des pavés contenant les solutions de façon garantie. De surcroît, dans la Figure 3.24, on peut constater que l'algorithme se transforme en localisation à l'estime garantie lorsque le signal GPS n'est plus disponible (cette figure correspond à un masquage simulé de 10s). On peut aussi constater que lorsque le GPS est récupéré, le pavé reste consistant et est sensiblement contracté.

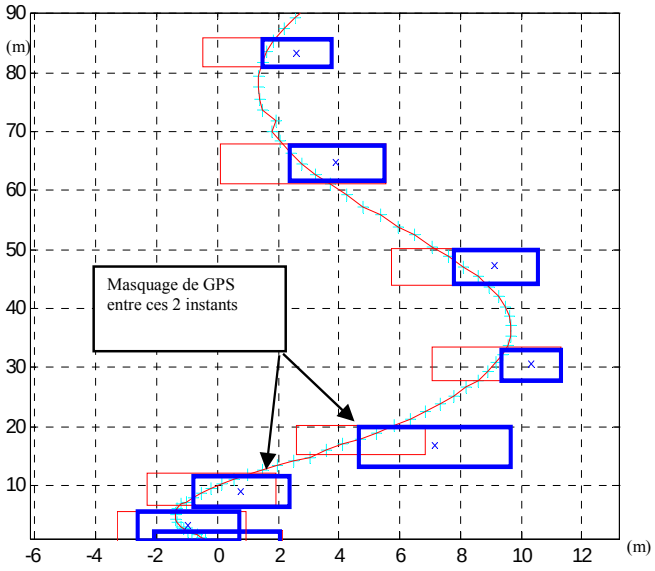


Figure 3.24 : Pavés des positions possibles estimées avec une situation de masquage (DGPS pavés traits fins, algorithme de Waltz pavés en traits foncés).

Un bon résultat pour l'algorithme de Waltz est aussi sa capacité à reconstruire, de façon très précise et garantie une variable qui n'est pas directement mesurée telle que l'angle de cap (voir Figure 3.25 où l'angle est estimé avec une largeur des intervalles moyenne de 5 degrés)

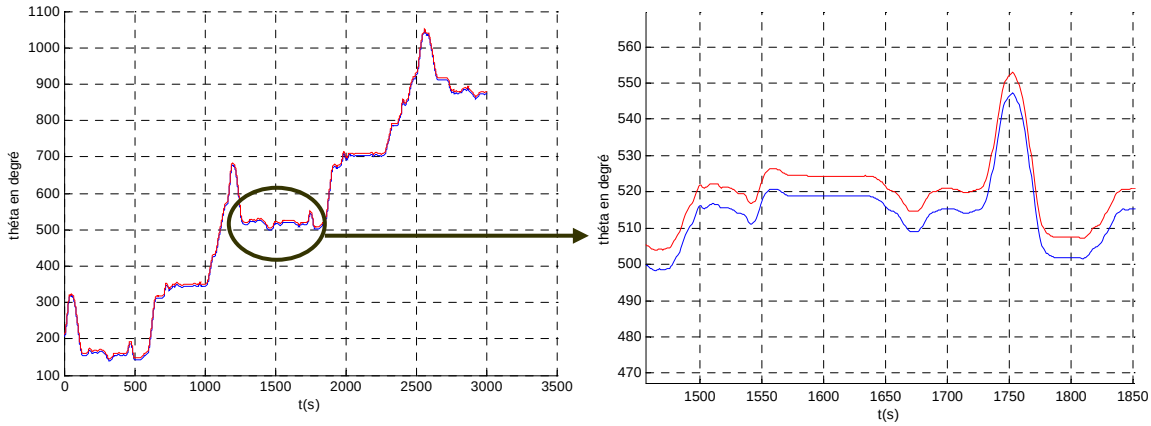


Figure 3.25 : Estimation de l'angle de cap (en degrés)

Après avoir étudié la méthode en termes de faisabilité et évalué des critères statistiques liés au choix de l'horizon, l'étape qui suit est de comparer les performances de la méthode avec l'algorithme classiquement utilisé pour résoudre le problème de localisation traitée notamment l'EKF. Cette étude est faite dans la section qui suit.

4.4.3 Comparaison avec un EKF

Dans cette partie, on reconsidère les mêmes données sur la piste du GIAT (à Versailles Satory) que précédemment avec une information supplémentaire qui est une référence précise. Cette dernière nous permettra de pouvoir comparer l'algorithme de Waltz et l'EKF grâce à des calculs d'erreurs. La référence est obtenue grâce à un GPS Thales bi-fréquence L1/L2 utilisé en mode post-traitement PPK[4], avec une fréquence de 1 Hz. Son principe de fonctionnement est d'enregistrer les données brutes GPS et grâce à une bonne connaissance de la constellation satellitaire et des erreurs dues à l'atmosphère, les positions pourront être retrouvées avec une grande précision (de l'ordre de quelques centimètres). Grâce à une bonne constellation des satellites durant les 10mn d'essais (avril 04), toutes les ambiguïtés de phases ont pu être résolues. La précision de la référence est donc centimétrique.

Figure 3.26 le véhicule expérimental avec les antennes Thales et AG132

La synchronisation entre la référence et les sorties des estimateurs a été faite grâce au temps GPS. La référence étant obtenue à une fréquence de 1Hz, la comparaison n'est faite précisément que pour les instants PPS. De surcroît, il a fallu prendre en compte le décalage entre les antennes GPS des deux récepteurs et l'origine du repère.

[4] Post-Processed Kinematic. Ces traitements ont été effectués par David Bétaille du LCPC de Nantes

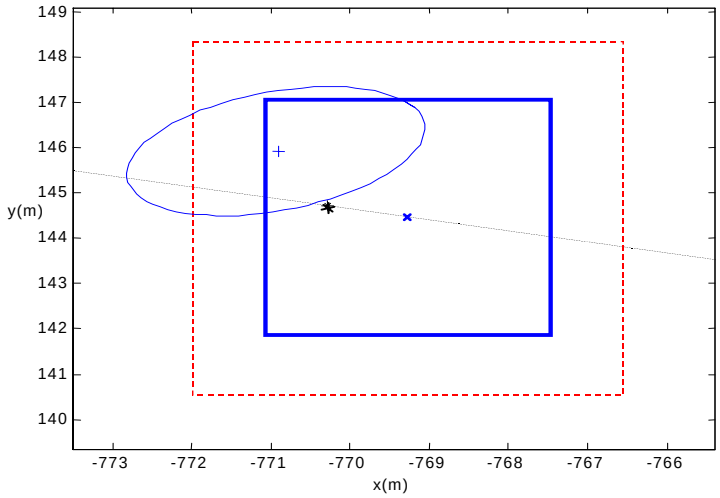


Figure 3.27 : illustration des problèmes de garanties de l'EKF.

La Figure 3.27 illustre le fait que l'EKF ne peut pas garantir une borne maximale d'erreur. On peut y voir que le point fourni par la référence PPK (il s'agit de l'étoile *) est en dehors de l'ellipse Gaussienne de certitude à 99% donnée par l'EKF. Par contre, le pavé obtenu par l'algorithme de Waltz en traits continus et pour $h = 3$ contient le point de référence PPK. En outre, on voit également que le pavé en traits pointillés correspondant au pavé obtenu par la position GPS est contracté par l'algorithme de Waltz pour donner le pavé en trait continu.

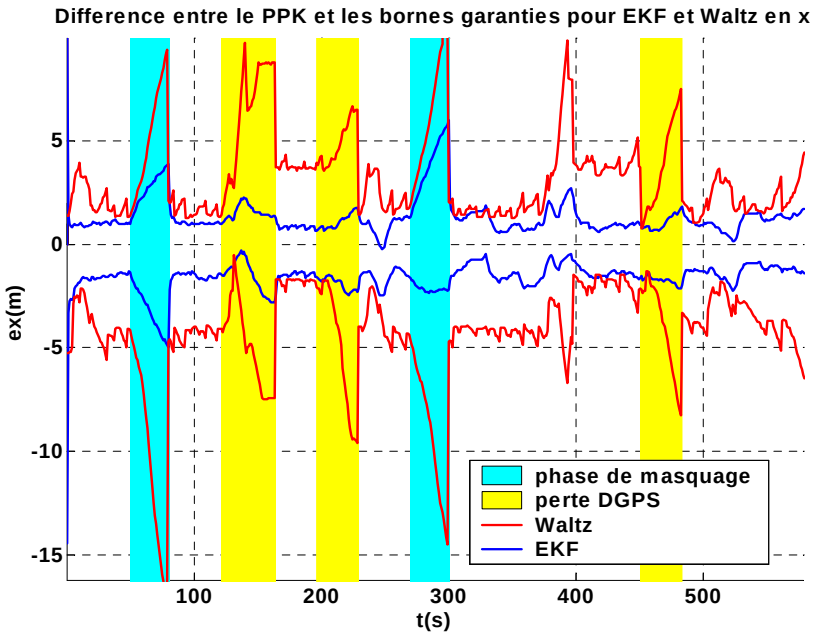


Figure 3.28 : comparaison entre les bornes d'erreur en x de l'EKF et de l'algorithme de Waltz.

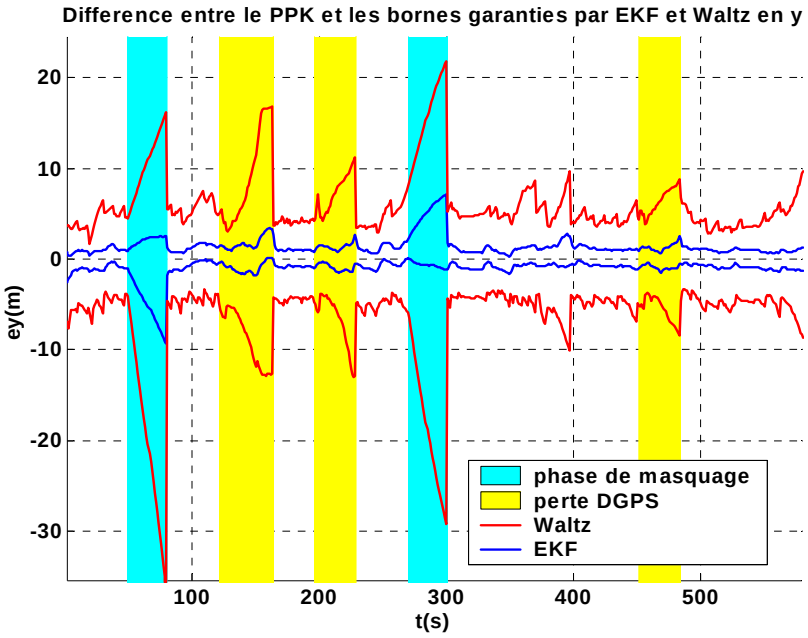


Figure 3.29 : comparaison entre les bornes d'erreur en *y* de l'EKF et de l'algorithme de Waltz.

Sur la Figure 3.28 et sur la Figure 3.29, on trace, pour chacun des estimateurs, la différence entre le point PPK et les deux bornes garanties (la borne inf. et la borne sup. pour l'algorithme de Waltz et l'estimation ± 3 fois l'écart type pour l'EKF). De plus, Il est représenté en bandes les instants où il y'a eu perte du différentiel et également des instants où un masquage de 30s a été simulé (pour t =50s et t = 270s). Rappelons que les pertes du différentiel traduisent l'absence du signal de correction satellitaire et que le masquage correspond à une insuffisance du nombre de satellites visibles.

Il apparaît sur ces résultats que l'EKF est moins pessimiste en ce qui concerne les bornes d'erreurs estimées (bornes qui ne peuvent malheureusement pas être garanties par l'EKF). Une des raisons probable en est la consistance uniquement locale atteinte pour l'algorithme de Waltz. Cependant, les valeurs des erreurs commises ayant les ordres de grandeurs comparables, on peut en conclure un intérêt certain pour les méthodes à approche bornée : il est donc possible de garantir le positionnement du véhicule tout en restant raisonnablement proche de l'EKF en termes d'ordre de grandeur de l'imprécision.

De surcroît, on peut vérifier que la valeur "0" appartient toujours à l'intervalle d'erreur pour l'algorithme de Waltz dans la Figure 3.28 et la Figure 3.29. Ceci confirme que les pavés contiennent bien les solutions (l'appartenance de "0" à l'intervalle d'erreur est équivalent à dire que le pavé contient le point PPK. Par contre, avec l'EKF, on ne peut pas garantir une

erreur maximale pour ses positions comme on peut le voir sur la Figure 3.28 (t≈250s) et sur la Figure 3.29 (t ≈120s, 400s)

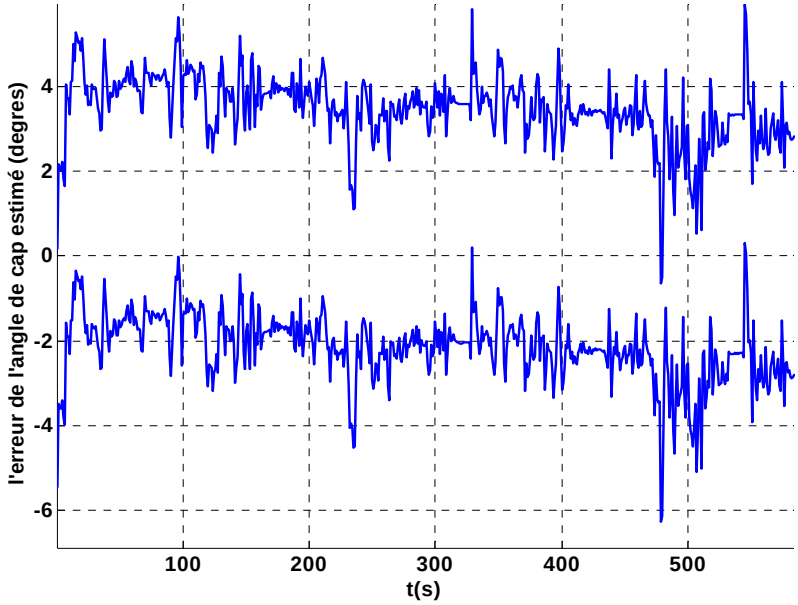


Figure 3.30 : bornes d'erreurs de l'angle de cap estimé par l'algorithme de Waltz.

La Figure 3.30 dessine les bornes d'erreurs correspondant à la différence entre l'estimation intervalle du cap et la valeur du cap estimé grâce à la référence PPK. On peut remarquer que pour trois instants la valeur "0" n'appartient pas à cette différence. Ceci est dû en grande partie aux bruits affectant l'angle de cap de référence, ce dernier ayant été construit manuellement à partir des points PPK. On peut cependant en déduire qu'avec l'algorithme de Waltz, l'angle de cap a pu être reconstruit de façon garantie avec une bonne moyenne de précision (5 degrés).

Conclusion

Dans ce chapitre, il a été introduit une nouvelle technique de localisation dynamique basée sur l'analyse par intervalle et plus spécifiquement sur les techniques de propagation de contraintes. Cette approche garantit que la position réelle du véhicule est contenue dans le pavé estimé au sortir de l'algorithme et cela même en cas de non linéarités. Après avoir fait des rappels pour bien introduire la méthode, cette dernière a été appliquée dans un premier temps à un problème réel. On a alors pu voir que la redondance des capteurs a permis de

diminuer considérablement la taille des pavés aux instants où les performances du GPS sont diminuées. De surcroît, la méthode est capable de reconstruire une variable non mesurée de façon garantie et avec une bonne précision. Dans un second temps, cette méthode a été comparée à un EKF grâce à une référence PPK centimétrique. De cette comparaison, il a été tiré principalement deux informations clefs : premièrement l'algorithme ensembliste est capable de garantir des erreurs maximales, ce qui n'est pas toujours le cas de l'EKF et, deuxièmement, la précision des pavés, bien qu'elle soit inférieure à celle des ellipses de certitude à 99% de l'EKF, reste cependant du même ordre de grandeur.

Ces résultats positifs et conformes aux attentes soulèvent cependant de nouveaux champs d'investigation. En effet, il reste encore quelques points particuliers qui méritent d'être plus étudiés :

- ***Le choix de l'horizon*** : ce choix peut s'avérer déterminant puisqu'il y a un compromis entre le temps de calcul et les performances de l'algorithme. Comme on l'a illustré dans le Tableau 3.3, les différentes expériences menées tendent à faire conclure qu'à partir d'un certain seuil, les contractions obtenues deviennent stables. Si ce résultat s'avérait exact, il suffirait, dans une perspective d'une application temps-réel, de faire une étude en post-traitement pour déterminer le dernier horizon à partir duquel les contractions restent sensiblement les mêmes.
- ***Le temps de calcul*** : par rapport aux différentes expériences menées, il apparaît que cette méthode est largement applicable dans un contexte temps réel. Cependant, il persiste le risque que pour un instant donné la boucle de l'algorithme de Waltz ne se termine pas à temps voulu ! Une solution simple pourrait être de fixer une limite à ne pas dépasser, ce qui permettrait d'obtenir une contraction (cette dernière ne serait pas malheureusement la meilleure qu'on puisse obtenir).
- ***La stratégie en cas d'apparition d'intervalles vides*** : comme nous l'avons expliqué dans la section 4.4.1, la stratégie qui semble être la plus viable consiste à détecter les capteurs défectueux ou les hypothèses faussées. Cependant dans un processus dynamique, il apparaît qu'une telle stratégie passe par une réflexion sur des algorithmes temps réel permettant d'atteindre une consistance globale.

A la lecture des deux derniers points cités plus haut et comme suite pertinente de cette première étude, nous nous sommes fixé comme objectif de mener des recherches sur des algorithmes qui pourrait permettre de se passer du double défaut de l'algorithme de Waltz : un

temps de calcul non déterminable a priori et la consistance locale en présence de cycles. Cette recherche a abouti à l'introduction d'une nouvelle approche par des "domaines de consistance" qui seront présentés dans le chapitre suivant. Ensuite, dans un dernier chapitre, on présentera une nouvelle technique de vectorisation dont l'objectif est d'utiliser les bonnes propriétés[5] de l'algorithme FALL/CLIMB qui permettent notamment d'accélérer et de connaître les temps de calcul.

[5] Comme "bonnes propriétés" de l'algorithme FALL/CLIMB, on pense notamment au de temps calcul optimisé et déterminable à priori

CHAPITRE 4. VERS UNE APPROCHE DE SATISFACTION DE CONTRAINTES VISANT LA CONSISTANCE GLOBALE

Introduction

Dans cette partie, on cherche à définir un concept théorique nouveau, susceptible de résoudre un problème souvent rencontré lorsqu'on résout un problème de satisfaction de contraintes. En effet, les algorithmes de propagation de contraintes aboutissent souvent à des consistances locales sauf dans certains cas, comme celui où le graphe de représentation des contraintes reliant les variables est sous forme d'un arbre. Dans le cas où des cycles interviennent, les méthodes reposent souvent sur l'algorithme de Waltz, amélioré par des heuristiques tendant à organiser le choix de l'ordre de contraction des variables. Dans le but d'améliorer la consistance locale qui découle de ces algorithmes et donc de tendre vers la consistance globale, les solutions existantes se basent, en général, sur un découpage des pavés. Malheureusement, les approches par découpage de pavé ont en commun la mauvaise propriété d'avoir un temps de calcul inconnu a priori, ce qui complique leur implémentation temps-réel.

Dans l'optique d'obtenir des consistances globales et de connaître les temps de calcul a priori, on propose dans cette partie d'introduire de nouveaux concepts et une nouvelle théorie à partir des "domaines de consistance". L'idée générale est de reconsidérer la notion de consistance d'un point de vue transverse. En effet, au lieu de rechercher si un point est consistant avec un système de contraintes, on va chercher dans les dimensions complémentaires l'ensemble de tous les vecteurs consistants avec ce point (voir Figure 4.1). Autour de cette idée, des propriétés intéressantes vont nous permettre d'évaluer ces ensembles et on montrera comment, à partir cette idée, on peut initier une formulation qui généralise les techniques de propagation de contraintes.

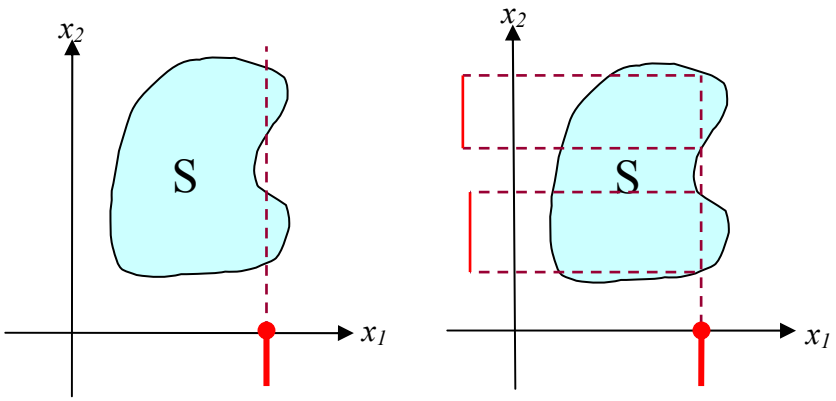


Figure 4.1 : l'idée de consistance d'un point et l'idée plus riche en informations de son domaine de consistance.

Ce chapitre est constitué essentiellement de quatre parties. Dans une première partie, on s'efforcera de rassembler des définitions et notations qui seront utiles pour la suite du chapitre et qui permettront d'introduire la notion de domaine de consistance.

Dans une seconde partie, on pourra définir cette dernière notion et en tirer d'intéressantes propriétés. Puis, une attention particulière sera observée pour montrer l'intérêt d'étudier les domaines de consistance. Ensuite, trois théorèmes reliant les domaines de consistance et les CSP seront donnés. Le premier théorème permet d'obtenir les solutions des CSP avec des domaines de consistance. Les deuxième et troisième théorèmes introduisent des règles de calcul de domaines de consistance utiles notamment en présence de cycles.

Dans une troisième partie, on montrera en quoi les calculs des domaines de consistance constituent une généralisation des techniques classiques de propagation de contraintes. Pour cela, la démarche exposée est d'écrire successivement les *projections de contraintes*, les *résolveurs de contraintes* (voir les définitions dans la section 1) et l'algorithme de Waltz, à l'aide de calculs de domaines de consistance.

Dans la dernière partie, on s'exercera à résoudre deux problèmes académiques en présence de cycles, l'un linéaire, l'autre non linéaire, qui permettront, d'une part, de dégager une méthodologie pour calculer les domaines de consistance et, d'autre part, de comparer les performances des algorithmes obtenus avec l'algorithme de Waltz.

1 NOTATIONS ET DEFINITIONS

1.1 Notations

Soient [x] un pavé de IR^n, $(f_i)_{i=1\cdots p}$ des fonctions réelles et $\mathrm{F}=(f_1,\cdots f_p)$. Pour chaque f_i, on note V_{f_i} l'ensemble des variables concernées par f_i. Considérons le CSP $\mathcal{H}$= (F(x)=*0*, $x\in[x]$), on note S=$\{x\in[x] \mid \mathrm{F}(x)=0\}$. Soient $\mathrm{I}=(1,\cdots,n)$ et $\mathrm{J}=(j_1,\cdots,j_m)$ un sous ensemble de I de cardinal m, on note $\mathrm{K}=\mathrm{I}\setminus\mathrm{J}$ son complémentaire dans I de cardinal *n-m*.

1.2 Sous-vecteur

Considérons un vecteur noté $x \in \mathrm{IR}^n$. On note x_J un sous-vecteur de x associé à J. Ce sous-vecteur est de dimension *m* et s'écrit : $x_J=(x(j_1),\cdots,x(j_m))$. x_J est donc défini comme le vecteur regroupant uniquement les projections du vecteurs x dans les dimensions $(j_1,\cdots,j_m)$. Remarquons que le sous-vecteur x_K est le sous-vecteur de x formé des composantes complémentaire à x_J. On écrira souvent $x=x_I\times x_K$ en supposant que le produit cartésien se fait dans le bon ordre des indices.

Exemple 4.1

■ $\left.\begin{array}{c} x=(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9) \\ \mathrm{J}=(2,3,5,7) \end{array}\right\}\ x_J=(x_2,x_3,x_5,x_7)$ ■

1.3 Sous-pavé

De même que pour les sous-vecteurs, soit [x] un pavé dans IR^n, le sous-pavé de [x] associé à J, noté $[x_J]\in \mathrm{IR}^m$, s'écrit : $[x_J]=([x(j_1)],\cdots,[x(j_m)])$.

1.4 Projection de contraintes selon une dimension

Considérons une contrainte f_i et considérons l'intervalle $[x_j]$. La projection de [x] selon la $j^{ième}$ dimension et la $i^{ième}$ contrainte, notée $\prod_{f_i,j}([x])$, est par définition l'ensemble des x_j dans $[x_j]$ consistants avec f_i. On écrit donc formellement :

$$\prod_{f_i,j}([x]) = \left\{ \tilde{x}_j \in [x_j] \middle| \exists \tilde{x}_k \in [x_k] \text{ pour tout } k \in I - \{i\} \text{ et } f_i(\tilde{x}_1, \cdots, \tilde{x}_n) = 0 \right\} \quad (4.1)$$

On définit, de manière plus générale, la projection $\prod_{G,J}([x])$ selon un système de contraintes G sur un sous-vecteur associé à J selon l'équation suivante :

$$\prod_{G,J}([x]) = \left\{ \tilde{x}_J \in [x_J] \middle| \exists \tilde{x}_k \in [x_k] \text{ pour tout } k \in I - J \text{ et } G(\tilde{x}_1, \cdots, \tilde{x}_n) = 0 \right\} \quad (4.2)$$

Précisons que lorsque G = F, on note la projection $\prod_J([x])$ pour plus de simplicité. Par ailleurs, on peut choisir, par soucis de clarté, de noter les projections en remplaçant les indices par des sous-vecteurs (par exemple $\prod_{G,x_J}([x])$ au lieu de $\prod_{G,\mathrm{J}}([x])$ ou encore $\prod_{x_J}([x])$ au lieu de $\prod_J([x])$).

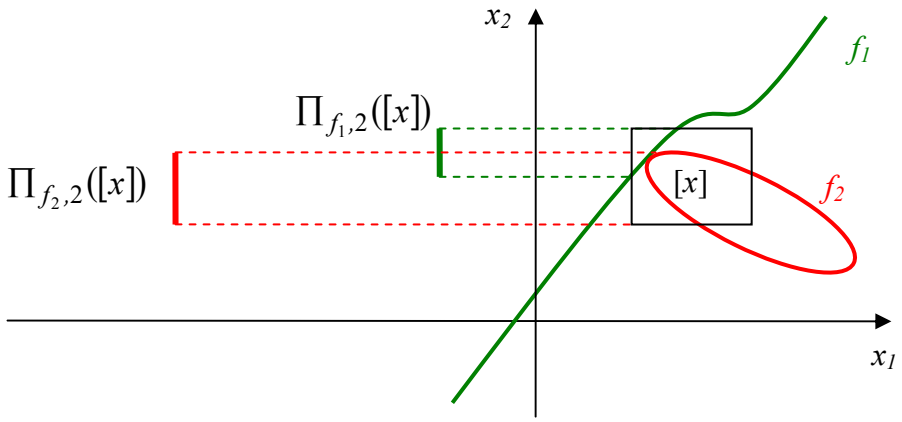


Figure 4.2: illustration de la projection de contraintes selon une dimension.

1.5 Résolveur d'une contrainte

On rappelle qu'un résolveur pour un CSP est un contracteur qui permet d'obtenir le pavé solution (vu au chapitre 3, section 2.4) et qu'un sous-résolveur est un contracteur qui permet d'obtenir les intervalles solutions pour un groupe d'intervalles du pavé initial. On propose d'étendre les résolveurs à la notion de "*résolveur d'une contrainte*". On définit un résolveur R_{f_i} pour une contrainte f_i comme un opérateur tel que $R_{f_i}([x])$ soit le plus grand pavé vérifiant : $R_{f_i}([x]) \subset [x]$ et $R_{f_i}([x])$ consistant avec f_i.

2 DOMAINE DE CONSISTANCE

2.1 Domaine de consistance associé à un sous-vecteur

Considérons d'abord le cas simple décrit par la Figure 4.3 où $\mathcal{H}$ est un CSP à 2 variables (x, y) et constitué de 2 contraintes (une bande et une ellipse, ici). Pour la valeur de x_0 choisie, l'ensemble $D_F(x_0)$ des y tels que (x_0, y) soit globalement consistant avec $\mathcal{H}$ est représenté sur la Figure 4.3.

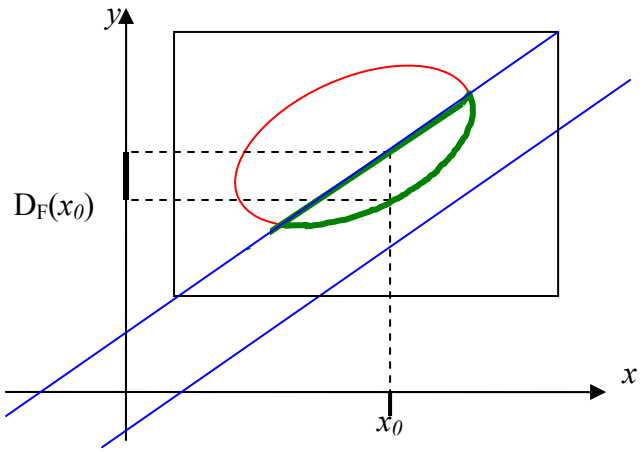


Figure 4.3: domaine de consistance associé à la valeur x_0.

Soit $\tilde{x}_J$ un sous-vecteur associé à J. On appelle domaine de consistance associé à $\tilde{x}_J$, l'ensemble $D_F(\tilde{x}_J)$, inclus dans l'ensemble des sous-vecteurs de IR^n associés à K (complémentaire de J dans I). $D_F(\tilde{x}_J)$ est vide ou bien vérifie les conditions suivantes :

1. $\forall\ \tilde{z} \in IR^n$ tel que $\tilde{z}_J = \tilde{x}_J$ et tel que $\tilde{z}$ vérifie toute les contraintes F, alors $\tilde{z}_K \in D_F(\tilde{x}_J)$ (autrement dit, tout vecteur globalement consistant ayant $\tilde{x}_J$ comme sous-vecteur a ses composantes complémentaires à $\tilde{x}_J$ incluses dans le domaine de consistance $D_F(\tilde{x}_J)$).

2. $\forall\, \tilde{z}_K \in D_F(\tilde{x}_J)$ le produit cartésien de $\tilde{z}_K$ et $\tilde{x}_J$ dans IR^n (dans le bon ordre) vérifie toutes les contraintes F de $\mathcal{H}$ (ou encore, tout vecteur constitué des sous-vecteurs $\tilde{z}_K \in D_F(\tilde{x}_J)$ et de $\tilde{x}_J$ est globalement consistant).

Pour un sous-vecteur donné $\tilde{x}_J$, $D_F(\tilde{x}_J)$ représente donc le plus grand ensemble de sous-vecteurs tels que leurs produits cartésiens avec $\tilde{x}_J$ donne un vecteur dans IR^n globalement consistant avec $\mathcal{H}$.

2.2 Domaine de consistance associé à un sous-pavé

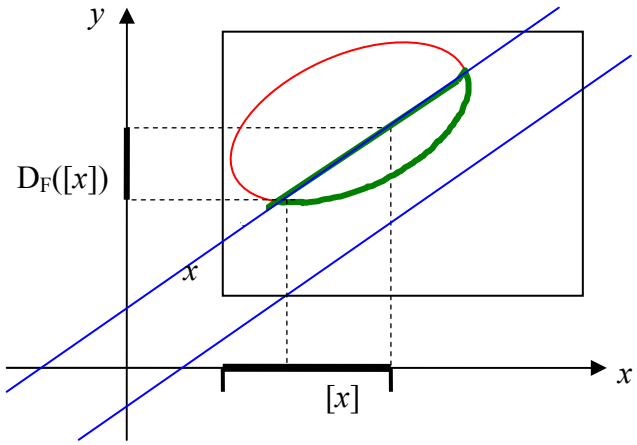


Figure 4.4 : domaine de consistance d'un intervalle.

Considérons maintenant le cas décrit par la Figure 4.4 où, cette fois ci, on cherche à caractériser, pour un intervalle [x] donné, l'ensemble des y pour lesquels $\exists\ \tilde{x} \in [x]$ tel que (x, y) soit globalement consistant avec $\mathcal{H}$. Cet ensemble, noté $D_F([x])$, est appelé domaine de consistance associé à l'intervalle [x].

On peut généraliser cette notion pour un ensemble de sous-vecteurs donné A_J associés à J non vide. $D_F(A_J)$ représente le plus grand ensemble de sous-vecteurs vérifiant :

$\forall\ \tilde{z}_K \in D_F(A_J), \exists\ \tilde{x}_J \in A_J$ tel que le produit cartésien avec $\tilde{x}_J$ donne un vecteur dans [x] globalement consistant. $D_F(A_J)$ (inclus dans l'ensemble des sous-vecteurs dans [x] associés à K) est vide ou bien vérifie :

1. si $\tilde{z}$ est globalement consistant et si ses composantes selon J sont incluses dans A_J alors les composantes de z selon K appartiennent au domaine de consistance $D_F(A_J)$, i.e. $\forall \tilde{z} \in IR^n$ tel que $\tilde{z}_J \in A_J$ et tel que $\tilde{z}$ vérifie toutes les contraintes F, alors $\tilde{z}_k \in D_F(A_J)$,

2. pour tout sous-vecteur dans $D_F(A_J)$, on peut trouver un sous-vecteur dans A_J tel que leur produit cartésien soit globalement consistant, i.e $\forall \tilde{z}_k \in D_F(A_J), \exists \tilde{z}_J \in A_J$ tel que le produit cartésien de z_K et z_J dans IR^n vérifie toutes les contraintes F.

2.3 Propriétés

On se propose dans cette partie d'étudier les propriétés des domaines de consistance. Il faut noter que, pour des soucis de clarté, les propriétés énoncées dans ce paragraphe sont écrites pour des pavés mais elles auraient pu également être écrites pour des ensembles de sous-vecteurs A_J quelconques.

1. Dans le cas particulier de deux sous-vecteurs x et y reliés par une contrainte f : $y-f(x) = 0$, on a $D_f([x])=f([x])$.

Preuve :

■ En effet, par définition $D_f([x])$ est le plus grand ensemble de y vérifiant $y = f(x)$ qui est $f([x])$ ■

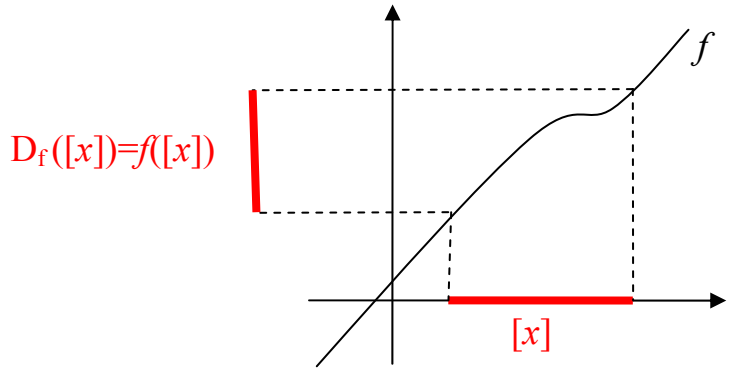


Figure 4.5 : domaine de consistance d'un intervalle pour le cas d'une unique contrainte.

2. Monotonie

Soient $[y_J]$ et $[z_J]$ des sous-pavés dans $\mathcal{H}$ associés à J tels que $[y_J] \subset [z_J]$ alors $D_F([y_J]) \subset D_F([z_J])$.

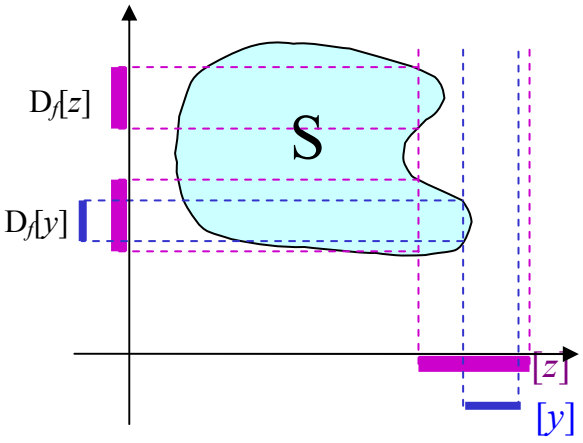


Figure 4.6 : illustration de la propriété de monotonie pour les domaines de consistance.

Preuve :

■ $\forall\ \tilde{y}_K \in D_F([y_J]), \exists\ \tilde{y}_J \in [y_J] \subset [z_J]$ tel que le produit cartésien de $\tilde{y}_K$ et $\tilde{y}_J$ dans IR^n soit globalement consistant

$\Rightarrow \tilde{y}_K \in D_F([z_J])\ \forall\ \tilde{y}_K \in D_F([y_J])$

$\Rightarrow D_F([y_J]) \subset D_F([z_J])$ ■

3. On a la relation suivante pour tout sous-pavé $[x_J]$ de $[x]$: $D_F([x_J]) = \bigcup_{\tilde{x}_J \in [x_J]} D_F(\tilde{x}_J)$ (le domaine de consistance associé à un sous-pavé est l'union des domaines de consistance de tous les sous-vecteurs de ce sous-pavé).

Preuve :

■ En effet, on a $D_F(\tilde{x}_J) \subset D_F([x_J])\ \forall\ \tilde{x}_J \in [x_J] \Rightarrow \bigcup_{\tilde{x}_J \in [x_J]} D_F(\tilde{x}_J) \subset D_F([x_J])$

<u>à l'inverse</u>, pour $\tilde{z}_K \in D_F([x_J])$, on a, par définition, $\exists\ \tilde{z}_J \in [x_J]$ tel que le produit cartésien de $\tilde{z}_K$ et $\tilde{z}_J$ dans IR^n soit globalement consistant

$\Rightarrow \tilde{z}_K \in D_F(\tilde{z}_J)$

or $D_F(\tilde{z}_J) \subset \bigcup_{\tilde{x}_J \in [x_J]} D_F(\tilde{x}_J) \Rightarrow \tilde{z}_K \in \bigcup_{\tilde{x}_J \in [x_J]} D_F(\tilde{x}_J)\ \forall\ \tilde{z}_K \in D_F([x_J])$

$\Rightarrow D_F([x_J]) \subset \bigcup_{\tilde{x}_J \in [x_J]} D_F(\tilde{x}_J)$ ■

4. Union de domaines de consistance

a. Soit $[z_J]$ un sous-pavé associé à J et soient $[z_{J,1}],\cdots,[z_{J,r}]$, *r* sous-pavés associés à J et constituant un recouvrement de $[z_J]$, c'est-à-dire $[z_J] = \bigcup_{1\le i\le r}[z_{J,i}]$ alors : $D_F([z_J]) = \bigcup_{1\le i\le r} D_F([z_{J,i}])$

Preuve :

■on a $[z_{J,i}] \subset [z_J] \;\forall\; i \in 1...r \Rightarrow D_F([z_{J,i}]) \subset D_F([z_J]) \;\forall\; i \in 1\ldots r \Rightarrow \bigcup_{1\le i\le r} D_F([z_{J,i}]) \subset D_F([z_J])$,

A l'opposé, pour $\tilde{z}_K \in D_F([z_J])$, on sait que, par définition, $\exists\ \tilde{z}_J \in [z_J] = \bigcup_{1\le i\le r}[z_{J,i}]$ tel que le produit cartésien de $\tilde{z}_K$ et $\tilde{z}_J$ dans IR^n soit globalement consistant.

or $\tilde{z}_J \in \bigcup_{1\le i\le r}[z_{J,i}] \Rightarrow \exists$ l tel que $\tilde{z}_J \in [z_{J,l}] \Rightarrow \tilde{z}_K \in D_F[z_{J,l}] \subset \bigcup_{1\le i\le r} D_F([z_{J,i}])$

$\Rightarrow \tilde{z}_K \in \bigcup_{1\le i\le r} D_F([z_{J,i}]) \;\forall\; \tilde{z}_K \in D_F([z_J]) \Rightarrow D_F([z_J]) \subset \bigcup_{1\le i\le r} D_F([z_{J,i}])$ ■

b. Ce résultat est aussi vrai pour une union infinie (dénombrable ou non) et la démonstration est identique, i.e. : $[z_J] = \bigcup_{\xi\in\Omega}[z_{J,\xi}] \Rightarrow D_F([z_J]) = \bigcup_{\xi\in\Omega} D_F([z_{J,\xi}])$

5. Intersection

a. Soient $[z_J],[z_{J,1}],\cdots,[z_{J,r}]$ des sous-pavés associés à J tels que $[z_J] = \bigcap_{1\le i\le r}[z_{J,i}]$ alors $D_F([z_J]) \subset \bigcap_{1\le i\le r} D_F([z_{J,i}])$. De plus, en général, l'égalité $D_F([z_J]) = \bigcap_{1\le i\le r} D_F([z_{J,i}])$ n'est pas vraie.

Preuve :

■En effet, $[z_J] \subset [z_{J,i}] \;\forall\; i \in 1...r \Rightarrow D_F([z_J]) \subset D_F([z_{J,i}]).\ \forall\ i \in 1...r$

avec la propriété de monotonie $\Rightarrow D_F([z_J]) \subset \bigcap_{1\le i\le r} D_F([z_{J,i}])$

L'inclusion réciproque est généralement fausse. On peut trouver un contre exemple comme celui illustré sur la Figure 4.7. Dans le cas où $r = n = 2$ et où il n'y a qu'une contrainte *f*, on a : $D_f([z_J]) = f([z_J])$ et $D_f([z_{J,1}]) \cap D_f([z_{J,2}]) = f([z_{J,1}]) \cap f([z_{J,2}])$, mais, en général on n'a pas : $f([z_{J,1}]) \cap f([z_{J,2}]) \subset f([z_J])$ (contre exemple sur la Figure 4.7)■

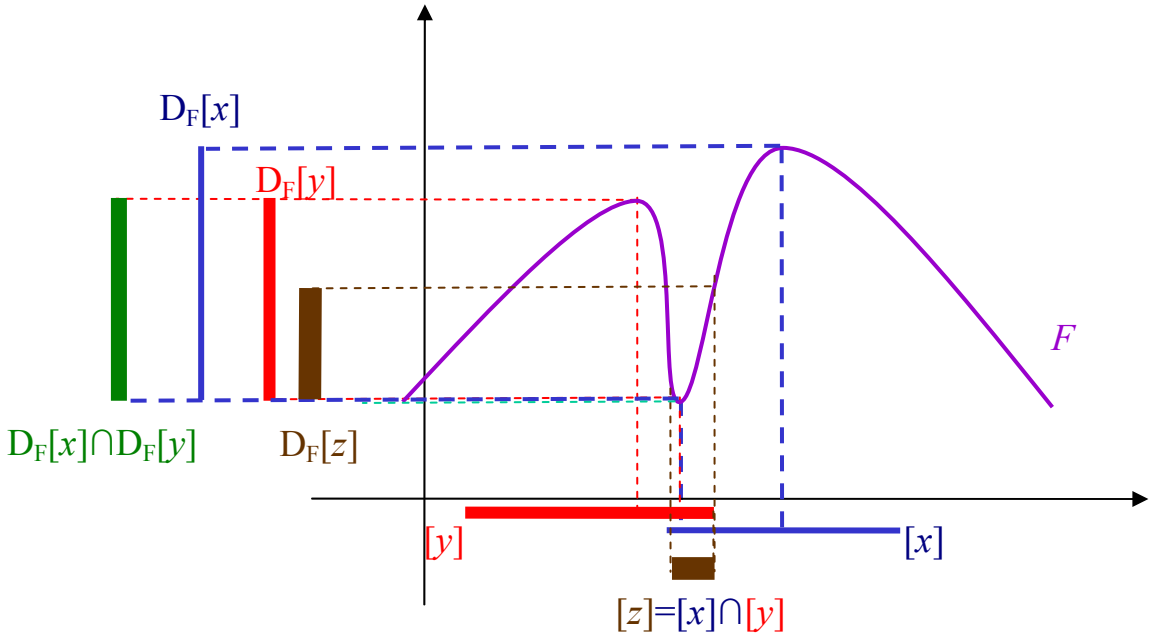


Figure 4.7 : domaine de consistance d'une intersection d'intervalles.

b. Ce résultat est aussi vrai pour une intersection infinie (dénombrable ou non) et la démonstration est identique i.e. : $[z_J] = \bigcap_{\xi\in\Omega}[z_{J,\xi}] \Rightarrow D_F([z_J]) \subset \bigcap_{\xi\in\Omega} D_F([z_{J,\xi}])$

6. Idempotence de l'opérateur D_F

Soit $[z_J]$ un sous-pavé associé à J, on a $[z_J] \subset D_F(D_F([z_J]))$. L'inclusion inverse n'est généralement pas vraie comme on peut le voir sur la Figure 4.8.

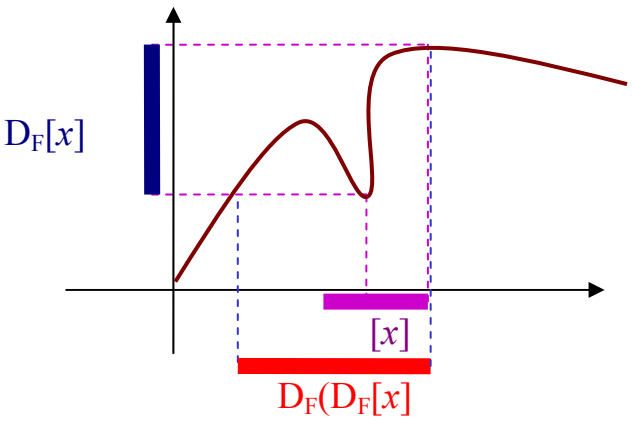


Figure 4.8 : propriété d'idempotence non vérifiée pour les domaines de consistance.

7. Système de contraintes

a. Si un système G(x)=0 est une sous partie du système F(x)=0, alors pour $[x_J]$ sous-pavé de [x], on a $D_F([x_J]) \subset D_G([x_J])$

Preuve :

■ $\tilde{x}_K \in D_F([x_J]) \Leftrightarrow \exists \tilde{x}_J \in [x_J]$ tel que le vecteur $(\tilde{x}_J \times \tilde{x}_K)$ dans IR^n vérifie $F(\tilde{x}_J \times \tilde{x}_K) = 0 \Rightarrow \tilde{x}_J \times \tilde{x}_K$ vérifie en particulier $G(\tilde{x}_J \times \tilde{x}_K) = 0 \Rightarrow \tilde{x}_K \in D_G([x_J])$■

b. Supposons que le système F(x)= 0 soit équivalent à un système qui s'écrit G(x)=0, alors pour $[x_J]$ sous-pavé de [x], on a $D_F([x_J]) = D_G([x_J])$

Preuve :

■ $\tilde{x}_K \in D_F([x_J]) \Leftrightarrow \exists \tilde{x}_J \in [x_J]$ tel que le vecteur $(\tilde{x}_J \times \tilde{x}_K)$ dans IR^n vérifie $F(\tilde{x}_J \times \tilde{x}_K) = 0 \Leftrightarrow \tilde{x}_J \times \tilde{x}_K$ vérifie $G(\tilde{x}_J \times \tilde{x}_K) = 0 \Leftrightarrow \tilde{x}_K \in D_G([x_J])$ ■

c. Le domaine de consistance calculé par rapport à un système de contraintes F est inclus dans l'intersection des domaines de consistance calculés par rapport aux contraintes f_i prises une à une, i.e. pour $[x_J]$ sous-pavé de [x], $D_F([x_J]) \subset \bigcap_{1 \le i \le p} D_{f_i}([x_J])$

De plus, l'égalité $D_F([x_J]) = \bigcap_{1 \le i \le p} D_{f_i}([x_J])$ n'est pas vraie en général.

Preuve :

■ En effet, $\forall i \in 1 \cdots p, f_i(x) = 0$ est une sous-partie du système $F(x) = 0$, d'où en appliquant la propriété 7.a) on a :

$D_F([x]) \subset D_{f_i}([x]) \forall i \in 1 \cdots p \Rightarrow D_F([x]) \subset \bigcap_{1 \le i \le p} D_{f_i}([x])$

L'égalité est fausse en général comme on peut le voir dans le contre exemple de la Figure 4.9.■

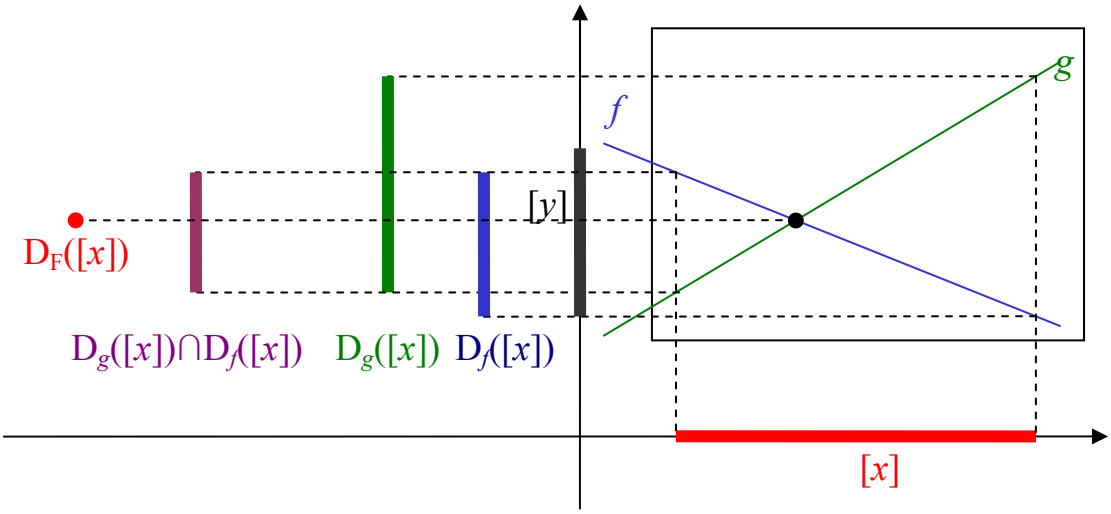


Figure 4.9 : exemple d'un domaine de consistance d'un système de contraintes strictement inclus dans l'intersection des domaines de consistance des contraintes prises une à une.

2.4 Intérêt des domaines de consistance

Après avoir défini les domaines de consistance et présenté une multitude de propriétés, on propose à travers deux exemples simples de montrer, en préambule, l'intérêt de l'introduction de cette notion. En effet, tout d'abord, dans un premier exemple, la consistance locale et la consistance globale pourront être reformulées à l'aide de domaines de consistance. Puis, dans un second exemple, une projection de contrainte sera écrite à l'aide de domaines de consistance.

Exemple 4.2 : ***Consistance locale et domaines de consistance***

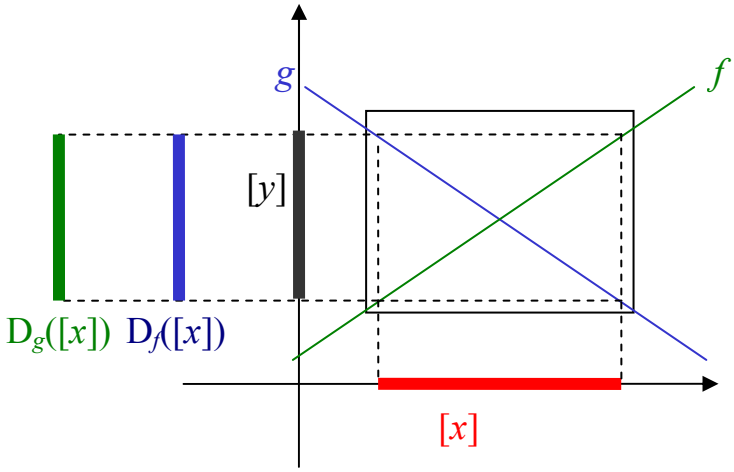


Figure 4.10 : illustration du lien entre la consistance locale et les domaines de consistance.

Considérons la Figure 4.10 traitant un cas en dimension 2 avec un système F de deux contraintes *f* et *g*. Ces deux contraintes sont linéaires et sont représentées par deux droites n'ayant qu'un point d'intersection. Par conséquent, il y a une solution unique au système de deux contraintes. Par contre, le couple d'intervalles [*x*] et [*y*] sont consistants avec les deux contraintes *f* et *g* prises une à une, ce qui est appelé par définition consistance locale du couple ([*x*], [*y*]).

En considérant les domaines de consistances calculés pour [*x*] respectivement avec les contraintes *f* et *g* , on obtient les deux intervalles identiques $D_f([x])$ et $D_g([x])$. Si par contre, on calcule le domaine de consistance pour [*x*] en considérant le système de deux contraintes, on obtient un seul point. Ce faisant, la consistance locale pour cet exemple est équivalente à la propriété 7.a. sur les systèmes de contraintes qui est ici : $D_F([x]) \subset D_f([x]) \cap D_g([x])$

Cette réécriture de la propriété de consistance locale peut s'avérer primordiale sachant qu'en présence de cycles, les techniques de consistances classiques sont souvent, au mieux, uniquement capable d'approcher les domaines globalement consistants sans toujours connaître la précision d'approximation atteinte.

Exemple 4.3 : projection de contrainte et domaine de consistance

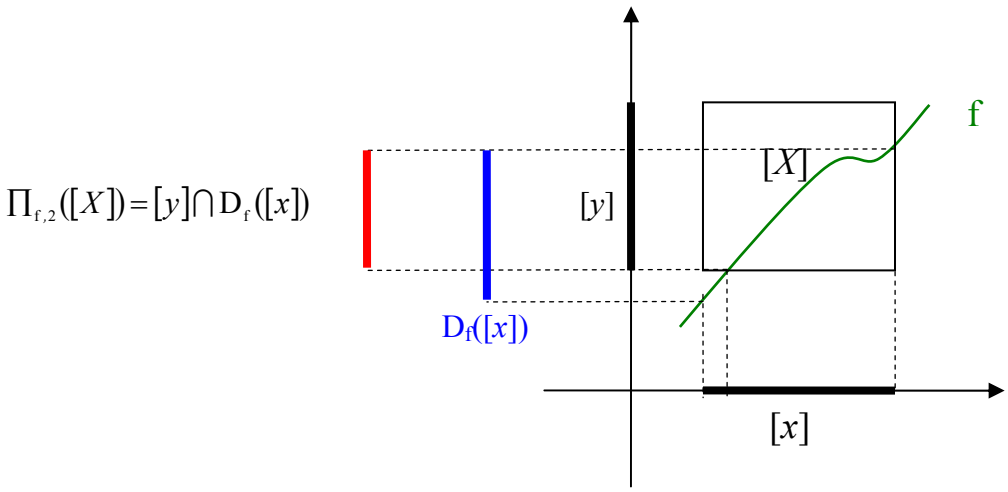


Figure 4.11 : illustration du lien entre les projections de contraintes et les domaines de consistance.

Dans cet exemple en dimension 2, on est en présence d'une contrainte f qui s'écrit $y = f(x)$. Notons $X = (x \times y)$ le vecteur variable du système. La projection $\Pi_{f,2}([X])$ de la contrainte f selon la dimension 1 (celle de *x*) s'écrit : $\Pi_{f,2}([X]) = [y] \cap f([x])$ et le domaine de

consistance $D_f([x])$ associé à [x] selon la contrainte f s'écrit : $D_f([x]) = f([x])$ (propriété 1). On peut alors écrire la projection $\prod_{f,2}([X])$ en fonction du domaine de consistance $D_f([x])$ selon (4.3):

$$\prod_{f,2}([X]) = [y] \cap D_f([x]) \tag{4.3}$$

Sur cet exemple, on a pu écrire la projection d'une contrainte sous forme de calcul de domaine de consistance. Les méthodes de propagation de contraintes étant basées sur des projections, la généralisation de ce résultat permettra de faire un lien direct entre les techniques classiques de consistance et les domaines de consistance.

En conclusion, les domaines de consistance offrent une nouvelle lecture de la consistance locale, tout en étant dans la continuité des techniques de satisfaction de contraintes existantes. De surcroît, de nombreuses propriétés sont vérifiées par les domaines de consistance. Pour toutes ces raisons, une étude plus poussée sur les domaines de consistance est justifiée. La prochaine section est dédiée à l'énoncé de théorèmes et de leurs corollaires.

2.5 Calcul des solutions des CSP grâce à des calculs de domaines de consistance

Dans cette partie, à travers un théorème, l'étude du lien entre les solutions d'un CSP et les domaines de consistance est faite. De plus deux autres théorèmes et leurs corollaires, allant vers des techniques de calcul des domaines de consistance seront montrés.

Théorème 1 : calcul du domaine globalement consistant à l'aide des domaines de consistance associés au complémentaire de chaque composante

On a pour tout $j \in 1 \ldots n$:

$$\Pi_j(S) = [x_j] \cap D_F\left(\underset{i \in I-\{j\}}{\times} [x_i]\right) \tag{4.4}$$

Où $\Pi_j(S)$ est la projection de la solution S selon la $j^{ième}$ composante et $\underset{i \in I-\{j\}}{\times} [x_i]$ est le sous pavé de [x] complémentaire à $[x_j]$.

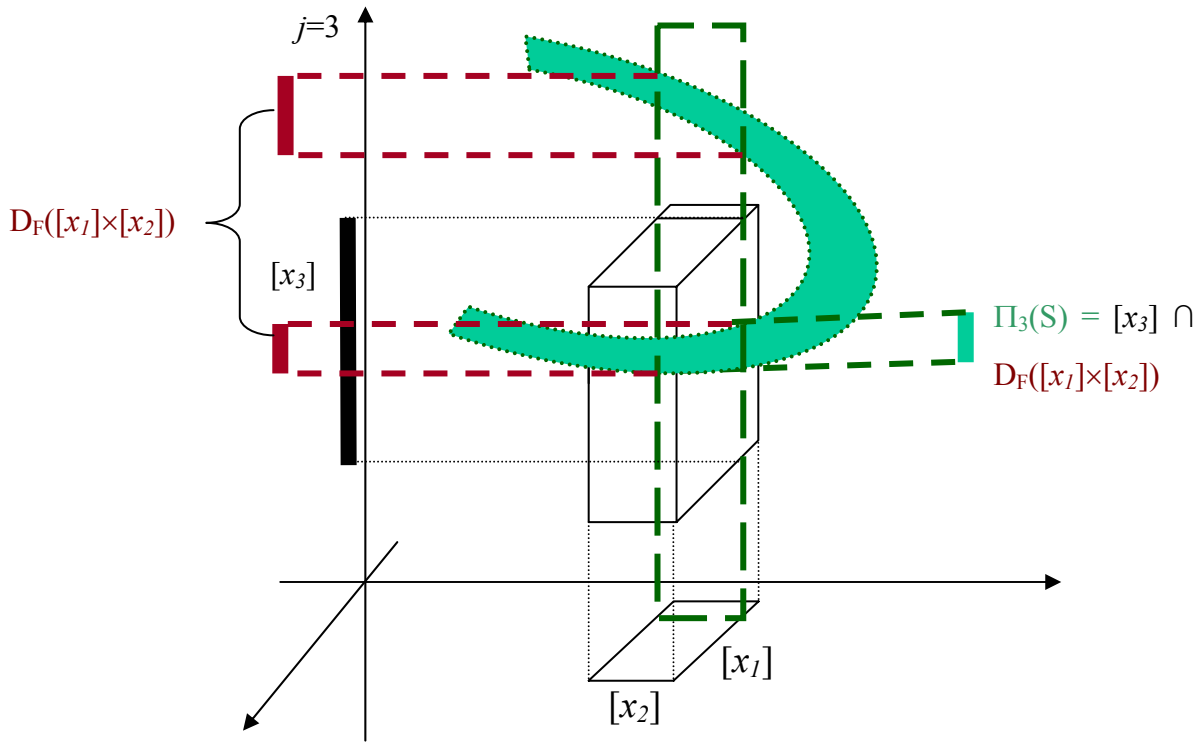


Figure 4.12 : illustration du calcul des projections, sur une dimension, des solutions des CSP à l'aide d'un calcul de domaine de consistance (la contrainte en forme de fer à cheval est dans le plan vertical).

Preuve

- ■ Montrons tout d'abord l'inclusion : $\Pi_j(S) \subset [x_j] \cap D_F\left(\underset{i \in I-\{j\}}{\times} [x_i]\right)$

Par définition $S \subset [x]$ donc $\Pi_j(S) \subset [x_j]$. Montrons de même que $\Pi_j(S) \subset D_F\left(\underset{i \in I-\{j\}}{\times} [x_i]\right)$.

Soit $\tilde{z}_j \in \Pi_j(S)$. Par définition, $\exists\ \tilde{s} \in S$ tel que $\tilde{z}_j = \tilde{s}_j$. Comme $\tilde{s} \in S$ $\Rightarrow \tilde{s}_j \in D_F\left(\underset{i \in I-\{j\}}{\times} \tilde{s}_i\right)$. Donc, on a $\tilde{z}_j = \tilde{s}_j \in D_F\left(\underset{i \in I-\{j\}}{\times} \tilde{s}_i\right)$.

De plus, comme $\underset{i \in I-\{j\}}{\times} (\tilde{s}_i) \subset \underset{i \in I-\{j\}}{\times} [x_i]$, on a : $D_F\left(\underset{i \in I-\{j\}}{\times} s_i\right) \subset D_F\left(\underset{i \in I-\{j\}}{\times} [x_i]\right)$,

Donc $\tilde{z}_j \in D_F\left(\underset{i \in I-\{j\}}{\times} [x_i]\right)\ \forall\ \tilde{z}_j \in \Pi_j(S)$. D'où $\Pi_j(S) \subset D_F\left(\underset{i \in I-\{j\}}{\times} [x_i]\right)$ et par suite

$\Pi_j(S) \subset [x_j] \cap D_F\left(\underset{i \in I-\{j\}}{\times} [x_i]\right)$

- Montrons l'inclusion réciproque $[x_j] \cap \mathrm{D_F}\left(\underset{i \in I-\{j\}}{\times} [x_i] \right) \subset \Pi_j(\mathrm{S})$.

  Soit $\tilde{z}_j \in [x_j] \cap \mathrm{D_F}\left(\underset{i \in I-\{j\}}{\times} [x_i] \right) \Rightarrow \tilde{z}_j \in [x_j]$ et $\exists\ \tilde{y}_K \in \underset{i \in I-\{j\}}{\times} [x_i]$ tel que le produit cartésien $(\tilde{y}_K \times \tilde{z}_J)$ dans $\mathrm{IR}^n \in \mathrm{S}$ (signalons que pour plus de commodité I-$\{j\}$ est aussi noté K, notamment, pour les vecteurs et sous-pavés qui lui sont associés)

  $\Rightarrow \tilde{z}_j \in \Pi_j(\mathrm{S}), \forall\ z_j \in [x_j] \cap \mathrm{D_F}\left(\underset{i \in I-\{j\}}{\times} [x_i] \right)$ d'où $[x_j] \cap \mathrm{D_F}\left(\underset{i \in I-\{j\}}{\times} [x_i] \right) \subset \Pi_j(\mathrm{S})$ ■

En répétant *n* fois cette opération pour toutes les composantes, on obtient un pavé globalement consistant avec S. Cependant, toute la difficulté réside dans le calcul du domaine de consistance $\mathrm{D_F}\left(\underset{i \in I-\{j\}}{\times} [x_i] \right)$, sachant que, d'un coté, le terme $\underset{i \in I-\{j\}}{\times} [x_i]$ peut exploser du fait de la taille du problème, et que, d'un autre coté, le système de contraintes F peut être très complexe avec la présence de cycles.

Par la suite, en se focalisant sur le domaine de consistance $\mathrm{D_F}\left(\underset{i \in I-\{j\}}{\times} [x_i] \right)$, deux théorèmes importants ont été obtenus. Tout d'abord, un premier théorème permet de passer par une étape très utile pour les cycles : la possibilité de calculer le domaine de consistance $\mathrm{D_F}\left(\underset{i \in I-\{j\}}{\times} [x_i] \right)$, en prenant les contraintes une à une, et en prenant cependant la précaution de prendre pour certaines variables des domaines ponctuels à la place des domaines intervalles. Ensuite, un second résultat consiste en un raisonnement récursif sur le système de contraintes permettant de simplifier le terme $\mathrm{D_F}\left(\underset{i \in I-\{j\}}{\times} [x_i] \right)$, en réduisant les contraintes à traiter et en remplaçant le domaine $\underset{i \in I-\{j\}}{\times} [x_i]$ par les solutions d'un CSP particulier.

Théorème 2 : choix judicieux de variables ponctuelles ou intervalles pour le calcul d'un domaine de consistance

Dans cette partie, l'objectif est de comparer le domaine de consistance calculé sur un ensemble de contraintes et l'intersection des domaines de consistance calculés pour chaque contrainte.

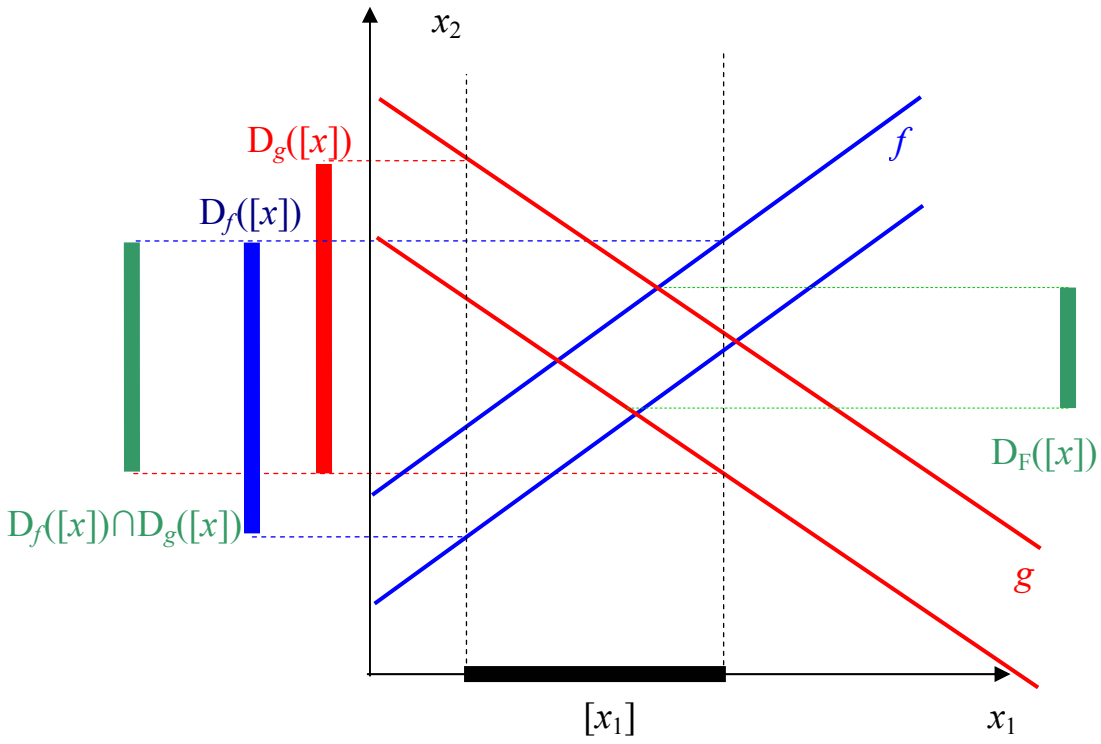


Figure 4.13 : calcul du domaine de consistance associé à un intervalle selon un système de contraintes et selon les contraintes prises une à une.

Considérons la Figure 4.13 et la Figure 4.14, où on a un CSP en dimension 2 et 2 contraintes f et g. En appliquant le ***Théorème 1***, on a : $\Pi_2(S) = [x_2] \cap D_F\left(\underset{i\in I-\{2\}}{\times} [x_i]\right)$ $= [x_2] \cap D_F([x_1])$. Pour le calcul du terme $D_F([x_1])$, on met en exergue le constat suivant : dans la Figure 4.13, le domaine de consistance $D_F([x_1])$, associé à l'intervalle $[x_1]$, calculé avec le système F est strictement contenu dans l'intersection des deux domaines de consistance $D_f([x])$ et $D_g([x])$ calculés respectivement avec les contraintes f et g. Par contre, dans la Figure 4.14, lorsqu'à la place de l'intervalle $[x_1]$ on considère cette fois ci un point x_0

dans $[x_1]$, on obtient l'égalité entre le domaine calculé selon le système et l'intersection des deux domaines de consistance.

Ce constat nous emmène à développer l'idée selon laquelle, pour calculer le terme $D_F\left(\underset{i\in I-\{j\}}{\times}[x_i]\right)$, il faut trier les variables qui apparaissent au moins deux fois dans les contraintes des autres variables. Ceci est formalisé de manière générale dans le ***Théorème 3***.

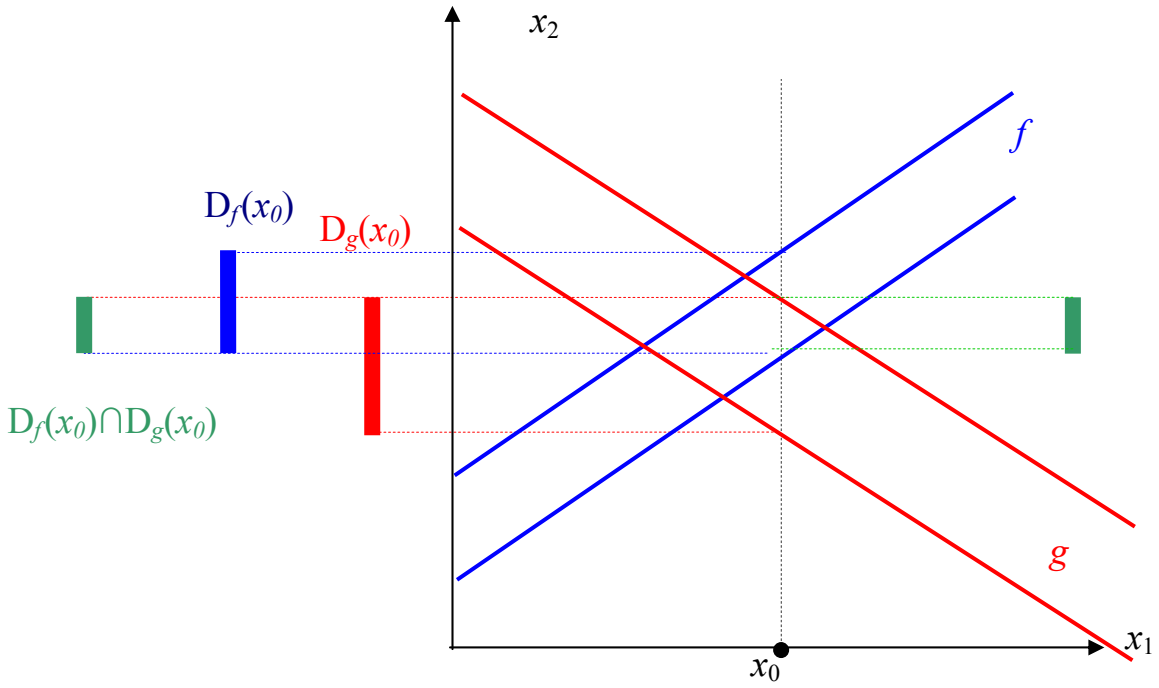


Figure 4.14 : calcul du domaine de consistance associé à un point selon un système de contraintes et selon les contraintes prises une à une.

Soit $[x_J]$ le sous-pavé de $[x]$ associé à J. On note à leur tour $[x_J^{1-}]$ et $[x_J^{1+}]$ deux sous-vecteurs de $[x_J]$ qui sont les composantes qui apparaissent respectivement une fois au plus et plus d'une fois dans le système de contraintes. On a pour tout $\tilde{x}_J^{1+}$ dans $[x_J^{1+}]$

$$D_F([x_J^{1-}]\times\tilde{x}_J^{1+}) = \bigcap_{i=1\cdots p} D_{f_i}([x_J^{1-}]\times\tilde{x}_J^{1+}) \qquad (4.5)$$

Preuve

■ Montrons l'inclusion dans les deux sens

$D_F([x_J^{1-}]\times\tilde{x}_J^{1+}) \subset \bigcap_{i=1\cdots p} D_{f_i}([x_J^{1-}]\times\tilde{x}_J^{1+})$:

$\forall\ i$=1…p, f_i est une sous partie du système F donc $\mathrm{D_F}\left(\left[x_J^{1-}\right]\times\tilde{x}_J^{1+}\right)\subset \mathrm{D}_{f_i}\left(\left[x_J^{1-}\right]\times\tilde{x}_J^{1+}\right)$ $\forall\ i$=1…p (propriété sur les systèmes de contraintes 7.a),

d'où : $\mathrm{D_F}\left(\left[x_J^{1-}\right]\times\tilde{x}_J^{1+}\right)\subset \bigcap_{i=1\cdots p} \mathrm{D}_{f_i}\left(\left[x_J^{1-}\right]\times\tilde{x}_J^{1+}\right)$

Réciproquement :

soit $\tilde{z}_K \in \bigcap_{i=1\cdots p} \mathrm{D}_{f_i}\left(\left[x_J^{1-}\right]\times\tilde{x}_J^{1+}\right) \Rightarrow \forall\ i$=1…$p$, $\tilde{z}_K \in \mathrm{D}_{f_i}\left(\left[x_J^{1-}\right]\times\tilde{x}_J^{1+}\right) \Rightarrow \forall\ i$=1…$p$, $\exists\ \tilde{z}^i \in \left[x_J^{1-}\right]$ tel que $f_i\left(\tilde{z}_K\times\tilde{z}^i\times\tilde{x}_J^{1+}\right)=0$

Or, les variables dans $\left[x_J^{1-}\right]$ n'apparaissent qu'une fois dans les contraintes f_i. On peut alors prendre un $\tilde{z}_J^{1-} \in \left[x_J^{1-}\right]$ tel que les composantes de $\tilde{z}_J^{1-}$ qui interviennent dans la contrainte f_i soient les mêmes que celles de $\tilde{z}^i$ pour tout i = 1…p. i.e. $\tilde{z}_J^{1-}$ vérifie alors

$\tilde{z}_J^{1-} \in \left[x_J^{1-}\right]$ et $f_i\left(\tilde{z}_K\times\tilde{z}_J^{1-}\times\tilde{x}_J^{1+}\right)=f_i\left(\tilde{z}_K\times\tilde{z}^i\times\tilde{x}_J^{1+}\right)\ \forall\ i$=1…$p$

$\Rightarrow \exists\ \tilde{z}_J^{1-} \in \left[x_J^{1-}\right]$ tel que $f_i\left(\tilde{z}_K\times\tilde{z}_J^{1-}\times\tilde{x}_J^{1+}\right)=0\ \forall\ i$=1…$p$

$\Rightarrow \exists\ \tilde{z}_J^{1-} \in \left[x_J^{1-}\right]$ tel que $\mathrm{F}\left(\tilde{z}_K\times\tilde{z}_J^{1-}\times\tilde{x}_J^{1+}\right)=0 \Rightarrow \tilde{z}_K \in \mathrm{D}_F\left(\tilde{z}_J^{1-}\times\tilde{x}_J^{1+}\right)$

$\Rightarrow \tilde{z}_K \in \mathrm{D_F}\left(\left[x_J^{1-}\right]\times\tilde{x}_J^{1+}\right)$ d'où : $\bigcap_{i=1\cdots p} \mathrm{D}_{f_i}\left(\left[x_J^{1-}\right]\times\tilde{x}_J^{1+}\right)\subset \mathrm{D_F}\left(\left[x_J^{1-}\right]\times\tilde{x}_J^{1+}\right)$ ■

La conséquence directe de ce théorème est que, pour calculer le domaine de consistance d'un pavé associé à un système d'équations, on prendra des points pour certaines variables pour ensuite faire l'union. Cette sélection des variables permet d'envisager de calculer les domaines de consistance en considérant toutes les contraintes à la fois.

Théorème 3 : simplification du calcul de domaines de consistance grâce à une récurrence sur les CSP à résoudre

On a vu que pour une dimension donnée *j*, le calcul de la projection des solutions du CSP sur *j* équivaut à la connaissance du domaine de consistance $\mathrm{D}_{\mathrm{F}}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right)$. Dans cette partie, on cherche, cette fois, à étudier le domaine de consistance $\mathrm{D}_{\mathrm{F}}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right)$ en se focalisant sur les contraintes concernant effectivement la variable x_j i.e. les $f_i \,|\, x_j \in V_{f_i}$. En effet, une idée naturelle est de faire la distinction entre, d'un coté, les contraintes concernant la dimension *j*, et, d'un autre coté, toutes les autres. L'enjeu est alors de savoir calculer $\mathrm{D}_{\mathrm{F}}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right)$ en fonction uniquement des contraintes reliées à la dimension *j,* mais aussi en fonction d'un domaine dépendant de l'autre ensemble des contraintes ne concernant pas la dimension *j*. Cal

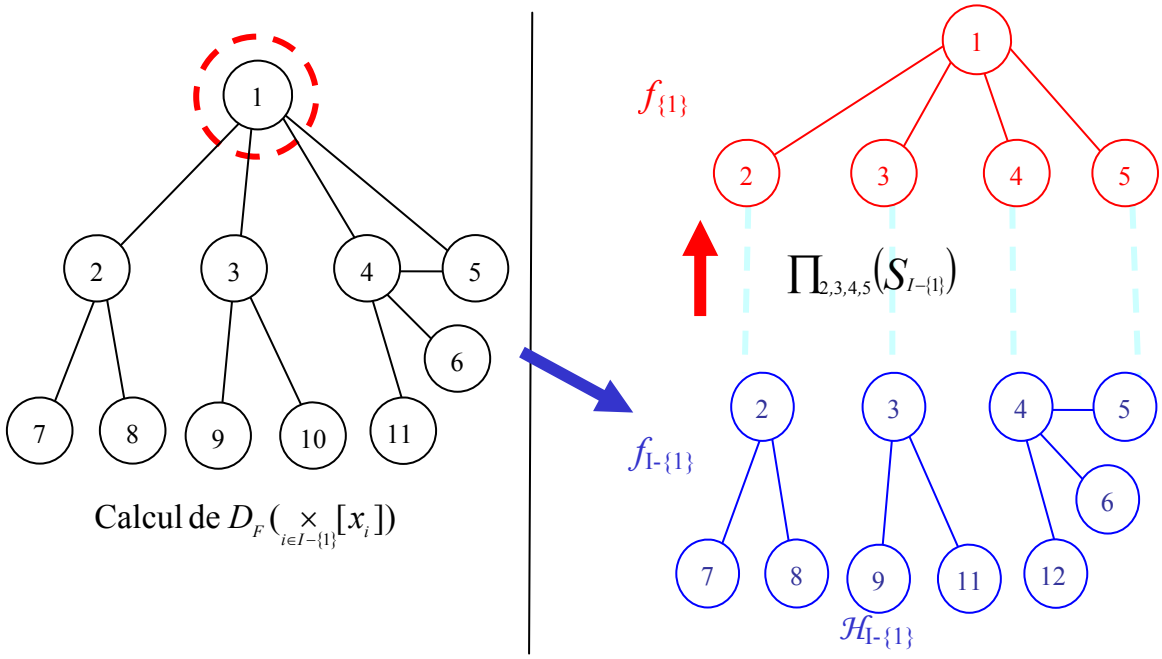


Figure 4.15 : calcul du domaine de consistance $\mathrm{D}_{\mathrm{F}}\left(\underset{i\in I-\{1\}}{\times}[x_i]\right)$ avec une stratégie permettant d'éclater le calcul en deux parties

La Figure 4.15 est une représentation tacite d'un graphe[1] de contraintes. On peut y voir que l'on désire rendre consistante la variable dans la dimension 1 en utilisant le domaine de consistance $\mathrm{D_F}\left(\underset{i\in I-\{1\}}{\times}[x_i]\right)$.

La variable dans la dimension 1 est reliée aux variables dans les dimensions 2, 3, 4 et 5 par l'ensemble des contraintes notée ici $f_{\{1\}}$. On cherche alors un résultat permettant d'exprimer le domaine $\mathrm{D_F}\left(\underset{i\in I-\{1\}}{\times}[x_i]\right)$ uniquement en fonction des variables connexes à "1" que sont "2", "3", "4" et "5" ainsi que des contraintes concernant "1" qui sont ici $f_{\{1\}}$. De surcroît, intuitivement, les domaines à utiliser dans les dimensions 2, 3, 4 et 5 pour calculer $\mathrm{D_F}\left(\underset{i\in I-\{1\}}{\times}[x_i]\right)$ sont liés au CSP $\mathcal{H}_{\mathrm{I}\text{-}\{1\}}$ qui est formé des variables autres que "1" (c'est-à-dire de "2" à "11") et des contraintes $f_{I-\{1\}}$ (qui sont les contraintes dans F autres que celles dans $f_{\{1\}}$).

Ainsi, pour résumer l'idée illustrée sur la Figure 4.15, si on obtient les solutions du CSP qui ne contient pas la variable "1", pour calculer le domaine de consistance associé aux dimensions complémentaire à 1, il suffit de projeter ces solutions sur les composantes connexes à "1" et de prendre en compte uniquement les contraintes faisant intervenir "1". Le ***Théorème 3*** formalise cette idée et les corollaires de ce théorème affinent, au fur et à mesure, les contraintes ainsi que les domaines à utiliser dans le calcul du domaine de consistance.

L'intérêt de ce théorème pour les calculs de domaine de consistance est donc, à la fois, de simplifier la complexité de l'ensemble des contraintes à prendre en compte et de diminuer le nombre de variables à prendre en compte. En effet pour le calcul du domaine de consistance, d'un coté, on considère uniquement un premier lot de contraintes, et d'un autre coté, on considère uniquement certaines variables avec des domaines précis dépendants des contraintes exclues du premier lot.

La première écriture du théorème ci-dessous traduit l'idée que « pour un CSP donné, le domaine de consistance d'un sous-pavé peut s'écrire comme celui des solutions d'un CSP réduit ».

---

[1] On reparlera dans les détails des techniques pour représenter les graphes pour les CSP dans le chapitre 5.

Adoptons les notations suivantes en préambule : soit le CSP noté $\mathcal{H}_{I-\{j\}}$, dans l'espace IR^{n-1} correspondant aux variables réduites aux $x_i \neq x_j$ et aux contraintes les reliant i.e. $\mathcal{H}_{I-\{j\}} = (f_{I-\{j\}}(y)=0,\ y \in \underset{i\in I-\{j\}}{\times}[x_i])$ où $f_{I-\{j\}}$ est le sous-ensemble des contraintes de F indépendantes de la variable x_j. Notons $\mathrm{S}_{\mathrm{I}-\{j\}}$ l'ensemble solution du CSP $\mathcal{H}_{I-\{j\}}$. On a le résultat :

$$\mathrm{D}_{\mathrm{F}}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right) = \mathrm{D}_{\mathrm{F}}\left(\mathrm{S}_{\mathrm{I}-\{j\}}\right) \tag{4.6}$$

En d'autres termes, le ***Théorème 3*** dit que pour un j donné, le calcul du domaine de consistance $\mathrm{D}_{\mathrm{F}}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right)$, par rapport au système de contraintes F, associé au sous-pavé de $[x]$ formé à partir des autres dimensions I-$\{j\}$ est équivalent au domaine de consistance, calculé à partir du système F, associé cette fois-ci à l'ensemble des sous-vecteurs $\mathrm{S}_{\mathrm{I}-\{j\}}$ solutions du CSP $\mathcal{H}_{I-\{j\}}$ (défini à partir des variables autres que j et les contraintes les reliant).

Preuve

■ On a $\mathrm{S}_{I-\{j\}} \subset \underset{i\in I-\{j\}}{\times}[x_i]$ donc d'après la propriété de monotonie $\mathrm{D}_{\mathrm{F}}\left(\mathrm{S}_{\mathrm{I}-\{j\}}\right) \subset \mathrm{D}_{\mathrm{F}}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right)$.

Montrons la réciproque : soit alors $\tilde{z}_j \in \mathrm{D}_{\mathrm{F}}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right)$

$\Leftrightarrow \exists\, \tilde{y}_K \in \underset{i\in I-\{j\}}{\times}[x_i]$ tel que $(\tilde{y}_K \times \tilde{z}_j)$ dans IR^n vérifie $\mathrm{F}(\tilde{y}_K \times \tilde{z}_j) = 0$, en particulier, $(\tilde{y}_K \times \tilde{z}_j)$ vérifie l'ensemble des contraintes $f_{I-\{j\}}$ dans $\mathcal{H}_{I-\{j\}}$, i.e. $f_{I-\{j\}}(\tilde{y}_K \times \tilde{z}_j) = 0$

$f_{I-\{j\}}$ ne concernant que les variables x_i, $i \neq j$, l'écriture $f_{I-\{j\}}(\tilde{y}_K \times \tilde{z}_j) = 0$ revient à $f_{I-\{j\}}(\tilde{y}_K) = 0$ d'où $\tilde{y}_K \in \mathrm{S}_{I-\{j\}}$.

Par suite $\forall\ \tilde{z}_j \in \mathrm{D}_{\mathrm{F}}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right), \exists\ \tilde{y}_K \in \mathrm{S}_{I-\{j\}}$ vérifiant $\mathrm{F}(\tilde{y}_K \times \tilde{z}_j) = 0$

$\Rightarrow \forall\ \tilde{z}_j \in \mathrm{D}_{\mathrm{F}}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right),\ \tilde{z}_j \in \mathrm{D}_{\mathrm{F}}(\mathrm{S}_{\mathrm{I}-\{j\}})$ d'où $\mathrm{D}_{\mathrm{F}}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right) \subset \mathrm{D}_{\mathrm{F}}\left(\mathrm{S}_{\mathrm{I}-\{j\}}\right)$■

Corollaire 1

Appelons $f_{\{j\}}$ l'ensemble des contraintes reliant la variable x_j aux autres variables dans $\mathcal{H}$, on a la relation suivante découlant du résultat précédent :

$$\mathrm{D_F}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right)=\mathrm{D}_{f_{\{j\}}}\left(\mathrm{S_{I-\{j\}}}\right) \tag{4.7}$$

Ce corollaire affine un peu plus le résultat du ***Théorème 3*** en remplaçant le système de contraintes F par le système $f_{\{j\}}$ qui est l'ensemble des contraintes concernant la variable j.

Preuve

■ Soit $\tilde{z}_j \in \mathrm{D_F}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right)$. D'après le ***Théorème 3***, $\mathrm{D_F}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right)=\mathrm{D_F}\left(\mathrm{S_{I-\{j\}}}\right)$.

Notons, pour plus de facilité de lecture, K = I-$\{j\}$. On a donc, $\exists\ \tilde{y}_K \in \mathrm{S_{I\text{-}\{j\}}}$ tel que le produit cartésien $\left(\tilde{y}_K \times \tilde{z}_j\right)$ dans IR^n vérifie $\mathrm{F}\left(\tilde{y}_K \times \tilde{z}_j\right)=0$

$\Leftrightarrow \exists\ \tilde{y}_K \mid f_{I-\{j\}}\left(\tilde{y}_K\right)=0$ et $\left(\tilde{y}_K \times \tilde{z}_j\right)$ vérifie $\mathrm{F}\left(\tilde{y}_K \times \tilde{z}_j\right)=0$

$\Leftrightarrow \exists\ \tilde{y}_K \mid f_{I-\{j\}}\left(\tilde{y}_K\right)=0$ et $\left(\tilde{y}_K \times \tilde{z}_j\right)$ vérifie $f_{\{j\}}\left(\tilde{y}_K \times \tilde{z}_j\right)=0$ et $f_{I-\{j\}}\left(\tilde{y}_K \times \tilde{z}_j\right)=0$

($f_{I-\{j\}}$ ne concernant que les dimensions dans I-$\{j\}$, on peut remplacer $f_{I-\{j\}}\left(\tilde{y}_K \times \tilde{z}_j\right)$ par $f_{I-\{j\}}\left(\tilde{y}_K\right)$)

$\Leftrightarrow \exists\ \tilde{y}_K \mid f_{I-\{j\}}\left(\tilde{y}_K\right)=0$ et $\left(\tilde{y}_K \times \tilde{z}_j\right)$ vérifie $f_{\{j\}}\left(\tilde{y}_K \times \tilde{z}_j\right)=0$ et $f_{I-\{j\}}\left(\tilde{y}_K\right)=0$

$\Leftrightarrow \exists\ \tilde{y}_K \mid f_{I-\{j\}}\left(\tilde{y}_K\right)=0$ et $\left(\tilde{y}_K \times \tilde{z}_j\right)$ vérifie $f_{\{j\}}\left(\tilde{y}_K \times \tilde{z}_j\right)=0$

$\Leftrightarrow \exists\ \tilde{y}_K \in \mathrm{S_{I\text{-}\{j\}}}$ et $\left(\tilde{y}_K \times \tilde{z}_j\right)$ vérifie $f_{\{j\}}\left(\tilde{y}_K \times \tilde{z}_j\right)=0$,

$\Leftrightarrow \tilde{z}_j \in \mathrm{D}_{f_{\{j\}}}\left(\mathrm{S_{I-\{j\}}}\right)$.

D'où $\mathrm{D_F}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right)=\mathrm{D}_{f_{\{j\}}}\left(\mathrm{S_{I-\{j\}}}\right)$ ■

Corollaire 2

Supposons de plus que l'ensemble de contraintes $f_{\{j\}}$ relie x_j à q variables d'indices notés $j_1, \ldots, j_q$ ($q \leq n$). On a alors :

$$\mathrm{D_F}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right) = \mathrm{D}_{f_{\{j\}}}\left(\prod\nolimits_{j_1\ldots j_q}\left(\mathrm{S_{I-\{j\}}}\right)\right) \qquad (4.8)$$

où on note $\prod_{j_1\ldots j_q}$ la projection sur l'espace des $\left(x_{j_1}\times\cdots\times x_{j_q}\right)$.

Ce corollaire, comme suite logique du ***Corollaire 1***, remplace l'ensemble des solutions $\left(\mathrm{S_{I-\{j\}}}\right)$ du CSP $\mathcal{H}_{I-\{j\}}$ par leurs projections $\prod_{j_1\ldots j_q}$ sur l'espace $j_1\times\cdots\times j_q$ qui rassemble les dimensions des variables reliées par une contrainte à la variable x_j.

Preuve

■ D'après le ***Corollaire 1***, $\mathrm{D_F}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right) = \mathrm{D}_{f_{\{j\}}}\left(\mathrm{S}_{I-\{j\}}\right)$. Donc, on peut écrire :

$$\tilde{z}_j \in \mathrm{D_F}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right) \Leftrightarrow \exists\ \tilde{y}_K \in \mathrm{S_{I\text{-}\{j\}}} \text{ tel que } f_{\{j\}}\left(\tilde{y}_K\times\tilde{z}_j\right)=0$$

On a posé que $j_1, \ldots, j_q$ sont les indices des variables reliés par une contrainte à la variable j, donc

$$\tilde{z}_j \in \mathrm{D_F}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right)$$

$\Leftrightarrow \exists\ \tilde{y}_K \in \mathrm{S_{I\text{-}\{j\}}}$ tel que $\prod_{j_1\ldots j_q}\left(\tilde{y}_K\right)\times\tilde{z}_j$ vérifie $f_{\{j\}}\left(\prod_{j_1\ldots j_q}\left(\tilde{y}_K\right)\times\tilde{z}_j\right)=0$.

En notant $\prod_{j_1\ldots j_q}\left(\tilde{y}_K\right) = \left(\tilde{y}_{j_1}\times\cdots\times\tilde{y}_{j_q}\right)$

$\Leftrightarrow \quad \exists \quad \left(\tilde{y}_{j_1}\times\cdots\times\tilde{y}_{j_q}\right)\in\prod_{j_1\ldots j_q}\left(\mathrm{S_{I-\{j\}}}\right) \quad | \quad \tilde{y}_{j_1}\times\cdots\times\tilde{y}_{j_q}\times\tilde{z}_J \quad$ vérifie

$f_{\{j\}}\left(\tilde{y}_{j_1}\times\cdots\times\tilde{y}_{j_q}\times\tilde{z}_J\right)=0$.

D'où $\mathrm{D_F}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right) = \mathrm{D}_{f_{\{j\}}}\left(\prod_{j_1\ldots j_q}\left(\mathrm{S_{I-\{j\}}}\right)\right)$ ■

La détermination de la projection des solutions $\prod_{j_1 \ldots j_q}\left(S_{I-\{j\}}\right)$ dans $\mathcal{H}_{I-\{j\}}$ n'est pas toujours évidente. Dans l'idéal, on aimerait pouvoir écrire ce terme sous la forme $\prod_{j_1}\left(S_{I-\{j\}}\right)\times\cdots\times\prod_{j_q}\left(S_{I-\{j\}}\right)$ (cette écriture est cependant fausse en générale). En pratique, on sait seulement calculer la projection sur une dimension. A travers les quelques problèmes que nous avons étudiés, il nous est apparu qu'il était judicieux et faisable d'exprimer cette projection $\prod_{j_1 \ldots j_q}\left(S_{I-\{j\}}\right)$ en fonction de la projection sur une dimension. Ces propos sont illustrés dans les deux exemples ci dessus.

Exemple 4.4

■ Considérons le CSP $\mathcal{H}$: ($x+y=z$ avec $(x, y, z) \in [-1\ 1]\times[-1\ 1]\times[0\ 0]$) constitué de 3 variables et d'une unique contrainte. La Figure 4.16 représente en deux dimensions ce CSP (y en fonction de x). On peut voir à travers ce cas simple que $\Pi_1(S)\times\Pi_2(S)$ est différent de $\Pi_{1\,2}(S)$. En effet, $\Pi_1(S)\times\Pi_2(S)$ est le pavé $[-1\ 1]\times[-1\ 1]$ alors que $\Pi_{1\,2}(S)$ est un segment de droite (voir sur la Figure 4.16).

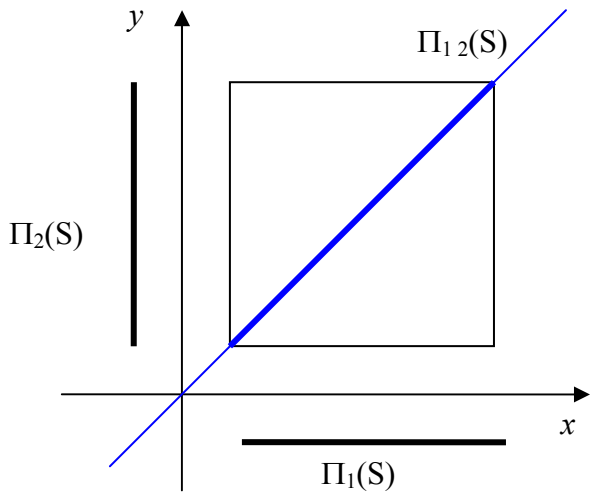


Figure 4.16 : illustration de la difficulté d'exprimer la projection multidimensionnelle des solutions en un produit cartésien de projections monodimensionnelles des solutions. ■

Exemple 4.5

■ Considérons le CSP $\mathcal{H}$: ($x+y=z$ avec $(x, y, z) \in [-1\ 1]\times[-1\ 1]\times[-1\ -0.5]$) constitué de 3 variables et d'une unique contrainte f qui est une bande cette fois-ci. La Figure 4.17 représente en deux dimensions ce CSP (y en fonction de x). On s'intéresse, par exemple, à exprimer $\Pi_{1\,2}(S)$ en fonction de $\Pi_1(S)$. Sur la Figure 4.17, pour chaque point $\tilde{x}$ dans $\Pi_1(S)$, on représente l'ensemble des $\tilde{y}$ tels que $(\tilde{x}\times\tilde{y})$ appartienne à $\Pi_{1\,2}(S)$. Pour cet exemple, on trouve alors, à chaque fois, des intervalles et on illustre que cet ensemble peut s'écrire sous la forme $\Pi_2\left(D_f(\tilde{x})\right)\cap[y]$.

Par la suite, on peut remarquer que si $\tilde{x}$ parcourt l'intervalle $\Pi_1(S)$ et si on considère tous les couples $(\tilde{x}\times\tilde{y})$ avec $\tilde{y} \in D_F(\tilde{x})\cap[y]$, on décrit entièrement $\Pi_{1\,2}(S)$, i.e. $\Pi_{1\,2}(S) = \bigcup_{\tilde{x}\in\Pi_1(S)} \left(\tilde{x}, \Pi_2\left(D_f(\tilde{x})\right)\cap[y]\right)$.

Cette écriture de $\Pi_{1\,2}(S)$ annonce le lemme 1 qui permet d'écrire une projection de solutions sur plusieurs dimensions en fonction d'une projection de solutions sur une dimension.

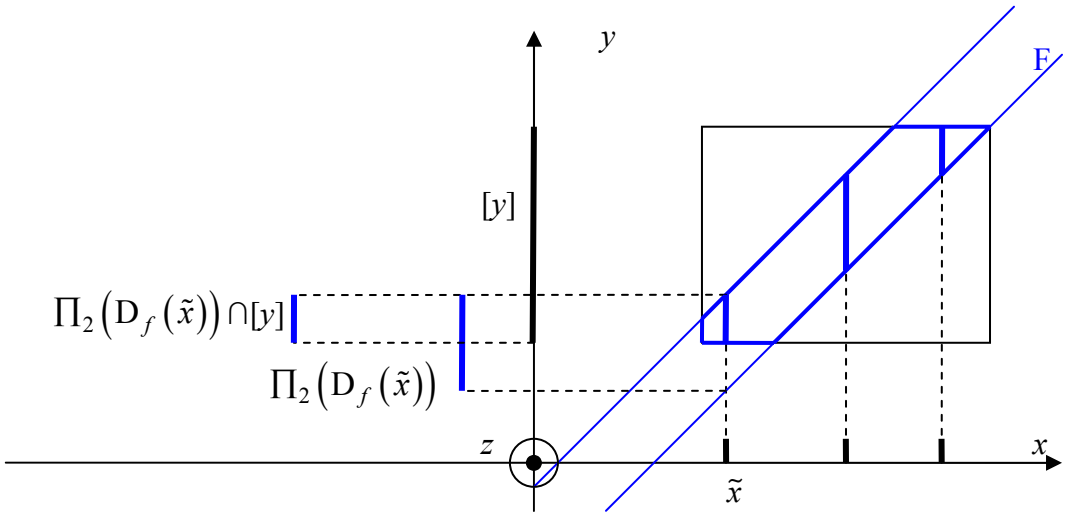


Figure 4.17 : illustration de la difficulté d'exprimer la projection multidimensionnelle des solutions en un produit cartésien de projections monodimensionnelles des solutions. ■

L'idée exprimée à travers cet exemple est écrite formellement au moyen du lemme ci-après. Ce lemme permettra une écriture finale du ***Théorème 3*** à travers un dernier corollaire.

Lemme

On choisit de formuler le résultat pour la première variable mais les résultats qui sont énoncés sont vrais en prenant n'importe quelle indice.

Considérons un CSP $\mathcal{H}$ et son ensemble solution S. On cherche une expression de la projection $\prod_{1\ldots q}(\mathrm{S})$ en fonction de la projection $\prod_{1}(\mathrm{S})$. L'idée est la suivante : pour $\tilde{x}_1$ solution de $\mathcal{H}$, l'ensemble des q−1-uplets $(\tilde{x}_2,\cdots,\tilde{x}_q)$ pour lesquels l'instanciation $(\tilde{x}_1,\cdots,\tilde{x}_q)$ est solution de $\mathcal{H}$ s'écrit : $\prod_{2\ldots q}(\mathrm{D_F}(\tilde{x}_1))\cap[x_2]\times\cdots\times[x_q]$. Par suite, l'ensemble des q-uplets $(\tilde{x}_1,\cdots,\tilde{x}_q)$ solutions de $\mathcal{H}$ est retrouvé lorsqu'on fait le produit cartésien de tous les $\tilde{x}_1$ solutions de $\mathcal{H}$ avec leur « ensemble d'instanciation compatible » $\prod_{2\ldots q}(\mathrm{D_F}(\tilde{x}_1))\cap[x_2]\times\cdots\times[x_q]$. On traduit cette idée de manière plus formelle par l'équation 4.9 suivante :

$$\prod_{1\ldots q}(\mathrm{S}) = \bigcup_{\tilde{x}_1\in\prod_1(\mathrm{S})}\left(\tilde{x}_1,\left(\prod_{2\ldots q}\left(\mathrm{D}_{f_{I-\{1\}}}(\tilde{x}_1)\right)\cap[x_2]\times\cdots\times[x_q]\right)\right) \qquad (4.9)$$

Notons qu'on présente ce lemme de façon simplifiée en considérant des indices triées de 1…q au lieu de q indices quelconques mais le résultat est aussi vrai pour une suite de q indices quelconques.

Preuve

■ Montrons l'égalité 4.9. On a les équivalences suivantes :

$(\tilde{x}_2,\cdots,\tilde{x}_q) \in \prod_{2\ldots q}(\mathrm{D_F}(\tilde{x}_1))\cap[x_2]\times\cdots\times[x_q]$

$\Leftrightarrow$ $(\tilde{x}_2,\cdots,\tilde{x}_q)\in[x_2]\times\cdots\times[x_q]$ et $\exists$ $\tilde{x}_{q+1},\cdots,\tilde{x}_n$ pris dans les dimensions complémentaires à {1, ..., q}, tels que l'instanciation $(\tilde{x}_2,\cdots,\tilde{x}_q,\tilde{x}_{q+1},\cdots,\tilde{x}_n)\in\mathrm{D_F}(\tilde{x}_1)$.

Or $(\tilde{x}_2,\cdots,\tilde{x}_q,\tilde{x}_{q+1},\cdots,\tilde{x}_n)\in\mathrm{D_F}(\tilde{x}_1)$ $\Leftrightarrow$ $(\tilde{x}_1,\tilde{x}_2,\cdots,\tilde{x}_q,\tilde{x}_{q+1},\cdots,\tilde{x}_n)$ vérifie toutes les contraintes F $\Leftrightarrow$ $(\tilde{x}_1,\tilde{x}_2,\cdots,\tilde{x}_q,\tilde{x}_{q+1},\cdots,\tilde{x}_n)\in\mathrm{S}$.

D'où, on peut réécrire les équivalences:

$\left(\tilde{x}_2,\cdots,\tilde{x}_q\right) \in \prod_{2\cdots q}\left(\mathrm{D_F}\left(\tilde{x}_1\right)\right)\cap\left[x_2\right]\times\cdots\times\left[x_q\right] \Leftrightarrow \left(\tilde{x}_2,\cdots,\tilde{x}_q\right)\in\left[x_2\right]\times\cdots\times\left[x_q\right]$ et $\exists$ $\tilde{x}_{q+1},\cdots,\tilde{x}_n$ pris dans I\{1, ..., q}, tels que l'instanciation $\left(\tilde{x}_1,\tilde{x}_2,\cdots,\tilde{x}_q,\tilde{x}_{q+1},\cdots,\tilde{x}_n\right) \in \mathrm{S}$ $\Leftrightarrow \left(\tilde{x}_1,\tilde{x}_2,\cdots,\tilde{x}_q\right) \in \prod_{1\cdots q}(\mathrm{S})$.

En définitive on peut donc écrire que :

$\left(\tilde{x}_2,\cdots,\tilde{x}_q\right) \in \prod_{2\cdots q}\left(\mathrm{D_F}\left(\tilde{x}_1\right)\right)\cap\left[x_2\right]\times\cdots\times\left[x_q\right] \Leftrightarrow \left(\tilde{x}_1,\tilde{x}_2,\cdots,\tilde{x}_q\right) \in \prod_{1\cdots q}(\mathrm{S})$.

D'où l'égalité (4.9) : $\prod_{1\cdots q}(\mathrm{S}) = \bigcup_{\tilde{x}_1\in\prod_1(\mathrm{S})}\left(\tilde{x}_1,\left(\prod_{2\cdots q}\left(\mathrm{D_F}\left(\tilde{x}_1\right)\right)\cap\left[x_2\right]\times\cdots\times\left[x_q\right]\right)\right)$ ■

Ce lemme nous permet d'écrire, grâce à un corollaire 3, le théorème 3 sous une forme plus avancée, notamment, en écrivant la projection sur l'espace des indices $j_1,\ldots, j_q$ en fonction d'un seule dimension. Rappelons que l'ensemble de contraintes $f_{\{j\}}$ relie x_j à q variables d'indices notés $j_1,\ldots, j_q$ ($q\leq n$). On a alors le ***Corollaire 3*** ci-dessous :

Corollaire 3

Pour plus de lisibilité, ce corollaire est formulé pour x_{j_1}, en sachant que les résultats qui seront énoncés restent vrais pour n'importe quel indice dans $j_1, \ldots, j_q$. Le domaine de consistance $\mathrm{D_F}\left(\underset{i\in I-\{j\}}{\times}\left[x_i\right]\right)$ vérifie :

$$\mathrm{D_F}\left(\underset{i\in I-\{j\}}{\times}\left[x_i\right]\right) = \mathrm{D}_{f_{\{j\}}}\left(\bigcup_{\tilde{x}_{j1}\in\prod_{j1}\left(\mathrm{S}_{\mathrm{I}-\{j\}}\right)}\left(\tilde{x}_{j_1},\left(\prod_{j_2\cdots j_q}\left(\mathrm{D}_{f_{I-\{j\}}}\left(\tilde{x}_{j_1}\right)\right)\cap\left[x_{j_2}\right]\times\cdots\times\left[x_{j_q}\right]\right)\right)\right)$$

Preuve

■D'après le ***Corollaire 2*** on a l'éq. (4.8) : $\mathrm{D_F}\left(\underset{i\in I-\{j\}}{\times}\left[x_i\right]\right) = \mathrm{D}_{f_{\{j\}}}\left(\prod_{j_1\cdots j_q}\left(\mathrm{S}_{\mathrm{I}-\{j\}}\right)\right)$. De plus, appliquons le ***lemme*** au CSP $\mathcal{H}_{I-\{j\}}$ qui est, rappelons le , constitué des contraintes $f_{I-\{j\}}$ et des variables autres que j. La projection $\prod_{j_1\cdots j_q}\left(\mathrm{S}_{\mathrm{I}-\{j\}}\right)$ peut alors s'écrire :

$$\prod_{j_1\cdots j_q}\left(\mathrm{S}_{\mathrm{I}-\{j\}}\right) = \bigcup_{\tilde{x}_{j1}\in\prod_{j1}\left(\mathrm{S}_{\mathrm{I}-\{j\}}\right)}\left(\tilde{x}_{j_1},\left(\prod_{j_2\cdots j_q}\left(\mathrm{D}_{f_{I-\{j\}}}\left(\tilde{x}_{j_1}\right)\right)\cap\left[x_{j_2}\right]\times\cdots\times\left[x_{j_q}\right]\right)\right) \quad (4.10)$$

En combinant (4.8) et (4.10), on obtient l'expression du ***Corollaire 3***. ■

3 LIEN ENTRE LES TECHNIQUES DE PROPAGATION DE CONTRAINTE ET LE CALCUL DE DOMAINES DE CONSISTANCE.

Dans cette partie, on met en exergue le lien entre les techniques de propagation de contraintes (ou algorithme de type Waltz), avec l'approche de calcul de solutions grâce aux domaines de consistance. Le but de cette étude est de pouvoir ramener un algorithme de Waltz quelconque à des calculs de domaines de consistance. De facto, l'enjeu deviendra double. Tout d'abord, on pourra présenter les calculs de domaine de consistance comme une généralisation des algorithmes de propagations de contrainte, mais surtout, on pourra montrer, qu'en présence de cycles, le calcul de domaines de consistance offre plus de possibilités d'obtenir les domaines globalement consistants.

La démarche adoptée est alors la suivante : on écrira d'abord toute projection de contrainte avec un calcul de domaine de consistance. Puis, l'étape suivante sera d'écrire également tout résolveur d'une contrainte comme un calcul de domaine de consistance. Enfin, on se basera sur une écriture formelle des algorithmes de Waltz en fonction de résolveurs de contraintes, pour faire le lien voulu.

3.1 Projection selon une contrainte et domaine de consistance

Les projections de contraintes constituent la base des algorithmes de propagation de contraintes. En règle générale, on cherche les plus petits intervalles possibles, englobant les projections de l'ensemble des solutions. On propose alors, dans cette partie, de définir des projections intervalles et de les écrire avec des domaines de consistance.

Considérons une contrainte f_i et $[x_j]$ un sous-pavé de [x]. Rappelons que, $\prod_{f_i,j}([x])$, la projection intervalle de [x] selon la $i^{ième}$ contrainte et la $j^{ième}$ dimension vérifie : $\prod_{f_i,j}([x]) = \left\{ x_j \in [x_j] \,\middle|\, \exists x_k \in [x_k] \text{ pour tout } k \in I - \{i\} \text{ et } f_i(x_1, \cdots, x_n) = 0 \right\}$. Posons $\mathcal{H}_i$ = ($f_i([x]) = 0$, $x \in [x]$) le CSP ayant pour seule contrainte f_i. On a alors $\prod_{f_i,j}([x])$ est équivalente à la projection de l'ensemble des solutions S_i du CSP $\mathcal{H}_i$ selon la $j^{ième}$ dimension notée $\Pi_j(S_i)$, i.e :

$$\Pi_{f_i,j}\left([x]\right) = \Pi_j\left(\mathrm{S}_i\right) \tag{4.11}$$

D'après le ***Théorème 1*** appliqué au CSP $\mathcal{H}_i$ on a :

$$\Pi_j(S_i) = \left[x_j\right] \cap \mathrm{D}_{f_i}\left(\underset{k\in I-\{j\}}{\times}\left[x_k\right]\right) \tag{4.12}$$

en combinant (4.11) et (4.12), on obtient :

$$\Pi_{f_i,j}\left([x]\right) = \left[x_j\right] \cap \mathrm{D}_{f_i}\left(\underset{k\in I-\{j\}}{\times}\left[x_k\right]\right) \tag{4.13}$$

De ce résultat, on peut conclure qu'un calcul de projection se ramène à un calcul de domaine de consistance. Les algorithmes de propagation de contraintes étant basés sur des projections, on est donc en mesure de faire le parallèle avec les calculs de domaines de consistance. Pour cela, on considère dans un deuxième temps dans le paragraphe suivant, les résolveurs de contraintes.

3.2 Résolveur d'une contrainte f_i et domaine de consistance

Rappelons que, par définition, un résolveur pour une contrainte permet de trouver le plus petit pavé [x] consistant avec la contrainte. En pratique, pour trouver les domaines intervalles consistants avec une contrainte f_i, celle-ci est projetée sur toutes les variables concernées pour contracter les domaines concernés. Les autres domaines restent insensibles à la contrainte. De manière formelle, pour une contrainte f_i, on peut écrire son sous résolveur noté R_{f_i}, sous la forme de projections intervalles de la manière suivante :

$$R_{f_i}[x] = \overset{n}{\underset{j=1}{\times}}\left(\left[\Pi_{f_i,j}\left([x]\right)\right]\right) \tag{4.14}$$

Or d'après (4.13) $\Pi_{f_i,j}\left([x]\right) = \left[x_j\right] \cap \mathrm{D}_{f_i}\left(\underset{k\in I-\{j\}}{\times}\left[x_k\right]\right)$

D'où

$$R_{f_i}[x] = \overset{n}{\underset{j=1}{\times}}\left(\left[\left[x_j\right] \cap \mathrm{D}_{f_i}\left(\underset{k\in I-\{j\}}{\times}\left[x_k\right]\right)\right]\right) \tag{4.15}$$

L'équation (4.15) permet d'exprimer le résolveur pour chaque contrainte avec un calcul équivalent de domaine de consistance. Cette écriture peut se faire à l'identique avec un algorithme de Waltz comme démontré dans la section 3.3 ci dessous.

3.3 Algorithme de propagation de contraintes (ou algorithme de Waltz)

Rappelons que le principe d'un algorithme de Waltz est d'utiliser la stratégie qui consiste en la répétition de la contraction des variables du CSP selon toutes les contraintes prises une à une, avec un ordre non déterminé a priori, tant qu'il y'a au moins une variable de contractée.

Pour écrire de manière formelle un algorithme de Waltz, notons $J = \{j_1, \ldots, j_p\}$ un ensemble d'indices, de rang p, dans $\{1, \ldots, p\}$ tel que $j_i \neq j_k$ pour $i \neq k$ i.e. J est toujours l'ensemble des indices de 1 à p dans un ordre quelconque. Considérons alors un algorithme de Waltz avec les contraintes prises consécutivement selon J. Un algorithme de Waltz s'écrit donc de manière formelle :

Tant que la contraction de $[x]$ est assez grande, répéter :

$$[x] = R_{f_{j_1}}\left(\cdots\left(R_{f_{j_p}}([x])\right)\cdots\right) \tag{4.16}$$

Fin tant que

avec $R_{f_{j_i}}[x] = \underset{j_i=1}{\overset{n}{\times}}\left(\left[\prod_{f_{j_i},j}([x])\right]\right)$ pour j_i de 1 à p . Or d'après (4.15) R_{f_i} pour tout i de 1 à p est équivalent à des calculs de domaines de consistance. D'où, d'après l'écriture d'un algorithme de Waltz selon (4.16), on peut conclure que l'algorithme de Waltz peut être exprimé à l'aide de calculs de domaines de consistance.

4 EXEMPLES DE RESOLUTION DE CYCLE GRACE A DES CALCULS DE DOMAINES DE CONSISTANCE

Nous proposons de résoudre deux exemples pour lesquels les algorithmes de propagation de contraintes aboutissent à une consistance locale. A l'aide des domaines de consistance, on va pouvoir développer des algorithmes répondant aux deux objectifs de temps de calcul et de

consistance globale. Nous avons choisi de traiter un exemple linéaire et un exemple non linéaire avec une classe de cycles représentée sur la Figure 4.18.

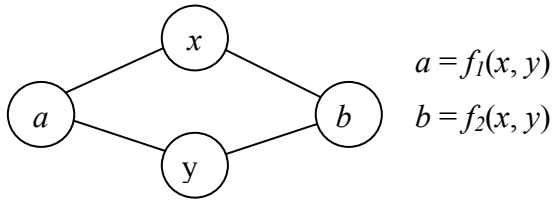


Figure 4.18 : exemple illustratif de cycle.

4.1 Méthodologie générale

On propose ici de fournir, en préambule des exemples traités, une démarche générale qui nous est apparue récurrente, pour la résolution de cycles par des domaines de consistance. En règle générale, le point de départ consiste à appliquer le ***Théorème 1***, qui permet de ramener le calcul des solutions sur une dimension *j* au domaine de consistance $\mathrm{D_F}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right)$ associé au sous-pavé $\underset{i\in I-\{j\}}{\times}[x_i]$ correspondant aux dimensions complémentaires à *j*. Toute la difficulté se reporte donc fort logiquement à ce calcul de domaine de consistance, qui se fait en général en suivant les 4 étapes suivantes :

- ***Etape 1 : évaluation des bornes du domaine de consistance*** $\mathrm{D_F}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right)$.

  Lors de cette étape, on cherche une expression littérale du domaine de consistance $\mathrm{D_F}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right)$. Pour cela, il faut dans certains cas, commencer par isoler les contraintes concernant la variable *j* des autres en se servant du ***Théorème 3***, notamment, dans les situations où toutes les contraintes ne concernent pas la dimension *j*. Ceci permet de ramener le problème au calcul du domaine de consistance associé à un ensemble réduit à une projection de solution $\prod_{j_1\ldots j_q}\left(\mathrm{S}_{\mathrm{I}-\{j\}}\right)$ d'un nouveau CSP $\mathcal{H}_{I-\{j\}}$, et pour un ensemble réduit de contraintes $f_{\{j\}}$. La première difficulté apparaissant ici, vient du calcul des projections de solutions $\prod_{j_1\ldots j_q}\left(\mathrm{S}_{\mathrm{I}-\{j\}}\right)$. En effet, selon la complexité du cycle à résoudre, cette projection peut s'avérer très difficile à faire. D'une part, il faut espérer que le nouveau CSP $\mathcal{H}_{I-\{j\}}$ puisse être résolu sans trop de difficultés. D'autre part, on part

souvent du fait que la projection de solutions sur une dimension est faisable pour appliquer le ***Corollaire 3***, du ***Théorème 3*** qui permet, d'exprimer cette projection de solution en fonction de la projection sur une des dimensions.

Après avoir dissocié si nécessaire, les contraintes concernant la dimension *j* des autres, il faut toujours passer par l'étape qui consiste à isoler les variables apparaissant plus de deux fois dans l'expression du domaine de consistance à calculer en utilisant la règle énoncée dans le ***Théorème 2***. Après avoir appliqué ce théorème, on peut calculer le domaine de consistance associé à un sous-pavé (ponctuel pour certaines dimensions), en faisant une intersection de tous les domaines de consistance de ce sous-pavé calculés pour les contraintes prises unes à unes. Sachant que le domaine de consistance, lorsque l'on considère une unique contrainte, revient à un calcul d'image (propriété 2), on obtient pour le domaine de consistance de ce sous-pavé une expression sous forme d'intersections d'intervalles qui s'exprime à l'aide des fonctions Sup et Inf sous un unique intervalle (exemple : [1 3]∩[2 4] = [Sup(1,2), Inf(3,4)]). Par suite, le domaine $\mathrm{D_F}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right)$ peut alors s'exprimer comme une réunion d'intervalles ayant pour borne des fonctions Sup et Inf. D'où l'on aboutit à une expression littérale de $\mathrm{D_F}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right)$ voulue dans cette première étape.

- ***Etape 2 : caractérisation des bornes de l'expression de*** $\mathrm{D_F}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right)$

Dans cette partie, on cherche à caractériser les bornes dans l'expression de $\mathrm{D_F}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right)$, en étudiant les fonctions Sup et Inf. Généralement, on cherche à trouver un recouvrement de l'espace où on évalue ces fonctions tel que, pour chaque composante du recouvrement, on pourra écrire le domaine de consistance $\mathrm{D_F}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right)$ sans les Inf et les Sup. Par suite, on utilise la propriété sur le domaine de consistance d'une réunion pour obtenir $\mathrm{D_F}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right)$ en faisant un calcul de ce domaine dans chaque composante du recouvrement calculé.

Il faut noter que ce recouvrement, selon les problèmes à résoudre peut s'avérer fastidieux à trouver, puisqu'il s'agit d'étudier des variations de fonctions.

- ***Etape 3 : élimination des intervalles impropres dans l'expression de*** $\mathrm{D_F}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right)$ ***pour le calcul d'une réunion d'intervalles***

A ce stade du calcul, on connaît un recouvrement (dans l'espace où on évalue les Sup et Inf), où on a une expression du domaine $\mathrm{D_F}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right)$ en ayant éliminé les Sup et Inf. Il faut à présent faire la réunion pour chaque élément du recouvrement. Une subtilité ici est d'éviter de faire les réunions d'intervalles sans avoir éliminé des intervalles dits impropres (borne Inf > borne Sup, comme [4, 1] par exemple). En effet, on doit impérativement éliminer les intervalles impropres pour éviter de faire des évaluations fausses comme dans l'exemple suivant : [4, 1] ∪ [5, 6] = [4 6] alors qu'en réalité [4, 1] ∪ [5, 6] = [5 6].

Pour éliminer ces intervalles, on utilise la même technique que dans l'étape 2 lorsqu'il était question de supprimer les Inf et Sup. On étudie ainsi les bornes pour connaître les parties où ces intervalles sont vides. Ceci donne de nouvelles parties dans les recouvrements où les domaines de consistances sont non vides. Il suffit alors dans chaque composante du recouvrement de ne considérer que les parties pour lesquelles les intervalles sont propres, lorsqu'on passera à l'étape qui suit qui est le calcul de l'union dans l'expression de $\mathrm{D_F}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right)$.

- ***Etape 4 : Calcul d'une expression finale de*** $\mathrm{D_F}\left(\underset{i\in I-\{j\}}{\times}[x_i]\right)$ ***en ayant supprimé les réunions***

Après avoir supprimé les intervalles impropres, la réunion des intervalles se fera en faisant une nouvelle étude des variations des fonctions intervenant dans les bornes. Lorsque cette étude est faite, on obtient un algorithme de mise en œuvre simple de manière générale, mais difficile à détailler.

Dans les deux exemples linéaire et non linéaire qui suivent, on propose de traiter les cycles rencontrés en suivant la démarche en 4 étapes présentée dans cette section. Les deux exemples sont décrits en détail dans les paragraphes 4.2 et 4.3.

4.2 Exemple Linéaire détaillé

Considérons un CSP $\mathcal{H} = (F(a, b, x, y) = 0 \mid (a \times b \times x \times y) \in [a_1, a_2] \times [b_1, b_2] \times [x_1, x_2] \times [y_1, y_2])$ où F est constitué des 2 contraintes $f_1 : y+x=a$ et $f_2 : y-x=b$.

Ce CSP contient un cycle et donc un algorithme de Waltz peut ne pas atteindre la consistance globale. De plus, son temps de calcul n'est pas connu à l'avance. On propose d'utiliser le formalisme des domaines de consistance pour résoudre ce CSP en temps connu a priori tout en atteignant la consistance globale.

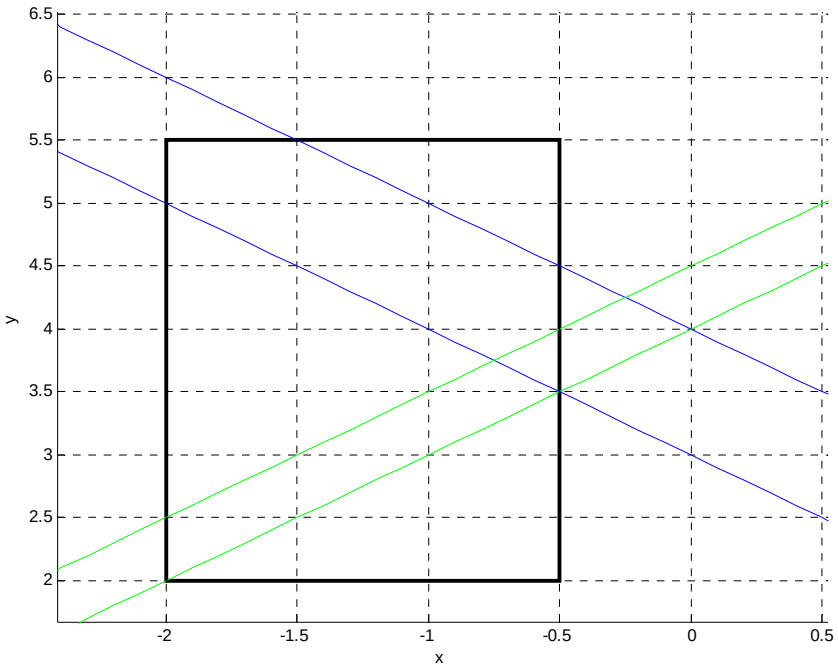


Figure 4.19 : vue en dimension 2 de l'exemple linéaire (pavé $[x]\times[y]$ et bandes de contraintes).

4.2.1 Détermination d'un résolveur basé sur les domaines de consistance

Considérons en premier lieu la variable y et calculons son ensemble de solutions $\Pi_4(S)$. On a choisi de traiter d'abord la variable y pour insister sur le fait qu'on peut indépendamment calculer les $\Pi_i(S)$ de façon aléatoire.

4.2.1.1 Calcul de $\Pi_4(S)$ (selon y)

Appliquons le ***Théorème 1*** pour déterminer $[y]$ globalement consistant avec $\mathcal{H}$. On a : $\Pi_4(S) = [y] \cap D_F([a]\times[b]\times[x])$. Calculer $\Pi_4(S)$ revient donc à calculer le domaine de consistance $D_F([a]\times[b]\times[x])$. Dans cette optique, le calcul a été décomposé selon 4 différentes étapes présentées dans la section 4.1.

1. *Etape 1 : évaluation des bornes de l'intervalle* $D_F([a]\times[b]\times[x])$

La variable x étant la seule à apparaître plus d'une fois dans les équations, d'après le ***Théorème 2***, pour un $x_0 \in [x]$, le domaine de consistance associé à $[a]\times[b]\times x_0$, $D_F([a]\times[b]\times x_0)$ peut être calculé grâce à une intersection des domaines obtenus en prenant les contraintes séparément :

$$D_F([a]\times[b]\times x_0) = D_{f_1}([a]\times[b]\times x_0) \cap D_{f_2}([a]\times[b]\times x_0)$$

$$= ([a]-x_0) \cap ([b]+x_0) = ([a_1, a_2]-x_0) \cap ([b_1, b_2]+x_0)$$

$$= ([a_1-x_0, a_2-x_0]) \cap ([b_1+x_0, b_2+x_0])$$

$$= [\mathrm{Sup}(a_1-x_0, b_1+x_0), \mathrm{Inf}(a_2-x_0, b_2+x_0)]$$

Connaissant la valeur des bornes du domaine de consistance $D_F([a]\times[b]\times x_0)$, pour un x_0 donné, l'étape suivante va être alors de caractériser ses bornes en fonction de la position de x_0 dans IR.

2. *Etape 2 : caractérisation des bornes en fonction de* x_0

On cherche à caractériser le domaine lorsque x_0 parcourt IR notamment en étudiant les bornes de l'intervalle. Faisons une étude de la différence de fonctions :

$$\Delta(x) = (a-x)-(x+b) = a-b-2x$$

Δ est décroissante et s'annule pour $x_{ab} = \frac{a-b}{2}$. Le Tableau 4.1 positionne les fonctions $a-x$ et $b+x$ selon x

	$]-\infty, x_{ab}]$	$[x_{ab}, +\infty[$
$\Delta(x)$	>0	<0
$\text{Sup}\{a-x, b+x\}$	$a-x$	$b+x$
$\text{Inf}\{a-x, b+x\}$	$b+x$	$a-x$

Tableau 4.1 : comparaison des fonctions de x : $a-x$ et $b+x$.

En remplaçant dans ce tableau le couple (a,b) par chaque couple (a_1,b_2) et (a_2,b_1) on obtient 3 secteurs (P_1, P_2, P_3) qui recouvrent IR et pour lesquels les bornes de $D_F([a]\times[b]\times x)$ sont connues (voir sur la Figure 4.20 et la Figure 4.21).

- Si $x_{a_1b_1} < x_{a_2b_2}$

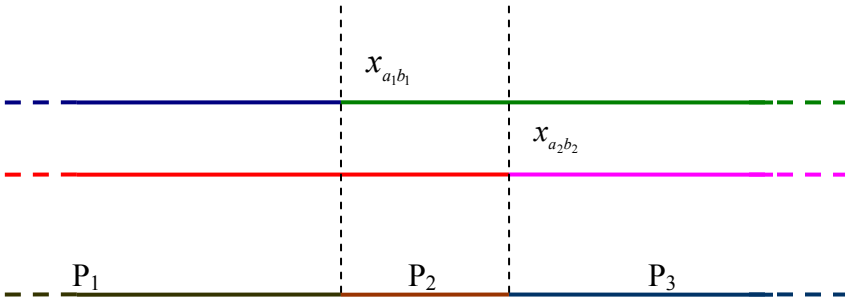


Figure 4.20 : représentation des 3 secteurs P_1, P_2 et P_3.

	$P_1 =]-\infty, x_{a_1b_1}]$	$P_2 = [x_{a_1b_1}, x_{a_2b_2}]$	$P_3 = [x_{a_2b_2}, +\infty[$
$\text{Sup}\{a_1-x, b_1+x\}$	a_1-x	b_1+x	b_1+x
$\text{Inf}\{a_2-x, b_2+x\}$	b_2+x	b_2+x	a_2-x
$D_F([a]\times[b]\times x)$	$[a_1-x, b_2+x]$	$[b_1+x, b_2+x]$	$[b_1+x, a_2-x]$

Tableau 4.2 : expression de $D_F([a]\times[b]\times x)$ dans les secteurs P_1, P_2 et P_3.

- Si $x_{a_2b_2} < x_{a_1b_1}$

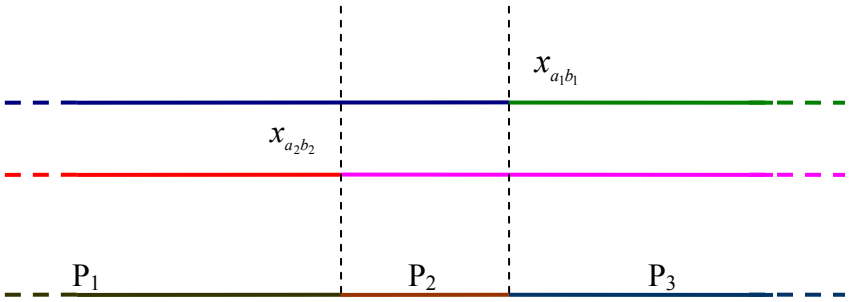


Figure 4.21 : représentation des 3 secteurs P_1, P_2, P_3.

En suivant le même raisonnement que pour le Tableau 4.2, le Tableau 4.3 donne les valeurs de $D_F([a]\times[b]\times x)$ prises dans les P_i. Remarquons que la différence entre le cas $x_{a_1b_1} < x_{a_2b_2}$ et le cas $x_{a_2b_2} < x_{a_1b_1}$ apparaît pour les x dans P_2.

	$P_1 =]-\infty,\ x_{a_2b_2}\ ]$	$P_2 = [\ x_{a_2b_2}\ ,\ x_{a_1b_1}\ ]$	$P_3 = [\ x_{a_1b_1}\ , +\infty[$
$D_F([a]\times[b]\times x)$	$[a_1–x, b_2+x]$	$[a_1–x, a_2–x]$	$[b_1+x, a_2–x]$

Tableau 4.3 : expression du domaine de consistance $D_F([a]\times[b]\times x)$ dans les espaces P_i.

On a donc une expression littérale du domaine de consistance $D_F([a]\times[b]\times x)$, sans les Sup et Inf, dans des secteurs précis de IR en fonction de x. Il faut alors faire la réunion pour obtenir $D_F([a]\times[b]\times[x])$.

3. *Etape 3 : élimination des x pour lesquels l'intervalle* $D_F([a]\times[b]\times x)$ *est impropre (vide) pour obtenir le domaine, réunion sur tous les* x, $D_F([a]\times[b]\times[x])$.

Connaissant la valeur de $D_F([a]\times[b]\times x)$ dans IR, on va utiliser les propriétés sur la réunion pour calculer le domaine de consistance $D_F([a]\times[b]\times[x])$. On a :

$$[a]\times[b]\times[x] = \bigcup_{i=1,2,3} [a]\times[b]\times([x]\cap P_i) \qquad (4.17)$$

(En effet les P_i formant un recouvrement de IR, on a $[x]=[x]\cap IR = [x]\cap \bigcup_{i=1,2,3} P_i = \bigcup_{i=1,2,3} ([x]\cap P_i)$

Pour calculer $D_F([a]\times[b]\times[x])$, utilisons la propriété sur le domaine de consistance de d'une réunion d'intervalles appliquée à (4.17) :

$$D_F([a]\times[b]\times[x]) = \bigcup_{i=1,2,3} D_F([a]\times[b]\times([x]\cap P_i)) = \bigcup_{i=1,2,3} \bigcup_{x\in[x]\cap P_i} D_F([a]\times[b]\times x) \qquad (4.18)$$

Remarque

■ Pour faire l'union d'intervalles propres et impropres, on doit impérativement éliminer les intervalles impropres pour éviter de faire des évaluations fausses comme dans l'exemple suivant : [4, 1] ∩ [5, 6] = [4 6]. ■

Il est à noter que le domaine de consistance peut être vide (si sa borne sup < borne inf). Par exemple, pour x dans P_1, $D_F([a]\times[b]\times x) = [a_1-x, b_2+x]$, ce domaine peut être impropre (ce qui signifie vide physiquement) si $a_1-x > b_2+x$. Pour P_1, on peut écrire l'équivalence suivante :

$$a_1-x>b_2+x \Leftrightarrow x<\frac{a_1-b_2}{2} \Leftrightarrow x<x_{a_1b_2} \qquad (4.19)$$

On note $P_{1,s}$ (s pour solution), l'ensemble des P_1 pour lesquels $[a_1-x, b_2+x]$ est non vide. D'après l'équation (4.19), on a $P_{1,s} = P_1\cap[\, x_{a_1b_2}, +\infty[$

On fait de même pour P_3 pour supprimer les intervalles impropres. On introduit ainsi $P_{3,s} = P_3\cap]-\infty, x_{a_2b_1}]$.

Pour P_2, on remarque que les intervalles sont toujours non vides (pour tout x, $b_1+x<b_2+x$ et $a_1-x<a_2-x$). On note alors de même $P_{2,s} = P_2$.

4. *Etape 4 : calcul de* $D_F([a]\times[b]\times[x])$ ***connaissant*** $D_F([a]\times[b]\times x)$ ***et les*** $P_{i,s}$

Rappelons que, d'après 4.18, on a $D_F([a]\times[b]\times[x]) = \bigcup_{i=1,2,3} \bigcup_{x\in[x]\cap P_i} D_F([a]\times[b]\times x)$. Après avoir éliminé les intervalles impropres, calculons l'union des $D_F([a]\times[b]\times x)$ dans les $P_{i,s}$.

- Pour P_1 :

$$\bigcup_{x\in[x]\cap P_1} D_F([a]\times[b]\times x) = \bigcup_{x\in[x]\cap P_1} [a_1-x,\ b_2+x] = \bigcup_{x\in[x]\cap P_{1,S}} [a_1-x,\ b_2+x]$$

$$= \left[\operatorname{Inf}\{a_1 - x\}_{x\in[x]\cap P_{1,s}}, \operatorname{Sup}\{b_2 + x\}_{x\in[x]\cap P_{1,s}} \right]$$

Notons $[x]\cap P_{1,S} = [\underline{x}_{1,s}, \overline{x}_{1,s}]$. On a du fait de la monotonie des fonctions b_2+x et a_2-x :

$$\bigcup_{x\in[x]\cap P_1} D_F([a]\times[b]\times[x]) = [a_1 - \overline{x}_{1,s},\ b_2 + \overline{x}_{1,s}] \tag{4.20}$$

- Pour P_2 :
  - Si $x_{a_1b_1} < x_{a_2b_2}$, en posant $[x]\cap P_{2,S} = [\underline{x}_{2,s}, \overline{x}_{2,s}]$ on a :

$$\bigcup_{x\in[x]\cap P_2} D_F([a]\times[b]\times[x]) = [b_1 + \underline{x}_{2,s},\ b_2 + \overline{x}_{2,s}] \tag{4.21}$$

  - Si $x_{a_2b_2} < x_{a_1b_1}$, en posant $[x]\cap P_{2,S} = [\underline{x}_{2,s}, \overline{x}_{2,s}]$ on a :

$$\bigcup_{x\in[x]\cap P_2} D_F([a]\times[b]\times[x]) = [a_1 - \overline{x}_{2,s},\ a_2 - \underline{x}_{2,s}] \tag{4.22}$$

- Pour P_3 :

En posant $[x]\cap P_{3,S} = [\underline{x}_{3,s}, \overline{x}_{3,s}]$, on a :

$$\bigcup_{x\in[x]\cap P_2} D_F([a]\times[b]\times[x]) = [b_1 + \underline{x}_{3,s},\ a_2 - \underline{x}_{3,s}] \tag{4.23}$$

En définitive, on sait calculer l'union sur des $D_F([a]\times[b]\times x)$ sur les P_i. Pour calculer $D_F([a]\times[b]\times[x])$, il suffit alors de faire une simple réunion. Ainsi :

Si $x_{a_1b_1} < x_{a_2b_2}$, d'après les équations (4.20), (4.21) et (4.23) donnant les expressions des domaines de consistance $\bigcup_{x\in[x]\cap P_i} D_F([a]\times[b]\times x)$, on a :

$$D_F([a]\times[b]\times[x]) = [a_1 - \overline{x}_{1,s},\ b_2 + \overline{x}_{1,s}] \cup [b_1 + \underline{x}_{2,s},\ b_2 + \overline{x}_{2,s}] \cup [b_1 + \underline{x}_{3,s},\ a_2 - \underline{x}_{3,s}]$$

Si $x_{a_2b_2} < x_{a_1b_1}$, d'après (4.20), (4.22) et (4.23) on a :

$$D_F([a]\times[b]\times[x]) = [a_1 - \overline{x}_{1,s},\ b_2 + \overline{x}_{1,s}] \cup [a_1 - \overline{x}_{2,s},\ a_2 - \underline{x}_{2,s}] \cup [b_1 + \underline{x}_{3,s},\ a_2 - \underline{x}_{3,s}] \tag{4.24}$$

On a donc obtenu un algorithme permettant d'obtenir l'intervalle y solution, résumé dans le Tableau 4.4. L'étape suivante est le calcul des solutions $\Pi_1(S)$ et $\Pi_2(S)$ et $\Pi_3(S)$ (dans les a, b et x).

Algorithme DCsolve : Input([x],[y],[a],[b]), Output([y])
% sous résolveur de l'exemple linéaire

% points particuliers

$x_{a_1b_1} = \frac{a_1 - b_1}{2}$, $x_{a_2b_2} = \frac{a_2 - b_2}{2}$, $x_{a_2b_1} = \frac{a_2 - b_1}{2}$, $x_{a_1b_2} = \frac{a_1 - b_2}{2}$

% calcul de P_i (Recouvrement de IR où on connaît précisément les expressions des domaines de consistance)

$P_1 =]-\infty, \min(x_{a_1b_1}, x_{a_2b_2})]$, $P_2 = [\min(x_{a_1b_1}, x_{a_2b_2}), \max(x_{a_1b_1}, x_{a_2b_2})]$,

$P_3 = [\max(x_{a_1b_1}, x_{a_2b_2}), +\infty[$

% Partie des P_i où les domaines de consistance sont non vides

$P_{1,s} = P_1 \cap [x_{a_1b_2}, +\infty[$, $P_{2,s} = P_2$, $P_{3,s} = P_3 \cap]-\infty, x_{a_2b_1}]$

% Intersection des $P_{i,s}$ avec l'intervalle x

$[x_{1,s}] = [x] \cap P_{1,s}$, $[x_{2,s}] = [x] \cap P_{2,s}$, $[x_{3,s}] = [x] \cap P_{3,s}$

% Calcul des domaines de consistance

Si $[x_{1,s}] \neq \phi$, $[y_{1,s}] = [y] \cap [a_1 - \overline{x}_{1,s}, b_2 + \overline{x}_{1,s}]$ Sinon $[y_{1,s}] = \phi$ Fin

Si $(x_{a_1b_1} < x_{a_2b_2})$

  Si $[x_{2,s}] \neq \phi$, $[y_{2,s}] = [y] \cap [b_1 + \underline{x}_{2,s}, b_2 + \overline{x}_{2,s}]$ Sinon $[y_{2,s}] = \phi$ Fin

Sinon

  Si $[x_{2,s}] \neq \phi$, $[y_{2,s}] = [y] \cap [a_1 - \overline{x}_{2,s}, a_2 - \underline{x}_{2,s}]$ Sinon $[y_{2,s}] = \phi$ Fin

Fin

Si $[x_{3,s}] \neq \phi$, $[y_{3,s}] = [y] \cap [b_1 + \underline{x}_{3,s}, a_2 - \underline{x}_{3,s}]$ Sinon $[y_{3,s}] = \phi$ Fin

% Calcul du domaine solution

$[y] \cap D_F([x],[a],[b]) = [y_{1,s}] \cup [y_{2,s}] \cup [y_{3,s}]$.

Tableau 4.4 : contracteur permettant d'obtenir les valeurs dans [y] globalement consistantes.

4.2.1.2 Calcul de $\Pi_3(S)$ (selon x)

Pour ce problème particulier, on peut utiliser la symétrie entre x et y pour en déduire $\Pi_3(S)$. En effet, le système de contraintes $\{y+x = a$ et $y–x = b\}$, peut s'écrire de manière équivalente $\{x+y = a$ et $x–y = –b\}$. Donc il suffit d'appliquer le même algorithme décrit pour y, à cette fois-ci, la variable x en jouant sur l'intervalle $[b]$ ($[b]$ étant égale à $[b_1, b_2]$ pour y, on le remplacera par son opposé $[–b_2, –b_1]$ pour écrire l'algorithme pour x).

On voit, à travers cet exemple, toute la richesse des propriétés des domaines de consistance qui permettent, dans le cas présent, d'utiliser un même algorithme pour x et y en jouant sur des paramètres d'entrées.

4.2.1.3 Calcul de $\Pi_1(S)$ (selon a)

Pour calculer les solutions $\Pi_1(S)$ et $\Pi_2(S)$ dans $[a]$ et $[b]$, un idée adaptée à cet exemple particulier, est d'utiliser la propriété sur les domaines de consistance de systèmes de contraintes équivalents. En effet, les deux systèmes suivant sont équivalents :

$$\begin{cases} y+x = a \\ y–x = b \end{cases} \qquad\qquad \begin{cases} a+b = 2y \\ a–b = 2x \end{cases}$$

Il suffit alors d'appliquer le même algorithme en inversant les rôles. Plus précisément, pour obtenir $\Pi_1(S)$, on joue avec la symétrie entre ces deux systèmes et on remplace dans l'algorithme du Tableau 4.1, les intervalles $[a_1, a_2]$, $[b_1, b_2]$, $[x_1, x_2]$, et $[y_1, y_2]$ respectivement par les intervalles $[2y_1,2y_2]$, $[2x_1,2x_2]$, $[b_1, b_2]$ et $[a_1, a_2]$.

4.2.1.4 Calcul de $\Pi_2(S)$ (selon b)

De même, pour obtenir $\Pi_2(S)$, on remplace dans l'algorithme du Tableau 4.4, les intervalles $[a_1, a_2]$, $[b_1, b_2]$, $[x_1, x_2]$, et $[y_1, y_2]$ respectivement par les intervalles $[2y_1,2y_2]$, $[-2x_2,-2x_1]$, $[b_1, b_2]$ et $[a_1, a_2]$.

4.2.2 Résultats numériques

On choisit dans cette partie de résoudre le CSP étudié, $\mathcal{H} = (F(a, b, x, y) = 0 \mid (a \times b \times x \times y) \in [a_1, a_2] \times [b_1, b_2] \times [x_1, x_2] \times [y_1, y_2])$ où F est constitué des 2 contraintes $y+x = a$ et $y-x = b$. Pour 3 exemples choisis, résolvons ce CSP aussi bien avec l'algorithme de Waltz qu'avec l'algorithme DCsolve basé sur le calcul de domaine de consistance précédemment étudié. On choisit de montrer des résultats sous forme graphique, en représentant les projections sur les espaces x et y des solutions. On représente de surcroît les contraintes qui sont, dans ce cas ci, chacune une bande de droites (x et y vérifient les bandes de droites : $y+x = [a]$ et $y-x = [b]$). Conformément à la théorie, on obtient les intervalles englobant exacts avec l'algorithme basé sur les domaines de consistance, tandis que l'algorithme de Waltz fournit des domaines localement consistants avec un pessimisme plus ou moins grand selon l'exemple choisi.

- CSP 1 : $[x] = [-3\ 3]$; $[y] = [1\ 6]$; $[a] = [3\ 5]$; $[b] = [2\ 4]$.

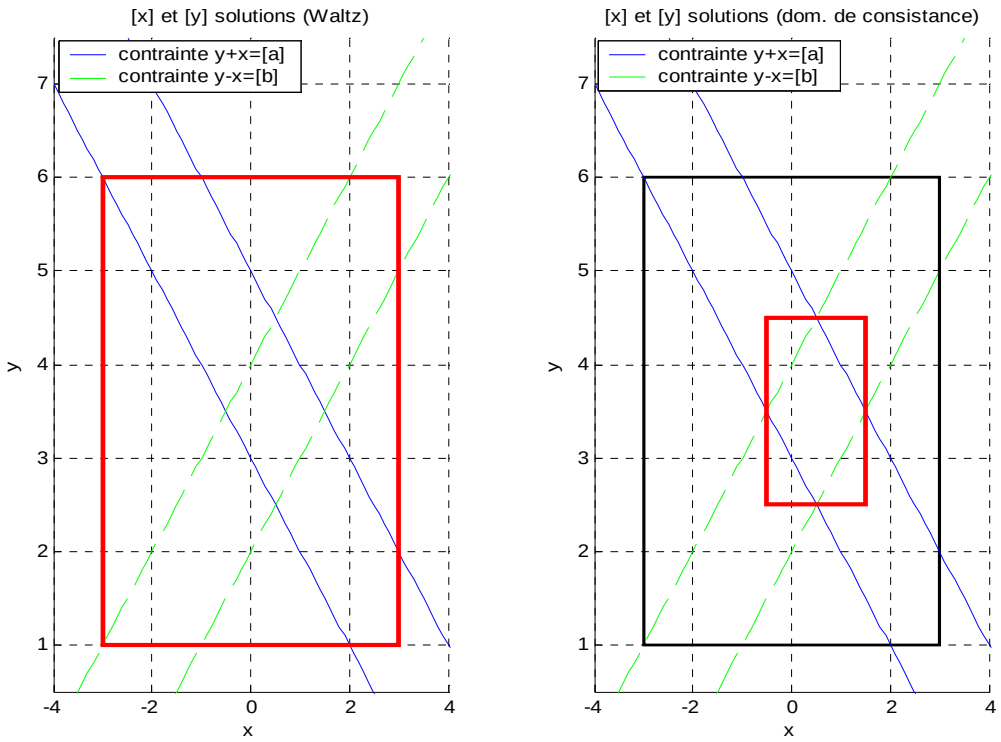


Figure 4.22 : comparaison entre l'algorithme de Waltz (à gauche) et l'algorithme DCsolve (à droite).

- CSP 2 : [x] = [-3 1]; [y] = [1 3] ; [a] = [3 5]; [b] = [2 4].

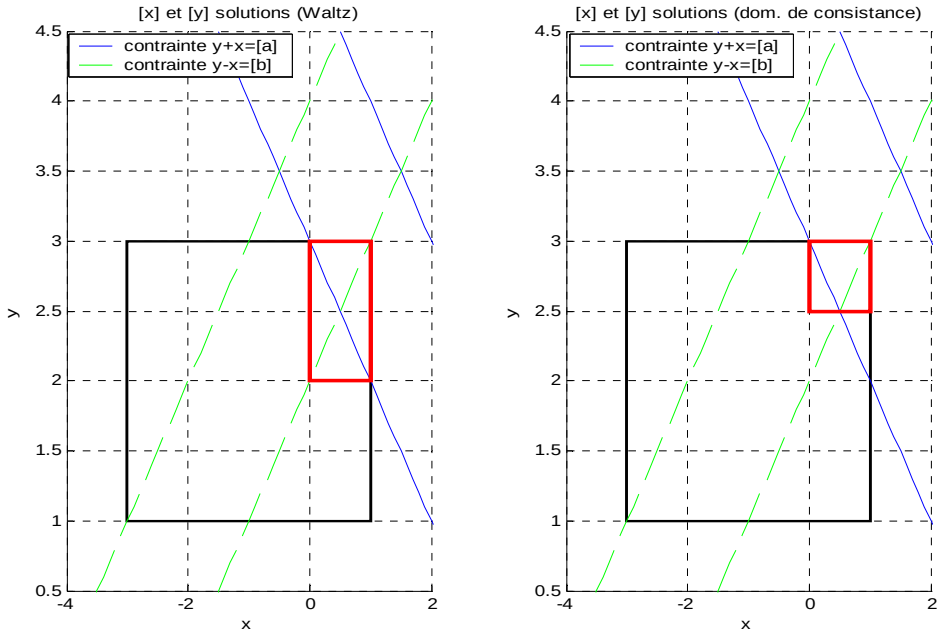


Figure 4.23 : comparaison entre l'algorithme de Waltz (à gauche) et l'algorithme DCsolve (à droite).

- CSP 3 : [x] = [-1 4]; [y] = [3 4] ; [a] = [3 5]; [b] = [2 4].

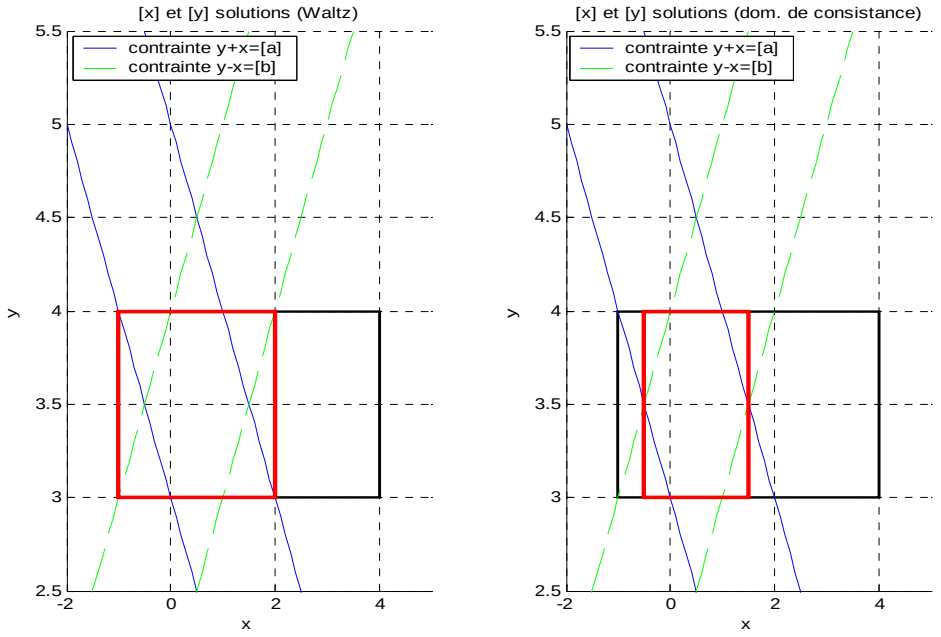


Figure 4.24 : comparaison entre l'algorithme de Waltz (à gauche) et l'algorithme DCsolve (à droite).

4.3 Exemple non linéaire détaillé

Considérons le CSP $\mathcal{H}$ = (F(a, b, x, y) = 0/ X=(a×b×x×y)∈[a_1, a_2] × [b_1, b_2] × [x_1, x_2] × [y_1, y_2]) où F est constitué des 2 contraintes f_1 : y+x−a = 0 et f_2 : y –ln(x)−b = 0

Ce CSP contient un cycle avec des contraintes non linéaires. On propose de le résoudre avec un calcul de domaine de consistance en comparant avec un algorithme de Waltz.

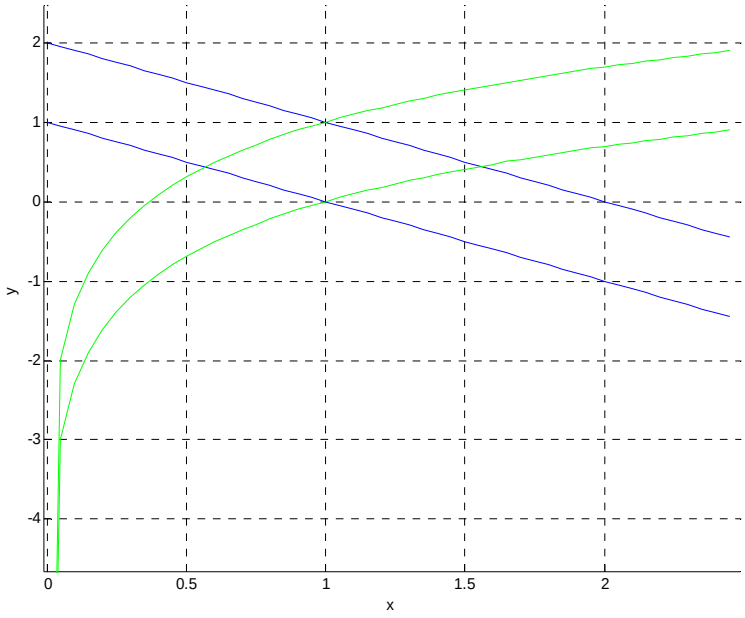


Figure 4.25 : vue sur deux dimensions (x, y) de l'exemple non linéaire.

4.3.1 Détermination d'un résolveur basé sur les domaines de consistance

On propose de commencer par déterminer Π_3(S) l'ensemble des solutions dans [x].

4.3.1.1 Calcul de Π_3(S) (selon x)

Appliquons le ***Théorème 1*** pour la projection sur [x]. On peut alors écrire : Π_3(S) = [x] ∩ D_F([a]×[b]×[y]).

La détermination de Π_3(S) passe par le calcul du domaine de consistance D_F([a]×[b]×[y]). Pour cela, nous distinguons 4 étapes principales :

1. *Etape 1 : évaluation des bornes de l'intervalle* $D_F([a]\times[b]\times y)$

La variable y apparaît deux fois dans les contraintes. En se conformant au ***Théorème 2***, il faut d'abord la considérer comme ponctuelle pour pouvoir calculer le domaine de consistance en considérant les contraintes une à une. Pour un $y_0 \in [y]$, le domaine de consistance associé à $[a]\times[b]\times y_0$, $D_F([a]\times[b]\times y_0)$ est donné par :

$$D_F([a]\times[b]\times y_0) = D_{f_1}([a]\times[b]\times y_0) \cap D_{f_2}([a]\times[b]\times y_0)$$

$$= ([a]-y_0) \cap (\exp(y_0-[b])) = ([a_1-y_0,\ a_2-y_0]) \cap [\exp(y_0-b_2),\ \exp(y_0-b_1)]$$

$$= [\mathrm{Sup}\{a_1-y_0,\ \exp(y_0-b_2)\},\ \mathrm{Inf}\{a_2-y_0,\ \exp(y_0-b_1)\}]$$

2. *Etape 2 : caractérisation des bornes en fonction de* y

On cherche à caractériser le domaine de consistance $D_F([a]\times[b]\times y_0)$, lorsque y_0 parcourt $[y]$. Pour déterminer les bornes, on compare des fonctions de y de la forme $a-y$ et $\exp(y-b)$. On étudie pour cela leur différence : $\Delta(y) = a-y-\exp(y-b)$. Δ est décroissante et ses valeurs parcourent IR. Posons y_{ab} la valeur qui annule Δ. Le Tableau 4.5 qui suit positionne le sup et l'inf des deux fonctions $a- y$ et $\exp(y-b)$ en fonction de y.

	$]-\infty, y_{ab}]$	$[y_{ab}, +\infty[$
Δ	>0	<0
$\mathrm{Sup}\{a-y, \exp(y-b)\}$	$a-y$	$\exp(y-b)$
$\mathrm{Inf}\{a-y, \exp(y-b)\}$	$\exp(y-b)$	$a-y$

Tableau 4.5 : comparaison des fonctions de y : $a-y$ et $\exp(y-b)$.

En remplaçant dans ce tableau le couple (a, b) par les couples (a_1, b_2) et (a_2, b_1), on obtient 3 secteurs (P_1, P_2, P_3) qui recouvrent IR et pour lesquels les bornes de $D_F([a]\times[b]\times y)$ sont connues.

- Si $y_{a_1b_2} < y_{a_2b_1}$

	$P_1 = [-\infty, y_{a_1b_2}]$	$P_2 = [y_{a_1b_2}, y_{a_2b_1}]$	$P_3 = [y_{a_2b_1}, +\infty]$
$D_F([a]\times[b]\times y)$	$[a_1-y, \exp(y-b_1)]$	$[\exp(y-b_2), \exp(y-b_1)]$	$[\exp(y-b_2), a_2-y]$

Tableau 4.6 : expression de $D_F([a]\times[b]\times y)$ dans les espaces P_1, P_2 et P_3.

- Si $y_{a_2b_1} < y_{a_1b_2}$

	$P_1 = [-\infty, y_{a_2b_1}]$	$P_2 = [y_{a_2b_1}, y_{a_1b_2}]$	$P_3 = [y_{a_1b_2}, +\infty]$
$D_F([a]\times[b]\times y)$	$[a_1-y, \exp(y-b_1)]$	$[a_1-y, a_2-y]$	$[\exp(y-b_2), a_2-y]$

Tableau 4.7 : expression de $D_F([a]\times[b]\times y)$ dans les espaces P_1, P_2 et P_3.

On peut remarquer que la différence entre le cas $y_{a_2b_1} < y_{a_1b_2}$ et le cas $y_{a_1b_2} < y_{a_2b_1}$ apparaît pour les y dans P_2.

3. ***Etape 3 : élimination des x pour lesquels l'intervalle*** $D_F([a]\times[b]\times y)$ ***est impropre* (*vide*)**

Pour calculer $D_F([a]\times[b]\times[y])$ utilisons la propriété sur la réunion de domaines de consistance. On a :

$$[a]\times[b]\times[y] = \bigcup_{i=1,2,3} [a]\times[b]\times([y]\cap P_i) \qquad \text{(les } P_i \text{ formant un recouvrement de IR)}$$

$$\Rightarrow D_F([a]\times[b]\times[y]) = \bigcup_{i=1,2,3} D_F([a]\times[b]\times([y]\cap P_i)) = \bigcup_{i=1,2,3} \bigcup_{y\in[y]\cap P_i} D_F([a]\times[b]\times y) \qquad (4.25)$$

Pour faire l'union $\bigcup_{i=1,2,3} \bigcup_{y\in[y]\cap P_i} D_F([a]\times[b]\times y)$, éliminons les $D_F([a]\times[b]\times y)$ vide (borne sup < borne inf), dans chacun des P_i.

- Pour P_1, on a : $\bigcup_{y\in[y]\cap P_i} D_F([a]\times[b]\times y) = \bigcup_{y\in[y]\cap P_i} [a_1-y, \exp(y-b_1)]$. Les intervalles se trouvant dans cette réunion peuvent être vide selon la position de y par rapport à $y_{a_1b_1}$ (qui annule $a_1-y-\exp(y-b_1)$), i.e. $[a_1-y, \exp(y-b_1)] = \phi \Leftrightarrow y \in]-\infty, y_{a_1b_1}]$

- Pour P_2, on doit étudier les deux cas $y_{a_2b_1} < y_{a_1b_2}$ et $y_{a_1b_2} < y_{a_2b_1}$. La remarque ici est que les deux intervalles $[\exp(y-b_2), \exp(y-b_1)]$ et $[a_1-y, a_2-y]$ sont $\neq \phi$ pour tout y dans P_2.

D'où l'ensemble $P_{2,s}$ des éléments de P_2 pour lesquels $[\exp(y-b_2), \exp(y-b_1)] \neq \phi$ (ou $[a_1-y, a_2-y] \neq \phi$) est dans tous les cas égal à P_2.

- Pour P_3, on définit de même la constante $y_{a_2b_2}$ positionnant le domaine pour laquelle $[\exp(y-b_2), a_2-y]$ est non vide en écrivant que : $[\exp(y-b_2), a_2-y] = \phi \Leftrightarrow y \in [\, y_{a_2b_2}, +\infty[$

4. *Etape 4 calcul de* $D_F([a]\times[b]\times[x])$ ***connaissant*** $D_F([a]\times[b]\times x)$ ***et les*** $P_{i,s}$

Rappelons que, d'après 4.25, on a $D_F([a]\times[b]\times[y]) = \bigcup_{i=1,2,3} \bigcup_{y\in[y]\cap P_i} D_F([a]\times[b]\times y)$. Après avoir éliminé les intervalles impropres, calculons l'union des $D_F([a]\times[b]\times x)$ dans les $P_{i,s}$.

- Pour P_1

Notons $P_{1,s}$ l'ensemble des éléments de P_1 pour lesquels $[a_1-y, \exp(y-b_1)]$ est non vide. $P_{1,s}$ s'écrit alors : $P_{1,s} = P_1 \cap [\, y_{a_1b_1}, +\infty[$ et on a,

$$D_F([a]\times[b]\times[y]) = \bigcup_{y\in[y]\cap P_1} D_F([a]\times[b]\times y) = \left[\mathrm{Inf}\{a_1 - y\}_{y\in[y]\cap P_{1s}}, \mathrm{Sup}\{exp(y-b_2)\}_{y\in[y]\cap P_{1s}}\right]$$

Notons $[y]\cap P_{1,s} = [\, \underline{y}_{P_1}, \overline{y}_{P_1}\,]$. On a du fait de la monotonie des fonctions $\exp(y-b_1)$ et a_2-y :

$$\bigcup_{y\in[y]\cap P_1} D_F([a]\times[b]\times[y]) = [a_1 - \overline{y}_{P_1}, \exp(\overline{y}_{P_1} - b_1)] \qquad (4.26)$$

- Pour P_2

En posant $[y]\cap P_{2,s} = [\, \underline{y}_{P_2}, \overline{y}_{P_2}\,]$, et en utilisant la monotonie des fonctions de y, $a-y$ et $\exp(y-b)$ on a :

Si, $y_{a_1b_2} < y_{a_2b_1}$, $\bigcup_{y\in[y]\cap P_2} D_F([a]\times[b]\times[y]) = [\exp(\underline{y}_{P_2} - b_2), \exp(\overline{y}_{P_2} - b_1)]$ (4.27)

Si $y_{a_2b_1} < y_{a_1b_2}$, $\bigcup_{y\in[y]\cap P_2} D_F([a]\times[b]\times[y]) = [a_1 - \overline{y}_{P_2}, a_2 - \underline{y}_{P_2}\,]$ (4.28)

- Pour P_3

En posant $P_{3,s} = P_3 \cap]-\infty, \ y_{a_2b_2}[$ et $[y] \cap P_{3,s} = [\underline{y}_{P_1}, \overline{y}_{p_1}]$, en utilisant toujours la monotonie des fonctions $\exp(y-b_1)$ et a_2-y on a :

$$\bigcup_{y \in [y] \cap P_3} D_F([a]\times[b]\times[y]) = [\exp(\underline{y}_{P_2} - b_2), \exp(\overline{y}_{p_2} - b_1)] \tag{4.29}$$

On a ainsi obtenu les expressions de $D_F([a]\times[b]\times[y])$ dans les P_i. Ces valeurs sont récapitulées dans le Tableau 4.8 et le Tableau 4.9.

- Si $y_{a_1b_2} < y_{a_2b_1}$

	$P_1 =]-\infty, \ y_{a_1b_2}[$	$P_2 = [\ y_{a_1b_2}, y_{a_2b_1}\ ]$	$P_3 = [\ y_{a_2b_1}, +\infty[$
$D_F([a]\times[b]\times[y])$	$[a_1 - \overline{y}_{p_1}, \exp(\overline{y}_{p_1} - b_1)]$	$[\exp(\underline{y}_{P_2} - b_2), \exp(\overline{y}_{p_2} - b_1)]$	$[\exp(\underline{y}_{P_3} - b_2), a_2 - \underline{y}_{P_3}]$

Tableau 4.8 : expression du domaine de consistance $D_F([a]\times[b]\times[y])$ dans les espaces P_i.

- Si $y_{a_2b_1} < y_{a_1b_2}$

	$P_1 =]-\infty, y_{a_2b_1}[$	$P_2 = [\ y_{a_2b_1}, y_{a_1b_2}\ ]$	$P_3 = [\ y_{a_1b_2}, +\infty[$
$D_F([a]\times[b]\times[y])$	$[a_1 - \overline{y}_{p_1}, \exp(\overline{y}_{p_1} - b_1)\ ]$	$[a_1 - \overline{y}_{p_2}, a_2 - \underline{y}_{P_2}]$	$[\exp(\underline{y}_{P_3} - b_2), a_2 - \underline{y}_{P_3}]$

Tableau 4.9 : expression du domaine de consistance $D_F([a]\times[b]\times[y])$ dans les espaces P_i.

Remarque

■ En se basant sur le Tableau 4.8 et le Tableau 4.9, on va pouvoir proposer un algorithme (dans le Tableau 4.10) permettant d'obtenir la projection des solutions dans $[x]$. Seulement, théoriquement, on doit savoir déterminer, pour un couple (a, b) donné, y_{ab} qui annule la fonction $\Delta(y) = a - y - \exp(y-b)$. Ne sachant pas déterminer précisément la solution à cette équation, la stratégie classique de dichotomie permet de converger exponentiellement vers la solution avec un intervalle contenant la solution selon une précision désirée. Malheureusement, les points particuliers y_{ab} sont ceux qui permettent de déterminer d'autres points particuliers $\overline{y}_{p_i}$ qui interviennent dans le calcul de $D_F([a]\times[b]\times[y])$ et une erreur sur les y_{ab} induit une erreur sur les $\overline{y}_{p_i}$ notamment dans les cas où l'intervalle $[y]$ contient un des

y_{ab}. Par exemple, considérons un intervalle [y] en supposant qu'il contienne $y_{a_1b_2}$ et qu'on est dans le cas où $y_{a_1b_2} < y_{a_2b_1}$. Si on désire calculer $D_F([a]\times[b]\times[y])$ pour l'intersection de [y] avec P_1, on utilise tour à tour l'expression de P_1 =]-∞, $y_{a_1b_2}$] et l'expression de $P_{1,s} = P_1 \cap [\, y_{a_1b_1}, +\infty]$. Le calcul de $D_F([a]\times[b]\times[y])$ se fait en utilisant l'expression [a_1– $\overline{y}_{p_1}$, exp($\overline{y}_{p_1} - b_1$)] (voir Tableau 4.8), en faisant l'intersection de [y] avec P_1 =]-∞, $y_{a_1b_2}$] puis avec $P_{1,s} = P_1 \cap [\, y_{a_1b_1}, +\infty]$ pour obtenir [$\underline{y}_{P_1}$, $\overline{y}_{P_1}$]. Mais étant incapable de déterminer exactement $y_{a_1b_1}$ qui dans cet exemple appartient à [y]∩P_2, il y a une erreur d'approximation à gérer ici.

La stratégie adoptée dans l'algorithme décrit par le Tableau 4.10 est alors la suivante : lorsqu'on fait un calcul de domaine de consistance avec une borne Inf de type $\underline{y}_{P_i}$, on aura pris le soin de faire, dans tous les cas possibles, une approximation inférieure de $\underline{y}_{P_i}$. De façon identique, pour les bornes Sup du type $\overline{y}_{P_i}$, on aura pris soin de faire, dans tous les cas possibles, une approximation supérieur de $\overline{y}_{P_i}$.

Une justification de cette stratégie est qu'en utilisant une dichotomie simple (plusieurs autres méthodes peuvent être également utilisées telles que celle de Newton), on peut atteindre très rapidement une très grande précision pour des approximations Inf et Sup, et de surcroît, le calcul de $D_F([a]\times[b]\times[y])$ se fait avec des valeurs de fonction continues sur les bornes $\underline{y}_{P_i}$ et $\overline{y}_{P_i}$. Ceci signifie que la précision est non seulement adaptable aux besoins mais son ordre de grandeur est facilement estimable.

Une autre justification est que le problème des approximations Sup et Inf est connu et inhérent au fonctionnement des calculateurs. En effet, comme conséquence de la précision limitée des machines dues à la représentation discrétisé des réels, les erreurs d'arrondi existent toujours. Ces erreurs d'arrondi entraînent des imprécisions qui doivent être prises en compte pour conserver l'aspect garanti de la méthode [*Defour et al., 2004*]. ■

Algorithme DCsolve 3 : Input([a] ,[b], [x], [y], ε), Output([x])

% DCsolve3 : sous résolveur de l'exemple non linéaire donnant le [x] *solution, 3 représente la dimension de x*

% points particuliers / zero est une fonction dichotomique permettant d'approcher le zéro d'une fonction et Δ(y, a, b) = a–y–exp(y–b), ε est le seuil de précision

$(\underline{y}_{a_1b_1}, \overline{y}_{a_1b_1}) = \text{zero}(\Delta(y, a_1, b_1), \varepsilon)$, $(\underline{y}_{a_2b_2}, \overline{y}_{a_2b_2}) = \text{zero}(\Delta(y, a_2, b_2), \varepsilon)$,

$(\underline{y}_{a_1b_2}, \overline{y}_{a_1b_2}) = \text{zero}(\Delta(y, a_1, b_2), \varepsilon)$, $(\underline{y}_{a_2b_1}, \overline{y}_{a_2b_1}) = \text{zero}(\Delta(y, a_2, b_1), \varepsilon)$

% les P_i Recouvrement de IR où on connaît précisément les expressions des domaines de consistance

$P_1 =]-\infty, \min(\overline{y}_{a_1b_2}, \overline{y}_{a_2b_1})]$, $P_2 = [\min(\underline{y}_{a_1b_2}, \underline{y}_{a_2b_1}), \max(\overline{y}_{a_1b_2}, \overline{y}_{a_2b_1})]$,

$P_3 = [\max(\underline{y}_{a_1b_2}, \underline{y}_{a_2b_1}), +\infty[$

% Partie des P_i où les domaines de consistance sont non vides

$P_{1,s} = P_1 \cap]-\infty, \overline{y}_{a_1b_1}]$, $P_{2,s} = P_2$, $P_{3,s} = P_3 \cap [\underline{y}_{a_2b_2}, +\infty[$

% Intersection des $P_{i,s}$ avec l'intervalle x

$[y_{1,s}] = [y] \cap P_{1,s}$, $[y_{2,s}] = [y] \cap P_{2,s}$, $[y_{3,s}] = [y] \cap P_{3,s}$

% Calcul des domaines de consistance

Si $[y_{1,s}] \neq \phi$, $[x_{1,s}] = [x] \cap [a_1 - \overline{y}_{p_1}, \exp(\overline{y}_{p_1} - b_1)]$ Sinon $[x_{1,s}] = \phi$ Fin

Si $(y_{a_1b_2} < y_{a_2b_1})$

  Si $[y_{2,s}] \neq \phi$, $[x_{2,s}] = [x] \cap [\exp(\underline{y}_{P_2} - b_2), \exp(\overline{y}_{p_2} - b_1)]$ Sinon $[x_{2,s}] = \phi$ Fin

Sinon

  Si $[y_{2,s}] \neq \phi$, $[x_{2,s}] = [x] \cap [a_1 - \overline{y}_{p_2}, a_2 - \underline{y}_{P_2}]$ Sinon $[x_{2,s}] = \phi$ Fin

Fin

Si $[y_{3,s}] \neq \phi$, $[x_{3,s}] = [x] \cap [\exp(\underline{y}_{P_3} - b_2), a_2 - \underline{y}_{P_3}]$ Sinon $[x_{3,s}] = \phi$ Fin

% Calcul du domaine solution

$[x] \cap D_F([y],[a],[b]) = [x_{1,s}] \cup [x_{2,s}] \cup [x_{3,s}]$.

Tableau 4.10 : contracteur permettant d'obtenir les x globalement consistants à un facteur de précision près.

4.3.1.2 Calcul de $\Pi_4(S)$

Le problème ne présentant pas de symétrie, on ne peut pas appliquer, comme dans le cas de l'exemple linéaire, une stratégie permettant de réutiliser l'algorithme du Tableau 4.16. On cherchera donc un algorithme propre à chaque dimension.

Empruntons la même démarche que pour le calcul de $\Pi_3(S)$ de la section 4.3.1.1 précédente. Tout d'abord, on calcule $\Pi_4(S)$ avec le ***Théorème 1*** $\Pi_4(S) = [y] \cap D_F([a]\times[b]\times[x])$, puis on calcule le terme : $D_F([a]\times[b]\times[x])$ en utilisant 4 étapes. Le raisonnement étant proche de celui fait précédemment, le cheminement sera moins détaillé.

1. ***Etape 1 : évaluation des bornes de l'intervalle*** $D_F([a]\times[b]\times x_0)$

$D_F([a]\times[b]\times x_0) = ([a]-x_0) \cap ([b]+\ln(x_0))$

$= ([a_1-x_0,\ a_2-x_0]) \cap ([b_1+\ln(x_0),\ b_2+\ln(x_0)])$

$= [\mathrm{Sup}\{a_1-x_0,\ b_1+\ln(x_0)\},\ \mathrm{Inf}\{a_2-x_0,\ b_2+\ln(x_0)\}]$

2. ***Etape 2 : caractérisation des bornes en fonction de*** x_0

Etudions $\Delta(x) = a-x-(b+\ln(x))$. Δ est décroissante et ses valeurs parcourent IR, et notons que Δ s'annule au point x_{ab}. Le tableau qui suit positionne $a-x$ et $b+\ln(x)$ selon a et b

	$]0,\ x_{ab}]$	$[x_{ab}, +\infty[$
Δ	>0	<0
$\mathrm{Sup}\{a-x \text{ et } b-\ln(x)\}$	$a-x$	$b+\ln(x)$
$\mathrm{Inf}\{a-x \text{ et } b-\ln(x)\}$	$b+\ln(x)$	$a-x$

Tableau 4.11 : comparaison des fonctions de x : $a-x$ et $b-\ln(x)$.

En remplaçant dans ce tableau le couple $(a,\ b)$ par chaque couple $(a_1,\ b_1)$ et $(a_2,\ b_2)$ on obtient 3 secteurs (P_1, P_2, P_3) qui recouvrent IR^+ et pour lesquels les bornes de $D_F([a]\times[b]\times x)$ sont connus. Le Tableau 4.12 et le Tableau 4.13 donnent les valeurs prises par $D_F([a]\times[b]\times x)$ selon les cas possibles.

- Si $x_{a_1b_1} < x_{a_2b_2}$

	$P_1 = [0,\ x_{a_1b_1}]$	$P_2 = [x_{a_1b_1}, x_{a_2b_2}]$	$P_3 = [x_{a_2b_2}, +\infty[$
$D_F([a]\times[b]\times x)$	$[a_1-x,\ b_2+\ln(x)]$	$[b_1+\ln(x),\ b_2+\ln(x)]$	$[b_1+\ln(x),\ a_2-x]$

Tableau 4.12 : expression du domaine de consistance $D_F([a]\times[b]\times x)$ dans les espaces P_i.

- Si $x_{a_2b_2} < x_{a_1b_1}$

	$P_1 =]0,\ x_{a_2b_2}]$	$P_2 = [x_{a_2b_2},\ x_{a_1b_1}]$	$P_3 = [x_{a_1b_1}, +\infty[$
$D_F([a]\times[b]\times x)$	$[a_1-x,\ b_2+\ln(x)]$	$[a_1-x,\ a_2-x]$	$[b_1+\ln(x),\ a_2-x]$

Tableau 4.13 : expression du domaine de consistance $D_F([a]\times[b]\times x)$ dans les espaces P_i.

3. *Etape 3 : élimination des x pour lesquels l'intervalle $D_F([a]\times[b]\times x)$ est impropre*

- Pour P_1

  Notons $P_{1,s}$ l'ensemble des éléments de P_1 pour lesquels $[a_1-x,\ b_2+\ln(x)]$ est non vide. $P_{1,s}$ s'écrit alors : $P_{1,s} = P_1 \cap [x_{a_1b_2}, +\infty[$.

- Pour P_2

  Notons $P_{2,s}$ l'ensemble des éléments de P_2 pour lesquels soit l'intervalle $[a_1-x,\ a_2-x]$, soit l'intervalle $[b_1+\ln(x),\ b_2+\ln(x)]$ est non vide. On a $P_{2,s} = P_2$.

- Pour P_3

  Notons $P_{3,s}$ l'ensemble des éléments de P_3 pour lesquels $[b_1+\ln(x),\ a_2-x]$ est non vide. $P_{3,s}$ s'écrit alors : $P_{3,s} = P_3 \cap]-\infty,\ x_{a_2b_1}]$.

4. *Etape 4 : calcul de* $D_F([a]\times[b]\times[x])$ ***connaissant*** $D_F([a]\times[b]\times x)$ **et les** $P_{i,s}$

Notons $[x]\cap P_{i,s} = [\underline{x}_{P_i}, \overline{x}_{P_i}]$

- Si $x_{a_1b_1} < x_{a_2b_2}$

	$P_1 =]0, x_{a_1b_1}]$	$P_2 = [x_{a_1b_1}, x_{a_2b_2}]$	$P_3 = [x_{a_2b_2}, +\infty[$
$D_F([a]\times[b]\times[x])$	$[a_1 - \overline{x}_{P_1}, b_2 + \ln(\overline{x}_{P_1})]$	$[b_1 + \ln(\underline{x}_{P_2}), b_2 + \ln(\overline{x}_{P_2})]$	$[b_1 + \ln(\underline{x}_{P_3}), a_2 + \underline{x}_{P_3}]$

Tableau 4.14 : expression du domaine de consistance $D_F([a]\times[b]\times[x])$ dans les espaces P_i.

- Si $x_{a_2b_2} < x_{a_1b_1}$

	$P_1 =]0, x_{a_2b_2}]$	$P_2 = [x_{a_2b_2}, x_{a_1b_1}]$	$P_3 = [x_{a_1b_1}, +\infty[$
$D_F([a]\times[b]\times[x])$	$[a_1 - \overline{x}_{P_1}, b_2 + \ln(\overline{x}_{P_1})]$	$[a_1 - \overline{x}_{P_2}, a_2 - \underline{x}_{P_2}]$	$[b_1 + \ln(\underline{x}_{P_3}), a_2 - \underline{x}_{P_3}]$

Tableau 4.15 : expression du domaine de consistance $D_F([a]\times[b]\times[x])$ dans les espaces P_i.

Remarque

■ Comme pour le cas du calcul de $\Pi_3(S)$ dans la section 4.3.1.1, on se trouve dans la situation où on ne peut pas calculer les x_{ab} zéros des fonctions $\Delta(x) = a - x - b - \ln(x)$. L'algorithme présenté dans le Tableau 4.10, utilise la dichotomie pour avoir des approximations inférieures et supérieures des points particuliers permettant de calculer le domaine de consistance $D_F([a]\times[b]\times[x])$. ■

Algorithme DCsolve 4 : Input([a] ,[b], [x], [y], ε), Output([y])

% DCsolve4 : sous résolveur de l'exemple non linéaire donnant le [y]*, 4 représente la dimension de y*

% points particuliers / zero est une fonction dichotomique permettant d'approcher le zéro d'une fonction et Δ(x, a, b) = a–x–b–ln(x), ε est le seuil de précision.

$(\underline{x}_{a_1b_1}, \overline{x}_{a_1b_1}) = \text{zero}(\Delta(x, a_1, b_1), \varepsilon)$, $(\underline{x}_{a_2b_2}, \overline{x}_{a_2b_2}) = \text{zero}(\Delta(x, a_2, b_2), \varepsilon)$,

$(\underline{y}_{a_1b_2}, \overline{y}_{a_1b_2}) = \text{zero}(\Delta(y, a_1, b_2), \varepsilon)$, $(\underline{x}_{a_2b_1}, \overline{x}_{a_2b_1}) = \text{zero}(\Delta(x, a_2, b_1), \varepsilon)$

% les P_i Recouvrement de IR où on connaît précisément les expressions des domaines de consistance

$P_1 =]0, \min(\overline{x}_{a_1b_1}, \overline{x}_{a_2b_2})]$, $P_2 = [\min(\underline{x}_{a_1b_1}, \underline{x}_{a_2b_2}), \max(\overline{x}_{a_1b_1}, \overline{x}_{a_2b_2})]$,

$P_3 = [\max(\underline{x}_{a_1b_1}, \underline{x}_{a_2b_2}), +\infty[$

% Partie des P_i où les domaines de consistance sont non vides

$P_{1,s} = P_1 \cap [\underline{x}_{a_1b_2}, +\infty[$, $P_{2,s} = P_2$, $P_{3,s} = P_3 \cap]-\infty, \underline{x}_{a_2b_1}]$

% Intersection des $P_{i,s}$ avec l'intervalle x

$[x_{1,s}] = [x] \cap P_{1,s}$, $[x_{2,s}] = [x] \cap P_{2,s}$, $[x_{3,s}] = [x] \cap P_{3,s}$

% Calcul des domaines de consistance

Si $[x_{1,s}] \neq \phi$, $[y_{1,s}] = [y] \cap [a_1 - \overline{x}_{P_1}, b_2 + \ln(\overline{x}_{P_1})]$ Sinon $[y_{1,s}] = \phi$ Fin

Si $(x_{a_1b_1} < x_{a_2b_2})$

  Si $[x_{2,s}] \neq \phi$, $[y_{2,s}] = [y] \cap [b_1 + \ln(\underline{x}_{P_2}), b_2 + \ln(\overline{x}_{P_2})]$ Sinon $[y_{2,s}] = \phi$ Fin

Sinon

  Si $[x_{2,s}] \neq \phi$, $[y_{2,s}] = [y] \cap [a_1 - \overline{x}_{P_2}, a_2 - \underline{x}_{P_2}]$ Sinon $[y_{2,s}] = \phi$ Fin

Fin

Si $[x_{3,s}] \neq \phi$, $[y_{3,s}] = [y] \cap [b_1 + \ln(\underline{x}_{P_3}), a_2 - \underline{x}_{P_3}]$ Sinon $[y_{3,s}] = \phi$ Fin

% Calcul du domaine solution

$[y] \cap D_F([x],[a],[b]) = [y_{1,s}] \cup [y_{2,s}] \cup [y_{3,s}]$.

Tableau 4.16 : contracteur permettant d'obtenir les y globalement consistants à un facteur de précision près.

4.3.1.3 Calcul de $\Pi_1(S)$

En appliquant le ***Théorème 1***, on peut écrire $\Pi_1(S) = [a] \cap D_F([b]\times[x]\times[y])$. Suivons la démarche en 4 étapes :

1. *Etape 1 : évaluation des bornes de l'intervalle* $D_F([b]\times[x]\times[y])$

Rappelons ici que le vecteur ($a\times b\times x\times y$) de variables initial est noté X et que par exemple $X(3)$ désigne x. Pour évaluer le domaine de consistance $D_F([b]\times[x]\times[y])$, on peut remarquer que des deux contraintes $f_1 : y+x-a = 0$ et $f_2 : y-\ln(x)-b = 0$ seule f_1 concerne la variable a. On peut alors utiliser le ***Corollaire 3*** du ***Théorème 3*** pour écrire le domaine de consistance $D_F([b]\times[x]\times[y])$ en fonction de la projection des solutions de la deuxième contrainte f_2 sur les dimensions 3 et 4 : $D_F([b]\times[x]\times[y]) = D_{f_1}\left(\prod_{3,4}\left(S_{I-\{1\}}\right)\right)$ et, de plus, la projection $\prod_{3,4}\left(S_{I-\{1\}}\right)$ peut être décomposée en fonction de la projection $\prod_3\left(S_{I-\{1\}}\right)$, par exemple, en utilisant la formule (4.9) (on prend la réunion du produit cartésien de toutes les solutions en x avec leur ensemble compatible dans l'espace des y qui est égale à $\prod_4\left(D_{f_2}\left(X(3)\right)\right)\cap\left[X(4)\right]$) i.e.

$$\prod_{3,4}\left(S_{I-\{1\}}\right) = \bigcup_{X(3)\in\prod_3\left(S_{I-\{1\}}\right)}\left(X(3),\left(\prod_4\left(D_{f_2}\left(X(3)\right)\right)\cap\left[X(4)\right]\right)\right)$$

$$\text{D'où : } D_F([b]\times[x]\times[y]) = D_{f_1}\left(\bigcup_{X(3)\in\prod_3\left(S_{I-\{1\}}\right)}\left(X(3),\left(\prod_4\left(D_{f_2}\left(X(3)\right)\right)\cap\left[X(4)\right]\right)\right)\right)$$

$$= D_{f_1}\left(\bigcup_{x\in\prod_3\left(S_{I-\{1\}}\right)} x\times\left(\prod_4\left(D_{f_2}(x)\right)\cap[y]\right)\right)$$

De plus $\prod_4\left(D_{f_2}(x)\right)$ est l'ensemble des y pour lesquels $\exists\ b \in [b]$ vérifiant $y = \ln(x)+b$

$$\Rightarrow \prod_4\left(D_{f_2}(x)\right) = \ln(x)+[b]. \text{ D'où : } D_F([b]\times[x]\times[y]) = D_{f_1}\left(\bigcup_{x\in\prod_3\left(S_{I-\{1\}}\right)}\left(x,[y]\cap\left([b]+\ln(x)\right)\right)\right)$$

Le CSP $\mathcal{H}_{I-\{1\}}$ ne contenant qu'une contrainte sans multi occurence, sa résolution est simple et obtenue avec une étape de propagation et une de rétro propagation. La projection

$\prod_3(S_{I-\{1\}})$ est égale à $[x] \cap \exp([y]-[b])$. Pour plus de simplicité, à la place de $\prod_3(S_{I-\{1\}})$ on peut alors mettre $[x]$, après avoir fait cette contraction. En utilisant la propriété de réunion, on a alors :

$$D_F([b]\times[x]\times[y]) = D_{f_1}\left(\bigcup_{x\in[x]}\left(x,\left([y]\cap[b]+\ln(x)\right)\right)\right) = \bigcup_{x\in[x]} D_{f_1}\left(x,\left([y]\cap[b]+\ln(x)\right)\right).$$

Puisque la contrainte f_1 s'écrit $a = y+x$ on a :

$$D_{f_1}\left(x,\left([y]\cap[b]+\ln(x)\right)\right) = x+([y]\cap[b]+\ln(x)) = [x+\text{Sup}\{b_1+\ln(x), y_1\}, x+\text{Inf}\{b_2+\ln(x), y_2\}]$$
$$= [\text{Sup}\{x+b_1+\ln(x), x+y_1\}, \text{Inf}\{x+b_2+\ln(x), x+y_2\}]$$

2. *Etape 2 : caractérisation des bornes en fonction de x*

Posons $x_{b_0y_0}$ la fonction qui annule $\Delta(x) = (x+b_0+\ln(x))-(x+y_0)$. Le Tableau 4.17 donne les le Sup et l'Inf des fonctions $x+b_0+\ln(x)$ et $x+y_0$, en fonction de $x_{b_0y_0}$.

	$]0,\ x_{b_0y_0}]$	$[x_{b_0y_0}, +\infty[$
$\text{Sup}\{x+b_0+\ln(x), x+y_0\}$	$x+y_0$	$x+b_0+\ln(x)$
$\text{Inf}\{x+b_0+\ln(x), x+y_0\}$	$x+b_0+\ln(x)$	$x+y_0$

Tableau 4.17 : comparaison des fonctions de x : $x+b_0+\ln(x)$ et $x+y_0$.

En remplaçant dans ce tableau b par b_1 et b_2, on obtient 3 secteurs (P_1, P_2, P_3) qui recouvrent IR^{+*} et pour lesquels les bornes de $D_{f_1}\left(x,\left([y]\cap[b]+\ln(x)\right)\right)$ sont connues. Le Tableau 4.18 et Tableau 4.19 donnent les valeurs prises par $D_{f_1}\left(x,\left([y]\cap[b]+\ln(x)\right)\right)$ selon les cas possibles.

- Si $x_{b_1y_1} \le x_{b_2y_2}$

	$P_1 = [0,\ x_{b_1y_1}]$	$P_2 = [x_{b_1y_1}, x_{b_2y_2}]$	$P_3 = [x_{b_2y_2}, +\infty[$
$D_{f_1}\left(x,\left([y]\cap[b]+\ln(x)\right)\right)$	$[x+y_1, x+b_2+\ln(x)]$	$[x+b_1+\ln(x), x+b_2+\ln(x)]$	$[x+b_1+\ln(x), x+y_2]$

Tableau 4.18 : expression du domaine de consistance $D_{f_1}\left(x,\left([y]\cap[b]+\ln(x)\right)\right)$ dans les P_i.

- Si $x_{b_2y_2} \leq x_{b_1y_1}$

	$P_1 =]0,\ x_{b_2y_2}[$	$P_2 = [\ x_{b_2y_2}, x_{b_1y_1}[$	$P_3 = [\ x_{b_1y_1}, +\infty[$
$D_{f_1}\left(x,\left([y]\cap[b]+\ln(x)\right)\right)$	$[x+y_1, x+b_2+\ln(x)]$	$[x+y_1, x+y_2]$	$[x+b_1+\ln(x), x+y_2]$

Tableau 4.19 : expression du domaine de consistance $D_{f_1}\left(x,\left([y]\cap[b]+\ln(x)\right)\right)$ dans les P_i.

3. ***Etape 3 : élimination des x pour lesquels l'intervalle $D_{f_1}\left(x,\left([y]\cap[b]+\ln(x)\right)\right)$ est impropre***

- Pour P_1

  Notons $P_{1,s}$ l'ensemble des éléments de P_1 pour lesquels $[x+y_1, x+b_2+\ln(x)]$ est non vide. $P_{1,s}$ s'écrit : $P_{1,s} = P_1 \cap [\ x_{b_2y_1}, +\infty[$.

- Pour P_2

  Notons $P_{2,s}$ l'ensemble des éléments de P_2 pour lesquels soit l'intervalle $[x+y_1, x+y_2]$, soit l'intervalle $[x+b_1+\ln(x), x+b_2+\ln(x)]$ est non vide. On a $P_{2,s} = P_2$.

- Pour P_3

  Notons $P_{3,s}$ l'ensemble des éléments de P_3 pour lesquels $[x+b_1+\ln(x), x+y_2]$ est non vide. $P_{3,s}$ s'écrit alors : $P_{3,s} = P_3 \cap]-\infty,\ x_{b_1y_2}[$.

4. **Etape 4 : calcul de $D_F([b]\times[x]\times[y])$ connaissant $D_{f_1}\left(x,\left([y]\cap[b]+\ln(x)\right)\right)$ et les $P_{i,s}$**

Notons $[x]\cap P_{i,s} = [\ \underline{x}_{P_i}, \overline{x}_{P_i}\ ]$. En constatant que les fonctions de x : $x+b_0+\ln(x)$ et $x+y_0$ sont croissantes dans IR, on peut en déduire les expressions de $D_{f_1}\left(x,\left([y]\cap[b]+\ln(x)\right)\right)$ dans les P_i en fonction des $\underline{x}_{P_i}$ et $\overline{x}_{P_i}$. Ces différentes expressions sont données dans le Tableau 4.20 et le Tableau 4.21.

- Si $x_{b_1y_1} \leq x_{b_2y_2}$

	$P_1 =]0,\ x_{b_1y_1}[$	$P_2 = [\ x_{b_1y_1}, x_{b_2y_2}\ ]$	$P_3 = [\ x_{b_2y_2}, +\infty[$
$D_F([b]\times[x]\times[y])$	$[\ \underline{x}_{P_1}+y_1,\ \overline{x}_{P_1}+b_2+\ln(x)]$	$[\ \underline{x}_{P_2}+b_1+\ln(x),$ $\overline{x}_{P_2}+b_2+\ln(x)]$	$[\ \underline{x}_{P_3}+b_1+\ln(x),\ \overline{x}_{P_3}+y_2]$

Tableau 4.20 : expression du domaine de consistance $D_F([b]\times[x]\times[y])$ dans les espaces P_i.

- Si $x_{b_2y_2} \leq x_{b_1y_1}$

	$P_1 =]0,\ x_{b_1y_1}]$	$P_2 = [x_{b_1y_1}, x_{b_2y_2}]$	$P_3 = [x_{b_2y_2}, +\infty[$
$D_F([b]\times[x]\times[y])$	$[\underline{x}_{P_1}+y_1,\ \overline{x}_{P_1}+b_2+\ln(x)]$	$[\underline{x}_{P_2}+y_1,\ \overline{x}_{P_2}+y_2]$	$[\underline{x}_{P_3}+b_1+\ln(x),\ \overline{x}_{P_3}+y_2]$

Tableau 4.21 : expression du domaine de consistance $D_F([b]\times[x]\times[y])$ dans les espaces P_i.

Algorithme DCsolve1 : Input([a] ,[b], [x], [y]), Output([a])

% DCsolve1 : sous résolveur donnant le [a] solution, 1 représente la dimension de a

%On résout le CSP $\mathcal{H}_{1-\{1\}}$ (1 seule contrainte f_2 : $y-\ln(x)-b=0$

$[y] = [y]\cap\ln([x])+[b]$; $[x] = [x]\cap\exp([y]-[b])$; $[b] = [b]\cap[y]-\ln([x])$

% points particuliers $b_0+ln(x)-y_0$

$x_{b_1y_1} = \exp(y_1 - b_1)$, $x_{b_2y_2} = \exp(y_2 - b_2)$, $x_{b_1y_2} = \exp(y_2 - b_1)$, $x_{b_2y_1} = \exp(y_1 - b_2)$

% les P_i Recouvrement de IR où on connaît les expressions des domaines de consistance

$P_1 =]0, \min(x_{b_1y_1}, x_{b_2y_2})]$, $P_2 = [\min(x_{b_1y_1}, x_{b_2y_2}), \max(x_{b_1y_1}, x_{b_2y_2})]$,

$P_3 = [\max(x_{b_1y_1}, x_{b_2y_2}), +\infty[$

% Partie des P_i où les domaines de consistance sont non vides

$P_{1,s} = P_1\cap[x_{b_2y_1}, +\infty[$, $P_{2,s} = P_2$, $P_{3,s} = P_3\cap]-\infty, x_{b_1y_2}]$

% Intersection des $P_{i,s}$ avec l'intervalle x

$[x_{1,s}]=[x]\cap P_{1,s}$, $[x_{2,s}]=[x]\cap P_{2,s}$, $[x_{3,s}]=[x]\cap P_{3,s}$

% Calcul des domaines de consistance

Si $[x_{1,s}]\neq\phi$, $[a_{1,s}]=[a]\cap[\underline{x}_{P_1}+y_1,\ \overline{x}_{P_1}+b_2+\ln(x)]$ Sinon $[y_{1,s}]=\phi$ Fin

Si $(x_{b_1y_1} \leq x_{b_2y_2})$

  Si $[x_{2,s}]\neq\phi$, $[a_{2,s}]=[a]\cap[b_1+\ln(\underline{x}_{P_2}), b_2+\ln(\overline{x}_{P_2})]$Sinon $[y_{2,s}]=\phi$ Fin

Sinon

  Si $[x_{2,s}]\neq\phi$, $[a_{2,s}]=[a]\cap[\underline{x}_{P_2}+b_1+\ln(x),\ \overline{x}_{P_2}+b_2+\ln(x)]$ Sinon $[y_{2,s}]=\phi$ Fin

Fin

Si $[x_{3,s}]\neq\phi$, $[a_{3,s}]=[a]\cap[\underline{x}_{P_3}+b_1+\ln(x),\ \overline{x}_{P_3}+y_2]$ Sinon $[y_{3,s}]=\phi$ Fin

% Calcul du domaine solution

$[a]\cap D_F([x],[y],[b]) = [a_{1,s}]\cup[a_{2,s}]\cup[a_{3,s}]$.

Tableau 4.22 : contracteur permettant d'obtenir les *a* globalement consistants.

4.3.1.4 Calcul de $\Pi_2(S)$

En appliquant le ***Théorème 1***, on peut écrire $\Pi_2(S) = [b] \cap D_F([a]\times[x]\times[y])$. Le calcul de $D_F([a]\times[x]\times[y])$ est assez similaire à celui de $D_F([b]\times[x]\times[y])$ précédemment résolu. Les différentes étapes ne seront donc pas aussi détaillées que pour le calcul de $\Pi_1(S)$ dans la section 4.3.1.3 précédente.

1. ***Etape 1 : évaluation des bornes de l'intervalle*** $D_F([a]\times[x]\times[y])$

Pour évaluer le domaine de consistance $D_F([a]\times[x]\times[y])$, on peut remarquer que des deux contraintes $f_1 : y+x-a = 0$ et $f_2 : y-\ln(x)-b = 0$ seule f_2 concerne la variable b. On peut alors utiliser le ***Corollaire 3*** du ***Théorème 3*** pour écrire :

$D_F([b]\times[x]\times[y]) = D_{f_2}\left(\Pi_{3,4}\left(S_{I-\{2\}}\right)\right)$ Et de plus la projection $\Pi_{3,4}\left(S_{I-\{2\}}\right)$ peut s'écrire :

$$\Pi_{3,4}\left(S_{I-\{1\}}\right) = \left(\bigcup_{X(3)\in\Pi_3\left(S_{I-\{1\}}\right)} \left(X(3), \left(\Pi_4\left(D_{f_1}\left(X(3)\right)\right) \cap \left[X(4)\right]\right)\right)\right) = \bigcup_{x\in\Pi_3\left(S_{I-\{1\}}\right)} \left(x, \left(\Pi_4\left(D_{f_1}(x)\right) \cap [y]\right)\right)$$

D'où : $D_F([a]\times[x]\times[y]) = D_{f_2}\left(\bigcup_{x\in\Pi_3\left(S_{I-\{1\}}\right)} \left(x, \left(\Pi_4\left(D_{f_1}(x)\right) \cap [y]\right)\right)\right)$

De plus $\Pi_4\left(D_{f_1}(x)\right)$ est l'ensemble des y pour lesquels $\exists\ b \in [b]$ vérifiant $y = -x+a$ i.e. $\Pi_4\left(D_{f_1}(x)\right) = -x + [a]$. D'où : $D_F([a]\times[x]\times[y]) = D_{f_2}\left(\bigcup_{x\in\Pi_3\left(S_{I-\{1\}}\right)} x, \left([y] \cap \left([a]-x\right)\right)\right)$

Le CSP $\mathcal{H}_{I-\{2\}}$ ne contenant qu'une contrainte sans multi occurrence, sa résolution est simple et obtenue avec une étape de propagation et une de rétro propagation. La projection $\Pi_3\left(S_{I-\{1\}}\right)$ est égale à $[x] \cap ([a]-[y])$. Pour plus de simplicité, à la place de $\Pi_3\left(S_{I-\{1\}}\right)$ on peut alors mettre [x], après avoir fait cette contraction. On peut alors écrire :

$D_F([a]\times[x]\times[y]) = D_{f_2}\left(\bigcup_{x\in[x]} \left(x, [y] \cap \left([a]-x\right)\right)\right) = \bigcup_{x\in[x]} D_{f_2}\left(x, [y] \cap \left([a]-x\right)\right)$ (en utilisant la propriété de réunion). Puisque la contrainte f_2 s'écrit $b = y-\ln(x)$ on a :

$$D_{f_2}\left(x, [y] \cap \left([a]-x\right)\right) = ([y] \cap ([a]-x)) - \ln(x) = [\mathrm{Sup}\{a_1-x, y_1\} - \ln(x),$$
$$\mathrm{Inf}\{a_2-x, y_2\} - \ln(x)] = [\mathrm{Sup}\{a_1-x-\ln(x), y_1-\ln(x)\}, \mathrm{Inf}\{a_2-x-\ln(x), y_2-\ln(x)\}]$$

2. ***Etape 2 : caractérisation des bornes en fonction de*** *x*

Posons $x_{a_0y_0}$ la fonction qui annule $\Delta(x) = (a_0 - x - \ln(x)) - (y_0 - \ln(x))$. Le Tableau 4.23 donne les Sup et Inf des fonctions $a_0 - x - \ln(x)$ et $y_0 - \ln(x)$, en fonction de $x_{a_0y_0}$. Remarquons qu'on suppose que $x_{a_0y_0} \geq 0$ (sinon, $]0,\ x_{a_0y_0}] = \varnothing$ et le Tableau 4.23 reste encore vrai.

	$]0,\ x_{a_0y_0}]$	$[x_{a_0y_0},\ +\infty[$
$\text{Sup}\{a_0 - x - \ln(x),\ y_0 - \ln(x)\}$	$a_0 - x - \ln(x)$	$y_0 - \ln(x)$
$\text{Inf}\{a_0 - x - \ln(x),\ y_0 - \ln(x)\}$	$y_0 - \ln(x)$	$a_0 - x - \ln(x)$

Tableau 4.23 : comparaison des fonctions de x : $a_0 - x - \ln(x)$ et $y_0 - \ln(x)$.

En remplaçant dans ce tableau le couple (a_0, y_0) par chaque couple (a_1, y_1) et (a_2, y_2), on obtient 3 secteurs (P_1, P_2, P_3) qui recouvrent IR^+ et pour lesquels les bornes de $D_{f_2}\left(x, [y] \cap ([a] - x)\right)$ sont connues. Le Tableau 4.24 et le Tableau 4.25 donnent les valeurs prises par $D_{f_2}\left(x, [y] \cap ([a] - x)\right)$ selon les cas possibles.

- Si $x_{a_1y_1} \leq x_{a_2y_2}$

	$P_1 = [0,\ x_{a_1y_1}]$	$P_2 = [x_{a_1y_1},\ x_{a_2y_2}]$	$P_3 = [x_{a_2y_2},\ +\infty[$
$D_{f_2}\left(x \times [y] \cap ([a] - x)\right)$	$[a_1 - x - \ln(x),\ y_2 - \ln(x)]$	$[y_1 - \ln(x),$ $y_2 - \ln(x)]$	$[y_1 - \ln(x),\ a_2 - x - \ln(x)]$

Tableau 4.24 : expression du domaine de consistance $D_{f_2}\left(x, [y] \cap ([a] - x)\right)$ dans les P_i.

- Si $x_{a_2y_2} \leq x_{a_1y_1}$

	$P_1 =]0,\ x_{a_2y_2}]$	$P_2 = [x_{a_2y_2},\ x_{a_1y_1}]$	$P_3 = [x_{a_1y_1},\ +\infty[$
$D_{f_2}(x \times [y] \cap ([a] - x))$	$[a_1 - x - \ln(x),\ y_2 - \ln(x)]$	$[a_1 - x - \ln(x),$ $a_2 - x - \ln(x)]$	$[y_1 - \ln(x),\ a_2 - x - \ln(x)]$

Tableau 4.25 : expression du domaine de consistance $D_{f_2}\left(x, [y] \cap ([a] - x)\right)$ dans les P_i.

3. *Etape 3 : élimination des x pour lesquels l'intervalle* $D_{f_2}\left(x,[y]\cap\left([a]-x\right)\right)$ *est impropre*

- Pour P_1

  Notons $P_{1,s}$ l'ensemble des éléments de P_1 pour lesquels $[a_1-x-\ln(x), y_2-\ln(x)]$ est non vide. $P_{1,s}$ s'écrit : $P_{1,s} = P_1\cap[\, x_{a_2y_1}, +\infty[$.

- Pour P_2

  Notons $P_{2,s}$ l'ensemble inclus dans P_2 tel que, soit l'intervalle $[y_1-\ln(x), y_2-\ln(x)]$, soit l'intervalle $[a_1-x-\ln(x), a_2-x-\ln(x)]$ est non vide. On a $P_{2,s} = P_2$.

- Pour P_3

  Notons $P_{3,s}$ l'ensemble des éléments de P_3 pour lesquels $[y_1-\ln(x), a_2-x-\ln(x)]$ est non vide. $P_{3,s}$ s'écrit alors : $P_{3,s} = P_3\cap]-\infty,\ x_{a_1y_2}\,]$.

4. *Etape 4 : calcul de* $D_F([a]\times[x]\times[y])$ *connaissant* $D_{f_2}\left(x,[y]\cap\left([a]-x\right)\right)$ *et les* $P_{i,s}$

Notons $[x]\cap P_{i,s} = [\,\underline{x}_{P_i}, \overline{x}_{P_i}\,]$. On peut en déduire les expressions de $D_{f_2}\left(x,[y]\cap\left([a]-x\right)\right)$ dans les P_i en fonction des $\underline{x}_{P_i}$ et des $\overline{x}_{P_i}$. Ces expressions sont données par le Tableau 4.26 et le Tableau 4.27.Si $x_{a_1y_1} \le x_{a_2y_2}$

	$P_1 =]0,\ x_{a_1y_1}\,]$	$P_2 = [\,x_{a_1y_1}, x_{a_2y_2}\,]$	$P_3 = [\,x_{a_2y_2}, +\infty[$
$D_F([a]\times[x]\times[y])$	$[a_1-\overline{x}_{P_1}-\ln(\overline{x}_{P_1}),$ $y_2-\ln(\underline{x}_{P_1})]$	$[y_1-\ln(\overline{x}_{P_2})$ $y_2-\ln(\underline{x}_{P_2})]$	$[y_1-\ln(\overline{x}_{P_3}),$ $a_2-\underline{x}_{P_3}-\ln(\underline{x}_{P_3})]$

Tableau 4.26 : expression du domaine de consistance $D_F([a]\times[x]\times[y])$ dans les espaces P_i.

- Si $x_{a_2y_2} \le x_{a_1y_1}$

	$P_1 =]0,\ x_{a_2y_2}\,]$	$P_2 = [\,x_{a_2y_2}, x_{a_1y_1}\,]$	$P_3 = [\,x_{a_1y_1}, +\infty[$
$D_F([a]\times[x]\times[y])$	$[a_1-\overline{x}_{P_1}-\ln(\overline{x}_{P_1}),$ $y_2-\ln(\underline{x}_{P_1})]$	$[a_1-\overline{x}_{P_2}-\ln(\overline{x}_{P_2})$ $a_2-\underline{x}_{P_2}-\ln(\underline{x}_{P_2})]$	$[y_1-\ln(\overline{x}_{P_3}),$ $a_2-\underline{x}_{P_3}-\ln(\underline{x}_{P_3})]$

Tableau 4.27 : expression du domaine de consistance $D_F([a]\times[x]\times[y])$ dans les espaces P_i.

Algorithme DCsolve2 : Input([a], [b], [x], [y]), Output([b])

% DCsolve2 : sous résolveur de l'exemple non linéaire donnant le [b] solution, 2 représente la dimension de b

%%On résout le CSP $\mathcal{H}_{1\text{-}\{2\}}$ (1 seule contrainte f_2 : $y+x-b=0$

$[y] = [y]\cap([a]-[x])$; $[x] = [x]\cap([a]-[y])$; $[b] = [b]\cap[y]+[x]$

%% points particuliers

$x_{a_1y_1} = a_1 - y_1$, $x_{a_2y_2} = a_2 - y_2$, $x_{a_1y_2} = a_1 - y_2$, $x_{a_2y_1} = a_2 - y_1$

%% les P_i Recouvrement de IR où on connaît précisément les expressions des domaines de consistance

$P_1 =]0, \min(x_{a_1y_1}, x_{a_2y_2})]$, $P_2 = [\min(x_{a_1y_1}, x_{a_2y_2}), \max(x_{a_1y_1}, x_{a_2y_2})]$,

$P_3 = [\max(x_{a_1y_1}, x_{a_2y_2}), +\infty[$

%% Partie des P_i où les domaines de consistance sont non vides

$P_{1,s} = P_1\cap[x_{a_2y_1}, +\infty[$, $P_{2,s} = P_2$, $P_{3,s} = P_3\cap]-\infty, x_{a_1y_2}]$

%% Intersection des $P_{i,s}$ avec l'intervalle x

$[x_{1,s}]=[x]\cap P_{1,s}$, $[x_{2,s}]=[x]\cap P_{2,s}$, $[x_{3,s}]=[x]\cap P_{3,s}$

%% Calcul des domaines de consistance

Si $[x_{1,s}]\neq\phi$, $[b_{1,s}]=[a]\cap[a_1-\overline{x}_{P_1}-\ln(\overline{x}_{P_1}), y_2-\ln(\underline{x}_{P_1})]$ Sinon $[y_{1,s}]=\phi$ Fin

Si $(x_{a_1y_1} \leq x_{a_2y_2})$

  Si $[x_{2,s}]\neq\phi$, $[b_{2,s}]=[a]\cap[y_1-\ln(\overline{x}_{P_2})\, y_2-\ln(\underline{x}_{P_2})]$ Sinon $[y_{2,s}]=\phi$ Fin

Sinon

  Si $[x_{2,s}]\neq\phi$, $[b_{2,s}]=[a]\cap[a_1-\overline{x}_{P_2}-\ln(\overline{x}_{P_2}), a_2-\underline{x}_{P_2}-\ln(\underline{x}_{P_2})]$ Sinon $[y_{2,s}]=\phi$ Fin

Fin

Si $[x_{3,s}]\neq\phi$, $[b_{3,s}]=[a]\cap[y_1-\ln(\overline{x}_{P_3}), a_2-\underline{x}_{P_3}-\ln(\underline{x}_{P_3})]$ Sinon $[y_{3,s}]=\phi$ Fin

%% Calcul du domaine solution

$[b]\cap D_F([x],[y],[a]) = [b_{1,s}]\cup[b_{2,s}]\cup[b_{3,s}]$.

Tableau 4.28 : contracteur permettant d'obtenir les variables dans [b] globalement consistants.

4.3.2 Résultats numériques

On choisit dans cette partie de résoudre le CSP étudié, $\mathcal{H} = (F(a, b, x, y) = 0/(a \times b \times x \times y) \in [a_1, a_2] \times [b_1, b_2] \times [x_1, x_2] \times [y_1, y_2])$ où F est constitué des 2 contraintes $f_1 : y+x-a = 0$ et $f_2 : y-\ln(x)-b = 0$. Pour trois cas particuliers, résolvons ce CSP aussi bien avec l'algorithme de Waltz qu'avec l'algorithme basé sur le calcul de domaine de consistance. On choisit de montrer comme pour l'exemple linéaire, des résultats sous forme de figures, en représentant les projections sur les espaces x et y des solutions. On représente de surcroît les contraintes qui sont dans ce cas ci, une bande linéaire en ce qui concerne la contrainte $y+x = [a]$ et une bande sous forme logarithmique pour la contrainte $y-\ln(x) = [b]$. Pour l'algorithme basé sur les domaines de consistance, on retrouve dans tous les cas de figure les intervalles englobant optimaux ce qui n'est jamais le cas pour l'algorithme de Waltz.

Exemple 4.6

$[x] = [-3\ 2]$; $[y] = [-5\ 5]$; $[a] = [-3\ 2]$; $[b] = [0\ 5]$

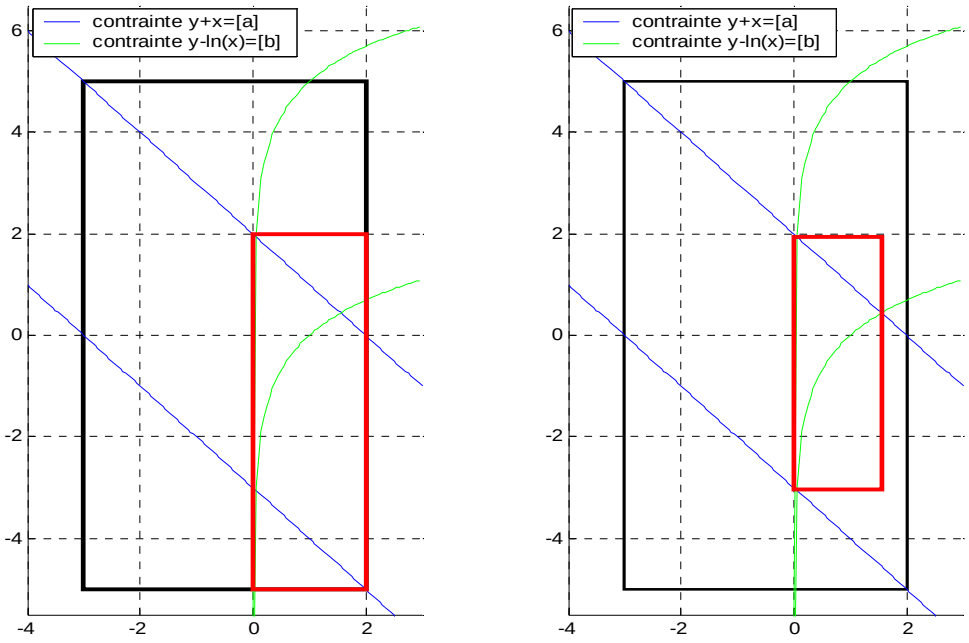


Figure 4.26 : comparaison entre l'algorithme de Waltz (à gauche) et l'algorithme DCsolve (à droite).

Exemple 4.7 : CSP 2 : $[x] = [-3\ 4]$; $[y] = [-2\ 6]$; $[a] = [-3\ 2]$; $[b] = [0\ 5]$.

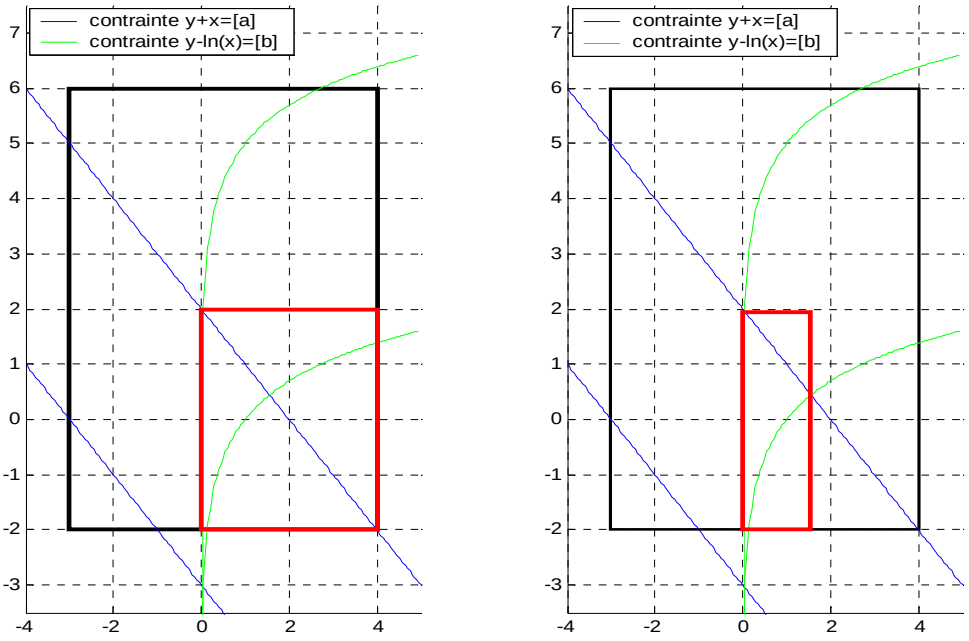


Figure 4.27 : comparaison entre l'algorithme de Waltz (à gauche) et l'algorithme DCsolve (à droite).

Exemple 4.8 : CSP 3 : [x] = [-3 4]; [y] = [-10 0] ; [a] = [-5 0]; [b] = [0 5].

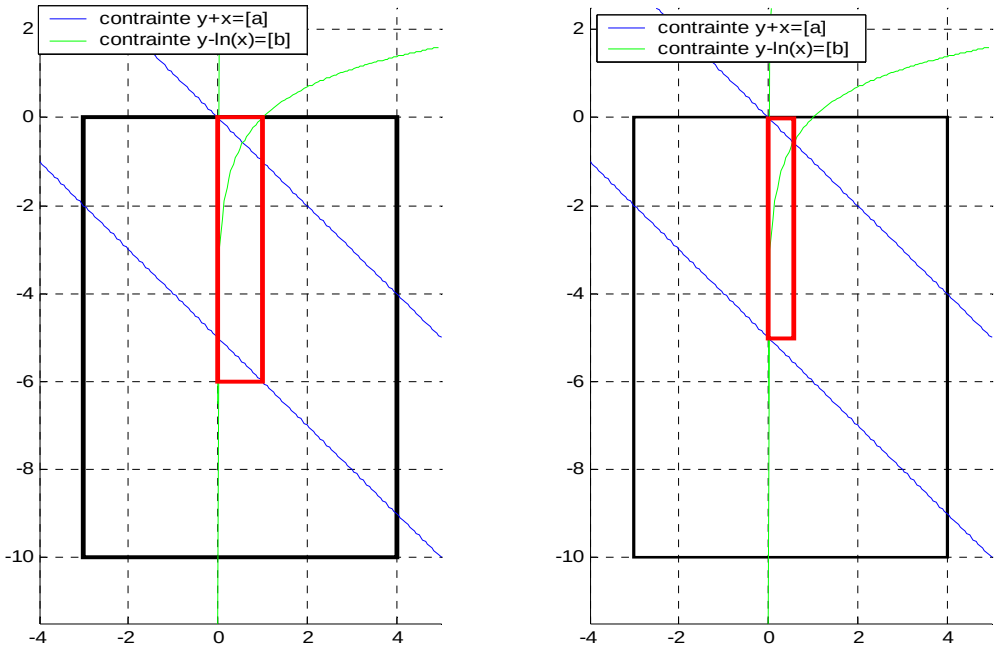


Figure 4.28 : comparaison entre l'algorithme de Waltz (à gauche) et l'algorithme DCsolve (à droite).

Conclusion

Le concept de domaine de consistance défini dans ce chapitre ouvre d'intéressantes perspectives pour les techniques permettant de résoudre des CSP en présence de cycles. Ce concept induit de nombreuses propriétés qui ont permis d'aboutir à des théorèmes importants. Il a aussi l'avantage de tout de suite se placer dans la continuité des méthodes existantes puisqu'il permet de réécrire tout algorithme de propagation de contraintes.

Une spécificité des méthodes basées sur les domaines de consistance est la possibilité de combiner des contraintes de façon rigoureuse. En effet, en cas de consistance locale, il est connu que des heuristiques cherchant à combiner les contraintes permettent d'améliorer sensiblement, dans certains cas, la consistance locale obtenue. Mais ce choix de combiner des contraintes reste purement intuitif sans qu'on puisse conclure réellement sur le niveau de consistance local atteint. En se basant sur les domaines de consistance, on évolue dans un cadre théorique, bien défini, permettant de raisonner avec des points dans certains cas critiques et d'utiliser toutes les connaissances et techniques existantes dans le domaine ponctuel.

A travers les exemples résolus dans ce chapitre, on a pu tracer une méthodologie en différentes étapes permettant d'obtenir des domaines globalement consistants. Le constat est que les algorithmes sont souvent simples dans leur écriture informatique mais demandent un effort de résolution venant du programmeur. A ce stade des algorithmes mies en oeuvre, on tend à penser que les méthodes basées sur les domaines de consistance sont éloignées d'une logique d'automatisation en vue de donner une solution pour n'importe quel CSP donné, logique d'automatisation qui existe déjà sous forme de *solveurs* pour des méthodes généralement basées sur des algorithmes de Waltz et des découpages (on peut citer, par exemple, [RealPaver ; ALIAS ; AQCS]).

Une des perspectives de ces travaux peut être, tout simplement, de changer les ambitions de cette méthode dans certaines situations où les cycles seraient plus complexes. En pareils cas, on pourrait, en lieu et place de chercher à calculer de façon exacte les domaines de consistance, chercher plutôt à les approximer, avec des méthodes numériques, tout en connaissant cependant l'erreur qu'on commet. En effet, la partie la plus délicate lorsque l'on

calcule un domaine de consistance correspond aux étapes 2 et 3 de la méthodologie présentée dans le paragraphe 4.1 de ce chapitre. Ces étapes consistent à déterminer une réunion de domaines de consistance calculés sur des points. Le calcul de domaine de consistance pour des points qui correspond à l'étape 1 se fait souvent sans trop de difficultés. On pourrait chercher à alléger ces étapes de réunions d'intervalles en changeant l'objectif de calcul exact en calcul approché avec une évaluation de l'erreur commise (on pense notamment que la partie qui consiste à calculer une réunion exacte de domaine de consistance peut être remplacée par une approximation en utilisant des méthodes numériques). Dans ces conditions, on n'atteindrait certes pas la consistance globale, mais d'une certaine façon, par rapport à une méthode de propagation de contraintes, on compenserait par une connaissance sur le pessimisme des résultats.

CHAPITRE 5 : VECTORISATION DANS LE CAS DE PROBLEMES DE SATISFACTION DE CONTRAINTES POUR DES ENSEMBLES CONTINUS

Introduction

Ce chapitre s'intéresse à la vectorisation pour des problèmes de satisfaction de contraintes. Vectoriser un CSP consiste à adopter une stratégie permettant de regrouper certaines variables dans un seul vecteur. Ce faisant, le problème traité est alors ramené à un nouveau problème ayant une structure particulière. Il peut s'agir, par exemple, de vouloir ramener les contraintes du problème à des contraintes binaires (concernant deux variables uniquement) ou encore, d'obtenir une représentation graphique sous forme d'arbre.

L'objectif de ce chapitre est d'introduire une nouvelle technique de vectorisation permettant, d'une part, d'isoler les cycles et, d'autre part, d'avoir une structure sous forme d'arbre. En effet, en présence de cycles, les temps de calcul des techniques habituelles de satisfaction de contraintes ne sont pas déterminables a priori. Sachant que, grâce aux techniques introduites dans le chapitre 4 précédent, la résolution de cycles en temps connu et a priori est faisable pour certains cas, on peut se poser diverses questions sur l'intérêt de la vectorisation. Est-il possible de résoudre les CSP en atteignant la consistance globale après avoir isolé et résolu les cycles ? Peut-on alors déterminer un temps de calcul a priori ? Pour répondre à ces questions, la structure naturelle que nous avons recherchée à obtenir est la représentation graphique sous forme d'un arbre. On se donne également pour objectif d'appliquer un algorithme FALL/CLIMB généralisé puisque cet algorithme est très efficace pour les structures sous forme d'arbre. Pour atteindre ces objectifs, diverses questions nous sont apparues non encore élucidées. On peut en citer quelques unes qui vont guider l'évolution de ce chapitre. Comment représenter un CSP sous forme de graphe sachant qu'au

mieux on sait exprimer un CSP sous forme d'un autre CSP avec des contraintes binaires et ternaires ? Qu'est ce qu'un cycle et comment traduire plus formellement cette idée qu'on en a de blocs de variables qui ont tendance à créer des boucles lors de l'exécution d'un algorithme de Waltz ? Comment écrire un CSP sous une nouvelle structure traduisant des blocs de variables reliés par des contraintes (chaque bloc constituant un cycle dans notre entendement) ? Ensuite, étant donnés deux blocs constituant concrètement chacun un cycle, comment rendre consistant un des vecteurs par rapport à l'autre ? Quelles conditions ces deux blocs doivent-ils vérifier entre eux dans l'éventualité d'appliquer un algorithme basé sur FALL/CLIMB ?

En essayant d'ordonnancer une démarche pour répondre à ces questions, le chapitre est organisé en plusieurs sous parties. D'abord, on montre dans un exemple précis, l'intérêt d'avoir une structure sous forme de graphe. Dans un second temps, on donne des éléments sur l'algorithme FALL/CLIMB pour lequel on va chercher une généralisation.

A ce stade, s'intercale, en préambule, une partie où on se pose des questions sur les notions nécessaires pour pouvoir définir une structure vectorisée. On s'intéresse aussi, dans ce préambule, à spécifier les cycles en étudiant les graphes associés aux CSP. Suite à ce préambule, on définit des notions qui vont nous permettre de définir les "*VSCP"* qui constituent la structure vectorisée d'un CSP qui nous est apparue la plus adéquate.

Après avoir défini la structure d'un VCSP, on engagera, dans une nouvelle section, une discussion qui nous permettra de fixer un certain nombre de concepts nécessaires pour pouvoir aboutir à une généralisation de l'algorithme FALL/CLIMB. On essaiera, notamment, de réfléchir sur des conditions à rechercher entre deux vecteurs de variables pour espérer pouvoir les rendre consistant l'un par rapport à l'autre. Ceci nous permettra d'introduire la notion de "*connexion à une branche*". On pourra alors présenter une version généralisée de l'algorithme FALL/CLIMB applicable à tout CSP, avec ou sans cycle.

Dans la dernière partie, on donne un exemple académique qui sera une concaténation des exemples de cycle résolus dans le chapitre 5 et qui permettra d'écrire concrètement l'algorithme FALL/CLIMB généralisé. Ensuite on l'appliquera également à un problème de localisation à l'estime sur des données réelles. Il s'agira, en particulier, d'estimer, de façon garantie, les déplacements élémentaires et rotations élémentaires d'un véhicule en déplacement.

1 INTERET DE VECTORISER UN CSP EN LA STRUCTURE D'UN ARBRE

Dans ce paragraphe, à travers un exemple illustratif, on justifie l'intérêt primordial de pouvoir réorganiser un graphe de contraintes sous la forme d'un arbre. Dans un premier temps, on rappelle le fonctionnement de l'algorithme FALL/CLIMB [Jaulin et al., 2001b, Benhamou et al., 1999] appliqué à un CSP illustratif et, ensuite, on met en exergue le gain en optimisation par rapport à l'algorithme de Waltz.

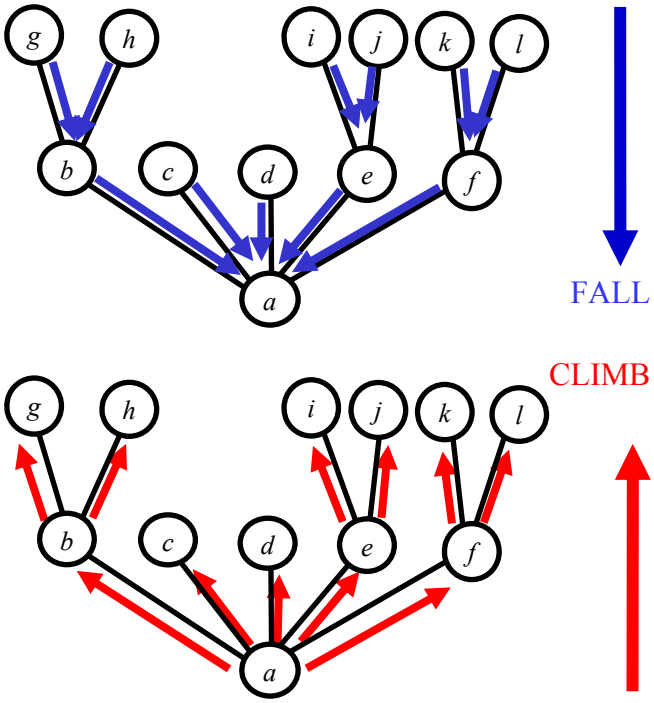


Figure 5.1 : algorithme de FALL/CLIMB.

Exemple 5.1

■ Considérons le CSP $\mathcal{H}$: (F(a, b,…, k, l)=0 | (a, b,…, k, l) ∈[a]×[b]×…×[k]×[l])), avec F le système de contraintes binaires suivant :

$\{b=2a\ ;\ c=3a\ ;\ d=4a\ ;\ e=5a\ ;\ f=6a\ ;\ g=2b\ ;\ h=3b\ ;\ i=2e\ ;\ j=3e\ ;\ k=2f\ ;l=3f\}$.

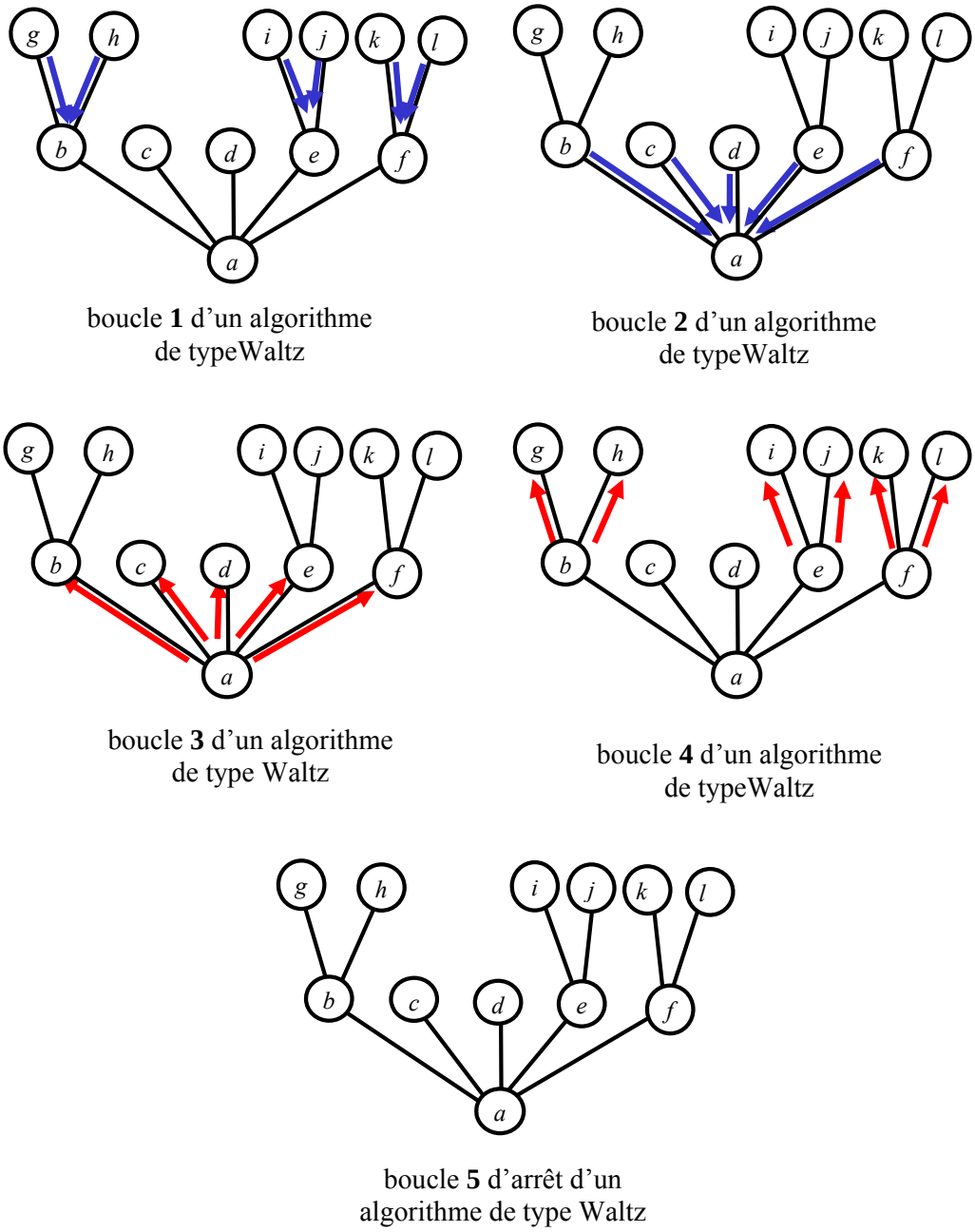
Les contraintes sont choisies binaires pour plus de simplicité. De surcroît, ce CSP est choisi de telle sorte que sa représentation graphique soit un arbre. Dans le cas d'un arbre, il suffit d'une contraction des feuilles vers la racine puis de la racine vers les feuilles pour

obtenir un graphe consistant (voir Figure 5.1). Ceci, comparé à un algorithme de Waltz, correspond à un tour, i.e. une étape de la boucle 'tant que' (voir chapitre 4, section 2.5.1).

L'algorithme FALL/CLIMB pour l'exemple considéré a alors la forme suivante :

- FALL :
- **1-** $[b]=[b]\cap\frac{[g]}{2}$ *;* **2-** $[b]=[b]\cap\frac{[h]}{3}$ *;* **3-** $[e]=[e]\cap\frac{[i]}{2}$ *;* **4-** $[e]=[e]\cap\frac{[j]}{3}$ *;* **5-** $[f]=[f]\cap\frac{[k]}{2}$ *;*
- **6-** $[f]=[f]\cap\frac{[l]}{3}$ *;* **7-** $[a]=[a]\cap\frac{[b]}{2}$ *;* **8-** $[a]=[a]\cap\frac{[c]}{3}$ *;* **9-** $[a]=[a]\cap\frac{[d]}{4}$ *;* **10-** $[a]=[a]\cap\frac{[e]}{5}$ *;*
- **11-** $[a]=[a]\cap\frac{[f]}{6}$ *;*
- CLIMB
- **1-** $[b]=[b]\cap 2[a]$ *;* **2-** $[c]=[c]\cap 3[a]$ *;* **3-** $[d]=[d]\cap 4[a]$; **4-** $[e]=[e]\cap 5[a]$; **5-** $[f]=[f]\cap 6[a]$;
- **6-** $[g]=[g]\cap 2[b]$ *;* **7-** $[h]=[h]\cap 3[b]$ *;* **8-** $[i]=[i]\cap 2[e]$ *;* **9-** $[j]=[j]\cap 3[e]$ *;* **10-** $[k]=[k]\cap 2[f]$ *;*
- **11-** $[l]=[l]\cap 3[f]$. ■

Considérons le graphe dans la Figure 5.2 qui représente toujours le CSP $\mathcal{H}$. Posons "k" la longueur de la plus longue chaîne de la racine aux feuilles de ce dernier (ici k=3). Un algorithme de Waltz classique consiste à prendre, dans un ordre quelconque, les mêmes projections que dans les étapes FALL et CLIMB décrites précédemment. Dans la Figure 5.2, on illustre alors le fait qu'un algorithme de Waltz ferait au plus 5 tours avant de s'arrêter (plus précisément $2(k-1)+1$ tours pour un arbre dont la chaîne de contraintes la plus longue est égale à k). A contrario, si la contraction était faite selon l'algorithme FALL/CLIMB un tour aurait suffit ! En effet, comme on peut le voir dans la Figure 5.2, dans une boucle d'algorithme de Waltz, à chaque étape, le niveau de contraction qu'on peut théoriquement garantir ne correspond qu'à l'un des niveaux des algorithmes FALL ou CLIMB.



boucle **1** d'un algorithme de typeWaltz

boucle **2** d'un algorithme de typeWaltz

boucle **3** d'un algorithme de type Waltz

boucle **4** d'un algorithme de typeWaltz

boucle **5** d'arrêt d'un algorithme de type Waltz

Figure 5.2 : scénario pessimiste d'un algorithme de Waltz.

On voit donc qu'organiser un graphe de contraintes peut être d'un intérêt notoire en terme de temps de calcul. L'objectif fixé dans la suite est d'étendre l'algorithme de FALL/CLIMB à un graphe quelconque, n'ayant pas forcément une représentation graphique sous forme d'arbre, notamment en vectorisant les variables de manière à traiter de façon isolée les cycles. Pour aboutir à cet objectif, on propose de rappeler les outils théoriques qui ont permis de démontrer l'algorithme FALL/CLIMB. On pourra de la sorte présenter une généralisation des résultats jusque là connus avec une démarche proche de la formulation existante.

2 RAPPELS SUR L'ALGORITHME FALL/CLIMB

L'idée de réorganiser le graphe d'un CSP pour améliorer les temps de calculs est très répandue principalement en ce qui concerne les CSP discrets (CSP avec des domaines initiaux qui sont des ensembles discrets). L'explication en est que l'objectif principal des algorithmes pour les CSP discrets est, non pas de trouver toutes les solutions, mais, le plus souvent, de trouver une solution. Ainsi, pour ces CSP discrets, différents niveaux de consistance sont définis puis visés par les algorithmes. Par exemple, la *k-consistance* (voir le livre [Dechter, 2003]**)** signifie que toute instanciation de $k-1$ variables consistantes entre-elles est extensible par n'importe quelle $k^{\text{ième}}$ variable (i.e. étant donnée une instanciation $(\tilde{x}_1,\cdots,\tilde{x}_{k-1})$ qui satisfait toutes les contraintes reliant leurs variables respectives et, étant donnée une variable donnée x_k, on peut trouver dans le domaine de x_k un $\tilde{x}_k$ tel que $(\tilde{x}_1,\cdots,\tilde{x}_k)$ satisfait toutes les contraintes reliant leurs variables respectives). Ces niveaux de consistance sont intéressants dans la mesure où dans une recherche d'une solution d'un CSP, on s'assure de l'extensibilité de solutions d'indice $k-1$ à une solution d'indice k. Par contre, cet exemple de k-consistance n'a pas de sens pratique pour les CSP continus où on cherche plutôt à englober les solutions sans pour autant être forcément capable de les déterminer.

Cette diversité dans les définitions de la consistance pour les CSP discrets entraîne aussi l'existence de différentes définitions de cycles [*Beri et al., 1983*] et donc différentes méthodes de vectorisation. En ce qui concerne les CSP continus, les méthodes de vectorisation ne sont pas aussi diversifiées que pour les CSP discrets. La méthode de vectorisation la plus connue pour les CSP continus a été présentée par [Jaulin et al., 2001b] et s'applique aux SCSP[1]. Nous proposons dans cette partie de rappeler la teneur de ces travaux, en adoptant les mêmes notations puis de démontrer, avec l'aide des domaines de consistance, qu'il existe une extension naturelle des résultats qui y sont démontrés.

2.1 Définition d'un SCSP

La représentation d'un CSP sous forme de SCSP (**S**et **C**onstraint **S**atisfaction **P**roblem) permet d'obtenir des "CSP vectoriels" avec des contraintes binaires et de pouvoir ainsi représenter le CSP sous la forme d'un graphe. La définition d'un SCSP est la suivante :

[1] La définition de ces CSP particuliers est donné dans le 2.1.

Soient $\mathcal{V} = \{v_1, \ldots, v_n\}$ un ensemble fini de vecteurs de variables de dimensions respectives $d_1, \ldots, d_n$ et ayant pour domaines d'appartenance respectifs $D_{v_1}, \cdots, D_{v_n}$. Pour v_i et v_j deux vecteurs dans $\mathcal{V}$, on définit une contrainte binaire $C_{i,j}$ reliant v_i et v_j (ayant par exemple la forme : $v_j = f_{i,j}(v_i)$).

Un SCSP est défini pour un triplet $\mathcal{H} = (\mathcal{V}, D, C)$ avec $\mathcal{V} = \{v_1, \ldots, v_n\}$ un ensemble fini de variables, $D = \{D_{v_1}, \cdots, D_{v_n}\}$ l'ensemble de domaines que décrivent ces variables et C un ensemble fini de contraintes binaires reliant les variables dans $\mathcal{V}$.

Exemple 5.2

1. ■ Considérons un CSP $\mathcal{H}_1 = \left(\mathrm{F}(\tilde{a}, \tilde{b}, \tilde{x}, \tilde{y}, \tilde{z}) = 0 \middle| (\tilde{a}, \tilde{b}, \tilde{x}, \tilde{y}, \tilde{z}) \in [a] \times [b] \times [x] \times [y] \times [z]\right)$ où F est constitué des 2 contraintes $y+x = a$ et $z = 2b$.

   On peut représenter ce CSP sous la forme d'un SCSP en choisissant $\mathcal{V} = \{v_1, v_2, v_3, v_4\}$ où $v_1 = (y, x)$, $v_2 = a$, $v_3 = b$, $v_4 = z$, $D = \{D_{v_1}, \cdots, D_{v_n}\}$ où $D_{v_1} = [x] \times [y]$, $D_{v_2} = [a]$, $D_{v_3} = [b]$, $D_{v_4} = [z]$ et $C = \{C_{1,2}, C_{3,4}\}$ où $C_{1,2} : y+x = a$, $C_{3,4} : z = 2b$.

   On peut remarquer que cette représentation sous forme de SCSP n'est pas unique puisque pour obtenir une contrainte binaire à partir de la contrainte $y+x = a$. on peut prendre $v_1 = (y, x)$ ou $v_1 = (a, x)$ ou encore $v_1 = (a, y)$.

2. Considérons un CSP $\mathcal{H}_2 = \left(\mathrm{F}(\tilde{a}, \tilde{b}, \tilde{x}, \tilde{y}) = 0 \middle| (\tilde{a}, \tilde{b}, \tilde{x}, \tilde{y}) \in [a] \times [b] \times [x] \times [y]\right)$ où F est constitué des 2 contraintes $y+x = a$ et $y - x = b$. On peut représenter ce CSP sous la forme d'un SCSP en choisissant $\mathcal{V} = \{v_1, v_2, v_3\}$ où $v_1 = (y, x)$, $v_2 = a$, $v_3 = b$, $D = \{D_{v_1}, D_{v_2}, D_{v_3}\}$ où $D_{v_1} = [x] \times [y]$, $D_{v_2} = [a]$, $D_{v_3} = [b]$ et $C = \{C_{1,2}, C_{1,3}\}$ où $C_{1,2} : y+x = a$, $C_{1,3} : y - x = b$. Le graphe de ce SCSP est représenté, sur la Figure 5.3 ■

2.2 Théorème de propagation

Après avoir défini les SCSP, on rappelle ici le théorème à la base de la démonstration de l'algorithme FALL/CLIMB. Ensuite, on montrera, dans la prochaine section, le besoin d'extension de ce théorème.

Soient $\mathcal{H}_a = (\mathcal{V}_a, D_a, C_a)$ avec $\mathcal{V}_a = \left(v_{a_1}, \cdots, v_{a_p}\right)$ et $\mathcal{H}_b = (\mathcal{V}_b, D_b, C_b)$ avec $\mathcal{V}_b = \left(v_{b_1}, \cdots, v_{b_q}\right)$ deux SCSP ayant toutes leurs variables distinctes. Supposons que v_{a_1} et v_{b_1} sont reliées par une contrainte f_{a_1,b_1}. Le nouveau SCSP $\mathcal{H}$ formé de l'ensemble des variables dans $\mathcal{H}_a$ et $\mathcal{H}_b$ noté x avec un domaine D qui rassemble leur domaines respectifs dans $\mathcal{H}_a$ et $\mathcal{H}_b$ et de l'ensemble des contraintes dans $\mathcal{H}_a$ et $\mathcal{H}_b$ auxquelles on rajoute la contrainte f_{a_1,b_1} est tel que :

Si v_{a_1} est $\mathcal{H}_a$-consistant et v_{b_1} est $\mathcal{H}_b$-consistant alors la projection de la contrainte sur la dimension de v_{a_1} suffit pour rendre cette dernière consistante avec $\mathcal{H}$ (idem pour v_{b_1}), i.e. $\prod_{v_{b_1}}(S) = \Pi_{f_{a1,b1},v_{b_1}}(D)$ et $\Pi_{v_{a_1}}(S) = \Pi_{f_{a1,b1},v_{a_1}}(D)$

Où S est la solution du CSP $\mathcal{H}$, $\Pi_{f_{a1,b1},v_{b_1}}(D)$ et $\Pi_{f_{a1,b1},v_{a_1}}(D)$ sont les projections des solutions dans D de la contrainte f_{a_1,b_1} sur respectivement v_{b_1} et v_{a_1} .■

2.3 Limite de la formulation sous forme de SCSP

Exemple 5.3

■ Reconsidérons le deuxième SCSP $\mathcal{H}_2$ de l'***Exemple 5.2*** (avec les deux contraintes $y+x = a$ et $y-x = b$), sa représentation sous forme de graphe donne la Figure 5.3.

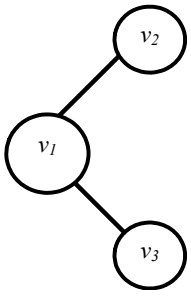


Figure 5.3 : représentation sous forme de graphe du SCSP $\mathcal{H}_2$.

Cette représentation du SCSP $\mathcal{H}_2$ est sous la forme d'un arbre. Mais, en réalité, cet exemple est également le cas linéaire traité dans le chapitre 4 précédent et qui contient un cycle. Un algorithme de Waltz est incapable de trouver une consistance globale et, a fortiori, un algorithme FALL/CLIMB avec une simple réorganisation de l'ordre des projections. ■

La vectorisation en SCSP est intéressante dans le sens où elle permet de représenter un système de contraintes binaires et elle est judicieuse lorsqu'il n'y a pas de cycle. Cependant, comme on l'a vu dans l'***Exemple 5.3***, cette représentation ne permet pas d'isoler les cycles ni de toujours les détecter. De plus, en présence de cycle, on ne peut pas appliquer l'algorithme FALL/CLIMB. On propose donc d'introduire une généralisation des SCSP permettant d'isoler les cycles tout en tirant bénéfice du processus de réorganisation des contraintes présent dans l'algorithme FALL/CLIMB.

3 PREAMBULE : DE QUELLES NOTIONS A-T-ON BESOIN POUR VECTORISER ?

Dans cette section, on cherche à dégager, à travers des exemples académiques, différentes notions qui formalisent l'idée que l'on se fait intuitivement d'un processus de vectorisation d'un CSP.

3.1 Illustration de notions indispensables à la vectorisation

Exemple 5.4

■ Considérons le CSP $\mathcal{H}$: $\left(\mathrm{F}\left(\tilde{a},\tilde{b},\tilde{c},\tilde{d},\tilde{e},\tilde{f},\tilde{g},\tilde{h},\tilde{i},\tilde{j},\tilde{k},\tilde{l}\right)=0 \middle| \left(\tilde{a},\cdots,\tilde{l}\right)\in[a]\times\cdots\times[l]\right)$, avec F le système de contraintes binaires suivant :

{(a=2b ; b=2c ; c=2d ; d=2a) ; (e=2f ; f=2g ; g=2h ; h=2e) ; (i=2j ; j=2k ; k=2l ; l=2i) et c=2e ; d=2j ;}

La représentation sous forme de graphe de ce CSP est donnée sur la Figure 5.4.

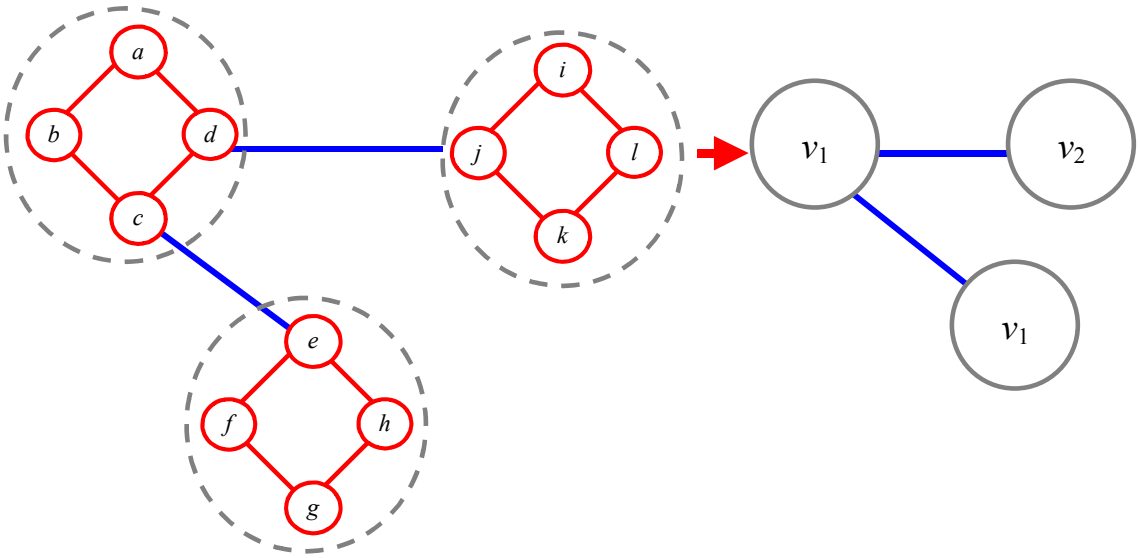


Figure 5.4 : exemple d'un graphe vectorisé en isolant les cycles. ■

A travers cette représentation graphique, on cherche à faire ressortir les idées qui seront formalisées par la suite dans le but de vectoriser et d'isoler les cycles. Après avoir vectorisé d'intuition cet exemple simple, on remarque, tout d'abord, que ce graphe est composé de deux classes essentielles à étudier que sont, d'une part, les "blocs" qui forment les variables vectorisées, et, d'autre part, les contraintes qui les relient, i.e. celles qui ne sont incluses dans aucun bloc (contraintes en traits foncés sur la Figure 5.4).

En ce qui concerne les blocs, pour les définir de manière analogue à ceux matérialisés sur la Figure 5.4, l'idée naturelle est de considérer une structure regroupant à la fois des variables et un système de contraintes reliant ces dernières. Par définition, cette structure est un CSP ! Ceci nous emmène à définir de manière logique ces blocs de variables comme étant des « sous CSP du CSP initial » ou « restrictions du CSP initial ». Une définition complète et détaillée de ces blocs de variables est donnée dans la section 4.4.

La deuxième classe de composantes à prendre en compte est l'ensemble des contraintes qui vont relier les blocs de variables vectorisés (les contraintes en trait foncé sur la Figure 5.4 à gauche). On propose de définir la notion de « restriction du CSP initial à un ensemble de contraintes ». Ceci permettra, à partir d'un sous-système quelconque du système de contraintes initial, de former un nouveau CSP avec les mêmes variables et un système de contraintes réduit. Une définition complète est donnée dans la section 4.3.

3.2 Adaptation des graphes aux problèmes de satisfaction de contraintes

Les graphes sont des outils qui servent essentiellement à modéliser et à résoudre de nombreux problèmes notamment en orientant l'intuition lors d'un raisonnement. Autour de ces graphes existent des résultats connus qui facilitent grandement de nombreux problèmes une fois modélisés. Un graphe est un couple G = (V, U) où V est un ensemble dont les éléments sont appelés des sommets ou nœuds et U est un sous ensemble de parties de V contenant chacune 2 éléments de l'ensemble. Le graphe est dit orienté si on prend des couples dans V×V à la place des parties, non orienté sinon. Des définitions plus détaillées sur les graphes sont données dans la partie C de l'annexe.

Cette définition d'un graphe permet de représenter tous les systèmes de contraintes binaires. Dans le cas d'un CSP, les contraintes ne sont généralement pas que binaires mais peuvent être ramenées à un système de contraintes binaires et ternaires grâce à l'introduction de variables intermédiaires.

Par convention, dans les travaux qui se sont orientés vers une résolution guidée par une représentation sous forme d'un graphe, une contrainte ternaire peut être représentée selon 3 possibilités comme le montre la Figure 5.5. Cette représentation est discutable puisque le choix d'une des 3 possibilités est fait intuitivement pour faciliter la lecture du graphe. On peut cependant se satisfaire de l'aspect primordial d'orientation du raisonnement qui est conservé avec l'adoption de cette convention.

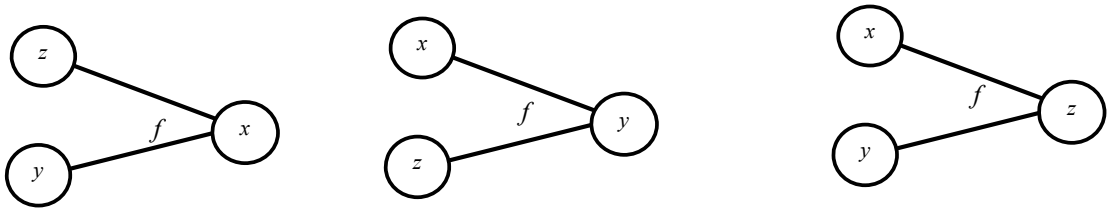


Figure 5.5 : représentations d'une contrainte ternaire.

Par ailleurs, dans la section 2.1, on a vu que l'introduction des SCSP permet de ramener le système de contraintes à un nouveau système binaire. On pourrait alors penser représenter les CSP sous formes de SCSP pour ensuite vectoriser ce dernier, une nouvelle fois, pour isoler les cycles. Cependant, il apparaît que la représentation d'une contrainte ternaire selon la

convention de la Figure 5.5 est plus simple que de ramener le système en des contraintes uniquement binaires par le biais de SCSP pour les deux raisons suivantes :

1. Prenons pour exemple la contrainte ternaire $z = x+y$. Pour obtenir une contrainte binaire, on peut vectoriser, selon un SCSP, de 3 façons : $\{v_1 = (y, x)$ et $v_2 = z\}$ ou $\{v_1 = (z, x)$ et $v_2 = y\}$ ou $\{v_1 = (z, y)$ et $v_2 = x\}$. Ce résultat est valable pour toute contrainte ternaire et on retrouve, de façon symétrique, à travers cette vectorisation, la convention représentée sur la Figure 5.5.

2. En outre dans l'***Exemple 5.3*** (avec les deux contraintes $f_1 : y+x = a$ et $f_2 : y -x = b$), il est apparu qu'avec une représentation sous forme d'un SCSP, on pouvait avoir des situations où la représentation est sous la forme d'un arbre malgré la présence d'un cycle. Il s'avère qu'en adoptant la convention de la Figure 5.5 pour représenter des contraintes ternaires, on ne retrouve plus ces situations (voir la Figure 5.6).

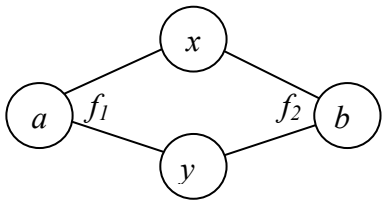


Figure 5.6 : représentation d'un cycle pour un système de deux contraintes ternaires en adoptant une convention pour représenter les contraintes ternaires.

Après avoir choisi d'adopter la convention représentée dans la Figure 5.5, on cherche à définir de façon simple un cycle. Par analogie aux graphes, on définira au préalable une chaîne de variables dans un CSP. Ainsi, on étend, dans les paragraphes 3.1 et 3.2, les définitions connues d'une chaîne et d'un cycle pour les graphes binaires, de façon mieux adaptée, aux CSP décomposés en contraintes binaires et ternaires.

3.2.1 Définition d'une chaîne

La définition d'une chaîne pour un graphe a pour objectif de traduire l'intuition que deux variables dans un CSP peuvent "s'influencer" mutuellement par un processus de contraction. Cette idée se traduit, alors, par le fait qu'on puisse aller d'une variable à l'autre en "empruntant" des arcs du graphe. Pour les CSP décomposés en contraintes binaires et ternaires, on reste fidèle à cette idée. Plus formellement, une chaîne pour un CSP est une suite de variables dans IR, $[v_1, \ldots, v_m]$ avec $m \geq 2$, telles que pour tout $1 \leq k \leq m$-1, il existe une

contrainte f_{i_k}, binaire ou ternaire, vérifiant $(v_k, v_{k+1}) \subset V_{f_{i_k}}$ (on rappelle que $V_{f_{i_k}}$ est défini, dans la section 1 du chapitre 4, comme l'ensemble des variables concernées par la contrainte f_{i_k}) i.e. deux variables consécutives quelconques dans $[v_1, \ldots, v_m]$ sont liées par une contrainte binaire ou ternaire dans F.

3.2.2 Définition d'un cycle

Dans cette partie, on cherche une définition d'un cycle conforme à l'intuition qu'on en a par rapport à la boucle que ces derniers engendrent lors de l'exécution d'un algorithme de Waltz. A travers l'***Exemple 5.5***, on illustre comment aboutir à une définition d'un cycle.

Exemple 5.5

■ Considérons les 2 contraintes $y+x = a$, $y - x = b$. La variable y peut être contractée par x de deux façons possibles car il existe deux chaînes de contraintes entre x et y n'ayant aucune contrainte en commun (plus précisément il existe deux contraintes différentes reliant x et y pour cet exemple). Concrètement, lors de l'exécution d'un l'algorithme de Waltz, pour rendre consistante x par rapport à y, on prend en compte les deux chaînes une à une alors qu'il faudrait les prendre en compte en même temps, d'où la boucle et la consistance locale.

Par ailleurs, il ne faut pas oublier que, du fait de la présence de contraintes ternaires, pour contracter la variable a par exemple, une simple projection de la contrainte $y+x = a$ ne suffit pas : il faut aussi prendre en compte le fait que les variables y et x sont reliées par la contrainte $y - x = b$.

En résumé, si on cherche à définir les cycles à partir des difficultés qui apparaissent lors de l'exécution d'un algorithme de Waltz, il faut prendre en compte :

- les variables qui peuvent "s'influencer" par deux chaînes différentes n'ayant pas de contrainte en commun (à l'exemple ici de x et y),
- Les variables dans les contraintes ternaires apparaissant dans les chaînes (à l'exemple ici de a et b). ■

D'après les conclusions tirées de l'***Exemple 5.5***, on peut dégager une définition des cycles uniquement à partir des chaînes. Considérons alors deux chaînes $[v_1, \ldots, v_m]$ et $[u_1, \ldots, u_p]$ ayant leurs deux extrémités ($v_1 = u_1$ *et* $v_m = u_p$) en commun. Soient, respectivement,

$(h_1, \ldots, h_{m-1})$ et $(g_1, \ldots, g_{p-1})$ les contraintes qui relient les variables dans les chaînes $[v_1, \ldots, v_m]$ et $[u_1, \ldots, u_p]$. Sous l'hypothèse qu'il n'y a pas de contrainte en commun entre les 2 chaînes i.e. $\{h_1, \ldots, h_{m-1}\} \cap \{g_1, \ldots, g_{p-1}\} = \varnothing$, un cycle est alors l'ensemble formé par les variables concernées par ces contraintes i.e. un cycle est l'ensemble des variables $V_{h_1} \cup \cdots \cup V_{h_{m-1}} \cup V_{g_1} \cup \cdots \cup V_{g_{p-1}}$

Exemple 5.6

■ Considérons un CSP $\mathcal{H} = \left(F\left(\tilde{a},\tilde{b},\tilde{c},\tilde{d},\tilde{e}\right)=0 \middle| \left(\tilde{a},\cdots,\tilde{e}\right) \in [a]\times\cdots\times[e]=0\right)$ où F est constitué des contraintes f_1 : $b+d = a$, f_2 : $b-d = c$ et f_3 : $a=2e$. Le graphe de ce CSP est représenté sur la Figure 5.7.

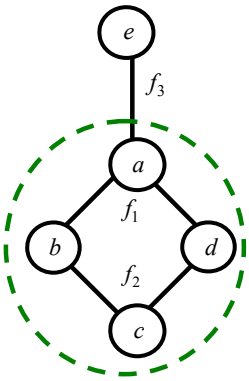


Figure 5.7 : un exemple de CSP contenant un cycle.

Comme exemple de chaîne, on peut prendre les ensembles de variables suivants :

- $[a, b, c]$: a et b liés par la contrainte f_1 et b et c liés par la contrainte f_2,
- $[e, a, b, c]$: e et a liés par la contrainte f_3, a et b liés par la contrainte f_1 et b et c liés par la contrainte f_2.

Comme exemple de cycle, on peut prendre les variables $[a, b, c, d]$ puisque les deux chaînes suivantes : $[b, d]$ avec b *et* d reliés par la contrainte f_1 et $[b, d]$ avec b *et* d reliés par la contrainte f_2 ont leurs deux extrémités en commun. Les variables concernées par les contraintes intervenant dans ces deux chaînes étant $V_{f_1} = \{a, b, d\}$ et $V_{f_2} = \{c, b, d\}$, on a donc bien $\{a, b, c, d\}$ qui est un cycle.

Par contre, on peut vérifier que la variable *e* n'appartient à aucun cycle puisqu'on ne peut pas trouver 2 chaînes contenant *e* et n'ayant pas de contrainte en commun (toute chaîne qui contient *e* contiendra f_3 comme contrainte) et *e* n'appartient à aucune contrainte ternaire. ■

4 INTRODUCTION A LA STRUCTURE DE VCSP

Dans cette partie, on donne les définitions permettant d'introduire les "VCSP" (pour "**V**ectorised **C**onstraint **S**atisfaction **P**roblem") qui sont la matérialisation de l'idée de vectoriser les CSP en isolant des cycles.

4.1 Restriction d'un système de contraintes F à un ensemble d'indices

On définit la restriction du système de contraintes F à l'ensemble d'indice J, que l'on note F_J, comme l'ensemble des contraintes $\left\{f_{j_1},\cdots,f_{j_p}\right\}$ i.e., l'ensemble des contraintes qui ont leur indice inclus dans J

Exemple 5.7

■ Considérons un exemple avec $p = 6$ contraintes et J = {1, 4, 6} alors $F_J = \{f_1, f_4, f_6\}$ ■

4.2 Restriction d'un système de contraintes F à un sous-vecteur

On considère, de la même façon, la restriction du système de contraintes à un sous-vecteur x_J associé à J, qui est définie comme l'ensemble des contraintes qui relient **exclusivement** des variables dans x_J et que l'on note F_{x_J}. En d'autres termes, une contrainte incluant d'autres variables hors de x_J, bien que concernant certaines variable dans x_J, ne sera pas dans F_{x_J}.

Exemple 5.8

■ Considérons un CSP $\mathcal{H} = \left(\mathrm{F}\left(\tilde{a},\tilde{b},\tilde{c},\tilde{d},\tilde{e},\tilde{f},\tilde{g},\tilde{h}\right)=0 \middle| \left(\tilde{a},\cdots,\tilde{h}\right)\in[a]\times\cdots\times[h]\right)$ avec $p = 6$, $n = 8$ et où F est constitué des contraintes suivantes :

$\{f_1 : a-2b=0 ; f_2 : a-3c=0 ; f_3 : a-4d=0$; et $f_4 : a-5e=0 ; f_5 : a-6e=0$ et $f_6 : f-g+h=0\}$.

Prenons le vecteur $x_J = (a, b, d)$ alors : $\mathrm{F}_{x_J} = \{f_1, f_3\}$.■

4.3 Restriction d'un CSP à un sous ensemble de contraintes

On propose de restreindre un CSP à un sous ensemble de contraintes donné F_J. On note $\mathcal{H}_{F_J}$ le sous problème ou le "sous CSP" du CSP $\mathcal{H}$ qui vérifie :

$$\mathcal{H}_{F_J} = \left(F_J(\tilde{x})=0,\ \tilde{x}\in[x]\right) \tag{5.1}$$

De même on note :

$$\mathrm{S}_{F_J} = \left\{\tilde{x}\ \in [x] \mid F_J(\tilde{x})=0\right\} \tag{5.2}$$

Exemple 5.9

■Considérons le CSP de l'***Exemple 5.8***. Sur la Figure 5.8, on représente le CSP $\mathcal{H}$ et sa restriction $\mathcal{H}_{F_J}$ à $F_J = \{f_1, f_3, f_4\}$.

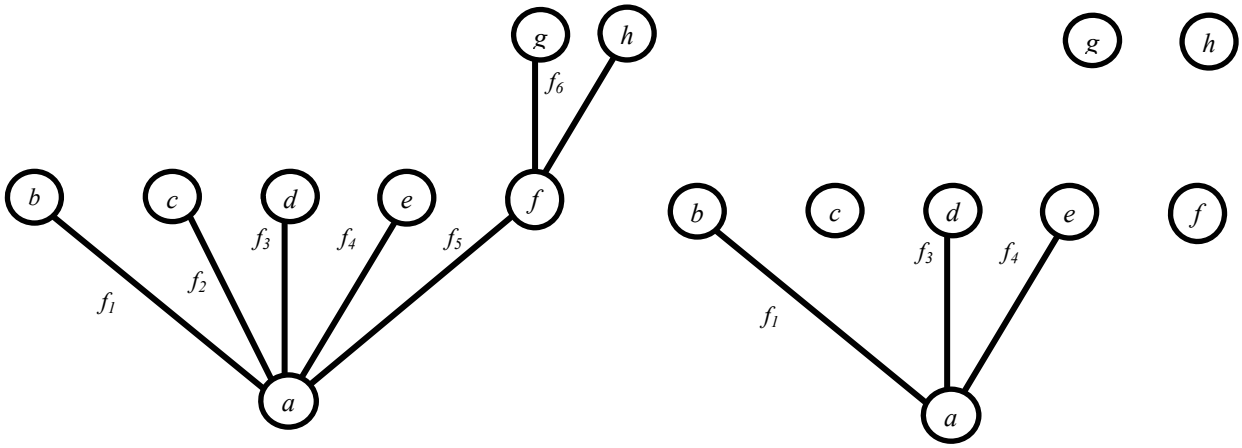


Figure 5.8: représentation graphique d'un CSP $\mathcal{H}$ et de sa restriction $\mathcal{H}_{F_J}$ à $F_J = \{f_1, f_3, f_4\}$. ■

4.4 Restriction d'un CSP à un sous-pavé

On se propose à présent de restreindre un CSP à un sous-pavé $[x_J]$, du pavé initial. Notons alors $\mathcal{H}_{x_J}$ "le sous-CSP" de $\mathcal{H}$ dont le pavé initial est $[x_J]$ et dont l'ensemble des contraintes est la restriction F_{x_J} de F au sous-vecteur x_J :

$$\mathcal{H}_{x_J} = \left(\mathrm{F}_{x_J}(\tilde{x}_J) = 0,\ \tilde{x}_J \in [x_J] \right) \tag{5.3}$$

De même on note :

$$\mathrm{S}_{x_J} = \left\{ \tilde{x}_J \in [x_J] \mid \mathrm{F}_{x_J}(\tilde{x}_J) = 0 \right\} \tag{5.4}$$

Exemple 5.10

■ Considérons le CSP de l'***Exemple 5.8***. Sur la Figure 5.9, on représente le CSP $\mathcal{H}$ et sa restriction $\mathcal{H}_{x_J}$ à $x_J = \{a, d, e, f, h\}$. On peut remarquer que la contrainte f_6 est ternaire et concerne aussi la variable $g \notin x_J$. f_6 n'apparaît donc pas $\mathcal{H}_{x_J}$.

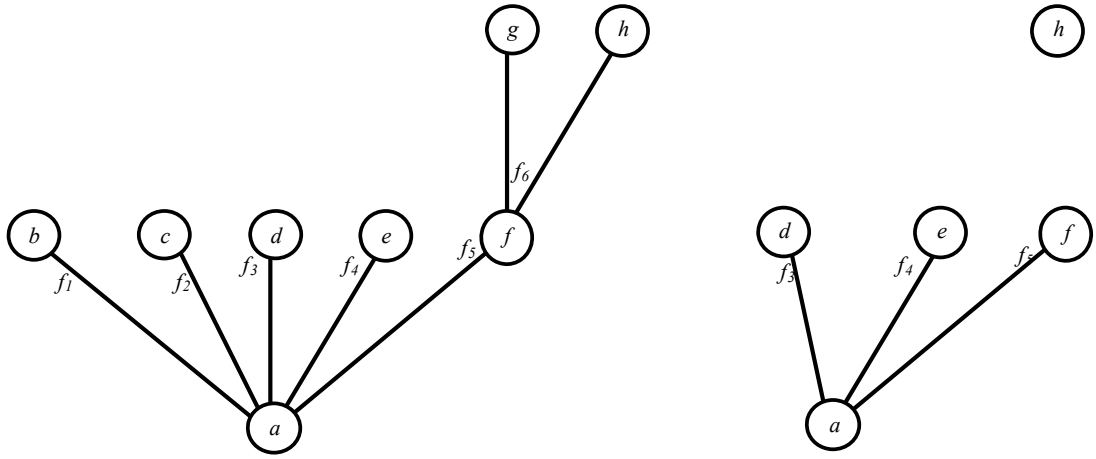


Figure 5.9 : représentation d'un CSP $\mathcal{H}$ et de sa restriction $\mathcal{H}_{x_J}$ à $x_J = \{a, d, e, f, h\}$. ■

4.5 Vectorised Constraints Satisfaction Problem

On s'intéresse à une vectorisation dans le but d'isoler des cycles. Pour cela, on définit un CSP vectorisé ou VCSP pour représenter le problème de la recherche dans un domaine initial de l'ensemble des variables vérifiant les contraintes entre vecteurs de variables et vérifiant également les contraintes à l'intérieur de ses vecteurs de variables. La démarche adoptée pour

décrire les VCSP est la suivante : pour un ensemble de vecteurs de variables donné qui décrit x, on associe au CSP $\mathcal{H}$ une unique représentation sous forme de VCSP suivant cet ensemble de sous-vecteurs. La formulation est alors la suivante :

Considérons $\mathcal{V} = \{v_1, \ldots, v_m\}$, un ensemble de m sous-vecteurs de x associés respectivement à $J_1, \ldots, J_m$, qui vérifient les deux conditions suivantes :

i) Pour tout $i \neq k$ on a $J_i \cap J_k = \varnothing$ (une composante x_i apparaît dans un seul vecteur de variables v_j)

ii) $\overset{m}{\underset{i=1}{\times}} v_i = x$ (les v_k regroupent toutes les variables de x)

Considérons les CSP $\mathcal{H}_{v_1}, \ldots, \mathcal{H}_{v_m}$ qui sont respectivement les restrictions du CSP $\mathcal{H}$ aux variables $v_1, \ldots, v_m$. Notons $F_J = \left\{ f_k \notin \left\{ F_{v_1}, \cdots, F_{v_m} \right\} \right\}$. Enfin, $\forall\ i = 1 \ldots m$, considérons S_{v_i} la solution de la restriction $\mathcal{H}_{v_i}$. On écrit alors la vectorisation du CSP $\mathcal{H}$ suivant $\mathcal{V}$, sous la forme :

$$\mathcal{H}_{\mathcal{V}} : ((\tilde{x} \in S_{F_J}) \text{ et } (\tilde{x}_{J_i} \in S_{v_i} \ \forall\ i = 1 \ldots m),\ \tilde{x} \in [x]) \qquad (5.5)$$

$\mathcal{H}_{\mathcal{V}}$ est le problème de la recherche, sur le pavé initial $[x]$, de l'ensemble des vecteurs $\tilde{x}$ qui vérifient les contraintes F_J (qui sont en dehors des restrictions $\mathcal{H}_{v_i}$) et dont, également, les sous-vecteurs $\tilde{x}_{J_i}$ vérifient les contraintes dans les restrictions $\mathcal{H}_{v_i}$.

Cette écriture est équivalente au CSP $\mathcal{H}$ puisqu'on retrouve le même ensemble de variables x avec les mêmes contraintes. En effet, sachant que $F_J = \left\{ f_k \notin \left\{ F_{v_1}, \cdots, F_{v_m} \right\} \right\}$, on a par définition :

$$(\tilde{x} \in S_{F_J}) \Leftrightarrow (f_k(\tilde{x}) = 0 \text{ pour tout } f_k \notin \left\{ F_{v_1}, \cdots, F_{v_m} \right\}) \qquad (5.6)$$

De plus $\forall\ i = 1 \ldots m$, on a par définition des S_{v_i} :

$(\tilde{x}_{J_i} \in \mathrm{S}_{v_i}) \Leftrightarrow (f_k(\tilde{x}_{J_i}) = 0$ pour tout $f_k \in \mathrm{F}_{v_i})$

$\Leftrightarrow (f_k(\tilde{x}) = 0$ pour tout $f_k \in \mathrm{F}_{v_i})$ (signalons que pour $f_k \in \mathrm{F}_{v_i}$, on peut écrire que $f_k(\tilde{x}) = f_k(\tilde{x}_{J_i})$ puisque la contrainte f_k ne concerne que les dimensions J_i qui sont celles du vecteur de variables v_i)

D'où : $(\tilde{x}_{J_i} \in S_{v_i}) \forall\, i = 1 \ldots m \Leftrightarrow (f_k(\tilde{x}) = 0$ pour tout $f_k \in \mathrm{F}_{v_i}) \forall\, i = 1 \ldots m$

On peut alors écrire que :

$$(\tilde{x}_{J_i} \in S_{v_i}) \forall\, i = 1 \ldots m \Leftrightarrow (f_k(\tilde{x}) = 0 \text{ pour tout } f_k \in \{\mathrm{F}_{v_1}, \cdots, \mathrm{F}_{v_m}\}) \qquad (5.7)$$

En combinant les équations (5.5), (5.6) et (5.7), on peut écrire :

$\mathcal{H}_{\mathcal{V}}$: $((\tilde{x} \in S_{F_J})$ et $(\tilde{x}_{J_i} \in S_{v_i} \forall\, i = 1 \ldots m),\ \tilde{x} \in [x])$

: $((f_k(\tilde{x}) = 0 \ \forall\ f_k \notin \{\mathrm{F}_{v_1}, \cdots, \mathrm{F}_{v_m}\})$ et $(f_k(\tilde{x}) = 0 \ \forall\ f_k \in \{\mathrm{F}_{v_1}, \cdots, \mathrm{F}_{v_m}\},\ \tilde{x} \in [x])$

: $(\mathrm{F}(\tilde{x}) = 0,\ \tilde{x} \in [x])$. D'où l'équivalence avec $\mathcal{H}$.

Exemple 5.11

■ Considérons les 4 variables x, y, z, w, reliées par les contraintes $\{x+y=z, x=2y, y=2z\}$ qui appartiennent à un pavé initial [P]. Le CSP représentant ce problème est alors :

$$\mathcal{H} = ((\tilde{x} + \tilde{y} - \tilde{z} = 0, \tilde{x} - 2\tilde{y} = 0, \tilde{y} - 2\tilde{z} = 0),\ (\tilde{x}, \tilde{y}, \tilde{z}, \tilde{w}) \in [\mathrm{P}])$$

En posant $v_1=(x,y)$, $v_2=(w,z)$ et $\mathcal{V} = (v_1, v_2)$, on peut vérifier les conditions i) et ii) sur $\mathcal{V}$. En effet, v_1 est associé à $\mathrm{J}_1 = \{1, 2\}$ et v_2 est associé à $\mathrm{J}_2 = \{3, 4\}$ et on a bien :

i) $\mathrm{J}_1 \cap \mathrm{J}_2 = \varnothing$

ii) $v_1 \cup v_2 = x$

Les restrictions F_{v_1}, F_{v_2}, F_J et le VCSP $\mathcal{H}_{\mathcal{V}}$ ont les expressions suivantes :

$\mathrm{F}_{v_1} = \{(x - 2y = 0)\}, \mathcal{H}_{v_1} = (\tilde{x} - 2\tilde{y} = 0, (\tilde{x}, \tilde{y}) \in [x] \times [y])$ et $\mathrm{S}_{v_1} = \{(\tilde{x}, \tilde{y}) \in [x] \times [y] \mid \tilde{x} - 2\tilde{y} = 0\}$

$\mathrm{F}_{v_2} = \{\}$, $\mathcal{H}_{v_2} = ((\tilde{w}, \tilde{z}) \in [w] \times [z])$ et $\mathrm{S}_{v_2} = [w] \times [z]$

$\mathrm{F}_J = \{f_i \notin \{\mathrm{F}_{v_1}, \mathrm{F}_{v_2}\}\} = (x + y - z = 0, y - 2z = 0)$ et $\mathrm{S}_{F_J} = \{\tilde{x} \in [x] \mid \mathrm{F}_J(\tilde{x}) = 0\}$

$\mathcal{H}_{\mathcal{V}}$: $((\tilde{x}_{J_1} \in \mathrm{S}_{v_1}$ et $\tilde{x}_{J_2} \in \mathrm{S}_{v_2})$ et $(\tilde{x} \in \mathrm{S}_{F_J}),\ \tilde{x} \in [x])$.■

5 CONDITIONS SUR LA CONNEXITE ENTRE DEUX VECTEURS D'UN GRAPHE

Après avoir défini une nouvelle structure permettant d'écrire un CSP vectorisé, dans cette partie, on cherche des conditions sur la connexité entre deux vecteurs dans un VCSP. L'objectif est de pouvoir appliquer une version généralisée de l'algorithme FALL/CLIMB. Dans un premier temps, on propose une section, constituée essentiellement d'exemples, en préambule des définitions qui vont être données.

5.1 Préambule : études des conditions sur les variables à vectoriser

Dans cette section, on se propose de lister, à travers des illustrations, des situations à éviter lorsqu'on vectorise un CSP, dans le but d'obtenir une condition bien adaptée à une exécution analogue à un algorithme FALL/CLIMB.

Remarquons tout d'abord qu'une propriété essentielle découlant du théorème de propagation et qui rend possible l'algorithme FALL/CLIMB est la suivante : étant donnés deux pavés $[v_1]$ et $[v_2]$ consistants entre eux, supposons que $[v_1]$ soit contracté par une autre contrainte (avec une tierce variable). On a alors $[v_1]$ qui reste consistant avec $[v_2]$ et on doit uniquement propager la contraction au pavé $[v_2]$ pour retrouver l'état de consistance entre les deux variables. Pour illustrer cette propriété, considérons x et y appartenant respectivement à [0 1] et [1 2] et qui vérifient $y=x+1$. Si x est contracté et appartient au nouveau domaine [0.5 1], lorsqu'on reconsidère la contrainte $y=x+1$, alors, seul y est contracté et son domaine devient [1.5 2].

Etudions l'exemple décrit sur la Figure 5.10. On matérialise deux vecteurs de variables connectés par deux contraintes. Dans cette exemple, on illustre les cas à éviter lorsqu'on vectorise un CSP avec comme objectif d'appliquer l'algorithme FALL/CLIMB.

Exemple 5.12

■ Soit le CSP $\mathcal{H}$= (F$(\tilde{a},\tilde{b},\tilde{c},\tilde{d},\tilde{e},\tilde{f},\tilde{g},\tilde{h})$= 0 | $(\tilde{a}\times\cdots\times\tilde{h})$ ∈ [a]×… ×[h]).

Avec F = {$f_1, f_2, f_3, f_4, f_5, f_6$} où : {$f_1$: a+b−d=0 ; f_2 : c−b−d =0 ; f_3 : e−f−h=0; et f_4 : g+f−h =0 ; f_5 : e−2a=0 et f_6 : g−2c=0}.

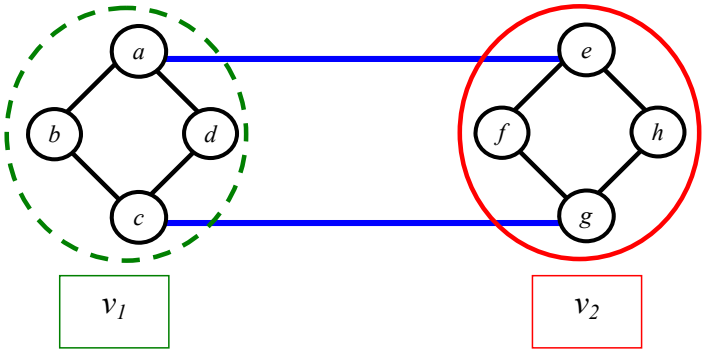


Figure 5.10 : deux vecteurs de variables regroupant des cycles et reliés par des contraintes créant, à leur tour, un cycle■

En considérant la Figure 5.10 et en raisonnant suivant la propriété précédemment énoncée, supposons que v_1 et v_2 soient consistants et que v_1 soit contracté par une autre contrainte. On peut alors remarquer que la présence d'un cycle entre variables de v_1 et v_2 est susceptible de faire se re-contracter la variable v_1. Pour étayer cette remarque, dans l'***Exemple 5.12***. On a v= (a, b, c, d) et v_2 = (e, f, g, h). [a, d, c, g, f, e] est un exemple de cycle contenant des variables de ces deux vecteurs (en effet [a, d, c] et [a, e, f, g, c] sont deux chemins ayant a et c comme extrémités. En supposant que v_1 et v_2 soient consistants, que v_1 soit contracté par une autre contrainte et qu'on veuille rendre consistant le vecteur v_2, le cycle [a, d, c, g, f, e] est susceptible de faire se re-contracter les variables a et c et donc, ce faisant, le vecteur v_1.

Il s'impose alors de créer une condition entre les variables de sorte qu'il n'y ait pas de cycle entre variables des deux vecteurs v_1 et v_2. Ceci nous conduit à définir la condition de "*connexité faible*" dans la section 5.2 suivante.

5.2 Notions de connexité

On donne ici une définition conforme à la condition explicitée dans la section précédente et on définit aussi la condition de *"connexité à une branche"* plus rigide. Considérons un CSP

$\mathcal{H}$ et soient v_1 *et* v_2 deux vecteurs de variables et $\mathcal{H}_{v_1}$ et $\mathcal{H}_{v_2}$ les deux restrictions du CSP $\mathcal{H}$ à ces sous-vecteurs.

- ***Connexité faible[2] entre deux restrictions d'un CSP $\mathcal{H}$.***

  On dit que deux restrictions sont *faiblement connectées* s'il existe une connexion entre elles (dans notre entendement, au moins une contrainte entre deux composantes des deux vecteurs) et si on ne peut pas trouver de cycle dans $\mathcal{H}$ contenant à la fois des variables dans $\mathcal{H}_{v_1}$ et dans $\mathcal{H}_{v_2}$. Autrement dit, s'il existe un cycle formé des variables $(x_{j_1},\cdots,x_{j_q})$ avec un k tel que x_{j_k} est inclus dans v_1 alors l'intersection de l'ensemble des variables de v_2 avec $\{x_{j_1},\cdots,x_{j_q}\}$ est vide (et vice versa). Par analogie, on dit de ces deux vecteurs v_1 et v_2 qu'ils sont *faiblement connectés* et on dit également qu'un VCSP est *à connexité faible* si toutes les connexions entre ses vecteurs sont faibles.

- ***Connexité à une branche entre deux restrictions d'un CSP.***

  On dit que deux restrictions sont *connectées par une branche* s'il n'existe qu'une seule et unique contrainte reliant une variable dans v_1 et une variable dans v_2 (i.e. $\exists!$ $(j, k) \mid v_1(j)$ et $v_1(k)$ sont reliés par une contrainte). Par analogie, on dit de ces deux vecteurs v_1 et v_2 qu'ils sont *connectés par une branche* et on dit également qu'un VSCP est *à connexité à une branche* si toutes les connexions entre ses vecteurs sont à une branche.

Exemple 5.13

■ Soit le CSP $\mathcal{H}_1 = \left(\mathrm{F}\left(\tilde{a},\tilde{b},\tilde{c},\tilde{d},\tilde{e},\tilde{f},\tilde{g},\tilde{h},\tilde{i},\tilde{j},\tilde{k}\right)=0 \middle| \left(\tilde{a},\cdots,\tilde{k}\right)\in[a]\times\cdots\times[k]\right)$, avec F = $\{f_1, f_2, f_3, f_4, f_5, f_6, f_7\}$ où :

$\{f_1 : a+b-d=0$; $f_2 : c-b-d=0$; $f_3 : e-f-h=0$; $f_4 : g+f-h=0$; $f_5 : i+j-k=0$; $f_6 : e-2a=0$ et $f_7 : c-2k=0\}$.

La représentation du graphe de ce CSP est donnée sur la Figure 5.11. Considérons les deux vecteurs v_1 =(a, *b*, *c*, *d*) et v_2 =(*e, f , g, h, i , j, k*). Ces deux vecteurs sont faiblement

[2] Le mot "faible" a été choisi par opposition à l'***Exemple 5.12*** de la Figure 5.10 qui constitue un cas de "*connexité forte*"

connectés puisqu'on ne peut pas trouver de cycle contenant à la fois des variables dans chacun de ces deux vecteurs.

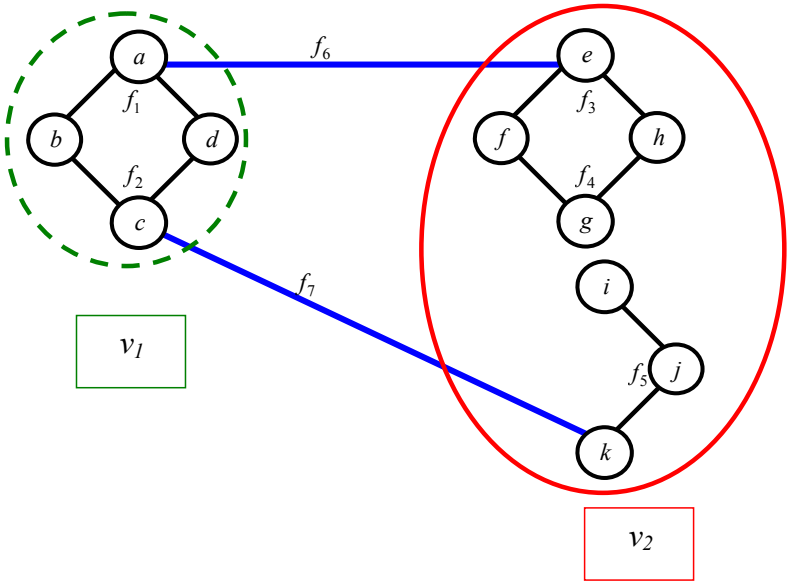


Figure 5.11 : deux vecteurs de variables faiblement connectés.

Soit le CSP $\mathcal{H}_2 = \left(\mathrm{F}\left(\tilde{a},\tilde{b},\tilde{c},\tilde{d},\tilde{e},\tilde{f},\tilde{g},\tilde{h}\right)=0 \middle| \left(\tilde{a},\cdots,\tilde{h}\right)\in[a]\times\cdots\times[h]\right)$|. Avec F = $\{f_1, f_2, f_3, f_4, f_5\}$ où : $\{f_1 : a+b-d=0 ; f_2 : c-b-d=0 ; f_3 : e-f-h=0; f_4 : g+f-h=0$ et $f_5 : e-2a=0\}$. On représente le graphe de ce CSP sur la Figure 5.12. En considérant les deux vecteurs v_1 =(a, b, c, d) et v_2 =(e, f, g, h), on peut vérifier qu'ils sont connectés par une branche.

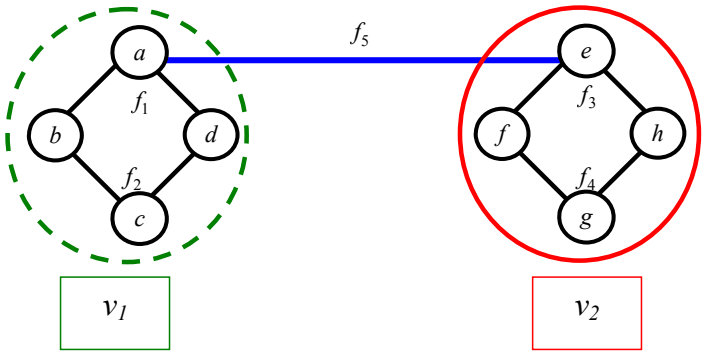


Figure 5.12 : deux vecteurs de variables connectés par une branche. ■

5.3 Discussion : connexité faible et connexité à une branche

On a vu, dans l'exemple de la section 5.1, qu'une condition minimale à respecter entre deux vecteurs pour espérer appliquer un algorithme FALL/CLIMB généralisé est la connexité faible entre vecteurs. Par ailleurs, on remarque que la condition sur la connexité à une branche est plus forte puisque que deux vecteurs connectés par une branche sont aussi faiblement connectés. Dans cette partie, on veut arriver à la conclusion que la meilleure stratégie pour vectoriser est de créer des vecteurs connectés par une branche (plutôt que la connexité faible).

Sur la Figure 5.13, on illustre que, grâce à la scission du vecteur en tirés, les deux vecteurs de variables initiaux (en traits pleins et tirés) qui n'ont pas un cycle en commun (faiblement connexes) peuvent être ramenés à une nouvelle distribution de vecteurs (trois nouveaux vecteurs sur la Figure 5.13) telle qu'on obtienne un graphe de vecteurs connectés par une branche et ayant la structure d'un arbre.

Exemple 5.14

■ Soit le CSP $\mathcal{H} = (\mathrm{F}(\tilde{a},\cdots,\tilde{l}) = 0 \mid (\tilde{a}\times\cdots\times\tilde{l}) \in [a]\times\ldots\times[l])$.

Avec F le système de contraintes $\{f_1 : a+b-d=0\ ; f_2 : c-b-d=0\ ; f_3 : e-f-h=0; f_4 : g+f-h=0\ ; f_5 : i+j-l=0\ ; f_6 : k-j-l=0\ ; f_7 : a+g=0\ ; f_8 : a-2k=0\}$.

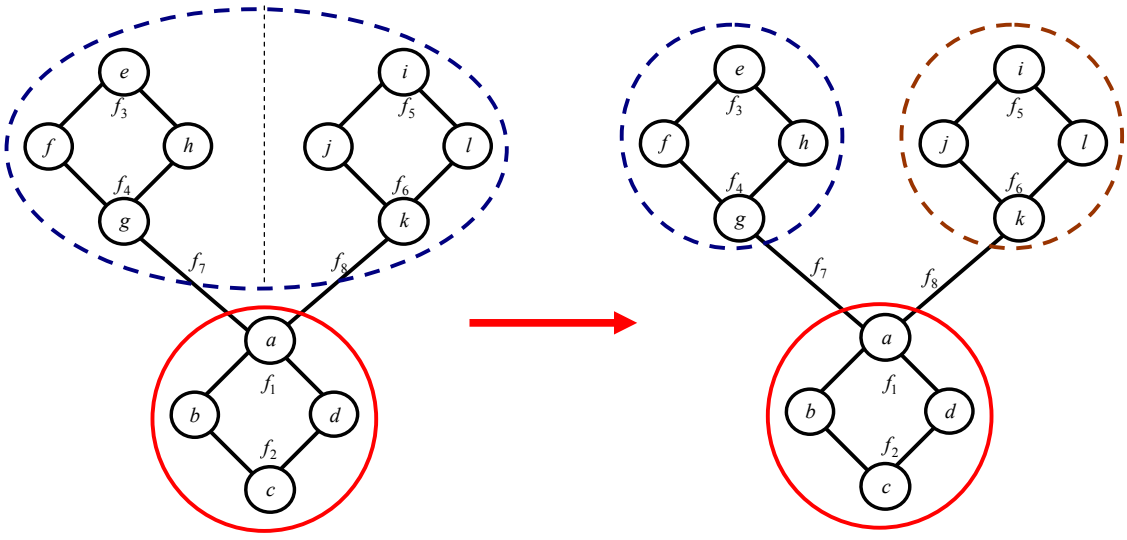


Figure 5.13 : exemple d'un vecteur scindé en deux nouvelles composantes pour obtenir des vecteurs connectés à une branche.

En outre, on peut aussi remarquer qu'en ramenant la condition de connexité faible à celle de la connexité à une branche, on réduit, d'une part, la taille des cycles à résoudre, et d'autre part, on diminue la complexité de l'algorithme à construire pour rendre consistant un vecteur par rapport à un autre.

Concernant ce dernier point, on peut voir cette diminution de la complexité sur la Figure 5.14 (où on représente toujours le graphe de l'***Exemple 5.14***), de la façon suivante : en se limitant uniquement aux 3 variables g, k et a et en faisant fi des autres variables, ces derniers forment un arbre, avec par exemple, la variable a comme racine et deux contraintes que sont $a+g=0$ et $a-2k=0$ qui relient g et a puis k et a. On peut remarquer que, pour rendre consistantes les variables g et k par rapport à la variable a, il faut, conformément à l'algorithme FALL/CLIMB, d'abord rendre consistante la variable a (étape FALL), puis seulement propager sur les variables g et k. En reprenant les vecteurs de variables de la Figure 5.14, sur la partie à gauche, on voit qu'on se trouve dans une situation où, obligatoirement, on utilise localement des propriétés de FALL/CLIMB pour rendre les deux vecteurs de variables consistants entre eux. Tandis que sur la partie à droite, la troncature d'un vecteur en deux permet :

- d'une part, de simplifier la procédure de contraction deux à deux du nouveau trio de vecteur
- d'autre part, les stratégies locales de contraction selon l'algorithme FALL/CLIMB sont intégrées dans un processus de réorganisation globale. On entend par là que dans un VCSP (contenant plusieurs autres vecteurs), ces deux vecteurs deviendraient 3 nouveaux autres vecteurs et que les stratégies d'ordre de contraction entre les 3 vecteurs seraient insérées dans un processus plus global propre au VCSP.

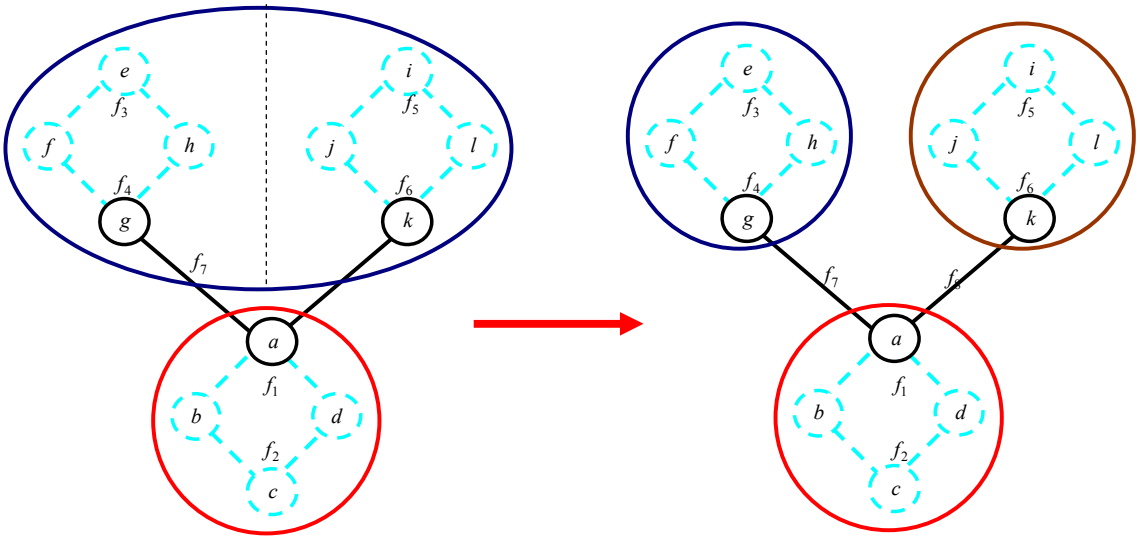


Figure 5.14 : intérêt de rigidifier la condition sur la connexité en introduisant la connexité à une branche.

La propriété de connexité à une branche est donc plus forte que celle de connexité faible mais elle a l'avantage de simplifier les procédures de contraction entre vecteurs de variables.

En résumé, on choisit comme critère de connexion entre vecteurs de variables la connexité à une branche pour les trois raisons suivantes :

- la connexité faible est un critère minimaliste pour espérer généraliser FALL/CLIMB à des vecteurs d'intervalles (comme illustré sur la section 5.1).
- à partir de vecteurs faiblement connectés, on peut toujours se ramener, grâce à des troncatures, à une nouvelle réorganisation de vecteurs connectés à une branche (propriété illustrée sur la Figure 5.13).
- le processus de contraction de deux vecteurs connectés par une branche est plus simple que celui de la contraction de deux vecteurs faiblement connectés (comme illustré sur la Figure 5.14).

Ce choix de la condition sur la connexité entre deux vecteurs étant fixé, la question légitime à se poser est comment rendre consistant un vecteur par rapport à un autre ? La réponse à cette question fait l'objet de la prochaine section.

5.4 Règles de contraction entre vecteurs connectés par une branche

Les règles de contraction entre vecteurs que l'on cherche dans cette partie sont à placer dans l'optique où l'on connaîtrait des résolveurs pour chacun des cycles matérialisés par les vecteurs de variables. En effet, toujours dans une logique de généralisation de l'algorithme FALL/CLIMB, il convient d'avoir des sous-résolveurs qui permettront de contracter un vecteur par rapport aux vecteurs qui lui sont connectés. Pour cela, on distinguera les vecteurs qui lui sont connectés de par une contrainte binaire, des coupless de vecteurs qui lui sont connectés par une contrainte ternaire.

Ainsi, la première question à laquelle on cherche à répondre ici est la suivante : étant donné deux vecteurs pour lesquels on a la condition de connexité à une branche (de par une contrainte binaire les liant) pour lesquels on connaît un résolveur pour chacune des restrictions à ces vecteurs, comment peut on rendre consistant un de ces vecteurs par rapport à son homologue ? La deuxième question à laquelle on cherche à répondre sera la suivante : étant donnés 3 vecteurs de variables pour lesquels on a la condition de connexité à une branche, de par une contrainte ternaire les liant, pour lesquels on connaît un résolveur pour chacune des restrictions à ces vecteurs, comment peut on rendre un de ces vecteurs consistant par rapport à ses deux homologues ?

Pour répondre à ces deux questions, il se trouve que la démarche est assez similaire et les sous-résolveurs qui seront proposés seront assez ressemblants. Par soucis de clarté et pour éviter la répétition et la surcharge, les démonstrations pour les contraintes ternaires seront faites dans la partie D de l'annexe de ce manuscrit.

Exemple 5.15

■ Soit le CSP $\mathcal{H} = \left(\mathrm{F}\left(\tilde{a},\tilde{b},\tilde{c},\tilde{d},\tilde{e},\tilde{f},\tilde{g},\tilde{h}\right)=0 \middle| \left(\tilde{a},\cdots,\tilde{h}\right)\in[a]\times\cdots\times[h]\right)$. Avec F = $\{f_1, f_2, f_3, f_4, f_5\}$ où : $\{f_1 : a+b-d=0 ; f_2 : c-b-d=0 ; f_3 : e-f-h=0$; et $f_4 : g+f-h=0 ; f_5 : a-2e=0\}$

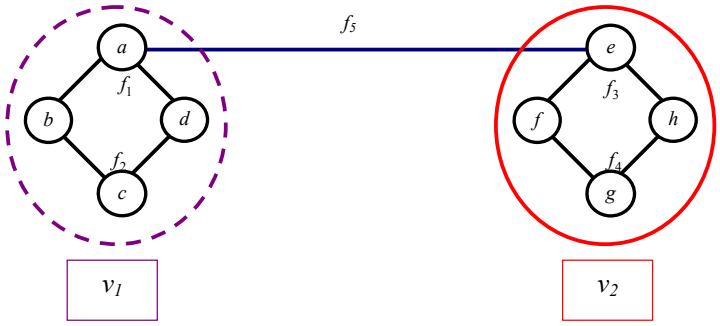


Figure 5.15 : comment rendre consistants deux vecteurs connectés par une branche, de par une contrainte binaire, l'un par rapport à l'autre ? ■

En se référant à la représentation sur la Figure 5.15, pour rendre consistante le vecteur v_1, intuitivement, il faut d'abord résoudre $\mathcal{H}_{v_2}$ pour connaître les solutions dans la variable *e* puis utiliser ces dernières pour rendre consistante la variable *a* et enfin seulement, résoudre le cycle que constitue $\mathcal{H}_{v_1}$. Pour traduire cette idée de manière formelle, on s'attachera, d'abord, pour plus de simplicité, à démontrer un lemme où on considère le vecteur v_1 comme une variable ponctuelle.

Exemple 5.16

Soit le CSP $\mathcal{H} = (\mathrm{F}(\tilde{a},\cdots,\tilde{l})=0 \mid (\tilde{a}\times\cdots\times\tilde{l}) \in [a]\times\ldots\times[l])$. Avec F le système de contraintes $\{f_1 : a+b-d=0 ; f_2 : c-b-d=0 ; f_3 : e-f-h=0; f_4 : g+f-h=0 ; f_5 : i+j-l=0 ; f_6 : k-j-l=0 ; f_7 : a+g-2k=0 \}$.

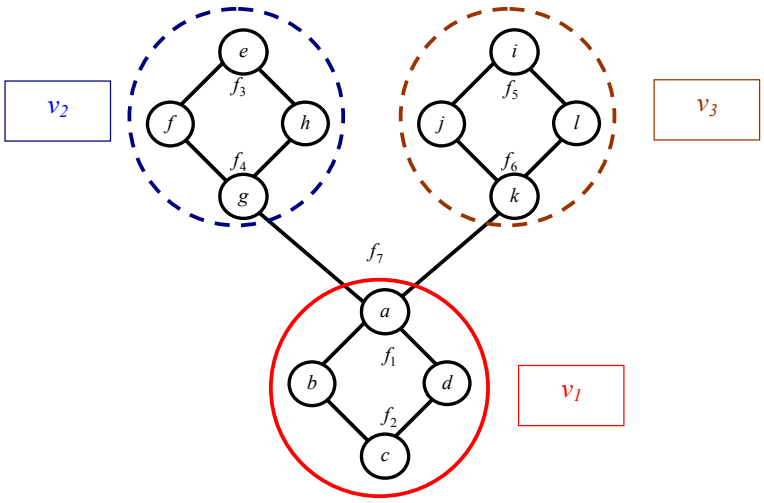


Figure 5.16 : comment rendre consistant un VCSP à connexité à une branche composé de 3 vecteurs reliés par une contrainte ternaire ? ■

En se référant à la représentation sur la Figure 5.16, pour rendre consistante le vecteur v_1, intuitivement, il faut d'abord résoudre $\mathcal{H}_{v_2}$ et $\mathcal{H}_{v_3}$ pour connaître les solutions dans les variables *g* et *k*, puis, utiliser ces dernières pour rendre consistante la variable *a* et enfin seulement, résoudre le cycle que constitue $\mathcal{H}_{v_1}$. Pour traduire cette idée de manière formelle, on s'attachera, également pour plus de simplicité, à démontrer un lemme où on considère le vecteur v_1 comme une variable ponctuelle.

5.4.1 Lemmes

5.4.1.1 couple de vecteurs connectés par une contrainte binaire

Considérons un CSP $\mathcal{H}$: (F(x) = 0, $x \in$ D), une variable scalaire $v_1 = x_j$ et le vecteur de variables v_2 qui est son complémentaire dans x i.e. $v_2 = x_{I-\{j\}}$. Si les deux restrictions $\mathcal{H}_{v_1}$ et $\mathcal{H}_{v_2}$ sont connectées par une branche, alors :

$$\prod_j (\mathrm{S}) = \prod_{f_{i,j},j} \left(\mathrm{D}'_x\right) \tag{5.8}$$

Où i est l'indice de la variable reliée à x_j par une contrainte notée $f_{i,j}$, et $\mathrm{D}'_x = \left(\mathrm{D}'_{x,1}, \cdots, \mathrm{D}'_{x,n}\right)$ est une contraction uniquement en i du domaine initial D de x et qui vérifie :

- Pour $k=i$, $\mathrm{D}'_{x,k} = \prod_i \left(S_{\mathrm{I}-\{j\}}\right)$
- Pour les k restants $\mathrm{D}'_{x,k} = \mathrm{D}_k$

Autrement dit, dans ce lemme, on dit qu'on peut calculer l'ensemble des solutions dans le domaine D_j de x_j en projetant sur la dimension j la relation $f_{i,j}$ qui relie x_j à v_2 et en remplaçant, cependant, le domaine D_i par $\prod_i \left(S_{\mathrm{I}-\{j\}}\right)$ (la projection des solutions du CSP $\mathcal{H}_{v_2}$ sur la $i^{\text{ème}}$ dimension).

Exemple 5.17

■ Soit le CSP $\mathcal{H} = \left(\mathrm{F}\left(\tilde{a},\tilde{b},\tilde{c},\tilde{d},\tilde{e}\right)=0 \middle| \left(\tilde{a}\times\cdots\times\tilde{e}\right)\in[a]\times\cdots\times[e]\right)$. Avec F = $\{f_1, f_2, f_3\}$ où : $\{f_1 : a+b-d=0\ ; f_2 : c-b-d=0\ ; f_3 : e-2b=0\}$

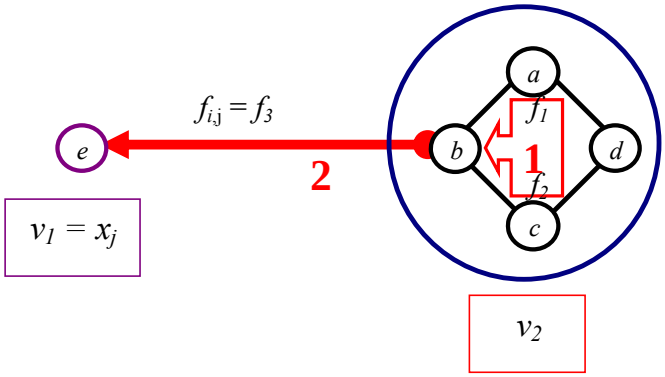


Figure 5.17 : illustration des 2 étapes de calcul pour rendre consistante une variable relié, par une contrainte binaire, à un vecteur de variables.

D'après ce lemme, pour rendre [e] consistant, il faut suivre les 2 étapes comme illustré sur la Figure 5.17. Consécutivement, on résout $\mathcal{H}_{v_2}$: ($\mathrm{G}\left(\tilde{a},\tilde{b},\tilde{c},\tilde{d}\right)=0$ | $\left(\tilde{a}\times\cdots\times\tilde{d}\right)$ ∈ [a]×…[d]), avec G = $\{f_1, f_2\}$ puis on projette la contrainte f_3 : $e-2b=0$ en considérant pour domaine de b la projection dans b des solutions du CSP $\mathcal{H}_{v_2}$.■

Preuve :

■ D'après le ***théorème 1*** (chapitre 4), on a : $\Pi_j(\mathrm{S}) = [x_j] \cap \mathrm{D_F}(\underset{i\in I-\{j\}}{\times}[x_i])$

On sait de plus que d'après le ***corollaire 2*** du ***théorème 3***, on a : $\mathrm{D_F}(\underset{i\in I-\{j\}}{\times}[x_i]) = \mathrm{D}_{f_{\{j\}}}(\prod_{j_1\cdots j_q}(\mathrm{S}_{I-\{j\}}))$. Sachant que, dans le cas présent, l'ensemble des variables (j_1, …, j_q) qui sont reliées par une contrainte à j est réduit à i, et l'ensemble des contraintes $f_{\{j\}}$ qui concernent j est réduit à la seule contrainte $f_{i,j}$, on peut donc écrire :

$$\mathrm{D_F}(\underset{i\in I-\{j\}}{\times}[x_i]) = \mathrm{D}_{f_{i,j}}(\prod_i(\mathrm{S}_{I-\{j\}}))$$

En reprenant l'expression $\Pi_j(\mathrm{S}) = [x_j] \cap \mathrm{D_F}(\underset{i\in I-\{j\}}{\times}[x_i])$, on obtient : $\Pi_j(\mathrm{S}) = [x_j] \cap \mathrm{D}_{f_{i,j}}(\prod_i(\mathrm{S}_{I-\{j\}}))$. Or, comme on l'a vu dans la section 3.1 du chapitre 4 (il s'agissait

notamment d'écrire une projection a l'aide de calculs de domaine de consistance), l'expression $[x_j] \cap D_{f_{i,j}}(\prod_i(S_{I-\{j\}}))$ correspond à la projection de la contrainte $f_{i,j}$ sur la $j^{\text{ième}}$ dimension en considérant le domaine des x_i possibles égal au domaine $\prod_i(S_{I-\{j\}})$. On peut alors écrire cette projection sous la forme $\prod_{f_{j,l,r},j}(D'_x)$ avec D'_x vérifiant :

- Pour $k=i$, $D'_{x,k} = \prod_i(S_{I-\{j\}})$
- Pour les k restants $D'_{x,k} = D_k$

D'où : $\prod_j(S) = \prod_{f_{i,j}}(D'_x)$.■

5.4.1.2 Triplet de vecteurs connectés par une branche

Considérons trois restrictions de CSP $\mathcal{H}_{v_1}, \mathcal{H}_{v_2}$ et $\mathcal{H}_{v_3}$ connectés uniquement par une contrainte ternaire que l'on note $f_{j,l,r}$ qui relient les variables d'indices j, l, r dans respectivement v_1, v_2 et v_3. On suppose que $v_1 = x_j$ est une variable scalaire. En considérant le CSP $\mathcal{H}$: ($F(x) = 0$, $x \in D$) qui regroupe ces variables respectivement avec leur domaine et toutes leurs contraintes plus la contrainte $f_{j,l,r}$, on peut écrire :

$$\prod_j(S) = \prod_{f_{j,l,r},j}(D'_x) \tag{5.9}$$

Où $\prod_{f_{j,l,r},j}(D'_x)$ et la projection de la contrainte $f_{j,l,r}$ sur le domaine $D'_x = (D'_{x,1}, \cdots, D'_{x,n})$. D'_x est une contraction uniquement en l et r du domaine initial D de x et qui vérifie :

- Pour $k=l$, $D'_{x,k} = \prod_l(S_{v_2})$
- Pour $k=r$, $D'_{x,k} = \prod_r(S_{v_3})$
- Pour les k restants $D'_{x,k} = D_k$

Autrement dit, dans ce lemme, on dit qu'on peut calculer l'ensemble des solutions dans le domaine D_j de x_j en projetant sur la dimension j la relation $f_{j,l,r}$ qui relie x_j à v_2 et v_3 et en

remplaçant, cependant, les domaine D_l et D_r par, respectivement $\prod_l\left(S_{v_2}\right)$ et $\prod_r\left(S_{v_3}\right)$ (la projection des solutions du CSP $\mathcal{H}_{v_2}$ sur la $l^{\text{ième}}$ dimension et la projection des solutions du CSP $\mathcal{H}_{v_3}$ sur la $r^{\text{ième}}$ dimension).

Exemple 5.18

■ Soit le CSP $\mathcal{H}=\left(\mathrm{F}\left(\tilde{a},\tilde{b},\tilde{c},\tilde{d},\tilde{e},\tilde{t},\tilde{w},\tilde{x},\tilde{z}\right)=0\middle|\left(\tilde{a},\tilde{b},\tilde{c},\tilde{d},\tilde{e},\tilde{t},\tilde{w},\tilde{x},\tilde{z}\right)\in[a]\times\cdots\times[z]\right)$. Avec F = $\{f_1, f_2, f_3, f_4, f_5,\}$ où : $\{f_1 : a+b-d=0 ; f_2 : c-b-d =0 ; f_3 : e-2b+x=0, f_4 : w+x-z=0, f_5 : t-w-z=0\}$

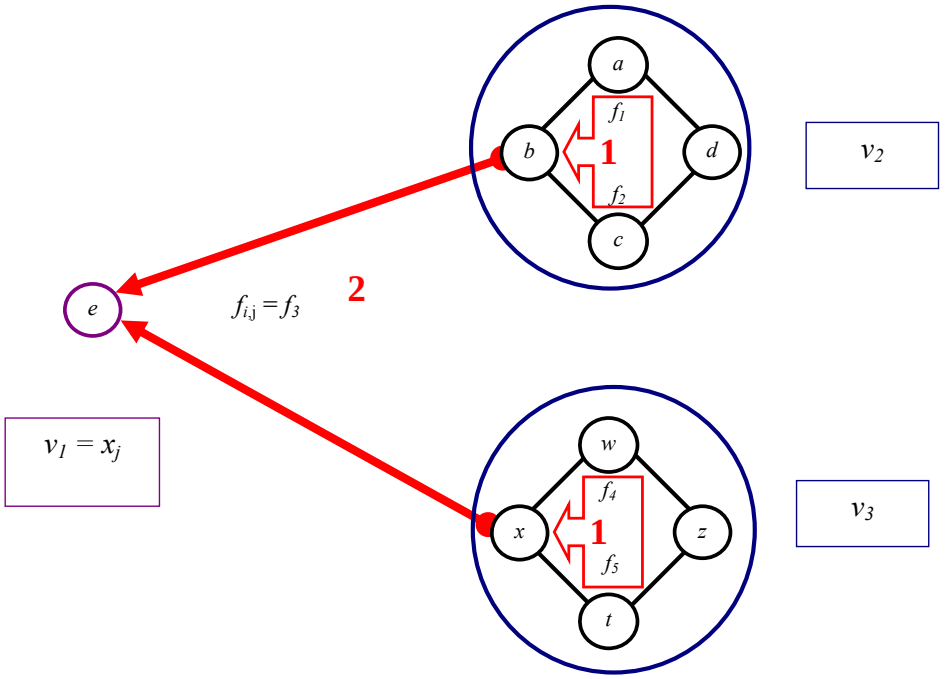


Figure 5.18 : illustration des 2 étapes de calcul pour rendre consistante une variable reliée, par une contrainte ternaire, avec un couple de vecteurs de variables.

D'après ce lemme, pour rendre [e] consistant, il faut suivre les 2 étapes comme illustré sur la Figure 5.18. Consécutivement, on résout $\mathcal{H}_{v_2}$: $(\mathrm{G}(\tilde{a},\tilde{b},\tilde{c},\tilde{d})=0 \mid (\tilde{a}\times\cdots\times\tilde{d}) \in [a]\times\ldots[d])$, avec G = $\{f_1, f_2\}$ et $\mathcal{H}_{v_3}$: $(\mathrm{H}(\tilde{t},\tilde{w},\tilde{x},\tilde{z})=0 \mid (\tilde{t}\times\cdots\times\tilde{z}) \in [t]\times\ldots[z])$, avec H = $\{f_3, f_4\}$. Puis, on projette la contrainte $f_{j,l,r}=f_3$: $e-2b+x=0$ en considérant, respectivement, pour domaine de b et de x , la projection dans b des solutions du CSP $\mathcal{H}_{v_2}$ et la projection dans x des solutions du CSP $\mathcal{H}_{v_3}$.■

On se propose d'étendre ces résultats, pour les contraintes binaires et ternaires, à une situation où, cette fois ci, on considère une variable vectorielle en lieu et place de la variable ponctuelle x_j.

5.4.2 Proposition de sous-résolveurs

5.4.2.1 Duo de vecteurs connectés par une contrainte binaire

Considérons un CSP $\mathcal{H} = (F(x) = 0, x \in D)$ et son VCSP $\mathcal{H}_{\mathcal{V}}$ pour $\mathcal{V} = (v_1, v_2)$. On suppose que les deux restrictions $\mathcal{H}_{v_1}$ et $\mathcal{H}_{v_2}$ sont connectées par une branche. On propose de caractériser l'ensemble des solutions du CSP $\mathcal{H}$ dans $[v_1]$. Par symétrie ce résultat sera applicable pour $[v_2]$. Posons alors l et r les indices dans v_1 et dans v_2 des deux variables reliées par une contrainte noté $f_{l,r}$ et notons n_1 la dimension de v_1. On a pour tous les indices k dans v_1 :

$$\prod_k(S) = \prod_k\left(S'_{v_1}\right) \tag{5.10}$$

Où S est la solution du CSP $\mathcal{H}$ et S'_{v_1} est la solution de la restriction $\mathcal{H}_{v_1}$ avec un domaine initial $D'_{v_1} = \left(D'_{v_1,1}, \cdots, D'_{v_1,n_1}\right)$ qui vérifie :

- Pour $k = l$, $D'_{v_1,k} = \prod_{f_{l,r},l}(D'_x)$
- Pour les k restants dans v_1, $D'_{v_1,k} = D_k$

Où D'_x est une contraction du domaine associé à x qui vérifie :

- Pour $k = r$, $D'_{x,k} = \prod_r(S_{v_2})$
- Pour les k restants dans x, $D'_{x,k} = D_k$

Pour résumer cette proposition, on dira que, grâce à la condition de connexité à une branche, pour rendre $\mathcal{H}$-consistant le vecteur v_1 par exemple, dans une première étape « on résoudra localement le cycle représenté par $\mathcal{H}_{v_2}$ pour obtenir en particulier le domaine solution pour la variable reliée à v_1 », puis dans une deuxième étape, « on transmettra la contraction à la variable v_1 par une simple projection». Arrivé à ce niveau de contraction,

dans une troisième étape, pour rendre v_1 $\mathcal{H}$-consistant, « il suffit de résoudre la restriction du CSP au vecteur v_1 ».

On choisit, pour la suite, de regrouper les deux premières étapes en un sous-résolveur $R_{v_2 \to v_1} : \mathrm{D}_{v_1} \to R_{v_2 \to v_1}\left(\mathrm{D}_{v_1}\right)$ opérant de l'espace de v_1 dans lui-même. Ce résolveur $R_{v_2 \to v_1}$ consiste donc à transmettre la contraction du vecteur de variables v_2 au vecteur v_1 par une projection, après avoir résolue la variable de v_2 reliée à v_1.

Notons R_{v_1} un sous-résolveur rendant consistant le vecteur v_1. Le sous-résolveur décrit par cette proposition, rendant consistant v_1 par rapport à v_2, peut s'écrire $R_{v_1} \circ R_{v_2 \to v_1} : \mathrm{D}_{v_1} \to R_{v_1} \circ R_{v_2 \to v_1}\left(\mathrm{D}_{v_1}\right)$. Ce résolveur $R_{v_1} \circ R_{v_2 \to v_1}$ opère de l'espace de v_1 dans lui-même.

On a ainsi construit un sous-résolveur que l'on notera par la suite $R_{v_1} \circ R_{v_2 \to v_1}$ tel que $R_{v_1} \circ R_{v_2 \to v_1}$ donne le domaine de v_1 consistant avec le domaine de v_2. La Figure 5.19 qui représente l'***Exemple 5.15*** illustre ces 2 premières étapes nécessaires pour $R_{v_1} \circ R_{v_2 \to v_1}$ et la troisième étape pour R_{v_1}.

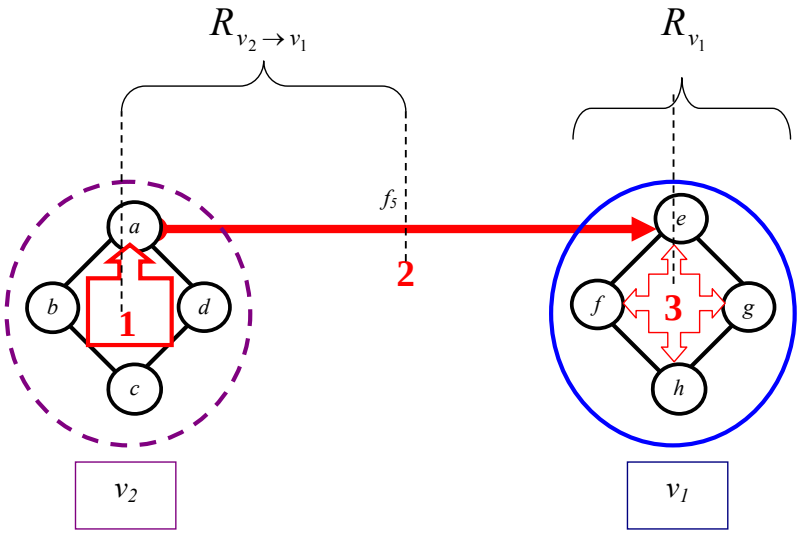


Figure 5.19 : illustration des 3 étapes de calcul pour rendre consistant v_1 connecté à une branche à v_2. ■

Preuve

■ La démonstration de cette proposition se fera par la preuve que l'inclusion dans les deux sens est vraie :

1. Pour tous les indices *k* dans v_1, montrons que $\prod_k(S) \subset \prod_k\left(S'_{v_1}\right)$

   Cette inclusion est évidente puisque, pour obtenir $\prod_k\left(S'_{v_1}\right)$, on contracte consécutivement des variables du CSP $\mathcal{H}$ avec des contraintes qui sont vérifiées par les solutions S. Donc ces solutions sont conservées.

2. Pour tous les indices *k* dans v_1, montrons que $\prod_k\left(S'_{v_1}\right) \subset \prod_k(S)$

   Pour plus de clarté, on suppose que les premiers indices sont rangés dans v_1 et le reste dans v_2.

   Soit $\tilde{x}_k \in \prod_k\left(S'_{v_1}\right)$. Par hypothèse, S'_{v_1} est la solution de la restriction $\mathcal{H}_{v_1}$ avec un domaine initial qui est $\mathrm{D}'_{v_1} \Rightarrow \forall\, i \in \{1, \ldots, n_1\}\text{-}\{k\}, \exists\, \tilde{x}_i \in \mathrm{D}'_{v_1,i} \mid \mathrm{F}_{v_1}\left(\tilde{x}_1, \cdots, \tilde{x}_{n_1}\right) = 0$.

   De plus, pour $i = l$, on a $\tilde{x}_i \in \mathrm{D}'_{v_1,i} = \prod_{f_{l,r},l}\left(\mathrm{D}'_x\right)$. Notons alors la variable $v_3 = v_2 \times v_1(i)$. Remarquons que la contrainte $f_{l,r}$ ne concerne , dans $\mathcal{H}_V$, que les deux variables dans les indices l et r qui appartiennent également au vecteur v_3. On peut donc écrire $\prod_{f_{l,r},l}\left(\mathrm{D}'_x\right) = \prod_{f_{l,r},l}\left(\mathrm{D}'_{v_3}\right)$ où D'_{v_3} est la restriction du domaine D'_x sur le vecteur de variables v_3. Ainsi donc, en considérant le CSP $\mathcal{H}_{v_3}$, et $v_1(i)$, on est dans les conditions du lemme 1 $\Rightarrow$ la projection $\prod_{f_{l,r},l}\left(\mathrm{D}'_{v_3}\right)$ permet d'obtenir les $\tilde{x}_i$ consistants avec le CSP $\mathcal{H}_{v_3}$. D'où l'existence d'un $\tilde{v}_2 \in [v_2]$ tel que $F_{v_3}\left(\tilde{x}_i, \tilde{v}_2\right) = 0$.

   D'où le n-uplet $\left(\tilde{x}_1, \cdots, \tilde{x}_{n_1}\right) \times \tilde{v}_2$ vérifie : $\mathrm{F}_{v_1}\left(\tilde{x}_1, \cdots, \tilde{x}_{n_1}\right) = 0$ et $\mathrm{F}_{v_3}(x_i, v_2) = 0$. Or, F_{v_1} regroupe toutes les contraintes dans la variable v_1 et F_{v_3} regroupe toutes les contraintes dans la variable v_2, plus la seule contrainte concernant des variables dans v_1 et v_2, d'où la conclusion que le n-uplet $\left(\tilde{x}_1, \cdots, \tilde{x}_{n_1}\right) \times \tilde{v}_2$ vérifie $\mathrm{F}\left(\left(\tilde{x}_1, \cdots, \tilde{x}_{n_1}\right) \times \tilde{v}_2\right) = 0$

   $\Rightarrow \left(\tilde{x}_1, \cdots, \tilde{x}_{n_1}\right) \times \tilde{v}_2 \in \mathrm{S} \Rightarrow \tilde{x}_k \in \prod_k(S) \Rightarrow \prod_k\left(S'_{v_1}\right) \subset \prod_k(S)$.■

5.4.2.2 Triplet de vecteurs connectés par une contrainte ternaire

Considérons un CSP $\mathcal{H} = (F(x) = 0, x \in D)$ et son VCSP $\mathcal{H}_{\mathcal{V}}$ pour $\mathcal{V} = (v_1, v_2, v_3)$. On suppose que $\mathcal{H}_{\mathcal{V}}$ est à connexité à une branche de par une contrainte ternaire reliant les trois vecteurs. On propose de caractériser l'ensemble des solutions du CSP $\mathcal{H}$ dans $[v_1]$. Par symétrie ce résultat sera applicable pour $[v_2]$ ou $[v_3]$. Posons alors j, l, et r les indices, respectivement, dans v_1, v_2 et v_3 des trois variables reliées par une contrainte noté $f_{j,q,r}$ et notons n_1 la dimension de v_1. On a pour tous les indices k dans v_1 :

$$\prod_k(S) = \prod_k\left(S'_{v_1}\right) \tag{5.11}$$

Où S est la solution du CSP $\mathcal{H}$ et S'_{v_1} est la solution de la restriction $\mathcal{H}_{v_1}$ avec un domaine initial $D'_{v_1} = \left(D'_{v_1,1}, \cdots, D'_{v_1,n_1}\right)$ qui vérifie :

- Pour $k = l$, $D'_{v_1,k} = \prod_{f_{j,l,r},j}(D'_x)$
- Pour les k restants dans v_1, $D'_{v_1,k} = D_k$

Où D'_x est une contraction du domaine associé à x qui vérifie :

- Pour $k=l$, $D'_{x,k} = \prod_l\left(S_{v_2}\right)$
- Pour $k=r$, $D'_{x,k} = \prod_r\left(S_{v_3}\right)$
- Pour les k restants dans x, $D'_{x,k} = D_k$

Pour résumer cette proposition, on dira que, grâce à la condition de connexité à une branche, pour rendre $\mathcal{H}$-consistant le vecteur v_1 par exemple, dans une première étape « on résoudra localement les cycle représentés, respectivement, par $\mathcal{H}_{v_2}$ et $\mathcal{H}_{v_3}$ pour obtenir en particulier les domaines solutions pour les variable reliées à v_1 », puis dans une deuxième étape, « on transmettra la contraction à la variable v_1 par une simple projection». Arrivé à ce niveau de contraction, dans une troisième étape, pour rendre v_1 $\mathcal{H}$-consistant, « il suffit de résoudre la restriction du CSP au vecteur v_1 ».

On choisit, pour la suite, de regrouper les deux premières étapes en un sous-résolveur $R_{(v_2,v_3)\to v_1}: \mathrm{D}_{v_1} \to R_{(v_2,v_3)\to v_1}\left(\mathrm{D}_{v_1}\right)$ opérant de l'espace de v_1 dans lui-même. Ce résolveur $R_{(v_2,v_3)\to v_1}$ consiste donc à transmettre la contraction du couple de vecteurs de variables (v_2, v_3) par rapport au vecteur v_1 par une projection, après avoir résolu les variables respectivement dans les vecteur v_2 et v_3 reliées à v_1.

Notons R_{v_1} un sous-résolveur rendant consistant le vecteur v_1. Le sous-résolveur décrit par cette proposition, rendant consistant v_1 par rapport à v_2 et v_3, peut s'écrire $R_{v_1} \circ R_{(v_2,v_3)\to v_1}: \mathrm{D}_{v_1} \to R_{v_1} \circ R_{(v_2,v_3)\to v_1}\left(\mathrm{D}_{v_1}\right)$. Ce résolveur $R_{v_1} \circ R_{(v_2,v_3)\to v_1}$ opère de l'espace de v_1 dans lui-même.

On a ainsi construit un sous-résolveur que l'on notera par la suite $R_{v_1} \circ R_{(v_2,v_3)\to v_1}$ tel que $R_{v_1} \circ R_{(v_2,v_3)\to v_1}$ donne le domaine de v_1 consistant avec le couple de vecteurs de (v_2, v_3). La Figure 5.20 qui représente l'***Exemple 5.16*** illustre ces 2 premières étapes nécessaires pour $R_{(v_2,v_3)\to v_1}$ et la troisième étape pour R_{v_1}.

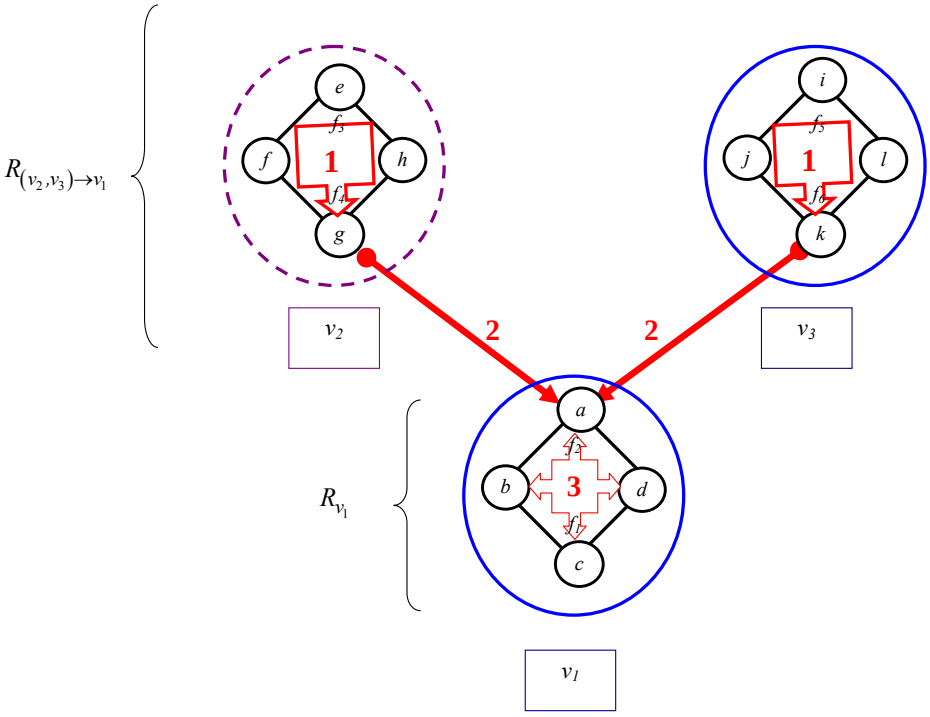


Figure 5.20 : illustration des 3 étapes de calcul pour rendre consistant v_1 connecté à deux autres vecteurs par une contrainte ternaire. ■

Conséquences

Cette proposition a deux conséquences. D'abord, étant donnés des vecteurs de variables v_1 et v_2 tels que les restrictions sur ces vecteurs $\mathcal{H}_{v_1}$ et $\mathcal{H}_{v_2}$ soient connectées par une branche, et en supposant connus deux résolveurs R_{v_1} et R_{v_2} pour respectivement $\mathcal{H}_{v_1}$ et $\mathcal{H}_{v_2}$, on sait alors construire des sous-résolveurs pour rendre consistant l'un par rapport à l'autre ces deux vecteurs. De même, pour un triplet de vecteurs reliés par une contrainte ternaire, et pour lesquels on suppose connus des résolveurs, on sait alors construire des sous-résolveurs pour rendre consistant un vecteur par rapport à l'autre couple de vecteur.

La deuxième conséquence est de pouvoir obtenir, à condition d'avoir vectorisé un CSP sous forme d'arbre présentant des vecteurs connectés par une branche, une consistance globale grâce à une adaptation de l'algorithme FALL/CLIMB, tout en minimisant le plus possible la difficulté des cycles à résoudre.

6 ALGORITHME FALL/CLIMB POUR UN VCSP

Dans cette partie, on décrit une généralisation des algorithmes FALL et CLIMB utilisée pour contracter un VSCP lorsqu'il remplit la condition sur la connexité à une branche, et lorsque sa représentation graphique est sous forme d'un arbre. La base de l'algorithme FALL/CLIMB pour les SCSP se trouve être le théorème de propagation grâce auquel, on démontre le théorème de rétro-propagation puis l'optimalité de l'algorithme (voir [Jaulin et al., 2001b]). C'est pourquoi, on se borne juste à donner une version adaptée aux VCSP du théorème de propagation, ce qui prouve entièrement l'algorithme de FALL/CLIMB qui sera alors proposé.

6.1 Théorème de propagation pour les VCSP

Dans cette partie on généralise le théorème de propagation nécessaire pour démontrer notamment l'algorithme FALL/CLIMB. Pour résumer, ce théorème permet, lorsque l'on connecte deux CSP distincts, sous certaines hypothèses, de donner des techniques pour résoudre le CSP ainsi obtenu. Dans notre cas on peut envisager de connecter deux CSP liés par une contrainte binaire ou encore trois CSP liés par une contrainte ternaire. Ainsi, on

propose d'écrire ce théorème sous deux déclinaisons, une pour les contraintes binaires et une pour les contraintes ternaires

6.1.1 Formulation pour des contraintes binaires

Posons $\mathcal{V}_a = \left(v_{a_1}, \cdots, v_{a_q}\right)$ et $\mathcal{V}_b = \left(v_{b_1}, \cdots, v_{b_r}\right)$, et soient $\mathcal{H}_{\mathcal{V}_a}$ et $\mathcal{H}_{\mathcal{V}_b}$ deux VCSP avec des vecteurs de variables distincts i.e. $v_{a,j} \neq v_{b,i}$ pour tout couple possible (*i, j*). On suppose que $\mathcal{H}_{\mathcal{V}_a}$ et $\mathcal{H}_{\mathcal{V}_b}$ sont connectées par une branche de par deux vecteurs de variables v_{a_1} et v_{b_1} et de par une contrainte notée f_{a_1,b_1}. Le nouveau VCSP $\mathcal{H}_{\mathcal{V}}$, où $\mathcal{V} = \mathcal{V}_a \bigcup \mathcal{V}_b$ avec les domaines $\mathrm{D}_a \cup \mathrm{D}_b$ et pour lequel, l'ensemble de ses contraintes regroupe les contraintes dans $\mathcal{H}_{\mathcal{V}_a}$ et $\mathcal{H}_{\mathcal{V}_b}$ plus la contrainte f_{a_1,b_1} vérifie :

Si v_{a_1} est $\mathcal{H}_{\mathcal{V}_a}$-consistant et v_{b_1} est $\mathcal{H}_{\mathcal{V}_b}$-consistant alors :

$\Pi_{v_{b_1}}(\mathrm{S}) = R_{v_{b_1}} \circ R_{v_{a_1} \to v_{b_1}}\left(\mathrm{D}_{v_{b_1}}\right)$ et $\Pi_{v_{a_1}}(\mathrm{S}) = R_{v_{a_1}} \circ R_{v_{b_1} \to v_{a_1}}\left(\mathrm{D}_{v_{a_1}}\right)$.Où :

- S est la solution du CSP $\mathcal{H}_{\mathcal{V}}$ et $\Pi_{v_{b_1}}(\mathrm{S})$ est la projection de S sur l'espace de v_{b_1},
- $R_{v_{b_1}} \circ R_{v_{a_1} \to v_{b_1}}$ et $R_{v_{a_1}} \circ R_{v_{b_1} \to v_{a_1}}$ sont, respectivement, les contracteurs qui permettent de rendre consistants v_{b_1} sachant v_{a_1} et v_{a_1} sachant v_{b_1}

La preuve de ce théorème est différente de celle du théorème de propagation donnée dans [Jaulin et al., 2001b]. On prend soin d'en donner une version réadaptée aux VCSP.

Preuve :

■ Avec la symétrie du problème, il suffit juste de prouver que, par exemple, $\Pi_{v_{a_1}}(\mathrm{S}) = R_{v_{a_1}} \circ R_{v_{b_1} \to v_{a_1}}\left(\mathrm{D}_{v_{a_1}}\right)$. La démonstration de cette égalité se fera par la preuve que l'inclusion dans les deux sens est vraie :

1. On a $\Pi_{v_{a_1}}(\mathrm{S}) \subset R_{v_{a_1}} \circ R_{v_{b_1} \to v_{a_1}}\left(\mathrm{D}_{v_{a_1}}\right)$ de façon triviale puisque les solutions du CSP $\mathcal{H}_{\mathcal{V}}$, par définition, satisfont à tous les opérateurs de contraction.

2. Notons (*l, r*) l'indice des variables respectivement dans v_{a_1} et v_{b_1} concernées par la contrainte f_{a_1,b_1}. Notons également $v_1 = v_{a_1}$, $v_2 = \left(v_{b_1} \times \cdots \times v_{b_q}\right)$ et $v_3 = \left(v_1 \times v_2\right)$. On cherche à prouver, dans un premier temps, que $R_{v_{a_1}} \circ R_{v_2 \to v_{a_1}}$ est équivalent à $R_{v_{a_1}} \circ R_{v_{b_1} \to v_{a_1}}$, en d'autres termes que le résolveur $R_{v_{a_1}} \circ R_{v_{b_1} \to v_{a_1}}$ rend le vecteur v_{a_1} consistant avec tous les vecteurs dans $\mathcal{V}_b$.

Les deux variables v_1 et v_2 sont uniquement reliées par la contrainte f_{a_1,b_1} et sont donc connectées par une branche (car f_{a_1,b_1} est une connexion à une branche entre les deux vecteurs v_{a_1} et v_{b_1}). D'après la proposition dans la section 5.4.2, pour tout indice *k* dans le vecteur v_1, on peut écrire :

$$\prod_k \left(S_{v_3}\right) = \prod_k \left(S'_{v_1}\right) \tag{5.12}$$

avec S_{v_3} est la solution du CSP $\mathcal{H}_{v_3}$, et S'_{v_1} est la solution de la restriction $\mathcal{H}_{v_1}$ avec un domaine initial $\mathrm{D}'_{v_1} = \left(\mathrm{D}'_{v_1,1}, \cdots, \mathrm{D}'_{v_1,n_1}\right)$ qui vérifie :

- Pour $k = l$, $\mathrm{D}'_{v_1,k} = \prod_{f_{l,r},l} \left(\mathrm{D}'_{v_3}\right)$
- Pour les *k* restants dans v_1, $\mathrm{D}'_{v_1,k} = \mathrm{D}_k$

Où D'_{v_3} est une contraction du domaine associé à v_3 qui vérifie :

- Pour $k = r$, $\mathrm{D}'_{v_3,k} = \prod_r \left(S_{v_2}\right)$
- Pour les *k* restants dans v_3, $\mathrm{D}'_{v_3,k} = \mathrm{D}_k$

Or, par hypothèse, la variable v_{b_1} est $\mathcal{H}_{\mathcal{V}_b}$-consistante donc en particulier pour $k=r$ on a $\mathrm{D}'_{v_3,k} = \prod_r (S_{v_2}) = \mathrm{D}_r$, d'où D'_{v_3} est exactement égale au domaine initial de v_3 i.e. D'_{v_3} vérifie $\mathrm{D}'_{v_3,k} = \mathrm{D}_k$ pour tout les indices *k* dans le vecteur v_3.

De même, posons $v_4 = v_{b_1}$ et $v_5 = v_{a_1} \times v_{b_1}$. En appliquant la proposition dans la section 5.4.2 une nouvelle fois au couple de vecteurs (v_1, v_4) (qui, précise-t-on, correspond le couple de vecteurs $\left(v_{a_1}, v_{b_1}\right)$), on peut écrire pour tout indice k dans v_1 :

$$\prod_k \left(S_{v_5}\right) = \prod_k \left(S''_{v_1}\right) \tag{5.13}$$

Avec S_{v_5} est la solution du CSP $\mathcal{H}_{v_5}$, et S''_{v_1} est la solution de la restriction $\mathcal{H}_{v_1}$ avec un domaine $\mathrm{D}''_{v_1} = \left(\mathrm{D}''_{v_1,1}, \cdots, \mathrm{D}''_{v_1,n_1}\right)$ qui est une contraction du domaine initial de v_1 qui vérifie :

- Pour $k = l$, $\mathrm{D}''_{v_1,k} = \prod_{f_{l,r},l}\left(\mathrm{D}''_{v_5}\right)$
- Pour les k restants dans v_1, $\mathrm{D}''_{v_1,k} = \mathrm{D}_k$

Où D''_{v_5} est une contraction du domaine associé à v_5 qui vérifie :

- Pour $k = r$, $\mathrm{D}''_{v_5,k} = \prod_r (S_{v_4})$
- Pour les k restants dans v_5, $\mathrm{D}''_{v_5,k} = \mathrm{D}_k$

Or, par hypothèse, la variable $v_4 = v_{b_1}$ est $\mathcal{H}_{v_b}$-consistante donc en particulier v_4 est aussi $\mathcal{H}_{v_4}$-consistante (en d'autre termes, le vecteur de variable v_{b_1} est résolu). Ainsi, pour $k=r$ on a $\prod_r \left(S_{v_4}\right) = \mathrm{D}_r$, d'où D''_{v_5} est exactement égale au domaine initial de v_5 i.e. D''_{v_5} vérifie $\mathrm{D}''_{v_5,k} = \mathrm{D}_k$ pour tout les indices k dans le vecteur v_5. Il s'avère alors que $\mathrm{D}''_{v_1} = \mathrm{D}'_{v_1}$ puisque ces deux domaines sont le domaine initial de v_1 contracté grâce à la contrainte f_{a_1,b_1} et le domaine D_r. Par suite on peut en conclure que $S'_{v_1} = S''_{v_1}$ puisqu'ils sont, tous les deux, définis comme les solutions de la restriction $\mathcal{H}_{v_1}$ avec respectivement les domaines égaux $\mathrm{D}''_{v_1} = \mathrm{D}'_{v_1}$. En considérant les égalités 5.12 et 5.13 on obtient pour tout indice k dans $v_1 = v_{a_1}$:

$$\prod_k \left(S_{v_3}\right) = \prod_k \left(S'_{v_1}\right) = \prod_k \left(S''_{v_1}\right) = \prod_k \left(S_{v_5}\right) \tag{5.14}$$

L'égalité 5.14 prouve que $R_{v_{a_1}} \circ R_{v_2 \to v_{a_1}}$ est équivalent à $R_{v_{a_1}} \circ R_{v_{b_1} \to v_{a_1}}$, ce qui signifie également que le sous-résolveur $R_{v_{a_1}} \circ R_{v_{b_1} \to v_{a_1}}$ qui rend le vecteur v_{a_1} consistant avec le vecteur v_{b_1}, le rend consistant avec le vecteur $v_2 = \left(v_{b_1} \times \cdots \times v_{b_q}\right)$.

A présent, sachant que $R_{v_{a_1}} \circ R_{v_2 \to v_{a_1}}$ est équivalent à $R_{v_{a_1}} \circ R_{v_{b_1} \to v_{a_1}}$, montrons que $R_{v_{a_1}} \circ R_{v_{b_1} \to v_{a_1}}\left(D_{v_{a_1}}\right) \subset \Pi_{v_{a_1}}(\mathrm{S})$. Soit $\tilde{v}_{a_1} \in R_{v_{a_1}} \circ R_{v_{b_1} \to v_{a_1}}\left(D_{v_{a_1}}\right) = R_{v_{a_1}} \circ R_{v_2 \to v_{a_1}}\left(\mathrm{D}_{v_{a_1}}\right) \Rightarrow \exists \tilde{v}_{b_1} \in \mathrm{D}_{b_1}, \cdots, \exists \tilde{v}_{b_r} \in \mathrm{D}_{b_r} \mid \tilde{v}_{a_1} \times \tilde{v}_{b_1} \times \cdots \times \tilde{v}_{b_r} \in S_{v_3}$ (rappelons que $v_3 = \left(v_{a_1} \times v_{b_1} \times \cdots \times v_{b_r}\right)$). De plus, par hypothèse, $\tilde{v}_{a_1}$ est $\mathcal{H}_{\mathcal{V}_a}$-consistant, donc $\exists \tilde{v}_{a_2} \in \mathrm{D}_{a_2}, \cdots, \exists \tilde{v}_{a_q} \in \mathrm{D}_{a_q}$ tel que $\left(\tilde{v}_{a_1}, \cdots, \tilde{v}_{a_q}\right)$ appartienne à $S_{\mathcal{V}_a}$. D'où le vecteur $\tilde{v}_{a_1} \times \cdots \times \tilde{v}_{a_q} \times \tilde{v}_{b_1} \times \cdots \times \tilde{v}_{b_r}$ vérifie toutes les contraintes dans $\mathcal{H}_{\mathcal{V}_a}$ et $\mathcal{H}_{\mathcal{V}_b}$ plus $f_{a_1,b_1} \Rightarrow \tilde{v}_{a_1} \times \cdots \times \tilde{v}_{a_q} \times \tilde{v}_{b_1} \times \cdots \times \tilde{v}_{b_r} \in S$. D'où, en particulier, $\tilde{v}_{a_1} \in \Pi_{v_{a_1}}(\mathrm{S}) \Rightarrow R_{v_{a_1}} \circ R_{v_{b_1} \to v_{a_1}}\left(\mathrm{D}_{v_{a_1}}\right) \subset \Pi_{v_{a_1}}(\mathrm{S})$.■

6.1.2 Formulation pour des contraintes ternaires

Posons $\mathcal{V}_a = \left(v_{a_1}, \cdots, v_{a_q}\right)$ $\mathcal{V}_b = \left(v_{b_1}, \cdots, v_{b_r}\right)$, $\mathcal{V}_c = \left(v_{c_1}, \cdots, v_{c_s}\right)$ et soient $\mathcal{H}_{v_a}$, $\mathcal{H}_{v_b}$ et $\mathcal{H}_{v_c}$ trois VCSP avec des vecteurs de variables distincts i.e. $v_{a,i} \neq v_{b,j} \neq v_{c,k}$ pour tout 3-uplet (i, j, k) possible. Supposons que ces trois VCSP soient deux à deux connectés par une branche, de par une contrainte ternaire f_{a_1,b_1,c_1} reliant les trois vecteurs de variables v_{a_1}, v_{b_1} et v_{c_1}. Le nouveau VCSP $\mathcal{H}_{\mathcal{V}}$, qui regroupe ces trois variables et leurs domaines respectifs et pour lequel l'ensemble de contraintes regroupe les contraintes dans $\mathcal{H}_{v_a}$, $\mathcal{H}_{v_b}$ et $\mathcal{H}_{v_c}$ plus la contrainte f_{a_1,b_1,c_1} vérifie :

Si v_{a_1} est $\mathcal{H}_{v_a}$-consistant, v_{b_1} est $\mathcal{H}_{\mathcal{V}_b}$-consistant et v_{c_1} est $\mathcal{H}_{v_c}$-consistant alors :

$$\Pi_{v_{a_1}}(S) = R_{v_{a_1}} \circ R_{\left(v_{b_1}, v_{c_1}\right) \to v_{a_1}}\left(D_{v_{a_1}}\right)$$

La démonstration de ce théorème est donnée dans la partie D de l'annexe.

6.2 TVCSP : structure sous forme d'arbre pour les VCSP

On appelle TVCSP (Tree Vectorised Constraint Satisfaction Constraint), un VCSP ayant une structure particulière sous forme d'arbre. Dans la partie C de l'annexe, on rappelle les définitions classiques sur les graphes. On peut réadapter, par analogie, quelques définitions classiques dérivant de la théorie sur les arbres de la façon suivante :

- On peut prendre l'origine d'un arbre, communément appelé racine, en choisissant un nœud quelconque. Pour un TVCSP $\mathcal{H}_{\mathcal{V}}$: (($\tilde{x} \in \mathrm{S}_{F_J}$) et ($\tilde{x}_{J_i} \in \mathrm{S}_{v_i} \forall\, i = 1 \ldots m$), $\tilde{x} \in [x]$), une racine correspondra à un des vecteurs choisi dans $\mathcal{V} = \{v_1, \cdots v_m\}$.
- La notation $(\mathcal{H}_{\mathcal{V}}, v_i)$ signifie que l'on choisit le vecteur v_i comme étant la racine du TVCSP $\mathcal{H}_{\mathcal{V}}$.
- On considérant un arbre $\mathcal{H}_{\mathcal{V}}$ de racine v_i et $v_1 \times \cdots \times v_q$ les variables reliées à v_i, en supprimant v_i, on obtient q nouveaux arbres. Chacun de ces arbres est, par définition, un sous-arbre de l'arbre $\mathcal{H}_{\mathcal{V}}$ que l'on note $\mathcal{H}_{\mathcal{V},k}$ avec k variant de *1* à q.

6.3 Algorithme de FALL/CLIMB pour un TVCSP

On considère ici, de surcroît, la classe des TVCSP avec une connexité à une branche. A partir du théorème de propagation sur les VCSP on démontre, de manière analogue à la démarche suivie dans [Jaulin et al., 2001b], que les algorithmes FALL/CLIMB décrits ci-après permettent d'obtenir les domaines exacts des solutions. Nous réécrivons ici, de façon adaptée, sur le Tableau 5.1 et le Tableau 5.2, la forme de cet algorithme pour des CSP vectorisés. Pour plus de précision sur la démonstration de cet algorithme, le lecteur pourra se reporter à la partie C de l'annexe.

Rappelons, au préalable, que $R_{v_k \to v_i}(\mathrm{D}_{v_i})$ signifie que l'on transmet la contraction du vecteur v_k au vecteur v_i avec un domaine égal à D_{v_i}. Rappelons également que, la notation $R_{v_i}(D_{v_i})$ signifie un résolveur pour v_i appliqué au domaine D_{v_i}. Les algorithmes FALL et CLIMB pour les TVCSP avec une connexité à une branche sont les suivants :

Algorithme FALL : Input($\mathcal{H}_{\mathcal{V}}$, v_i), Output($\mathcal{H}_{\mathcal{V}}$, v_i)

DEBUT

SI $\mathcal{H}_{\mathcal{V}}$ est une feuille, ALORS retourner $\mathcal{H}_{\mathcal{V}}$ FIN

$v_{racine} = v_i$; $v_1, \cdots, v_q$ = vecteurs liés à v_{racine} par une contrainte binaire

$(v_{11}, v_{12}), \cdots, (v_{r1}, v_{r2})$ = couples de vecteurs liés à v_{racine} par une contrainte ternaire

% contraintes binaires

POUR $k = 1$ à q, FAIRE

FALL($\mathcal{H}_{\mathcal{V},k}$, v_k)

$D_{v_{racine}} = R_{v_k \to v_{racine}}\left(D_{v_{racine}}\right)$

FIN

% contraintes ternaires

POUR $k = 1$ à r, FAIRE

FALL($\mathcal{H}_{\mathcal{V},k}$, $v_{k,1}$)

FALL($\mathcal{H}_{\mathcal{V},k}$, $v_{k,2}$)

$D_{v_{racine}} = R_{(v_{k,1}, v_{k,2}) \to v_{racine}}\left(D_{v_{racine}}\right)$

FIN

$D_{v_{racine}} = R_{v_{racine}}\left(D_{v_{racine}}\right)$

FIN

Tableau 5.1 : algorithme FALL écrit pour les TVCSP avec une connexité à une branche.

Algorithme CLIMB : Input($\mathcal{H}_{\mathcal{V}}$, v_i), Output([$\mathcal{H}_{\mathcal{V}}$, v_i])

DEBUT

SI $\mathcal{H}_{\mathcal{V}}$ est une feuille, ALORS retourner $\mathcal{H}_{\mathcal{V}}$ FIN

$v_{racine} = v_i$; $v_1, \cdots, v_q$ = vecteurs liés à v_{racine} par une contrainte binaire

$(v_{11}, v_{12}), \cdots, (v_{r1}, v_{r2})$ = couples de vecteurs liés à v_{racine} par une contrainte ternaire

% contraintes binaires

POUR $k = 1$ à q, FAIRE

$D_{v_k} = R_{v_k} \circ R_{v_{racine} \to v_k}\left(D_{v_k}\right)$

CLIMB($\mathcal{H}_{\mathcal{V},k}$, v_k)

FIN

% contraintes ternaires

POUR $k = 1$ à r, FAIRE

$D_{v_{k,1}} = R_{(v_{racine}, v_{k,2}) \to v_{k,1}}\left(D_{v_{k,1}}\right)$

$D_{v_{k,2}} = R_{(v_{racine}, v_{k,1}) \to v_{k,2}}\left(D_{v_{k,2}}\right)$

CLIMB($\mathcal{H}_{\mathcal{V},k}$, $v_{k,1}$)

CLIMB($\mathcal{H}_{\mathcal{V},k}$, $v_{k,2}$)

FIN

FIN

Tableau 5.2 : algorithme CLIMB écrit pour les TVCSP avec une connexité à une branche.

Remarque

1. ■ Par rapport à l'algorithme FALL/CLIMB pour les SCSP, pour généraliser aux VCSP, on remplace les projections de contraintes par des sous-résolveurs qui ont la forme $R_{v_k} \circ R_{v_i \to v_k}$ ou $D_{v_{k,3}} = R_{(v_{k,1}, v_{k,2}) \to v_{k,3}}$. Il est à noter que ces sous-résolveurs permettent de rendre consistantes, respectivement, un duo de vecteurs de variables connectés par une branche, de par une contrainte binaire, ou un trio de vecteurs de variables connectés par une branche, de par une contrainte ternaire, en contractant uniquement les variables pertinentes.

2. Dans l'algorithme FALL, le terme " $D_{v_{racine}} = R_{v_{racine}}\left(D_{v_{racine}}\right)$ " est à l'extérieur de la boucle. Ceci traduit une idée de factorisation d'une étape qu'il n'est pas nécessaire de répéter à l'intérieur des boucles de FALL.

6.4 Exemples de l'algorithme FALL/CLIMB appliqué à des CSP vectorisés

6.4.1 Exemple académique

Dans cette partie, on propose de résoudre un exemple académique en utilisant la méthode de vectorisation présentée dans ce chapitre. Considérons un CSP $\mathcal{H}$ avec comme ensemble de variables (a, b, c, d, e, f, u, v, w, x, y) et comme système de contraintes F : $\begin{cases} x+y=a \\ x-y=b \end{cases}, \begin{cases} v+w=c \\ v-w=d \end{cases}, \begin{cases} u+z=e \\ u-\ln(z)=f \end{cases}, e=b+d, g=2f$ dont le graphe est représenté sur la Figure 5.21. Le système de contraintes décrit est choisi de telle manière que l'on puisse le vectoriser sous forme de TVCSP. De surcroît, les cycles à résoudre sont volontairement choisis comme étant ceux résolus dans le chapitre 4.

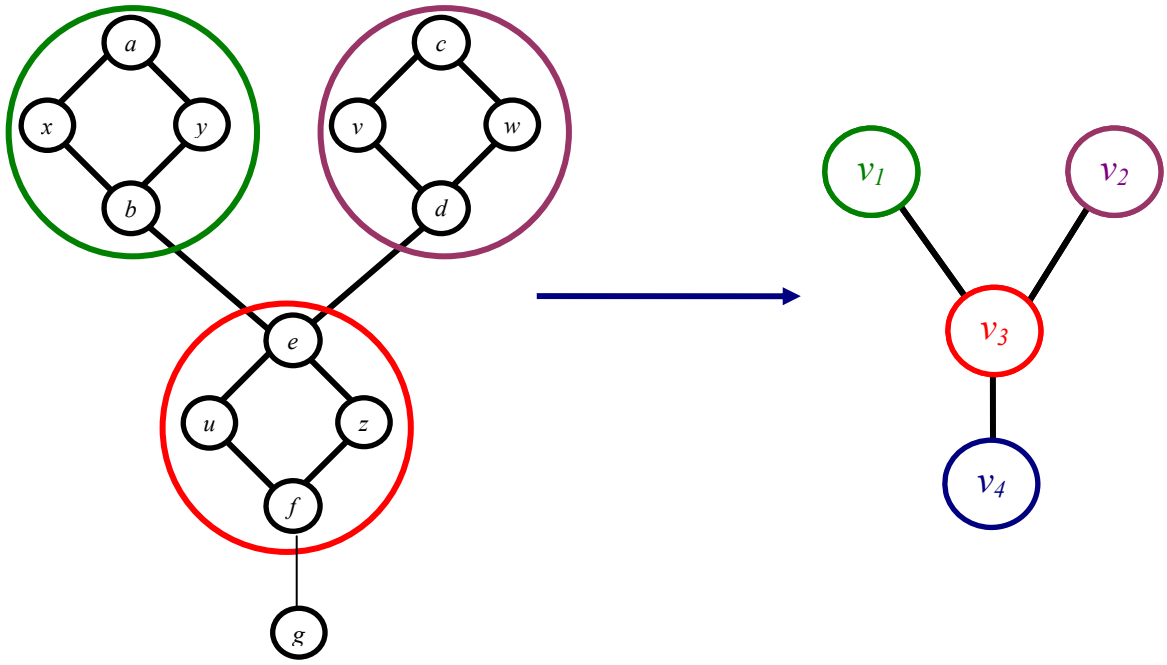


Figure 5.21 : exemple d'un TVSCP.

Posons $\mathcal{V}=(v_1,v_2,v_3,v_4)$ où $v_1=(a,b,x,y)$, $v_2=(c,d,v,w)$, $v_3=(e,f,u,z)$ et $v_4=g$. Le VCSP $\mathcal{H}_{\mathcal{V}}$ défini à partir de $\mathcal{H}$ et de $\mathcal{V}$ s'écrit $\mathcal{H}_{\mathcal{V}}$: (($x_{J_1}\in \mathrm{S}_{v_1}$, $x_{J_2}\in \mathrm{S}_{v_2}$, $x_{J_3}\in \mathrm{S}_{v_3}$, $x_{J_4}\in \mathrm{S}_{v_4}$) et ($x \in \mathrm{S}_{F_J}$), $x\in[x]$). Avec S_{v_1} l'ensemble des $(\tilde{a},\tilde{b},\tilde{x},\tilde{y})$ vérifiant le système d'équation $\begin{cases} x+y=a \\ x-y=b \end{cases}$, S_{v_2} l'ensemble des $(\tilde{c},\tilde{d},\tilde{v},\tilde{w})$ vérifiant le système d'équation $\begin{cases} v+w=c \\ v-w=d \end{cases}$, S_{v_3} l'ensemble des $(\tilde{e},\tilde{f},\tilde{u},\tilde{z})$ vérifiant le système d'équation $\begin{cases} u+z=e \\ u-\ln(z)=f \end{cases}$, S_{v_4} l'ensemble des $(\tilde{f},\tilde{g})$ vérifiant l'équation $f=2g$ et $\mathrm{S}_{F_J}=\{e=b+d\}$.

Le VCSP ainsi construit a la structure d'un arbre comme on peut le voir sur la Figure 5.21. De surcroît ce TVCSP est également à connexité à une branche. Appliquons l'algorithme FALL/CLIMB à ce TVCSP, en ayant choisi v_3 comme racine. On peut détailler les étapes de l'algorithme FALL/CLIMB de la façon suivante :

%FALL

1- $D_{v_3}=R_{(v_1,v_2)\to v_3}(D_{v_3})$; 2- $D_{v_3}=R_{v_3}(D_{v_3})$ 3- $D_{v_4}=R_{v_3\to v_4}(D_{v_4})$; 4- $D_{v_4}=R_{v_4}(D_{v_4})$

%CLIMB

5- $D_{v_3}=R_{v_3}\circ R_{v_4\to v_3}(D_{v_3})$; 6- $D_{v_1}=R_{v_1}\circ R_{(v_3,v_2)\to v_1}(D_{v_1})$; 7- $D_{v_2}=R_{v_2}\circ R_{(v_3,v_1)\to v_2}(D_{v_2})$.

Détaillons, par exemple, les différentes étapes intervenant dans le calcul de $R_{(v_1,v_2)\to v_3}(D_{v_3})$. D'abord, D_{v_3} est le domaine initial $[e]\times[f]\times[u]\times[z]$ et $R_{(v_1,v_2)\to v_3}$ est défini comme un sous-résolveur rendant la variable *e* du vecteur $v_3=(e,f,u,z)$ consistante par rapport au vecteur $v_1=(a,b,x,y)$ et au vecteur $v_2=(c,d,v,w)$. Sachant que les deux vecteurs sont reliés par la contrainte $e=b+d$ et en appliquant la proposition du paragraphe 5.4.2, on obtient $R_{v_1\to v_3}(D_{v_3})$ grâce aux 2 étapes de calcul suivantes (comme illustrées sur la Figure 5.22) :

1. calcul de la projection des solutions de la restriction $\mathcal{H}_{v_1}$ sur la variable *b* qui est celle reliée avec v_3 (voir la Figure 5.21) et de la projection des solutions de la restriction $\mathcal{H}_{v_2}$

sur la variable d qui est celle reliée avec v_3. Pour cette étape, on peut, comme exemple de sous-résolveur, utiliser ceux qui ont été présentés dans la section 4 du chapitre 4.

2. projection de la contrainte $\{e = b + d\}$. A ce stade, si par exemple, les domaines sont toujours des intervalles, il s'agit de faire $[e] = [e] \cap [b] + [d]$.

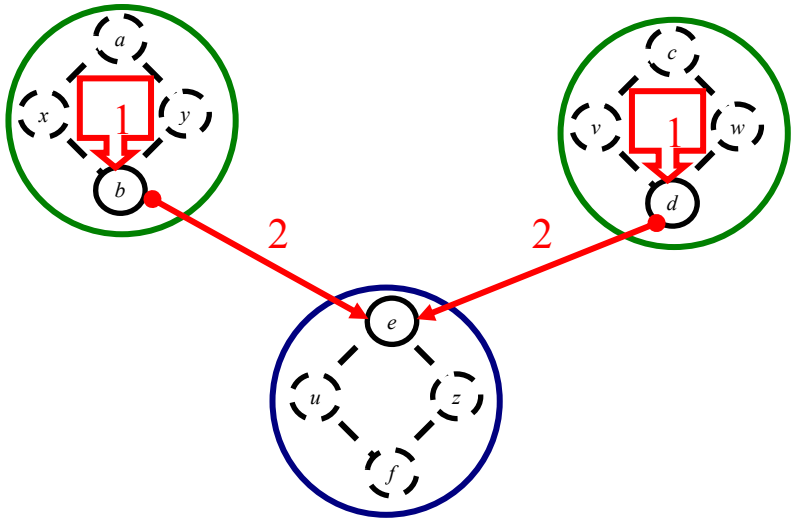


Figure 5.22 : illustration des 2 étapes de calculs pour l'obtention de $R_{v_1 \to v_3}(D_{v_3})$

6.4.2 Application à un problème de fusion de données statique

On propose dans cette partie d'appliquer l'algorithme généralisé de FALL/CLIMB pour résoudre le problème d'estimation garantie de la cinématique d'un véhicule. Pour cela, on dispose des mesures des 4 capteurs odométriques des roues ainsi que d'une mesure de l'angle au volant.

6.4.2.1 Modèle

Pour tirer bénéfice de la redondance des capteurs, on utilise un modèle d'Ackerman établi sous hypothèse de roulement sans glissement (voir sur la Figure 5.23).

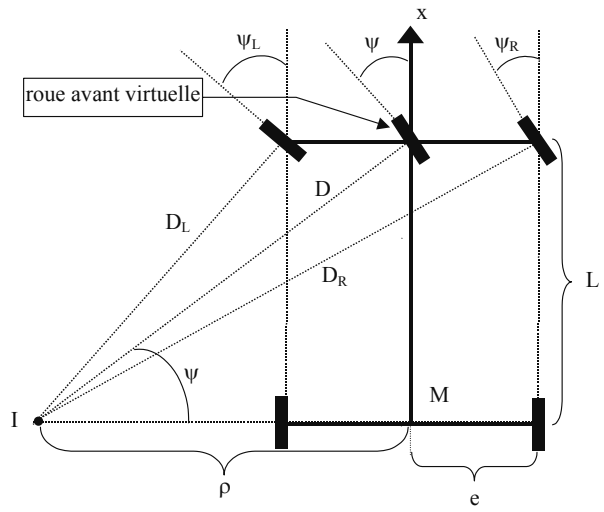


Figure 5.23 : modèle d'Ackerman d'un véhicule.

On note δ_s et δ_θ respectivement la distance parcourue par le point M et la variation de cap entre deux échantillonnages. De simples considérations géométriques permettent d'établir le modèle suivant [Bonnifait, 2003] :

$$\begin{cases} \tan(\psi) & = & L.\dfrac{\delta_\theta}{\delta_s} \\ \delta_{RL} & = & \delta_s - e.\delta_\theta \\ \delta_{RR} & = & \delta_s + e.\delta_\theta \\ \delta_{FL}.cos(\psi_L) & = & \delta_s - e.\delta_\theta \\ \delta_{FR}.cos(\psi_R) & = & \delta_s + e.\delta_\theta \end{cases} \qquad (5.15)$$

Où δ_{RL}, δ_{RR}, δ_{FL}, δ_{FR}, sont respectivement les distances parcourues par les roues arrières droites et gauches, et des roues avant droites et gauches. ψ_L et ψ_R sont les orientations des roues gauches et droites, et ψ l'angle d'une roue virtuelle centrale (cf. Figure 5.23).

Le système redondant (5.15) entraîne la présence de cycles dans le graphe des contraintes. On propose de résoudre le problème de façon garantie en utilisant l'algorithme généralisé de FALL/CLIMB introduit dans ce chapitre.

6.4.2.2 Formulation sous forme de CSP

A chaque instant, on résout le CSP $\mathcal{H}$ avec $x = (\delta s, \delta_\theta, \delta_{RR}, \delta_{RL}, \delta_{FR}, \delta_{FL}, \psi, \psi_R, \psi_L, e, L)$ et F le système d'équations (5.15). Pour résoudre ce CSP, la première étape consiste à introduire

des variables intermédiaires afin d'obtenir des contraintes uniquement ternaires et binaires. On introduit 4 nouvelles variables intermédiaires a_1, a_2, a_3 et a_4 :

$$\begin{cases} a_1 = L.\delta_\theta \\ a_2 = \delta_{FR}.cos(\psi_R) \\ a_3 = \delta_{FL}\,cos(\psi_L) \\ a_4 = e.\delta_\theta \end{cases} \quad (5.16)$$

Le système d'équations (5.15) devient alors :

$$\begin{cases} \tan(\psi) = \dfrac{a_1}{\delta_s} \\ \delta_{RL} = \delta_S - a_4 \\ \delta_{RR} = \delta_S + a_4 \\ a_2 = \delta_S - a_4 \\ a_3 = \delta_S + a_4 \end{cases} \quad (5.17)$$

En combinant les systèmes (5.16) et (5.17), on obtient un nouveau système de contraintes binaires et ternaires pour les CSP $\mathcal{H}$. La Figure 5.24 représente la représentation sous forme de graphe de $\mathcal{H}$.

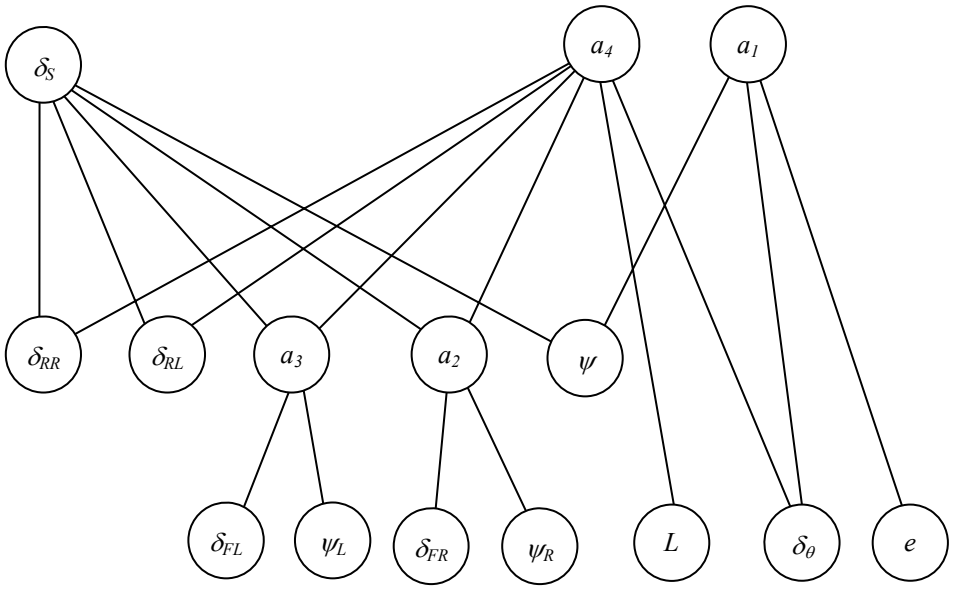


Figure 5.24 : représentation sous forme de graphe du CSP formulé pour le problème d'estimation de la cinématique du véhicule.

Ce CSP présente un cycle qui peut être regroupé dans un vecteur de variables (δs, δ_θ, δ_{RR}, δ_{RL}, ψ, e, L, a_1, a_2, a_3, a_4) (on peut voir notamment qu'on peut créer une multitude de chemins

différents ayant pour extrémités δs et δ_θ). Notons alors $v_1 = (\delta s, \delta_\theta, \delta_{RR}, \delta_{RL}, \psi, e, L, a_1, a_2, a_3, a_4)$, $v_2 = \delta_{FL}$, $v_3 = \psi_L$, $v_4 = \delta_{FR}$, $v_5 = \psi_R$

et $\mathcal{V} = \{v_1, \ldots, v_5\}$. Le VCSP $\mathcal{H}_{\mathcal{V}}$ constitué à partir de $\mathcal{H}$ et du vecteur $\mathcal{V}$ a une représentation graphique sous forme d'un arbre et est à connexité à une branche.

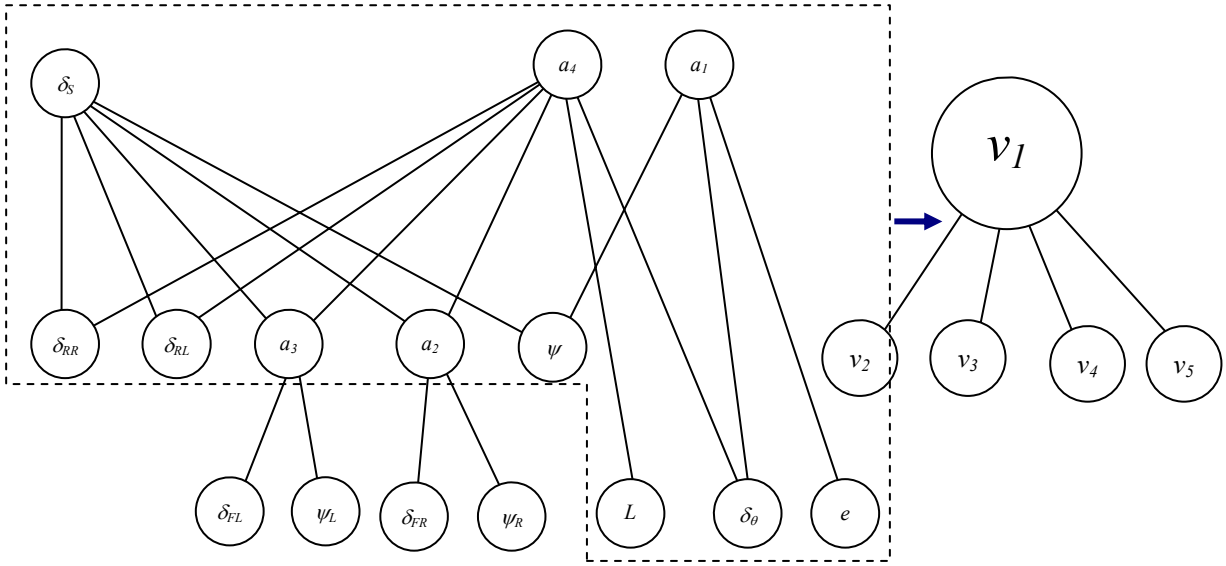


Figure 5.25 : représentation sous forme d'un TVCSP à connexion à une branche CSP.

%FALL

1- $D_{v_1} = R_{(v_2,v_3)\to v_1}\left(D_{v_1}\right)$; 2- $D_{v_1} = R_{(v_4,v_5)\to v_1}\left(D_{v_1}\right)$; 3- $D_{v_1} = R_{v_1}\left(D_{v_1}\right)$;

%CLIMB

4- $D_{v_2} = R_{v_2} \circ R_{(v_1,v_3)\to v_2}\left(D_{v_2}\right)$; 5- $D_{v_3} = R_{v_3} \circ R_{(v_1,v_2)\to v_3}\left(D_{v_3}\right)$;

3- $D_{v_4} = R_{v_4} \circ R_{(v_1,v_5)\to v_4}\left(D_{v_4}\right)$; 4- $D_{v_5} = R_{v_5} \circ R_{(v_1,v_4)\to v_5}\left(D_{v_5}\right)$.

Cet algorithme a été résolu et testé sur des données réelles. Pour plus de détails sur le résolveur du cycle présent dans cet exemple, le lecteur est renvoyé à l'article [Gning et al., 2004]. Dans le prochain paragraphe, on donne des résultats de la méthode sur des données réelles en comparaison avec un algorithme de Waltz.

6.4.2.3 Résultats expérimentaux

L'algorithme décrit dans la section précédente a été appliqué sur des données expérimentales réelles en comparaison avec l'algorithme de Waltz. Grâce au véhicule STRADA de l'Heudiasyc, les données odométriques ainsi que les mesures de l'angle au volant ont été enregistrées à 100 Hz. Ces données ont été ré-échantillonnées à 5 Hz de façon à être synchrones de celles d'un récepteur GPS (selon le même protocole expérimental présenté dans la section 4 du chapitre 3).

Des estimations garanties du déplacement élémentaire $[\delta s]$ entre deux instants d'échantillonnage sont données sur la Figure 5.26 (la vitesse du véhicule est d'environ 70 km/h). La méthode basée sur les domaines de consistance (courbe avec des '+') donne une estimation plus précise qu'une méthode basée sur l'algorithme de Waltz (courbe avec des '*'). En effet, l'imprécision est réduite d'au moins 25 %, ce qui est très significatif.

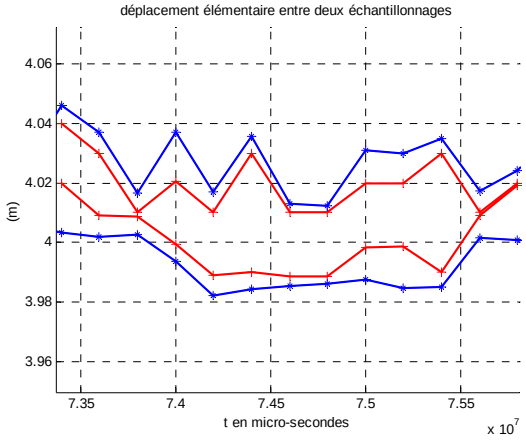


Figure 5.26 : estimations garanties de δs (vectorisation des cycles '+' et Waltz '*')

Ce résultat est conforme à la théorie puisque la méthode basée sur les domaines de consistance atteint la consistance globale et fournit donc l'estimation à erreur bornée la plus petite vérifiant toutes les contraintes. A l'inverse, une méthode basée sur l'algorithme de Waltz induit un pessimisme supplémentaire.

En terme de temps de calcul et pour fixer l'ordre de grandeur, une itération de la méthode prend 280 ms sous Matlab avec un PC équipé d'un processeur Intel Pentium IV à 1,7 Ghz. Ce résultat est extrêmement encourageant pour une implémentation temps-réel à une fréquence de 5 Hz. Pour l'algorithme de Waltz, avec un seuil choisi de 10^{-3}, le temps de calcul est inférieur avec une moyenne de 35*ms* environ et un nombre de boucle variant entre 3 et 4.

Mais l'avantage de la méthode de résolution des cycles, en plus d'une meilleure contraction, est de pouvoir borner a priori le temps de calcul sans altérer les performances des pavés obtenus. De surcroît, la taille du cycle résolu explique en partie la durée du temps de calcul.

Conclusion

Dans ce chapitre, de nouvelles méthodes destinées à améliorer les techniques de satisfaction de contraintes ont été introduites. Ces techniques sont une suite cohérente du chapitre précédent où nous avons cherché de nouvelles techniques pour permettre de résoudre les difficultés en terme de temps de calcul ainsi que les limites de performances rencontrées dès l'apparition de cycles. Nous avons ainsi cherché à définir une structure organisée de façon à isoler les cycles et à utiliser les bonnes propriétés de l'algorithme FALL/CLIMB. Nous nous sommes ainsi attachés, dans un premier temps, à illustrer l'intérêt particulier de savoir organiser l'ordre de contraction pour la résolution d'un CSP. Nous avons pu conclure que cela permet un gain en temps de calcul très appréciable, mais aussi, une connaissance a priori sur le temps de calcul.

Par la suite, on s'est intéressé aux techniques de vectorisation déjà existantes en montrant cependant leurs limites en présence de cycles, notamment pour la résolution et la détection de cycle.

Puis, il nous est alors apparu nécessaire de définir un cadre théorique nous permettant de caractériser les cycles en nous basant sur les difficultés et limites auxquelles on est confronté lorsque l'on effectue un algorithme de Waltz. Pour cela, on s'est basé sur la définition de chaînes de contraintes qui ont la particularité de prendre en compte les contraintes ternaires.

Ensuite, par des illustrations et des exemples académiques, nous nous sommes évertués à suivre un cheminement progressif nous permettant de définir les classes nécessaires pour aboutir à une structure de CSP vectorisés. Nous avons ainsi naturellement défini la notion de restriction d'un CSP à un vecteur de variables et celle de restriction d'un CSP à un système de contraintes. Ceci nous a permis de définir une nouvelle structure, que nous avons appelé VCSP, sous forme d'un CSP avec des variables vectorielles reliées par des contraintes internes et des contraintes entre vecteurs de variables.

Avec en ligne de mire la perspective de généraliser l'algorithme FALL/CLIMB, la suite logique de cette nouvelle structure a été de savoir sous quelles conditions on serait en mesure de construire des contracteurs nous permettant de rendre les vecteurs de variables consistants l'un par rapport à l'autre. Nous avons ainsi abouti à la condition forte de connexité à une branche pour laquelle nous avons pu proposer des contracteurs.

Enfin, dans un dernier travail, le théorème de propagation à la base de l'algorithme FALL/CLIMB a été généralisé à la nouvelle structuration vectorisée. Ceci a alors permis d'aboutir au schéma d'organisation de contraintes voulu.

La méthode de vectorisation donnée est donc très avantageuse puisqu'elle peut s'appliquer à un CSP quelconque à condition de pouvoir résoudre tous les cycles une fois isolés. Les perspectives de la méthode présentée seraient d'utiliser des propriétés connues sur la théorie des graphes pour être capable, à partir d'un système de contraintes quelconque, de générer automatiquement les cycles ainsi qu'un TVSCP avec une connexité à une branche. En effet, ayant défini la structure de TVCSP à connexité à une branche et ayant su généraliser l'algorithme FALL/CLIMB pour cette structure, une réflexion légitime serait de savoir générer à partir d'un CSP quelconque un TVCSP avec une connexité à une branche.

Une interrogation qui mériterait également des réponses serait de connaître l'apport que pourrait avoir cette vectorisation sur un algorithme de Waltz. En effet, un algorithme de Waltz appliqué sur un CSP vectorisé (on sous-entend ici une gestion locale des boucles au niveau des cycles) n'est-il pas plus efficace en terme de rapidité qu'un algorithme de Waltz classiquement exécuté ? L'intérêt de répondre à cette interrogation serait d'aider à accélérer des outils de résolution de CSP existants (solveurs).

Pour terminer, le couplage des outils théoriques introduits dans ce chapitre et le chapitre précédent – domaines de consistance et vectorisation des CSP – ouvre une nouvelle perspective, des plus ambitieuses, de savoir résoudre certaines classes de CSP présentant plusieurs cycles de façon globale et en temps connu.

CONCLUSION GENERALE

Au cours de cette thèse, nous nous sommes intéressés à appliquer les techniques de propagation de contraintes sur les intervalles à des problèmes de fusion de données dans le but de localiser de façon garantie un véhicule à l'aide de capteurs proprioceptifs et de récepteurs GPS.

Cette thèse est globalement constituée de trois grandes parties. Les deux premiers chapitres constituent le cadre général ainsi que l'état de l'art des techniques existantes. Le chapitre 3 constitue l'application de la méthode de propagation de contraintes à un problème réel de localisation dynamique d'un véhicule avec la présentation des résultats expérimentaux obtenus. Enfin, une dernière partie décrit de nouvelles méthodes de satisfaction de contraintes que nous préconisons pour améliorer les performances des méthodes existantes notamment en présence de cycles.

Dans la première partie du manuscrit, nous avons mis en avant l'intérêt croissant des systèmes de localisation pour des domaines clefs tels que la sécurité, les systèmes d'assistance à la conduite et la gestion des infrastructures. Ensuite, nous avons décliné les différents concepts de localisation et nous avons présenté une étude sur les capteurs présents dans le monde de l'automobile pour justifier le choix porté sur l'utilisation de l'hybridation GPS/gyromètre/odométrie. Puis, nous avons présenté quelques concepts sur la fusion de données ainsi que le formalisme de modélisation d'un système qui l'accompagne le plus souvent. Nous avons ensuite réalisé un état de l'art de diverses techniques utilisées pour résoudre les problèmes de fusion de données. Nous avons ainsi placé les méthodes de satisfaction de contraintes sur les intervalles dans la classe des méthodes ensemblistes beaucoup moins classiques que les approches ponctuelles et en particuliers bayésiennes.

Dans la deuxième partie, apparaît la première contribution de cette thèse à savoir l'application des méthodes de propagation de contraintes seules à un problème de localisation dynamique d'un véhicule sur des données réelles.

Les résultats les plus encourageants obtenus sont la garantie effectivement vérifiée en cas de fonctionnement normal des capteurs et d'hypothèses sur le modèle conformes à la réalité.

En effet, grâce à une référence centimétrique, nous avons pu vérifier qu'effectivement, les pavés estimés contenaient bien les vraies solutions de positionnement. De surcroît, comparés au filtrage de Kalman étendu (EKF), cette méthode ensembliste garantit toujours la localisation avec une précision qui est certes moins bonne que celle par l'EKF mais qui est comparable à l'EKF en terme d'ordre de grandeur. En outre, en se confrontant à ces données réelles, nous avons soulevé des interrogations nouvelles auxquelles nous avons cherché à apporter des réponses.

Tout d'abord, est apparu le choix de l'horizon à fixer dans un processus dynamique avec la possibilité théorique, mais irréaliste, de reculer jusqu'à l'origine du temps (instant t_1 où les premières mesures sont effectuées). Les études nous ont permis de conclure qu'à partir d'un certain horizon, les contractions restent stagnantes. Nous en avons conclu qu'il suffisait de faire des études préalables, adaptées au problème à résoudre, pour estimer l'horizon de temps à choisir.

Ensuite, l'implémentation en post-traitement de l'algorithme de propagation de contraintes nous a fait conclure, qu'en moyenne, les temps de calculs étaient très probants. Le temps réel est ainsi possible à condition de contourner quelques cas limites qui pourraient survenir notamment du fait de la boucle intervenant dans l'algorithme de Waltz. La solution préconisée est de limiter cette boucle dans le temps avec, cependant, une probable perte sur les performances atteignables par l'algorithme de Waltz.

Un autre point étudié est la stratégie à adopter en cas de survenue d'intervalles vides. Cette situation traduit soit des mesures incohérentes, soit des hypothèses fausses. Nous avons préconisé trois sortes d'adaptation des algorithmes, à savoir, réinitialiser l'algorithme, augmenter la taille des intervalles des mesures ou adopter une stratégie de détection de capteurs défaillants ou d'hypothèses violées. Les deux premières adaptations consistent à accorder plus de confiance à la dynamique de l'estimateur en cours ou, a contrario, à faire confiance au fonctionnement des capteurs extéroceptifs. Dans un cas où dans l'autre, l'objectif premier de garantie de l'algorithme peut être remis en cause. C'est pourquoi nous pensons qu'il vaut mieux considérer la troisième stratégie qui vise à détecter les pannes en cas d'intervalles vides. Malheureusement, le processus dynamique et la consistance locale ne nous mettent pas dans les conditions de pouvoir raisonnablement détecter l'origine des pannes.

Nous nous sommes alors tournés vers une contribution visant à chercher de nouveaux outils permettant d'obtenir des consistances globales et en temps connu. Ceci permet à la fois de résoudre les problèmes de temps réel, d'accroître la précision des pavés retournés, mais aussi, de pouvoir envisager de faire un outil de diagnostic des inconsistances dans un processus dynamique et donc de pouvoir conserver l'aspect primordial de garantie de la méthode.

C'est ainsi que nous avons introduit la notion de domaine de consistance. Grâce à cette vue transverse de la notion de consistance, nous avons abouti à de nombreuses propriétés ainsi qu'à des résultats intéressants qui permettent d'étudier dans les détails les cycles afin de les résoudre.

Les premières impressions sur les exemples traités tendent à nous faire dire qu'avec les domaines de consistance, nous tenons un nouvel outil avec lequel des cycles peuvent être résolus au prix d'un effort préalable qui ne semble pas faisable dans un processus automatisable. Les difficultés pour résoudre les cycles peuvent être certes très importantes et demander un effort de réflexion, mais, cependant, le résultat final, à savoir obtenir des domaines globalement consistants en un temps connu, justifie amplement des études plus abouties sur ces notions.

Toujours dans l'objectif d'améliorer les temps de calcul, nous avons proposé une nouvelle contribution dans le sens où il s'agit de savoir vectoriser même en présence de cycles. L'existence d'un ordre des contraintes en cas d'absence de cycles était déjà acquise mais le principe était alors inutilisable en présence de cycles. Il a été défini une nouvelle structure permettant de regrouper des vecteurs de variables reliés par des cycles. Puis, un processus de contraction des vecteurs de variables entre elles a été défini à la condition d'une connexion à une branche entre elles. Pour terminer l'algorithme FALL/CLIMB a été généralisé grâce à une réécriture adaptée du théorème de propagation.

Ainsi, la réorganisation des contraintes couplée à la capacité de résoudre les cycles de façon optimale et en temps connu permettent d'envisager de connaître le temps de calcul et de l'améliorer sensiblement par rapport à un algorithme de Waltz, mais également, de résoudre de façon globale les problèmes de satisfactions de contraintes même en présence de cycles.

PERSPECTIVES

Ce travail aboutit à de nombreuses perspectives. Le premier point est que nous n'avons pas eu le temps d'appliquer entièrement les méthodes de satisfaction de contraintes, introduites dans cette thèse, au problème de localisation présenté. Ce travail aurait été un point intéressant, notamment, en comparaison avec les résultats obtenus avec l'algorithme de Waltz appliqué au problème de localisation.

Précisons cependant, que ce manuscrit fait délibérément la part belle à une démarche d'explications détaillées et illustrées sur des exemples simples des notions introduites. Ce choix de se focaliser sur l'explication de la méthode plutôt que sur l'application se justifie par les premiers retours lors, notamment, de soumission d'articles.

Dans la rubrique des perspectives théoriques, il est envisageable d'améliorer le calcul des domaines de consistance et, également, de trouver des techniques permettant une représentation sous forme de TVCSP, avec une connexité à une branche, pour un CSP quelconque.

En ce qui concerne les domaines de consistance, on peut envisager de développer des techniques permettant d'approximer les domaines de consistance avec une connaissance sur les erreurs commises. L'avantage serait alors de simplifier l'obtention des algorithmes basés sur les domaines de consistance et de pouvoir fournir des domaines solutions en connaissant le pessimisme engendré par les approximations faites.

En ce qui concerne les TVCSP avec une connexion à une branche, une analogie avec les composantes connexes des graphes permet d'envisager l'existence d'algorithmes permettant d'obtenir automatiquement, à partir d'un CSP quelconque, cette organisation de variables sous forme de TVCSP. Cela faciliterait le travail d'isolation des cycles et, dans un autre volet, on pourrait tenter de répondre que cela accélérerait un algorithme de Waltz, si on exécutait des boucles qu'au niveau des vecteurs de variables présentant un cycle ! Ce dernier point pourrait être intéressant pour les solveurs.

Références bibliographiques

Articles

[*Ackerman, 1996*] J. Ackerman. "*Yaw disturbance by robust decoupling of car steering*" *IFAC World Congress, San Francisco*, july 1996

[*Alamo et al., 2005*]. Alamo, T., Bravo, J.M. & Camacho E.F. "*Guaranteed state estimation by zonotopes*" *Automatica, Volume 41, Issue 6, Pages 1035-1043, June 2005.*

[*Benhamou et al., 1994*] Benhamou, F., MacAllester, D. And van Hentenryck, P.. "*CLP(intervals) revisited*", Proceedings of the International Logic Programming Symposium, *Ithaca, NY ,pp.124-138*, 1994.

[*Benhamou et al., 1999*] F. Benhamou, F. Goualard, L. Granvilliers, and J.F. Puget. "Revising hull and box consistency", *Proceedings of the Intervational Conference on Logic Programming*, Las Cruces, NM, pp.230-244. 1999.

[*Beri., 1983*] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. "*On the Desirability of Acyclic Database Schemes*". *J. ACM*, 30(3):479-513, 1983.

[*Bertsekas et al., 1968*]. Bertsekas, D.P., Rhodes, I.B.. "*Recursive state estimation for a set-membership description of uncertainty*". IEEE Trans. Automat. Control 16 117–128, 1968.

[*Bonnifait et al., 2003*] Ph. Bonnifait, P. Bouron, D. Meizel and P. Crubillé. "*Dynamic Localization of Car-like vehicles using Data Fusion Of Redundant ABS sensors*". The Journal Of Navigation, Vol. 56, pp. 1-13. 2003.

[*Boreinstein et al, 96*] J. Boreinstein, L. Feng – "*Gyrodometry : a new method for combining data from gyros and odometry in mobile robots*" – Proc. of the IEEE International Conference on Robotics and Automation, Minneapolis, Minnesota - April 1996.

[*Chernousko, 1981*]. F.L. Chernousko, "*What is ellipsoidal modelling and how to use it for control and state estimation*?". In: I. Elishakoff (Ed.), Whys and Hows In Uncertainty Modelling, Springer,Wien, , pp. 127–188, 1999.

[*Chernousko, 1981*]. Chernousko, F. L.. "*Optimal guaranteed estimates of indeterminacies with the aid of ellipsoids*". I–III. Engineering Cybernetics, 18, 3–5, 1981.

[*Chernousko, 1994*]. Chernousko, F. L.. "*State estimation for dynamic systems*". Boca Raton: CRC Press, 1994.

[*Cleary, 1987*] Cleary, J. C. Logical arithmetic, *Future Computing Systems* 2, 125-149, 1987.

[*Clowes, 1971*] Clowes, M.B.. On seeing things, Artificial intelligence **2**: 179-185, 1971

[*Combastel, 2003*]. Combastel, C.. "*A state bounding observer based on zonotopes*". Proceedings of European control conference. Cambridge, UK, 2003.

[*Davis, 1987*] Davis, E.. "*Constraint propagation with interval labels*", Artificial Intelligence, 32 (3), 162-175, 1987.

[*Dechter, 2003*] Dechter, R., "*Constraint Processing*" *Published by Morgan Kaufmann* (2003)

[*Defour et al., 2004*] Defour, D., Hanrot, G., Lefèvre, V., Muller, J.-M., Revol, N., Zimmermann, P.. "*Proposal for a Standardization of Mathematical Function Implementation in Floating-Point Arithmetic*". Numerical Algorithms, 37(1):367-375, 2004.

[*Durieu et al., 2001*]. Durieu, C., Walter, E., & Polyak, B. T.. "*Multi-input multi-output ellipsoidal state bounding*". Journal of Optimization Theory and Applications, 111, 273–303, 2001.

[*Fargeon et al, 93*] C. Fargeon et J.-Ph. Quin. "Robotique mobile". Ouvrage collectif. Teknea. Toulouse. 1993.

[*Fogel et al., 1982*]. Fogel, E., & Huang, Y. F.. "*On the value of information in system identification-bounded noise case*". Automatica, 18, 229–238, 1982.

[*Freuder, 1978*] Freuder, E.C.. Synthesizing constraint expression, Communications of the ACM 21(11): 958-966, (1978).

[*Gordon et al., 1993*]. Gordon, N. J., Salmond, D. J., and Smith, A. F. M., "*Novel approach to nonlinear/non-Gaussian Bayesian state estimation*". IEEE Proceedings, vol. 2, no. 140, pp. 107-113, 1993.

[*Hanson, 1968*] Hanson, R. J.. Interval aritmetic as a closed arithmetic system on a computer, Technical Memorandum197, Jet Propulsion Laboratory, Section 314, California Institute of Technology, Pasadena, CA, 1968.

[*Hyvönen, 1992*] Hyvönen, E.. "*Constraint reasoning based on interval arithmetic; the tolerence propagation approach*", Artificial Intelligence 58(1-3), 71-112, 1992.

[*Gning, 2004*] Gning A. Bonnifait P. "*Guaranteed Dynamic Localization using Constraints Propagation Techniques on Real Intervals*". IEEE International Conference on Industrial Technology (ICIT 04), Hammamet, Tunisie, Décembre 2004.

[*Ito et al., 2000*]. Ito, K. & Xiong, K.. "*Gaussian filters for nonlinear filtering problems*". In *IEEE Transaction on Automatic Control*, volume 45 of 5, May 2000.

[*Jaulin et al, 2001a*] L. Jaulin., M. Kieffer, O. Didrit and E. Walter. "*Applied Interval Analysis*", *Springer-Verlag*, 2001.

[*Jaulin et al, 2001b*] L. Jaulin, M. Kieffer, I. Braems and E. Walter. "*Guaranteed non linear estimation using constraint propagation on sets*". International Journal of Control, volume 74, no 18, 1772-1782, 2001.

[*Jaulin, 2002*], Jaulin, L.. "*Consistency techniques for the localization of a satellite*". 1st International Workshop on Global Constrained Optimization and Constraint Satisfaction, Sophia Antipolis (Nice), France. October 2-4, 2002.

[*Jaulin, 2002*] L. Jaulin. "Nonlinear bounded-error state estimation of continuous-time systems". *Automatica*, 38, 1079-1082. 2002.

[*Julier et al., 1996*]. Julier, S.J. & Uhlmann, J.K.. "A general method for approximating nonlinear transformation of probability distribution". Technical report, University of Oxford, November 1996.

[*Kahan, 1968*] Kahan, W.."*A more complete interval arithmetic*", Lecture notes for a summer course, University of Toronto Canada,1968.

[*Kalman 1960*]. Kalman, R. E.. "*A New Approach to Linear Filtering and Prediction Problems*", Transactions of the ASME–Journal of Basic Engineering, 82 (Series D): 35-45. 1960.

[*Kalman et al., 1961*]. Kalman R. E. and Bucy R. "*A New Approach to Linear Filtering and Prediction Theory*". Trans. ASME, Journal of Basic Engineering, 83:95_108, 1961.

[*Kühn, 1998*]. Kühn, W.. "*Rigorous computed orbits of dynamical systems without the wrapping effect*". *Computing*, *61*(1), 47–67, 1998.

[*Kurzhanskii, 1977*]. Kurzhanskii, A. B.. "*Control and observation under uncertainty*" (in Russian). Moscow: Nauka, 1977.

[*Lesecq et al. 2002*]. , Lesecq, S. ; Barraud, A. "Une approche factorisée plus simple et numériquement stable pour l'estimation ensembliste ellipsoïdale" APII-JESA, Vol. 36, n°4, 2002, pp. 505-518.

[*Lohner, 1987*]. Lohner, R. "*Enclosing the solutions of ordinary initial and boundary value problems*", in Computer Arithmetic: Scientific Computation and Programming Languages, BG Teubner, Stuttgart, Germany, pp. 225-286, 1987.

[***Mackworth, 1977***] Mackworth, A.K.. Consistency in networks of relations, *Artificial Intelligence* **8**(1):99118, 1977.

[*Malan et al, 1992*] Malan, S.A., Milanese, M., Taragna, M. And Garloff, J.. B^3 "*algorithm for robust performance analysis in presence of mixed parametric and dynamic perturbations*", Proceedings of the 31st IEEE Conference on Decision and control, Tucson, AZ, pp.128-133, 1992

[*Milanese et al., 1996*]. Milanese, M., Norton, J. P., Piet-Lahanier, H., & Walter, E. (Eds.). "*Bounding approaches to system identification*". New York : Plenum, 1996.

[*Moore, 1966*] Moore, R. E. "*Interval Analysis*", Prentice-Hall, Englewood Cliffs, NJ, 1966.

[*Moore, 1979*] Moore, R. E.. "*Methods and applications of a rational function of n variables over a bounded region*". Computing **16**: 1-15, 1979.

[*Neumaier, 1990*] Neumaier, A.. "*Interval Methods for Systems of Equations*", Cambridge University Press, 1990.

[*Norgaard et al., 2000*]. Norgaard, M., Poulsen, N.K. & Ravn, O. . "*Advances in derivative-free state estimation for non-linear systems*". (IMM-REP-1998-15), April 2000.

[*Puig et al., 1981*]. Puig, V., Cugueró, P., Quevedo, J., 2001. "*Worst-case estimation and simulation of uncertain discrete-time systems using zonotopes*". Proceedings of European control conference, Portugal.

[*Ragot et al., 1990*]. Ragot, J., Darouach, M., Maquin, D. & Bloch, G.. "*Validation de données et diagnostic*". Traité des Nouvelles Technologies, Hermès, 1990.

[*Ratschek et al, 1984*] Ratschek, H. and Rokne, J.. "*Computer Methods for the range of functions"*, Ellis Horwood, Chichester, UK, (1984).

[*Sam-Haroud, 1995*] Sam-haroud, D.. "*Constraint consistency techniques for continuous domains"*, PhD dissertation 1423, Swiss federal Institute of technology in lausanne, Switzerland, 1995.

[*Sagnac, 1913*] G. Sagnac, "L'éther lumineux démontré par l'effet du vent relatif d'éther dans un interféromètre en rotation uniforme", C. R. Acad. Sci., 157, p 708, (1913)

[*Scheweppe, 1968*]. Schweppe, F. C.. "*Recursive state estimation: Unknown but bounded errors and system inputs*". IEEE Transactions on Automatic Control, 13, 22–28, 1968.

[*Waltz, 1975*] Waltz, D.. "*Generating semantic descriptions from drawings of scenes with shadows*", in P.P. Winston(ed.); The psychology of computer Vision, McGraw-Hill ; New York, NY pp. 19-91, 1975.

[*Witsenhausen, 1968*]. Witsenhausen, H. S.. "*Sets of possible states of linear systems given perturbed observations*". IEEE Transactions on Automatic Control, 13(5), 556–558, 1968.

Rapports de theses et de DEA

[***Gning, 2002***] Gning, E. A.. "*Fusion de donnée multi sensorielle par propagation de contraintes sur les intervalles*". Rapport DEA, Université de Technologie de Compiègne, 2002.

[Jaulin, 1994] Jaulin, L.. "*Solution Globale et Garantie de Problèmes Ensemblistes ; Application a L'estimation Non Linéaire et a la Commande Robuste*". PhD thesis, Université Paris XI Orsay, 1994.

Sites Web

http://www.arcos2004.com/

[*AQCS*]http://www.mpi-sb.mpg.de/~ratschan/AQCS/AQCS.html

[*ALIAS*] http://www-sop.inria.fr/coprin/logiciels/ALIAS/ALIAS.html

[*Realpaver*] http : //www.sciences.univ-nantes.fr/info/perso/permanents/granvil/realpaver.

Annexes

A. Principe de quelques contracteurs

Dans cette section de l'annexe, quelques contracteurs connus sont présentés. Ces contracteurs sont généralement des adaptations aux intervalles d'algorithmes existants pour des systèmes d'équations dans IR tels que les techniques d'élimination de Gauss ou les algorithmes de Newton.

A.1 Méthode d'élimination de Gauss généralisée aux intervalles

Considérons un CSP $\mathcal{H}$ constitué de l'ensemble des variables : $x=(a_{11},\cdots,a_{nn},p_1,\cdots,p_n,b_1,\cdots b_n)$ qui satisfont au système linéaire :

$$\begin{pmatrix} \mathrm{A}\in[\mathrm{A}], b\in[b], p\in[p] \\ Ap-b=0 \end{pmatrix} \tag{A.1}$$

Avec $\mathrm{A}=\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \vdots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}, b=\begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}, \ p=\begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix}$

Cette écriture du CSP $\mathcal{H}$ représente une classe des problèmes présentant des systèmes carrés linéaires avec des équations intervalles. La généralisation de la méthode de Gauss permet alors de contracter le pavé [p].

Afin de généraliser la méthode d'élimination de Gauss classique, le principe de triangulation de la matrice A est adapté aux intervalles. La différence sera que le test pour un pivot intervalle sera de vérifier que l'intervalle ne contient pas zéro. L'algorithme sera le suivant :

POUR $i = 1$ à $n-1$

  SI $0 \in [a_{ii}]$

    $[p_i] = \mathrm{IR}^n$

  SINON

    POUR $j = i+1$ à n

      $[\alpha_j] = \dfrac{[a_{ji}]}{[a_{ii}]}$;

      $[b_j] = [b_j] - [\alpha_j] * [b_i]$

      POUR $k = i+1$ à n

        $[a_{jk}] = [a_{jk}] - [\alpha_j] * [a_{ik}]$

      FIN

    FIN

  FIN

  POUR $i = n$ à 1

  $$[p_i] = \frac{\left([b_i] - \sum_{j=i+1} [a_{ij}] * [p_j]\right)}{a_{ii}}$$

  FIN

FIN

Tableau A.1 : algorithme Elimination de Gauss généralisée.

A.2 Méthodes du point fixe

Pour un système de contraintes F, le principe de ces méthodes est de trouver une fonction ψ vérifiant :

$$F(x) = 0 \Leftrightarrow \psi(x) = x \qquad (A.2)$$

Connaissant une fonction d'inclusion de ψ, pour contracter $\mathcal{H}$ il suffit alors de remplacer $[x]$ par $[x] \cap [\psi]([x])$.

Comme illustrations de ces méthodes du point fixe, on présente ci-dessous le contractor de Krawczyk.

Exemple du Contracteur de Krawczyk

Considérons le CSP $\mathcal{H}$: (F([x]) = 0, $x \in$ [x]), avec le nombre de contraintes qui est égale à la dimension du problème ($p = n$) et F différentiable. Pour toute matrice M inversible on a :

$$F(x) = 0 \Leftrightarrow x\text{–}M.F(x) = x \qquad (A.3)$$

La fonction $\psi(x) = x\text{–}M.F(x)$ vérifie l'équivalence $F(x) = 0 \Leftrightarrow \psi(x) = x$. En utilisant la fonction d'inclusion centrée de ψ qui s'écrit :

$$[\psi]([x]) = \psi(x_0)+[J_\psi]([x]).([x]\text{–}x_0) \qquad (A.4)$$

Où $[J_\psi]$ est une fonction d'inclusion pour la Jacobienne de ψ et x_0 = mid([x]), on obtient le contracteur classiquement appelé contractor de Krawczyk.

Remarque :

L'expression de la fonction d'inclusion centrée pour une fonction f et x_0 = mid([x]) est $[f]([x]) = f(x_0)+[J_f]([x]).([x]\text{–}x_0)$. Cette expression découle du théorème de la valeur moyenne puisque pour tout x dans [x], $\exists\ \xi \in [x]$ qui vérifie :

$$f(x)\text{–}f(x_0)=J_f(\xi).(x\text{–}x_0) \qquad (A.5)$$

Le contractor de Krawczyk s'écrit C : $[x] \rightarrow [x]\cap\{\psi(x_0)+[J_\psi]([x])*([x]\text{–}x_0\}$ et en remplaçant $\psi(x)$ par x–M.F(x) , on obtient :

$$C : [x] \rightarrow [x]\cap(x_0\text{–}M.F(x_0)+(I\text{–}M\left[J_f\right]([x]))*([x]\text{–}x_0)) \qquad (A.6)$$

Où I est la matrice identité et $\left[J_f\right]$ une fonction d'inclusion pour la Jacobienne de F. La matrice M est souvent prise comme l'inverse $J_f^{-1}(x_0)$ du Jacobien de f en x_0 et son rôle peut être vu comme une matrice de préconditionnement.

Pour ce contracteur, l'efficacité de la contraction obtenue est meilleure lorsque l'intervalle [x] est de petite taille. En effet, l'erreur qui découle de l'approximation intervalle de J_f est

atténuée lorsque $[x]$ est de petite taille, de par le produit avec la différence $[x]–x_0$. Cette dernière est alors petite et centrée en 0. De surcroît, J_f étant continue, lorsque $[x]$ est de petite longueur, $M\left[J_f\right]([x])$ est très proche de I et donc, $I–M\left[J_f\right]([x])$ est très proche de la matrice nulle.

$x_0 = \text{mid}([x])$
$M = J_f^{-1}(x_0)$
$[J_\psi] = I–M\left[J_f\right]([x])$
$[r] = x_0–M\,f(x_0)+[J_\psi]*([x] – x_0)$
$[x] = [x]\cap[r]$

Tableau A.2 : contracteur de Krawczyk

B. Bibliothèque de fonctions élémentaires

Dans cette partie, figurent les algorithmes des opérateurs élémentaires qui ont été indispensables pour nos programmes. On montre également des fonctions d'inclusion de quelques fonctions classiques à savoir cos, sin, $\sin^{-1}$, $\cos^{-1}$. Pour les autres fonctions classiquement utilisées telles que tan, ln, exp. etc., leur monotonie permet d'effectuer de simples calculs sur les bornes pour obtenir leurs fonctions d'inclusion.

B.1 Réunion de 2 vecteurs d'intervalles

Algorithme réunion : Entrée($[x]$, $[y]$), Sortie($[z]$)
si $[x] = \varnothing$ alors $[z] = [y]$ si $[y] = \varnothing$ alors $[z] = [x]$ sinon $[z] =[\min(x_1, y_1), \max(x_2, y_2)]$ %les x_i et y_i sont des vecteurs

B.2 Intersection de 2 vecteurs d'intervalles

Algorithme inter : Entrée([x], [y]), Sortie([z])
si [x] = ∅ ou [y] = ∅ alors [z] = ∅ sinon n = taille du vecteur [x] [z] = [max(x_1, y_1) min(x_2, y_2)] pour $i = 1$ à n si $z_2(i) < z_1(i)$ [z] = ∅ %une coordonnée vide entraîne z vide

B.3 Somme de 2 vecteurs d'intervalles

Algorithme somme : Entrée([x], [y]), Sortie([z])
[z] =[x_1+y_1, x_2+y_2];

B.4 Différence de 2 vecteurs d'intervalles

Algorithme moins : Entrée([x], [y]), Sortie([z])
[z]=[$x_1 - y_2$, $x_2 - y_1$];

B.5 Produit de 2 vecteurs d'intervalles

Algorithme fois : Entrée ([x], [y]), Sortie([z])
v = (x_1y_1, x_1y_2, x_2y_1, x_2y_2) [z] =[min(v) max(v)]

B.6 Inverse d'un vecteur d'intervalle

Algorithme inverse : Entrée([x]), Sortie([z])
n = taille du vecteur [x] pour $i = 1$ à n % boucle pour traiter chaque composante du vecteur [x] si [x](i) contient un ouvert de 0 alors [z](i) = [$-\infty$, ∞] sinon [z](i) = [$1/x_2(i)$, $1/x_1(i)$] %il ne peut y'avoir division par 0

B.7 Division de 2 vecteurs d'intervalles

Algorithme divise : Entrée([x], [y]), Sortie([z])
[z] = fois([x],inverse([y]))

B.8 Sinus d'un vecteur d'intervalle

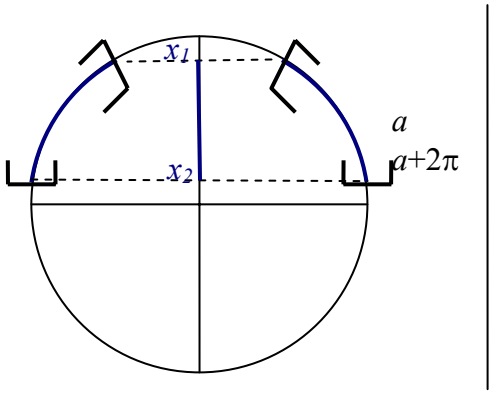
Algorithme sinus : Entrée([x]), Sortie([z])
n = taille du vecteur [x] Pour $i = 1$ *à* n $v_1 = x_1(i)$ mod(2π) %mod =modulo $v_2 = v_1+x_2(i)-x_1(i)$ %[v] représente [x] dans un voisinage proche de [0,2π] [z](i) = [min(sin($x_1(i)$),sin($x_2(i)$)), max(sin($x_1(i)$),sin($x_2(i)$))]; si $3\pi/2 \in$ [v] ou $7\pi/2 \in$ [v] alors $z_1(i) = -1$; si $\pi/2 \in$ [v]) ou $5\pi/2 \in$ [v] alors $z_2(i)$=1;

B.9 Cosinus d'un vecteur d'intervalle

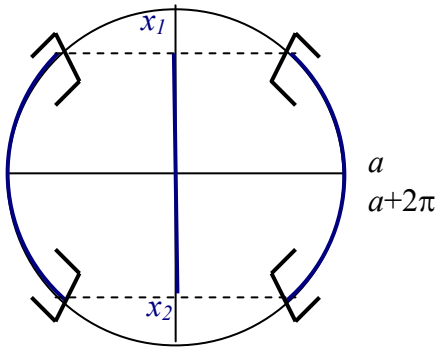
Algorithme cosinus : Entrée([x]), Sortie([z])
n = taille du vecteur [x] Pour $i = 1$ *à* n $v_1 = x_1(i)$ mod(2π) %mod =modulo $v_2 = v_1+x_2(i)-x_1(i)$ %[v] représente [x] dans un voisinage proche de [0,2π] [z](i) = [min(cos($x_1(i)$), cos($x_2(i)$)), max(cos($x_1(i)$), cos($x_2(i)$))]; si $\pi \in$ [v] ou $3\pi \in$ [v] alors $z_1(i) = -1$; si $\pi \in$ [v]) ou $2\pi \in$ [v] alors $z_2(i)$=1;

B.10 Inverse du sinus d'un intervalle

Le principe de notre algorithme est de donner un argument supplémentaire à la fonction $\sin^{-1}$ à savoir un réel a, qui indique que l'on cherchera l'inverse du sinus de [x] dans l'intervalle $[a, a + 2\pi]$. Notre algorithme qui est récursif, retourne selon le cas une liste d'au plus trois intervalles.



Cas où l'algorithme retourne 2 intervalles



Cas où l'algorithme retourne 3 intervalles

Algorithme asin : Entrée([x], a), Sortie([z])
si $a = -\pi/2$ alors % *l'algo est récursif* $[z](1)=[\sin^{-1}(x_1), \sin^{-1}(x_2)]$ $[z](2)=[\pi-\sin^{-1}(x_2), \pi-\sin^{-1}(x_1)]$ sinon $b = a+2\pi-\text{mod}(a,2\pi)-\pi/2$ si $b < a$ alors $b = b + 2\pi$ %*b représente -π/2 à 2π près dans l'intervalle [a, a + 2π]* $[v] = b+\pi/2+\text{asin}(x, -\pi/2)$ %*v = sin⁻¹([x]) % dans [b + π/2, b + π/2 + 2π]* si $v_1(1) - a > 2\pi$ alors $v_1(1) = v_1(1)-2\pi$ %*on ramène v₁(1) dans [a, a + 2π]* si $v_1(2) - a > 2\pi$ alors $v_1(2) = v_1(2)-2\pi$ si $v_2(1) - a > 2\pi$ alors $v_2(1) = v_2(1)-2\pi$ si $v_2(2) - a > 2\pi$ alors $v_2(2) = v_2(2)-2\pi$ si $v_1(2) < v_1(1)$ alors %*cas où a ∈ [v₁]* $[z_1] = [v_1(1), a+2\pi]$ $[z_2] = [a, v_1(2)]$ $[z_3] = [v_2]$

sinon

$[z_1] = [v_1]$

si $v_2(2) < v_2(1)$ alors %cas où $a \in [v_2]$

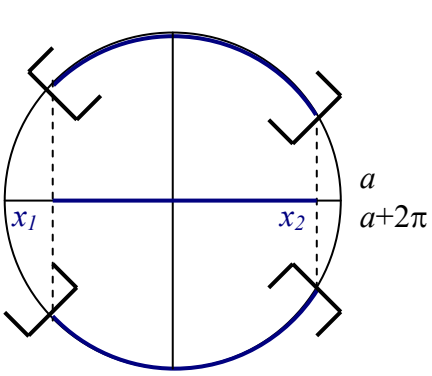
$[z_2] = [v_2(1), a + 2\pi]$

$[z_3] = [a, v_2(2)]$

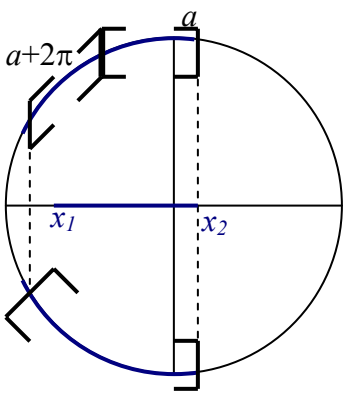
sinon $[z_2] = [v_2]$

B.11 Inverse du cosinus d'un intervalle

Comme pour l'algorithme de $\sin^{-1}$, $\cos^{-1}$ va recevoir un argument a lui indiquant de travailler à l'intérieur de l'intervalle $[a, a + 2\pi]$. L'algorithme et récursif et retourne selon le cas une liste de 2 ou 3 intervalles



Cas où l'algorithme retourne 2 intervalles



Cas où l'algorithme retourne 3 intervalles

Algorithme acos : Entrée($[x]$, a), Sortie($[z]$)
si $a = 0$ alors %*l'algo est récursif* $[z](1) = [\cos^{-1}(x_2), \cos^{-1}(x_1)]$ $[z](2) = [\pi - \cos^{-1}(x_1), \pi - \cos^{-1}(x_2)]$ sinon $b = a + 2\pi - \mathrm{mod}(a, 2\pi)$ %*b représente 0 à 2π près dans* $[a, a + 2\pi]$ $[v] = b + \mathrm{acos}(x, -\pi/2)$ %$v = \sin^{-1}([x])$ *dans* $[b, b + 2\pi]$

si $v_1(1) - a > 2\pi$ alors $v_1(1) = v_1(1) - 2\pi$ %*on ramène* $v_1(1)$ *dans* $[a, a + 2\pi]$

si $v_1(2) - a > 2\pi$ alors $v_1(2) = v_1(2) - 2\pi$

si $v_2(1) - a > 2\pi$ alors $v_2(1) = v_2(1) - 2\pi$

si $v_2(2) - a > 2\pi$ alors $v_2(2) = v_2(2) - 2\pi$

si $v_1(2) < v_1(1)$ alors %*cas où* $a \in [v_1]$

$[z_1] = [v_1(1), a + 2\pi]$

$[z_2] = [a, v_1(2)]$

$[z_3] = [v_2]$

sinon

$[z_1] = [v_1]$

si $v_2(2) < v_2(1)$ alors %cas où $a \in [v_2]$

$[z_2] = [v_2(1), a + 2\pi]$

$[z_3] = [a, v_2(2)]$

sinon $[z_2] = [v_2]$

C. Résolution du cycle de l'exemple statique

C.1 Description du cycle étudié

Rappelons que le cycle est constitué par le vecteur $v_1 = (\delta s, \delta_\theta, \delta_{RR}, \delta_{RL}, \psi, e, L, a_1, a_2, a_3, a_4)$:

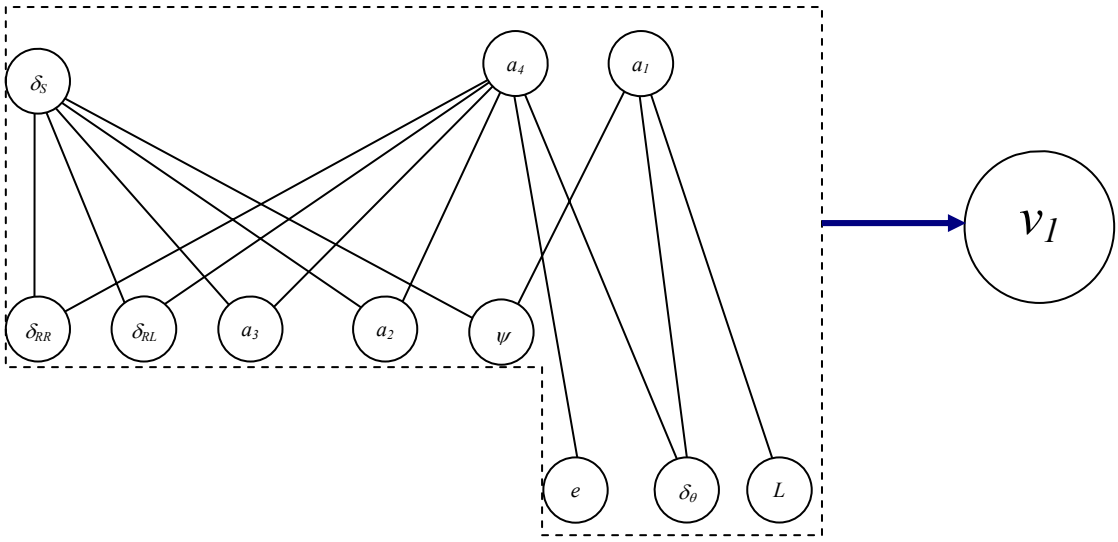


Figure E.1 : représentation sous forme d'un TVCSP à connexion à une branche CSP.

Le système de contraintes, noté ici F, correspondant au cycle v_1 est le suivant :

$$\begin{cases} a_1 = L.\delta_\theta \\ a_4 = e.\delta_\theta \\ \tan(\psi) = \dfrac{a_1}{\delta_S} \\ \delta_{RL} = \delta_S - a_4 \\ \delta_{RR} = \delta_S + a_4 \\ a_2 = \delta_S - a_4 \\ a_3 = \delta_S + a_4 \end{cases} \quad (7)$$

On propose dans cette annexe, à titre d'exemple, de calculer la solution pour δs. La méthode est similaire pour $[\delta_\theta]$.

C.2 Calcul de [δs] globalement consistant

Posons I les indices de 1 à 11 du vecteur (δs, δ_θ, δ_{RR}, δ_{RL}, ψ, e, L, a_1, a_2, a_3, a_4). D'après le Théorème 1 (chap 4), on a :

$$\Pi_1(S) = [\delta s] \cap D_F(x_{I\setminus\{1\}})$$

$$= [\delta s] \cap D_F([\delta_\theta], [\delta_{RR}], [\delta_{RL}], [\psi], [e], [L], [a_1], [a_2], [a_3], [a_4])$$

où

- $\Pi_1(S)$ est la projection de la solution S sur l'espace de δs
- $D_F(x_{I-\{1\}})$ est le domaine de consistance des composantes de x autres que δs

Le problème est, à présent, de calculer $D_F(x_{I-\{1\}})$. Sachant que δs n'est relié qu'aux variables δ_{RR}, δ_{RL}, ψ, a_1, a_2, a_3, a_4 d'indices respectifs dans I : 3, 4, 5, 8, 9, 10, 11 dans x, et en utilisant le Lemme 2 du Théorème 3 (chap. 4) , on peut écrire :

$$D_F(x_{I-\{1\}}) = D_{F_{\{1\}}}(\Pi_{3,4,5,8,9,10,11}(S_{I-\{1\}}))$$

où $S_{I-\{1\}}$ est l'ensemble des solutions du CSP $\mathcal{H}_{I-\{1\}}$ constitué uniquement des variables autres que δs et des contraintes autres que celles faisant intervenir δs, comme illustré sur la

Figure E.2. $F_{\{1\}}$ est l'ensemble des contraintes concernant la variable 1 (qui est ici δ_S) qui sont $\tan(\psi)=\frac{a_1}{\delta_S}$, $\delta_{RL}=\delta_S-a_4$, $\delta_{RR}=\delta_S+a_4$, $a_2=\delta_S-a_4$ et $a_3=\delta_S+a_4$.

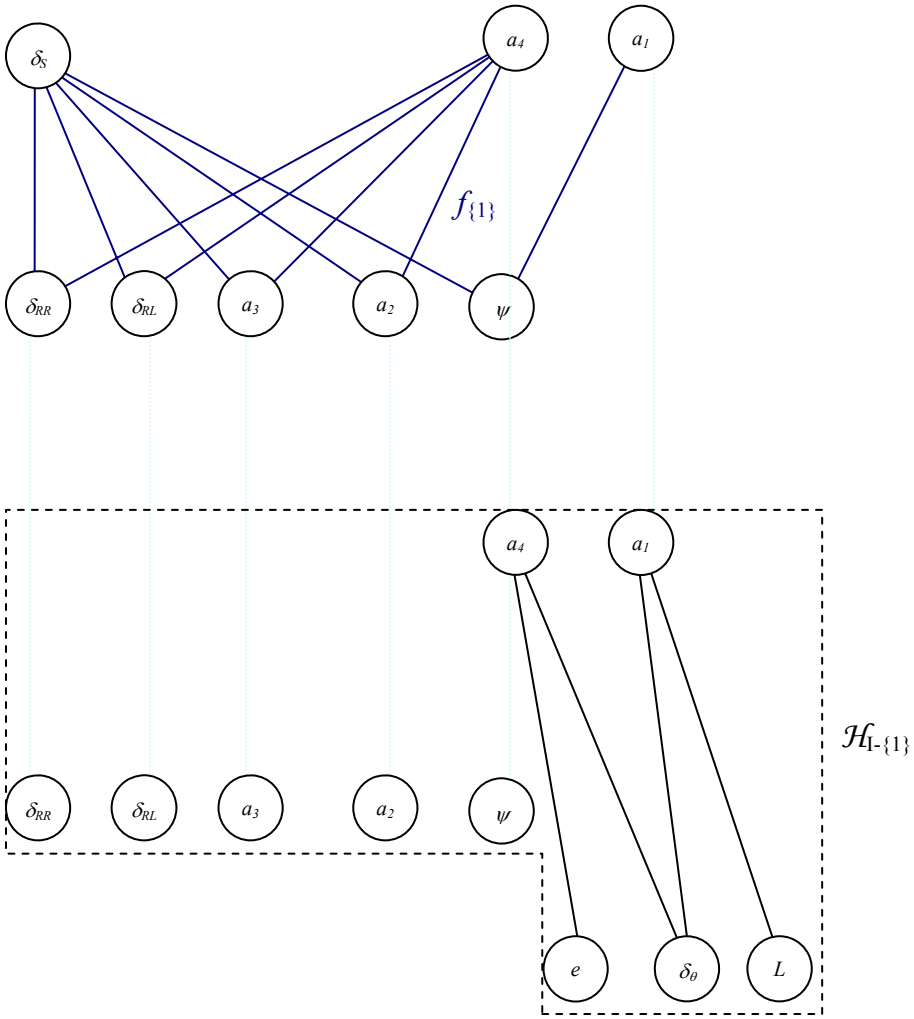


Figure E.2 : illustration du Théorème 2 (chap. 4) appliqué au vecteur v_1.

Pour calculer le domaine de consistance $D_{F_{\{1\}}}(\Pi_{3,\,4,\,5,\,8,\,9,\,10,\,11}(S_{I-\{1\}}))$, on procède par les 4 étapes qui ont été introduites dans le chapitre 4 qui sont décrites ci-après .

1. *Etape 1 : évaluation des bornes de l'intervalle* $D_F([a]\times[b]\times[x])$

Calculons, tout d'abord, la projection $\Pi_{3,\,4,\,5,\,8,\,9,\,10,\,11}(S_{I-\{1\}})$. Il s'avère que le CSP $\mathcal{H}_{I-\{1\}}$ est une foret (un ensemble d'arbres) pour lequel on peut atteindre la consistance globale avec l'algorithme FALL/ CLIMB classique.

Par ailleurs, $\mathcal{H}_{I-\{1\}}$ a une structure telle que les solutions dans les variables δ_{RR}, δ_{RL}, a_2, a_3 et ψ sont indépendantes entre elles et indépendantes également solutions dans les a_1 et a_4 (voir la Figure E.2). De surcroît, en tenant compte des deux contraintes du CSP $\mathcal{H}_{I-\{1\}}$ que sont $a_1 = L.\delta_\theta$ et $a_4 = e.\delta_\theta$ et sachant que, pour ce problème, δ_θ est initialisé à [−∞, ∞], les solutions dans les a_1 et a_4 sont également indépendant et parcourent IR pour les 2 variables. On peut donc écrire :

$$\Pi_{3,\,4,\,5,\,8,\,9,\,10,\,11}(S_{I-\{1\}}) = \Pi_3(S_{I-\{1\}})\times\Pi_4(S_{I-\{1\}})\times\Pi_5(S_{I-\{1\}})\times\Pi_8(S_{I-\{1\}})\times\Pi_9(S_{I-\{1\}})\times$$

$$\Pi_{10}(S_{I-\{1\}})\times\Pi_{11}(S_{I-\{1\}})$$

Le calcul des projections de $S_{I-\{1\}}$ sur ces sept espaces se fait de la manière suivante :

$\Pi_3(S_{I-\{1\}}) = [\delta_{RR}]$, $\Pi_4(S_{I-\{1\}}) = [\delta_{RL}]$, $\Pi_5(S_{I-\{1\}}) = [\psi]$, $\Pi_8(S_{I-\{1\}}) = [a_1]$, $\Pi_9(S_{I-\{1\}}) = [a_2]$,

$\Pi_{10}(S_{I-\{1\}}) = [a_3]$, $\Pi_{11}(S_{I-\{1\}}) = [a_4]$.

Finalement, $\Pi_{3,\,4,\,5,\,8,\,9,\,10,\,11}(S_{I-\{1\}}) = [\delta_{RR}]\times[\delta_{RL}]\times[\psi]\times[a_1]\times[a_2]\times[a_3]\times[a_4]$. A ce stade, il reste à calculer le domaine de consistance : $D_{F_{\{1\}}}(\Pi_{3,\,4,\,5,\,8,\,9,\,10,\,11}(S_{I-\{1\}}))$. Pour cela, on utilise le Théorème 2 pour pouvoir exprimer le domaine de consistance du système $F_{\{1\}}$ comme une intersection des domaines de consistance des contraintes constituants $F_{\{1\}}$. Ainsi, pour un scalaire $a_4 \in [a_4]$, le domaine de consistance associé à l'ensemble $([\delta_{RR}]\times[\delta_{RL}]\times[\psi]\times[a_1]\times[a_2]\times[a_3]\times a_4)$ est donné par :

$$D_{F_{\{1\}}}([\delta_{RR}]\times[\delta_{RL}]\times[\psi]\times[a_1]\times[a_2]\times[a_3]\times a_4) = \frac{[a_1]}{tan([\psi])} \cap([\delta_{RL}]+a_4)\cap([\delta_{RR}]-a_4)$$

$$\cap([a_2]+a_4)\cap([a_3]-a_4).$$ Regroupons les termes qui sont fonctions de a_4.

$$= \frac{[a_1]}{tan([\psi])} \cap([\underline{\delta}_{RL}+a_4,\ \overline{\delta}_{RL}+a_4])\cap([\underline{\delta}_{RR}-a_4,\ \overline{\delta}_{RR}-a_4])\cap([\underline{a}_2+a_4,\ \overline{a}_2+a_4])\cap([\underline{a}_3-a_4,\ \overline{a}_3-a_4])$$

$$= \frac{[a_1]}{tan([\psi])} \cap[\mathrm{Sup}\{\underline{\delta}_{RL}+a_4,\ \underline{\delta}_{RR}-a_4,\ \underline{a}_2+a_4,\ \underline{a}_3-a_4\},\ \mathrm{Inf}\{\overline{\delta}_{RL}+a_4,\ \overline{\delta}_{RR}-a_4,\ \overline{a}_2+a_4,\ \overline{a}_3-a_4\}]$$

Le calcul de cet intersection se simplifie, en posant $a_5 = \text{Sup}\{\underline{\delta}_{RL},\ \underline{a}_2\}$ et $a_6 = \text{Sup}\{\underline{\delta}_{RR},\ \underline{a}_3\}$. On a : $\text{Sup}\{\underline{\delta}_{RR}-a_4,\ \underline{\delta}_{RL}+a_4,\ \underline{a}_2+a_4,\ \underline{a}_3-a_4\} = \text{Sup}\{a_5+a_4,\ a_6-a_4\}$. De même en posant $a_7=\text{Inf}\{\overline{\delta}_{RL}, \overline{a}_2\}$ et $a_8=\text{Inf}\{\overline{\delta}_{RR},\ \overline{a}_3\}$, on a alors:

$$D_{F_{\{1\}}}([\delta_{RR}]\times[\delta_{RL}]\times[\psi]\times[a_1]\times[a_2]\times[a_3]\times a_4) = \frac{[a_1]}{tan([\psi])} \cap [\text{Sup}\{a_5+a_4,\ a_6-a_4\},\ \text{Inf}\{a_7+a_4,\ a_8-a_4\}]$$

2. ***Etape 2 : caractérisation des bornes de*** $D_{F_{\{1\}}}([\delta_{RR}]\times[\delta_{RL}]\times[\psi]\times[a_1]\times[a_2]\times[a_3]\times a_4)$ ***en fonction de*** a_4

Intéressons nous au terme $[\text{Sup}\{a_5+a_4,\ a_6-a_4\},\ \text{Inf}\{a_7+a_4,\ a_8-a_4\}]$. Par suite, nous proposons deux tableaux récapitulant le sup et l'inf de ces fonctions selon la position de a_4 sur IR (avec $a_{5,6} = \frac{a_5 - a_6}{2}$ et $a_{7,8} = \frac{a_7 - a_8}{2}$).

	$a_4 \in [-\infty,\ a_{5,6}]$	$a_4 \in [a_{5,6},\ +\infty]$
$\text{Sup}\{a_5+a_4,\ a_6-a_4\}$	a_6-a_4	a_5+a_4

Tableau E.3 : valeur prise par $\text{Sup}\{a_5+a_4,\ a_6-a_4\}$ en fonction de a_4.

	$[-\infty,\ a_{7,8}]$	$[a_{7,8},\ +\infty]$
$\text{Inf}\{a_7+a_4,\ a_8-a_4\}$	a_7+a_4	a_8-a_4

Tableau E.4 : valeur prise par $\text{Sup}\{a_5+a_4,\ a_6-a_4\}$ en fonction de a_4.

Grâce à ces deux tableaux, on connaît un recouvrement d'intervalles de IR l'intérieur desquelles on connaît les bornes de l'intervalle $[\text{Sup}\{a_5+a_4,\ a_6-a_4\},\ \text{Inf}\{a_7+a_4,\ a_8-a_4\}]$ (comme illustré sur la Figure E.3).

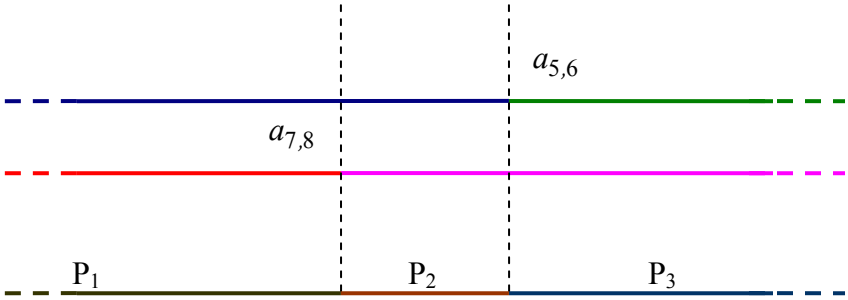


Figure E.3 : représentation des 3 secteurs P_1, P_2, P_3 (dans le cas où $a_{7,8} < a_{5,6}$).

En définitive, grâce au Tableau E.3 et au Tableau E.4, on peut déduire le Tableau E.5 et le Tableau E.5 qui donnent respectivement l'expression du domaine de consistance $D_{F_{\{1\}}}([\delta_{RR}]\times[\delta_{RL}]\times[\psi]\times[a_1]\times[a_2]\times[a_3]\times a_4)$ en fonction des P_i.

- Cas où $a_{7,8} < a_{5,6}$

	$P_1 = [-\infty,\ a_{7,8}]$	$P_2 = [a_{7,8}, a_{5,6}]$	$P_3 = [a_{5,6},\ +\infty]$
$D_{F_{\{1\}}}([\delta_{RR}]\times[\delta_{RL}]\times[\psi]\times[a_1]\times[a_2]\times[a_3]\times a_4)$	$[a_6-a_4,\ a_7+a_4]$	$[a_6-a_4,\ a_8-a_4]$	$[a_5+a_4,\ a_8-a_4]$

Tableau E.5 : expression de $D_{F_{\{1\}}}([\delta_{RR}]\times[\delta_{RL}]\times[\psi]\times[a_1]\times[a_2]\times[a_3]\times a_4)$ en fonction de a_4.

- Cas où $a_{7,8} < a_{5,6}$

	$P_1 = [-\infty,\ a_{5,6}]$	$P_2 = [a_{5,6}, a_{7,8}]$	$P_3 = [a_{7,8},\ +\infty]$
$D_{F_{\{1\}}}([\delta_{RR}]\times[\delta_{RL}]\times[\psi]\times[a_1]\times[a_2]\times[a_3]\times a_4)$	$[a_6-a_4,\ a_7+a_4]$	$[a_5+a_4,\ a_7+a_4]$	$[a_5+a_4,\ a_8-a_4]$

Tableau E.6 : expression de $D_{F_{\{1\}}}([\delta_{RR}]\times[\delta_{RL}]\times[\psi]\times[a_1]\times[a_2]\times[a_3]\times a_4)$ en fonction de a_4.

3. ***Etape 3 : élimination des a_4 pour lesquels l'intervalle*** $[\mathrm{Sup}\{a_5+a_4,\ a_6-a_4\},\ \mathrm{Inf}\{a_7+a_4,\ a_8-a_4\}]$ ***est impropre (vide).***

Après avoir obtenu les expressions de $[\mathrm{Sup}\{a_5+a_4,\ a_6-a_4\},\ \mathrm{Inf}\{a_7+a_4,\ a_8-a_4\}]$, en comparant les bornes, on peut, grâce à un raisonnement similaires sur les nouvelles bornes obtenues, trouver, à nouveau, un recouvrement $(P_{i,S})_{i=1\ldots3}$ de IR de manière à connaître les espaces dans lesquels $\mathrm{Sup}\{a_5+a_4,\ a_6-a_4\} \le \mathrm{Inf}\{a_7+a_4,\ a_8-a_4\}$ sont non vides i.e. les espaces où $[\mathrm{Sup}\{a_5+a_4,\ a_6-a_4\},\ \mathrm{Inf}\{a_7+a_4,\ a_8-a_4\}]$ est non vide.

- dans P_1,
  $[\mathrm{Sup}\{a_5+a_4,\ a_6-a_4\},\ \mathrm{Inf}\{a_7+a_4,\ a_8-a_4\}] = [a_6-a_4,\ a_7+a_4] \Rightarrow$ l'ensemble $P_{1,S}$ des a_4 pour lesquels $[a_6-a_4,\ a_7+a_4]$ est non vide est $P_{1,S} = P_1 \cap [\,a_{6,7}, +\infty[$.
- dans P_2,
  $[\mathrm{Sup}\{a_5+a_4,\ a_6-a_4\},\ \mathrm{Inf}\{a_7+a_4,\ a_8-a_4\}] = [a_6-a_4,\ a_8-a_4]$ ou $[\mathrm{Sup}\{a_5+a_4,\ a_6-a_4\},\ \mathrm{Inf}\{a_7+a_4,\ a_8-a_4\}] = [a_5+a_4,\ a_7+a_4]$. On a alors, dans les deux cas, $P_{2,S} = P_2$
- dans P_3,
  $[\mathrm{Sup}\{a_5+a_4,\ a_6-a_4\},\ \mathrm{Inf}\{a_7+a_4,\ a_8-a_4\}]=[a_5+a_4,\ a_8-a_4] \Rightarrow$ l'ensemble $P_{2,S}$ des a_4 pour lesquels$[a_5+a_4,\ a_8-a_4]$ est non vide est $P_{1,S} = P_1 \cap]-\infty, a_{5,8}\,]$.

4. ***Etape 4 : calcul de*** $D_{F_{\{1\}}}([\delta_{RR}]\times[\delta_{RL}]\times[\psi]\times[a_1]\times[a_2]\times[a_3]\times[a_4])$ ***connaissant*** $D_{F_{\{1\}}}([\delta_{RR}]\times[\delta_{RL}]\times[\psi]\times[a_1]\times[a_2]\times[a_3]\times a_4)$ ***et les*** $P_{i,s}$

On a $D_{F_{\{1\}}}([\delta_{RR}]\times[\delta_{RL}]\times[\psi]\times[a_1]\times[a_2]\times[a_3]\times[a_4]) = \bigcup_{a_4 \in \mathrm{IR}} \left(\frac{[a_1]}{tan([\psi])} \cap [\mathrm{Sup}\{a_5+a_4,\ a_6-a_4\},\right.$

$\left.\mathrm{Inf}\{a_7+a_4,\ a_8-a_4\}]\right) = \frac{[a_1]}{tan([\psi])} \cap \bigcup_{a_4 \in [a_4]} [\mathrm{Sup}\{a_5+a_4,\ a_6-a_4\},\ \mathrm{Inf}\{a_7+a_4,\ a_8-a_4\}]$

$$= \frac{[a_1]}{tan([\psi])} \cap \bigcup_{i=1\cdots3} \bigcup_{a_4 \in P_i \cap [a_4]} [\mathrm{Sup}\{a_5+a_4,\ a_6-a_4\},\ \mathrm{Inf}\{a_7+a_4,\ a_8-a_4\}]$$

$$= \frac{[a_1]}{tan([\psi])} \cap \bigcup_{i=1\cdots3} \bigcup_{a_4 \in P_{i,S} \cap [a_4]} [\mathrm{Sup}\{a_5+a_4,\ a_6-a_4\},\ \mathrm{Inf}\{a_7+a_4,\ a_8-a_4\}]$$

Or, d'après l'étape 3, on connaît l'expression de $[\mathrm{Sup}\{a_5+a_4,\ a_6-a_4\},\ \mathrm{Inf}\{a_7+a_4,\ a_8-a_4\}]$ dans les $(P_{i,S})_{i=1\ldots3}$. Par exemple dans $P_{1,S}$, $[\mathrm{Sup}\{a_5+a_4,\ a_6-a_4\},\ \mathrm{Inf}\{a_7+a_4,\ a_8-a_4\}] = [a_6-a_4,\ a_7+a_4]$ d'où $\bigcup_{a_4 \in P_{1,S} \cap [a_4]} [\mathrm{Sup}\{a_5+a_4,\ a_6-a_4\},\ \mathrm{Inf}\{a_7+a_4,\ a_8-a_4\}] = \bigcup_{a_4 \in P_{1,S} \cap [a_4]} [a_6-a_4,\ a_7+a_4]$.

En utilisant la monotonie des fonctions a_6-a_4 et a_7+a_4, et en posant $\left[\ a_{4,1}\right] = P_{1,S} \cap [a_4] = \left[\underline{a}_{4,1}, \overline{a}_{4,1}\right]$, on a alors $\bigcup_{a_4 \in P_{1,S} \cap [a_4]} [\mathrm{Sup}\{a_5+a_4,\ a_6-a_4\},\ \mathrm{Inf}\{a_7+a_4,\ a_8-a_4\}] = \left[a_5+\overline{a}_{4,1}, a_6-\underline{a}_{4,1}\right]$ (l'union est vide dans le cas où $\left[\ a_{4,1}\right] = \varnothing$).

On obtient une expression de $D_{F_{\{1\}}}([\delta_{RR}]\times[\delta_{RL}]\times[\psi]\times[a_1]\times[a_2]\times[a_3]\times[a_4])$ En faisant de même pour $P_{2,S}$, et $P_{3,S}$.

D. Définitions et concepts de base de la théorie des graphes

La théorie des graphes est née en 1736 quand Euler démontra qu'il était impossible de traverser chacun des sept ponts de la ville russe de Königsberg (aujourd'hui Kaliningrad) une fois exactement et de revenir au point de départ. Les graphes sont des outils qui servent essentiellement à modéliser et à résoudre de nombreux problèmes notamment en orientant l'intuition lors d'un raisonnement. Autour de ces graphes existent de puissants résultats connus qui facilitent grandement de nombreux problèmes une fois modélisés. Nous rappelons dans cette partie, quelques définitions qui ont été généralisées dans le chapitre 4.

D.1 Graphes orientés et non orientées

- ***Un graphe non orienté*** est un couple G = (X, U) où X est un ensemble dont les éléments sont appelés des sommets ou noeuds et où U est un sous ensemble de parties appelées arêtes de X. Chaque arête est constituée de 2 éléments de l'ensemble X. Cette classe de graphe correspond à des cas où on est en face d'un problème où on s'intéresse uniquement à l'existence d'une relation entre deux composantes d'un système.
- ***Un graphe orienté*** est défini de la même manière comme un couple G = (X, U) à la différence près qu'on parle d'arc à la place d'arêtes. Un arc dans U est un couple de 2 d'éléments de X. L'orientation qui apparaît à travers les arcs vient d'un désir de modéliser des cas où une variable est en relation avec une autre, sans que le contraire soit vrai.

D.2 Quelques concepts usuels sur les graphes

- On définit une boucle comme un arc (ou arête pour les cas non orienté) reliant un même sommet.
- Un graphe partiel G' de G est un graphe ayant même ensemble de sommets que G et dont l'ensemble des arcs (resp. des arêtes) est inclus dans l'ensembles des arcs (resp. des arêtes) de G, i.e. G' = (X, U') avec U' un sous ensemble de U.
- Le sous graphe de G associé à un sous ensemble A de X est par définition le graphe G_A défini par le graphe ayant pour sommets les éléments dans A et ayant pour arcs (resp. arêtes) les arêtes reliant exclusivement des sommets dans A, i.e. $G_A = (A, U \cap A \times A)$
- Un chemin (resp. une chaîne) pour un graphe orienté (resp. non orienté) est une suite de sommets $[x_0, \ldots, x_p]$ pour laquelle on vérifie que pour tout $i \in 1 \ldots p-1$, l'arc (resp. l'arête)

(x_i, x_{i+1}) appartient au graphe. La longueur d'un chemin (resp. d'une chaîne) est définie comme son nombre d'arcs.

- Un circuit (resp. un cycle) pour un graphe orienté (resp. non orienté) est un chemin (resp. une chaîne) ayant le même sommet comme extrémité initiale et extrémité finale.
- Un graphe est dit connexe s'il existe un chemin entre deux sommets quelconques du graphe
- On définit la connexité forte pour les graphes orientés (resp. faible pour les graphes non orientés) comme la relation d'équivalence basée sur l'existence de chaîne (resp. cycle) entre deux sommets. Ainsi deux sommets x et y sont connectés si $x=y$ ou s'il existe une chaîne (resp. cycle) les reliant.

D.3 Graphes particuliers : arbres et forets

Dans cette partie, on s'intéresse à la structure d'arbre très recherchée pour les graphes non orientés. Signalons qu'il existe également la structure d'arborescence qui correspond aux arbres pour les graphes orientés et qui ne seront pas détaillés ici.

Par définition, un arbre (T, U) est un graphe non orienté, connexe et sans cycle. Une forêt est un graphe sans cycle (donc un ensemble d'arbres). Autours des arbres existent un certain vocabulaire utile pour en extraire d'intéressantes propriétés listées dans les différents points ci-dessous :

- On définit la racine comme le nœud d'origine d'un arbre. On peut définir une racine à partir de n'importe quel nœud de l'arbre.
- Considérons un graphe ayant pour racine x_1, pour chaque nœud du graphe x_i, on peut définir le sous-arbre T_{x_i} à partir de l'arbre T ayant x_1 pour racine. En effet, en considérant $x_2,\cdots,x_{q+1}$ qui sont les q variables reliées à la racine x_1 et supprimant les arcs qui les relient à x_1, on obtient q nouveaux arbres $T_{x_2},\cdots,T_{x_{q+1}}$ ayant pour racines respectivement $x_2,\cdots,x_{q+1}$. On obtient ainsi, en raisonnant de la même manière pour les arbres $T_{x_2},\cdots,T_{x_{q+1}}$, les sous-arbres définis à partir des ramifications des nœuds x_i.
- En ayant choisi un nœud comme racine d'un arbre T, on peut définir la relation d'ascendance et de descendance entre les nœuds du graphe. Ainsi un nœud x_1 est un

ascendant à un noeud x_2 (ou un nœud x_2 est un descendant à un nœud x_1) si le sous arbre T_{x_1} contient x_2.

- Un nœud x_i dont le sous-arbre T_{x_i} correspondant ne contient que le nœud x_i lui-même, est appelé feuille.

E. Algorithme FALL CLIMB pour les VCSP

E.1 Démonstrations des résultats annoncés pour les contraintes ternaires

Dans cette partie, on propose de donner les démonstrations pour les résultats donnés dans le chapitre 5. Il s'agissait de faciliter la lecture de ce chapitre en plaçant ici des démonstrations qui sont redondantes avec celles concernant les contraintes binaires.

E.1.1 Démonstration du lemme sur les triplets de vecteurs connectés par une branche

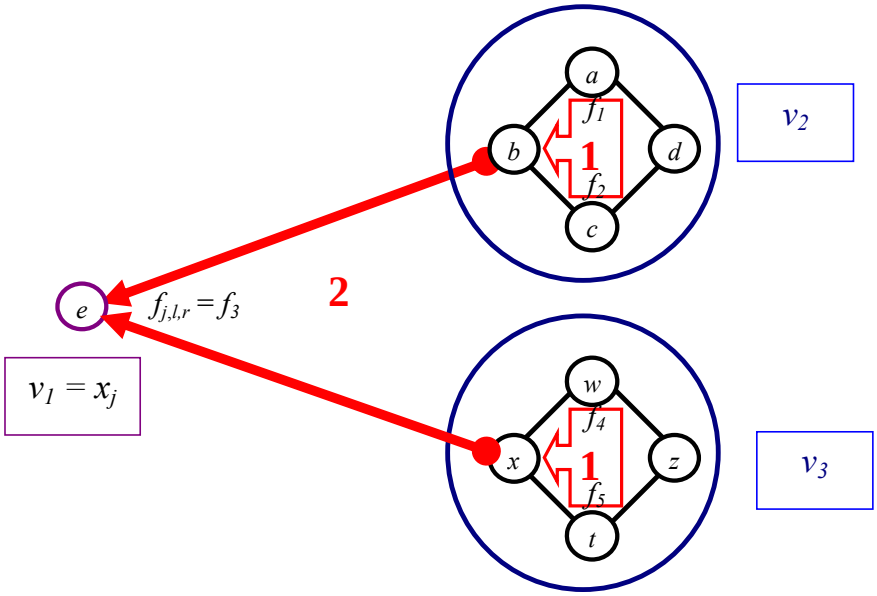


Figure E.4 : illustration des 2 étapes de calcul pour rendre consistante une variable reliée , par une contrainte ternaire, avec un couple de vecteurs de variables.

Considérons trois restrictions de CSP $\mathcal{H}_{v_1}$, $\mathcal{H}_{v_2}$ et $\mathcal{H}_{v_3}$ connectés par une branche, de par une contrainte ternaire que l'on note $f_{j,l,r}$ qui relient les variables d'indices *j*, *l*, *r* dans respectivement v_1, v_2 et v_3 .On suppose que $v_1 = x_j$ est une variable scalaire. En considérant le CSP $\mathcal{H}$: (F(x) = 0, $x \in$ D) qui regroupe ces variables respectivement avec leur domaine et toutes leurs contraintes plus la contrainte $f_{j,l,r}$, on peut écrire :

$$\prod_j (S) = \prod_{f_{j,l,r},j} \left(\mathrm{D}'_x \right) \qquad \text{(D.8)}$$

Où $\prod_{f_{j,l,r}} \left(\mathrm{D}'_x \right)$ et la projection de la contrainte $f_{j,l,r}$ sur le domaine $\mathrm{D}'_x = \left(\mathrm{D}'_{x,1}, \cdots, \mathrm{D}'_{x,n} \right)$. D'_x est une contraction uniquement en *l* et *r* du domaine initial D de *x* et qui vérifie :

- Pour $k=l$, $\mathrm{D}'_{x,k} = \prod_l \left(S_{v_2} \right)$
- Pour $k=r$, $\mathrm{D}'_{x,k} = \prod_r \left(S_{v_3} \right)$
- Pour les *k* restants $\mathrm{D}'_{x,k} = \mathrm{D}_k$

Preuve :

■ D'après le ***théorème 1*** (chapitre4) on peut écrire : $\Pi_j(\mathrm{S}) = [x_j] \cap \mathrm{D_F}(\overset{\times}{_{i \in I-\{j\}}} [x_i])$

On sait de plus que d'après le ***corollaire 2*** du ***théorème 3*** (chapitre4), on a : $\mathrm{D_F}\left(\underset{i \in I-\{j\}}{\times} [x_i] \right) = \mathrm{D}_{f_{\{j\}}} \left(\prod_{j_1 \ldots j_q} \left(\mathrm{S}_{\mathrm{I}-\{j\}} \right) \right)$. Sachant que, dans le cas présent, l'ensemble des variables (j_1, ..., j_q) qui sont reliées par une contrainte à *j* est réduit à *l* et *r*, et l'ensemble des contraintes f_j qui concernent *j* est réduit à la seule contrainte $f_{j,l,r}$, on peut donc écrire, $\mathrm{D_F}\left(\underset{i \in I-\{j\}}{\times} [x_i] \right)$ selon l'expression : $\mathrm{D_F}\left(\underset{i \in I-\{j\}}{\times} [x_i] \right) = \mathrm{D}_{f_{j,l,r}} \left(\prod_{lr} \left(\mathrm{S}_{\mathrm{I}-\{j\}} \right) \right)$.

De plus, $\mathcal{H}_{I-\{j\}}$ ne contenant, par définition, que la contrainte $f_{j,l,r}$ qui est la seule contrainte entre v_2, v_3, les deux variables d'indices *l* et *r* ne sont donc pas connectées dans

$\mathcal{H}_{I-\{j\}}$ (on entend par connexion ici qu'il n'y pas de chaîne allant d'une variable à l'autre). Cela implique que les solutions de $\mathcal{H}_{I-\{j\}}$ dans *l* et *r* sont indépendantes , i.e.

$$\prod_{lr}\left(\mathrm{S}_{\mathrm{I}-\{j\}}\right)=\prod_{l}\left(\mathrm{S}_{\mathrm{I}-\{j\}}\right)\times\prod_{r}\left(\mathrm{S}_{\mathrm{I}-\{j\}}\right).$$

Par suite on peut écrire $\mathrm{D}_{\mathrm{F}}\left(\underset{i\in I-\{j\}}{\times}\left[x_i\right]\right)=\mathrm{D}_{f_{j,l,r}}\left(\prod_{l}\left(\mathrm{S}_{\mathrm{I}-\{j\}}\right)\times\prod_{r}\left(\mathrm{S}_{\mathrm{I}-\{j\}}\right)\right)$ et en reprenant l'expression $\Pi_j(\mathrm{S}) = [x_j] \cap \mathrm{D}_{\mathrm{F}}(\underset{i\in I-\{j\}}{\times}[x_i])$, on a : $\Pi_j(\mathrm{S}) = [x_j]\cap \mathrm{D}_{f_{j,l,r}}\left(\prod_{l}\left(\mathrm{S}_{\mathrm{I}-\{j\}}\right)\times\prod_{r}\left(\mathrm{S}_{\mathrm{I}-\{j\}}\right)\right)$. Or, comme on l'a vu dans la section 3.1 du chapitre 4 (il s'agissait notamment d'écrire une projection a l'aide de calculs de domaine de consistance), l'expression $[x_j]\cap \mathrm{D}_{f_{j,l,r}}\left(\prod_{l}\left(\mathrm{S}_{\mathrm{I}-\{j\}}\right)\times\prod_{r}\left(\mathrm{S}_{\mathrm{I}-\{j\}}\right)\right)$ correspond à la projection de la contrainte $f_{j,l,r}$ sur la $j^{\text{ième}}$ dimension en considérant le domaine des x_l possibles égal au domaine $\prod_{l}\left(\mathrm{S}_{\mathrm{I}-\{j\}}\right)$ et celui des x_r possibles égal au domaine $\prod_{r}\left(\mathrm{S}_{\mathrm{I}-\{j\}}\right)$.

En outre $\mathcal{H}_{I-\{j\}}$ étant composé des deux restrictions de CSP $\mathcal{H}_{v_2}$ et $\mathcal{H}_{v_3}$ non connectés (la contrainte $f_{j,l,r}$ n'appartenant pas à $\mathcal{H}_{I-\{j\}}$), on a les solutions à $\mathcal{H}_{I-\{j\}}$ dans v_2 et v_3 sont dissociés. On peut donc écrire $\mathrm{S}_{\mathrm{I}-\{j\}}=S_{v_2}\times S_{v_3}$. D'où $\prod_{l}\left(\mathrm{S}_{\mathrm{I}-\{j\}}\right)=\prod_{l}\left(S_{v_2}\times S_{v_3}\right)=\prod_{l}\left(S_{v_2}\right)$ et $\prod_{r}\left(\mathrm{S}_{\mathrm{I}-\{j\}}\right)=\prod_{r}\left(S_{v_2}\times S_{v_3}\right)=\prod_{r}\left(S_{v_3}\right)$

Pour conclure, on peut alors écrire $\prod_{j}(S)$ sous la forme voulue $\prod_{f_{j,l,r},j}\left(\mathrm{D}'_x\right)$ avec D'_x vérifiant :

- Pour $k=l$, $\mathrm{D}'_{x,k} = \prod_{l}\left(S_{v_2}\right)$
- Pour $k=r$, $\mathrm{D}'_{x,k} = \prod_{r}\left(S_{v_3}\right)$
- Pour les *k* restants $\mathrm{D}'_{x,k} = \mathrm{D}_k$ ■

E.1.2 Démonstration de la proposition sur les triplets de vecteurs connectés par une contrainte ternaire

Considérons un CSP $\mathcal{H} = (F(x) = 0, x \in D)$ et son VCSP $\mathcal{H}_{\mathcal{V}}$ pour $\mathcal{V} = (v_1, v_2, v_3)$. On suppose que $\mathcal{H}_{\mathcal{V}}$ est à connexité à une branche de par une contrainte ternaire reliant les trois vecteurs. On propose de caractériser l'ensemble des solutions du CSP $\mathcal{H}$ dans $[v_1]$. Par symétrie ce résultat sera applicable pour $[v_2]$ ou $[v_3]$. Posons alors j, l, et r les indices, respectivement, dans v_1, v_2 et v_3 des trois variables reliées par une contrainte noté $f_{j,l,r}$ et notons n_1 la dimension de v_1. On a pour tous les indices k dans v_1 :

$$\prod_k(S) = \prod_k\left(S'_{v_1}\right) \tag{E.9}$$

Où S est la solution du CSP $\mathcal{H}$ et S'_{v_1} est la solution de la restriction $\mathcal{H}_{v_1}$ avec un domaine initial $D'_{v_1} = \left(D'_{v_1,1}, \cdots, D'_{v_1,n_1}\right)$ qui vérifie :

- Pour $k = j$, $D'_{v_1,k} = \prod_{f_{j,l,r},j}(D'_x)$
- Pour les k restants dans v_1, $D'_{v_1,k} = D_k$

Où D'_x est une contraction du domaine associé à x qui vérifie :

- Pour $k=l$, $D'_{x,k} = \prod_l\left(S_{v_2}\right)$
- Pour $k=r$, $D'_{x,k} = \prod_r\left(S_{v_3}\right)$
- Pour les k restants dans x, $D'_{x,k} = D_k$

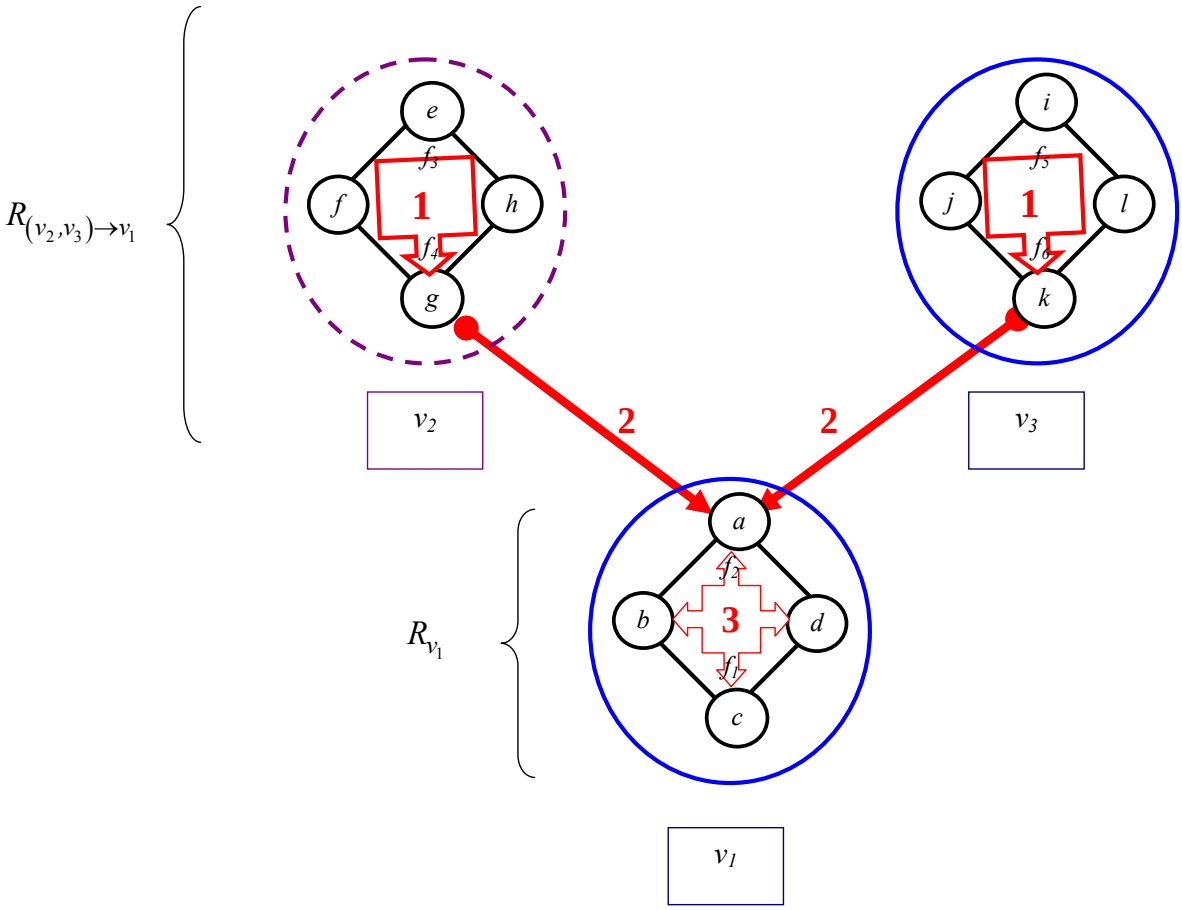


Figure E.5 : illustration des 3 étapes de calcul pour rendre consistant v_1 connecté à deux autres vecteurs par une contrainte ternaire. ■

Preuve :

■ La démonstration de cette proposition se fera par la preuve que l'inclusion dans les deux sens est vraie :

5. Pour tous les indices k dans v_1, montrons que $\prod_k(S) \subset \prod_k\left(S'_{v_1}\right)$

   Cette inclusion est évidente puisque, pour obtenir $\prod_k\left(S'_{v_1}\right)$, on contracte consécutivement des variables du CSP $\mathcal{H}$ avec des contraintes qui sont vérifiées par les solutions S. Donc ces solutions sont conservées.

6. Pour tous les indices k dans v_1, montrons que $\prod_k\left(S'_{v_1}\right) \subset \prod_k(S)$

   Pour plus de clarté, on suppose que les premiers indices sont rangés dans v_1 et le reste dans v_2.

Soit $\tilde{x}_k \in \prod_k \left(S'_{v_1}\right)$. Par hypothèse, S'_{v_1} est la solution de la restriction $\mathcal{H}_{v_1}$ avec un domaine initial qui est D'_{v_1}

$\Rightarrow \forall\, i \in \{1, \ldots, n_1\}\text{-}\{k\}, \exists\, \tilde{x}_i \in \mathrm{D}'_{v_1,i} \mid \mathrm{F}_{v_1}\left(\tilde{x}_1, \cdots, \tilde{x}_{n_1}\right) = 0$.

De plus, pour $i = j$, on a $\tilde{x}_i \in \mathrm{D}'_{v_1,i} = \prod_{f_{j,l,r},j}\left(\mathrm{D}'_x\right)$. Notons alors la variable v_4=v_2×v_3×v_1(j). Remarquons que la contrainte $f_{j,l,r}$ ne concerne, dans $\mathcal{H}_V$, que les trois variables dans les indices j, l et r qui appartiennent aussi au vecteur v_4. On peut donc écrire $\prod_{f_{j,l,r},j}\left(\mathrm{D}'_x\right) = \prod_{f_{j,l,r},j}\left(\mathrm{D}'_{v_4}\right)$ où D'_{v_4} est la restriction du domaine D'_x sur le vecteur de variables v_4. Ainsi donc, en considérant les CSP $\mathcal{H}_{v_2}$ et $\mathcal{H}_{v_3}$, et $v_1(i)$, on est dans les conditions du lemme précédent (section E.1.1) $\Rightarrow$ la projection $\prod_{f_{j,l,r},j}\left(\mathrm{D}'_{v_4}\right)$ permet d'obtenir les $\tilde{x}_j$ consistants avec les CSP $\mathcal{H}_{v_2}$ et $\mathcal{H}_{v_3}$. D'où l'existence d'un $\tilde{v}_2 \in [v_2]$ et d'un $\tilde{v}_3 \in [v_3]$ tel que $\mathrm{F}_{v_4}\left(\tilde{x}_i, \tilde{v}_2, \tilde{v}_3\right) = 0$.

D'où le n-uplet $\left(\tilde{x}_1, \cdots, \tilde{x}_{n_1}\right) \times \tilde{v}_2 \times \tilde{v}_3$ vérifie : $\mathrm{F}_{v_1}\left(\tilde{x}_1, \cdots, \tilde{x}_{n_1}\right) = 0$ et $\mathrm{F}_{v_4}\left(\tilde{x}_i, \tilde{v}_2, \tilde{v}_3\right) = 0$. Or, F_{v_1} regroupe toutes les contraintes dans la variable v_1 et $\mathrm{F}_{v_4}\left(\tilde{x}_i, \tilde{v}_2, \tilde{v}_3\right) = 0$ regroupe toutes les contraintes dans les variable v_2 et v_3, plus la seule contrainte concernant des variables dans v_1 et v_2, d'où la conclusion que le n-uplet $\left(\tilde{x}_1, \cdots, \tilde{x}_{n_1}\right) \times \tilde{v}_2 \times \tilde{v}_3$ vérifie $\mathrm{F}\left(\left(\tilde{x}_1, \cdots, \tilde{x}_{n_1}\right) \times \tilde{v}_2 \times \tilde{v}_3\right) = 0$

$\Rightarrow \mathrm{F}\left(\left(\tilde{x}_1, \cdots, \tilde{x}_{n_1}\right) \times \tilde{v}_2 \times \tilde{v}_3\right) = 0 \in \mathrm{S} \Rightarrow \tilde{x}_k \in \prod_k(S) \Rightarrow \prod_k\left(S'_{v_1}\right) \subset \prod_k(S)$.■

E.1.3 Démonstration du théorème de propagation formulé pour des contraintes ternaires

Posons $\mathcal{V}_a = \left(v_{a_1},\cdots,v_{a_q}\right)$ $\mathcal{V}_b = \left(v_{b_1},\cdots,v_{b_r}\right)$, $\mathcal{V}_c = \left(v_{c_1},\cdots,v_{c_s}\right)$ et soient $\mathcal{H}_{v_a}$, $\mathcal{H}_{v_b}$ et $\mathcal{H}_{v_c}$ trois VCSP avec des vecteurs de variables distincts i.e. $v_{a,i} \neq v_{b,j} \neq v_{c,k}$ pour tout 3-uplet (i, j, k) possible. Supposons que ces trois VCSP soient deux à deux connectés par une branche, de par une contrainte ternaire f_{a_1,b_1,c_1} reliant les trois vecteurs de variables v_{a_1}, v_{b_1} et v_{c_1}. Le nouveau VCSP $\mathcal{H}_{\mathcal{V}}$, qui regroupe ces trois variables et leurs domaines respectifs et pour lequel l'ensemble de contraintes regroupe les contraintes dans $\mathcal{H}_{v_a}$, $\mathcal{H}_{v_b}$ et $\mathcal{H}_{v_c}$ plus la contrainte f_{a_1,b_1,c_1} vérifie :

Si v_{a_1} est $\mathcal{H}_{v_a}$-consistant, v_{b_1} est $\mathcal{H}_{v_b}$-consistant et v_{c_1} est $\mathcal{H}_{v_c}$-consistant alors :

$$\prod_{v_{a_1}}(S) = R_{v_{a_1}} \circ R_{\left(v_{b_1},v_{c_1}\right)\to v_{a_1}}\left(D_{v_{a_1}}\right)$$

Preuve :

■ La démonstration de cette égalité se fera par la preuve que l'inclusion dans les deux sens est vraie :

7. On a $\prod_{v_{a_1}}(S) \subset R_{v_{a_1}} \circ R_{\left(v_{b_1},v_{c_1}\right)\to v_{a_1}}\left(D_{v_{a_1}}\right)$ de façon triviale puisque les solutions du CSP $\mathcal{H}_{\mathcal{V}}$, par définition, satisfont à tous les opérateurs de contraction.

8. Notons (j, *l, r*) l'indice des variables, respectivement, dans v_{a_1}, v_{b_1} et v_{c_1} concernées par la contrainte f_{a_1,b_1,c_1}. Notons également $v_1 = v_{a_1}$, $v_2 = \left(v_{b_1},\cdots,v_{b_r}\right)$, $v_3 = \left(v_{c_1},\cdots,v_{c_s}\right)$ et $v_4 = \left(v_1 \times v_2 \times v_3\right)$. On cherche à prouver, dans un premier temps, que $R_{v_{a_1}} \circ R_{(v_2,v_3)\to v_{a_1}}$ est équivalent à $R_{v_{a_1}} \circ R_{\left(v_{b_1},v_{c_1}\right)\to v_{a_1}}$, en d'autres termes que le sous-résolveur $R_{v_{a_1}} \circ R_{\left(v_{b_1},v_{c_1}\right)\to v_{a_1}}$ rend le vecteur v_{a_1} consistant avec tous les vecteurs dans $\mathcal{V}_b$ et dans $\mathcal{V}_c$.

   Le triplet de vecteurs de variables $\left(v_1, v_2, v_3\right)$ ne sont reliées, entre eux, que par la contrainte f_{a_1,b_1,c_1} et sont donc connectés par une branche (car f_{a_1,b_1,c_1} est une connexion à

une branche entre les trois vecteurs v_{a_1}, v_{b_1} et v_{c_1}). D'après la proposition dans la section E.1.2, pour tout indice *k* dans le vecteur v_1, on peut écrire :

$$\prod_k \left(S_{v_4}\right) = \prod_k \left(S'_{v_1}\right) \tag{E.10}$$

avec S_{v_4} est la solution du CSP $\mathcal{H}_{v_4}$, et S'_{v_1} est la solution de la restriction $\mathcal{H}_{v_1}$ avec un domaine initial $\mathrm{D}'_{v_1} = \left(\mathrm{D}'_{v_1,1}, \cdots, \mathrm{D}'_{v_1,n_1}\right)$ qui vérifie :

- Pour $k = j$, $\mathrm{D}'_{v_1,k} = \prod_{f_{j,l,r},j}\left(\mathrm{D}'_{v_4}\right)$
- Pour les *k* restants dans v_1, $\mathrm{D}'_{v_1,k} = \mathrm{D}_k$

Où D'_{v_4} est une contraction du domaine associé à v_4 qui vérifie :

- Pour $k = l$, $\mathrm{D}'_{v_4,k} = \prod_l \left(S_{v_2}\right)$
- Pour $k = r$, $\mathrm{D}'_{v_4,k} = \prod_r \left(S_{v_3}\right)$
- Pour les *k* restants dans v_4, $\mathrm{D}'_{v_4,k} = \mathrm{D}_k$

Or, par hypothèse, la variable v_{b_1} est $\mathcal{H}_{v_b}$-consistante donc en particulier pour $k=r$ on a $\mathrm{D}'_{v_4,k} = \prod_l \left(S_{v_2}\right) = \mathrm{D}_l$, de même pour $k=r$, du fait de la $\mathcal{H}_{v_c}$-consistance de v_{c_1}, on a $\mathrm{D}'_{v_4,k} = \prod_r (S_{v_2}) = \mathrm{D}_r$.

D'où D'_{v_4} est exactement égale au domaine initial de v_4 i.e. D'_{v_4} vérifie $\mathrm{D}'_{v_4,k} = \mathrm{D}_k$ pour tout les indices *k* dans le vecteur v_4.

De même, posons $v_5 = v_{b_1}$, $v_6 = v_{c_1}$ et $v_7 = v_{a_1} \times v_{b_1} \times v_{c_1}$. En appliquant la proposition dans la section E.1.2 une nouvelle fois au triplet de vecteurs (v_1, v_5, v_6) (qui, précise-t-on, correspond au triplet de vecteurs $\left(v_{a_1}, v_{b_1}, v_{c_1}\right)$), on peut écrire pour tout indice *k* dans v_1 :

$$\prod_k \left(S_{v_7}\right) = \prod_k \left(S''_{v_1}\right) \tag{E.11}$$

Avec S_{v_7} est la solution du CSP $\mathcal{H}_{v_7}$, et S''_{v_1} est la solution de la restriction $\mathcal{H}_{v_1}$ avec un domaine $\mathrm{D}''_{v_1} = \left(\mathrm{D}''_{v_1,1}, \cdots, \mathrm{D}''_{v_1,n_1}\right)$ qui est une contraction du domaine initial de v_1 qui vérifie :

- Pour $k = j$, $\mathrm{D}''_{v_1,k} = \prod_{f_{j,l,r},j}\left(\mathrm{D}''_{v_7}\right)$
- Pour les k restants dans v_1, $\mathrm{D}''_{v_1,k} = \mathrm{D}_k$

Où D''_{v_7} est une contraction du domaine associé à v_7 qui vérifie :

- Pour $k = l$, $\mathrm{D}''_{v_7,k} = \prod_l (S_{v_5})$
- Pour $k = r$, $\mathrm{D}''_{v_7,k} = \prod_r (S_{v_6})$
- Pour les k restants dans v_7, $\mathrm{D}''_{v_7,k} = \mathrm{D}_k$

Or, par hypothèse, la variable $v_5 = v_{b_1}$ est $\mathcal{H}_{v_b}$-consistante donc en particulier v_5 est aussi $\mathcal{H}_{v_b}$-consistante (en d'autre termes, le vecteur de variable v_{b_1} est résolu) . Ainsi, pour $k=l$ on a $\mathrm{D}''_{v_7,k} = \prod_l (S_{v_5}) = \mathrm{D}_l$. De façon analogue, on peut écrire pour $k=r$, $\mathrm{D}''_{v_7,k} = \prod_r (S_{v_6}) = \mathrm{D}_r$ d'où D''_{v_7} est exactement égale au domaine initial de v_7 i.e. D''_{v_7} vérifie $\mathrm{D}''_{v_7,k} = \mathrm{D}_k$ pour tout les indices k dans le vecteur v_7. Il s'avère alors que $\mathrm{D}''_{v_1} = \mathrm{D}'_{v_1}$ puisque ces deux domaines sont le domaine initial de v_1 contracté grâce à la contrainte f_{a_1,b_1,c_1} et les domaine D_r et D_l. Par suite on peut en conclure que $S'_{v_1} = S''_{v_1}$ puisqu'ils sont, tous les deux, définis comme les solutions de la restriction $\mathcal{H}_{v_1}$ avec respectivement les domaines égaux $\mathrm{D}''_{v_1} = \mathrm{D}'_{v_1}$. En considérant les égalités E.10 et E.11 on obtient pour tout indice k dans $v_1 = v_{a_1}$:

$$\prod_k \left(S_{v_4}\right) = \prod_k \left(S'_{v_1}\right) = \prod_k \left(S''_{v_1}\right) = \prod_k \left(S_{v_7}\right) \qquad \text{(E.12)}$$

L'égalité E.12 prouve que $R_{v_{a_1}} \circ R_{(v_2,v_3)\to v_{a_1}}$ est équivalent à $R_{v_{a_1}} \circ R_{\left(v_{b_1},v_{c_1}\right)\to v_{a_1}}$, ce qui signifie également que le sous-résolveur $R_{v_{a_1}} \circ R_{\left(v_{b_1},v_{c_1}\right)\to v_{a_1}}$ qui rend le vecteur v_{a_1}

consistant avec le vecteur v_{b_1} et v_{c_1}, le rend consistant avec les vecteur $v_2 = \left(v_{b_1},\cdots,v_{b_r}\right)$ et $v_3 = \left(v_{c_1},\cdots,v_{c_s}\right)$.

A présent, sachant que $R_{v_{a_1}} \circ R_{(v_2,v_3)\to v_{a_1}}$ est équivalent à $R_{v_{a_1}} \circ R_{\left(v_{b_1},v_{c_1}\right)\to v_{a_1}}$, montrons que $R_{v_{a_1}} \circ R_{\left(v_{b_1},v_{c_1}\right)\to v_{a_1}}\left(D_{v_{a_1}}\right) \subset \prod_{v_{a_1}}(S)$.

Soit $\tilde{v}_{a_1} \in R_{v_{a_1}} \circ R_{\left(v_{b_1},v_{c_1}\right)\to v_{a_1}}\left(D_{v_{a_1}}\right) = R_{v_{a_1}} \circ R_{(v_2,v_3)\to v_{a_1}} \Rightarrow \quad \exists \tilde{v}_{b_1} \in \mathrm{D}_{b_1},\cdots,\exists \tilde{v}_{b_r} \in \mathrm{D}_{b_r}$ et $\exists \tilde{v}_{c_1} \in \mathrm{D}_{c_1},\cdots,\exists \tilde{v}_{c_s} \in \mathrm{D}_{c_s} \mid \quad \tilde{v}_{a_1} \times \tilde{v}_{b_1} \times \cdots \times \tilde{v}_{b_r} \times \tilde{v}_{c_1} \times \cdots \times \tilde{v}_{c_s} \in S_{v_4}$ (rappelons que $v_4 = \left(v_{a_1} \times v_{b_1} \times \cdots \times v_{b_r} \times v_{c_1} \times \cdots \times v_{c_s}\right)$). De plus, par hypothèse, $\tilde{v}_{a_1}$ est $\mathcal{H}_{\mathcal{V}_a}$-consistant, donc $\exists \tilde{v}_{a_2} \in \mathrm{D}_{a_2},\cdots,\exists \tilde{v}_{a_q} \in \mathrm{D}_{a_q}$ tel que $\left(\tilde{v}_{a_1},\cdots,\tilde{v}_{a_q}\right)$ appartienne à $S_{\mathcal{V}_a}$. D'où le vecteur $\tilde{v}_{a_1} \times \cdots \times \tilde{v}_{a_q} \times \tilde{v}_{b_1} \times \cdots \times \tilde{v}_{b_r} \times \tilde{v}_{c_1} \times \cdots \times \tilde{v}_{c_s}$ vérifie toutes les contraintes dans $\mathcal{H}_{\mathcal{V}_a}$ et $\mathcal{H}_{\mathcal{V}_b}$ plus $f_{a_1,b_1,c_1} \quad \Rightarrow \quad \tilde{v}_{a_1} \times \cdots \times \tilde{v}_{a_q} \times \tilde{v}_{b_1} \times \cdots \times \tilde{v}_{b_r} \times \tilde{v}_{c_1} \times \cdots \times \tilde{v}_{c_s} \in S$. D'où, en particulier, $\tilde{v}_{a_1} \in \Pi_{v_{a_1}}(\mathrm{S}) \Rightarrow R_{v_{a_1}} \circ R_{\left(v_{b_1},v_{c_1}\right)\to v_{a_1}}\left(D_{v_{a_1}}\right) \subset \prod_{v_{a_1}}(S)$.■

E.2 Démonstration de l'algorithme FALL/CLIMB

L'objectif de ce paragraphe est de détailler la preuve de l'algorithme FALL/CLIMB présentée dans le chapitre 5. Rappelons que la base de cet algorithme est le théorème de propagation et son corollaire qui est le théorème de retro-propagation. Cette partie est placée dans l'annexe afin de donner, aux lecteurs désireux, toutes les étapes précises permettant de démontrer l'algorithme FALL/CLIMB. Précisons que, à part quelques détails propres à la généralisation des SCSP en VCSP, le cheminement qui sera rapporté ici est tiré de l'article [Jaulin et al., 2001b].

E.2.1 Théorèmes de propagation et de retropropagation

Dans cette partie on rappelle les théorèmes de propagation donnés dans le chapitre 5 (pour les contraintes binaires et les contraintes ternaires. On écrit, également, les théorème de retro propagation dans leurs versions pour les VCSP (contraintes binaires et ternaires).

Théorème de propagation : formulation pour des contraintes binaires

Posons $\mathcal{V}_a = \left(v_{a_1}, \cdots, v_{a_q}\right)$ et $\mathcal{V}_b = \left(v_{b_1}, \cdots, v_{b_r}\right)$, et soient $\mathcal{H}_{\mathcal{V}_a}$ et $\mathcal{H}_{\mathcal{V}_b}$ deux VCSP avec des vecteurs de variables distincts i.e. $v_{a,j} \neq v_{b,i}$ pour tout couple possible (*i, j*). On suppose que $\mathcal{H}_{\mathcal{V}_a}$ et $\mathcal{H}_{\mathcal{V}_b}$ sont connectées par une branche de par deux vecteurs de variables v_{a_1} et v_{b_1} et de par une contrainte notée f_{a_1,b_1}. Le nouveau VCSP $\mathcal{H}_{\mathcal{V}}$, où $\mathcal{V} = \mathcal{V}_a \bigcup \mathcal{V}_b$ avec les domaines $\mathrm{D}_a \cup \mathrm{D}_b$ et pour lequel, l'ensemble de ses contraintes regroupe les contraintes dans $\mathcal{H}_{\mathcal{V}_a}$ et $\mathcal{H}_{\mathcal{V}_b}$ plus la contrainte f_{a_1,b_1} vérifie :

Si v_{a_1} est $\mathcal{H}_{\mathcal{V}_a}$-consistant et v_{b_1} est $\mathcal{H}_{\mathcal{V}_b}$-consistant alors :

$\Pi_{v_{b_1}}(\mathrm{S}) = R_{v_{b_1}} \circ R_{v_{a_1} \to v_{b_1}}\left(\mathrm{D}_{v_{b_1}}\right)$ et $\Pi_{v_{a_1}}(\mathrm{S}) = R_{v_{a_1}} \circ R_{v_{b_1} \to v_{a_1}}\left(\mathrm{D}_{v_{a_1}}\right)$.Où :

- S est la solution du CSP $\mathcal{H}_{\mathcal{V}}$ et $\Pi_{v_{b_1}}(\mathrm{S})$ est la projection de S sur l'espace de v_{b_1},
- $R_{v_{b_1}} \circ R_{v_{a_1} \to v_{b_1}}$ et $R_{v_{a_1}} \circ R_{v_{b_1} \to v_{a_1}}$ sont, respectivement, les contracteurs qui permettent de rendre consistants v_{b_1} sachant v_{a_1} et v_{a_1} sachant v_{b_1}

La preuve de ce théorème est donnée dans la section 6 du chapitre 5. Il est appelé théorème de propagation parce qu'il sert essentiellement à démontrer l'algorithme FALL.

Théorème de propagation : formulation pour des contraintes ternaires

Posons $\mathcal{V}_a = \left(v_{a_1},\cdots,v_{a_q}\right)$ $\mathcal{V}_b = \left(v_{b_1},\cdots,v_{b_r}\right)$, $\mathcal{V}_c = \left(v_{c_1},\cdots,v_{c_s}\right)$ et soient $\mathcal{H}_{v_a}$, $\mathcal{H}_{v_b}$ et $\mathcal{H}_{v_c}$ trois VCSP avec des vecteurs de variables distincts i.e. $v_{a,i} \neq v_{b,j} \neq v_{c,k}$ pour tout 3-uplet (i, j, k) possible. Supposons que ces trois VCSP soient deux à deux connectés par une branche, de par une contrainte ternaire f_{a_1,b_1,c_1} reliant les trois vecteurs de variables v_{a_1}, v_{b_1} et v_{c_1}. Le nouveau VCSP $\mathcal{H}_{\mathcal{V}}$, qui regroupe ces trois variables et leurs domaines respectifs et pour lequel l'ensemble de contraintes regroupe les contraintes dans $\mathcal{H}_{v_a}$, $\mathcal{H}_{v_b}$ et $\mathcal{H}_{v_c}$ plus la contrainte f_{a_1,b_1,c_1} vérifie :

Si v_{b_1} est $\mathcal{H}_{\mathcal{V}_b}$-consistant et v_{c_1} est $\mathcal{H}_{v_c}$-consistant alors :

$$\prod_{v_{a_1}}(S) = R_{v_{a_1}} \circ R_{\left(v_{b_1},v_{c_1}\right)\to v_{a_1}}\left(D_{v_{a_1}}\right)$$

La preuve de ce théorème est donnée dans la section E.1.3.

Le deuxième théorème qui est celui de retro-propagation sert, à son tour, à démontrer l'algorithme CLIMB. Il découle directement du théorème de propagation. On donne ci-après sa formulation pour les VCSP ainsi que sa preuve (pour les contraintes binaires et les contraintes ternaires).

Théorème de retro-propagation pour les VCSP :formulation pour les contraintes binaires

Posons $\mathcal{V}_a = \left(v_{a_1},\cdots,v_{a_q}\right)$ et $\mathcal{V}_b = \left(v_{b_1},\cdots,v_{b_r}\right)$, et soient $\mathcal{H}_{\mathcal{V}_a}$ et $\mathcal{H}_{\mathcal{V}_b}$ deux VCSP avec des vecteurs de variables distincts i.e. $v_{a,j} \neq v_{b,i}$ pour tout couple possible (i, j). Soient v_{a_1} et v_{b_1} deux vecteurs de variables respectivement dans $\mathcal{H}_{\mathcal{V}_a}$ et $\mathcal{H}_{\mathcal{V}_b}$ qui sont connectés par une branche par une contrainte notée f_{a_1,b_1}. Le nouveau VCSP $\mathcal{H}_{\mathcal{V}}$, où $\mathcal{V} = \mathcal{V}_a \bigcup \mathcal{V}_b$ avec leurs domaines respectifs dans $\mathcal{H}_{\mathcal{V}_a}$ et $\mathcal{H}_{\mathcal{V}_b}$ et pour lequel, l'ensemble de ses contraintes regroupe les contraintes dans $\mathcal{H}_{\mathcal{V}_a}$ et $\mathcal{H}_{\mathcal{V}_b}$ plus la contrainte f_{a_1,b_1} vérifie :

Si v_{a_1} est $\mathcal{H}_{\mathcal{V}}$-consistant et v_{b_1} est $\mathcal{H}_{\mathcal{V}_b}$-consistant alors :

$$\Pi_{v_{b_1}}(\mathrm{S}) = R_{v_{b_1}} \circ R_{v_{a_1}\to v_{b_1}}\left(\mathrm{D}_{v_{b_1}}\right)$$

Preuve :

■ $\mathcal{H}_{\mathcal{V}_a}$ est un sous CSP du CSP $\mathcal{H}_{\mathcal{V}}$, on a donc v_{a_1} $\mathcal{H}_{\mathcal{V}}$-consistant $\Rightarrow$ v_{a_1} $\mathcal{H}_{\mathcal{V}_a}$-consistant. Les deux CSP $\mathcal{H}_{\mathcal{V}_a}$ et $\mathcal{H}_{\mathcal{V}_b}$ et les deux variables v_{a_1} et v_{b_1} remplissent les conditions du théorème de propagation pour les VCSP. En particulier, on peut écrire

$$\Pi_{v_{b_1}}(\mathrm{S}) = R_{v_{b_1}} \circ R_{v_{a_1} \to v_{b_1}} \left(\mathrm{D}_{v_{b_1}} \right) \blacksquare$$

Théorème de retro-propagation pour les VCSP :formulation pour les contraintes ternaires

Posons $\mathcal{V}_a = \left(v_{a_1}, \cdots, v_{a_q} \right)$ $\mathcal{V}_b = \left(v_{b_1}, \cdots, v_{b_r} \right)$, $\mathcal{V}_c = \left(v_{c_1}, \cdots, v_{c_s} \right)$ et soient $\mathcal{H}_{v_a}$, $\mathcal{H}_{v_b}$ et $\mathcal{H}_{v_c}$ trois VCSP avec des vecteurs de variables distincts i.e. $v_{a,i} \neq v_{b,j} \neq v_{c,k}$ pour tout 3-uplet (i, j, k) possible. Supposons que ces trois VCSP soient deux à deux connectés par une branche, de par une contrainte ternaire f_{a_1,b_1,c_1} reliant les trois vecteurs de variables v_{a_1}, v_{b_1} et v_{c_1}. Le nouveau VCSP $\mathcal{H}_{\mathcal{V}}$, qui regroupe ces trois variables et leurs domaines respectifs et pour lequel l'ensemble de contraintes regroupe les contraintes dans $\mathcal{H}_{v_a}$, $\mathcal{H}_{v_b}$ et $\mathcal{H}_{v_c}$ plus la contrainte f_{a_1,b_1,c_1} vérifie :

Si v_{a_1} est $\mathcal{H}_{\mathcal{V}}$–consistant, v_{b_1} est $\mathcal{H}_{\mathcal{V}_b}$-consistant et v_{c_1} est $\mathcal{H}_{\mathcal{V}_c}$-consistant alors :

$$\prod_{v_{b_1}}(S) = R_{v_{b_1}} \circ R_{\left(v_{a_1}, v_{c_1}\right) \to v_{b_1}} \left(D_{v_{b_1}} \right) \quad \prod_{v_{c_1}}(S) = R_{v_{c_1}} \circ R_{\left(v_{b_1}, v_{a_1}\right) \to v_{c_1}} \left(D_{v_{c_1}} \right)$$

■ $\mathcal{H}_{\mathcal{V}_a}$ est un sous CSP du CSP $\mathcal{H}_{\mathcal{V}}$, on a donc v_{a_1} $\mathcal{H}_{\mathcal{V}}$-consistant $\Rightarrow$ v_{a_1} $\mathcal{H}_{\mathcal{V}_a}$-consistant. Les trois CSP $\mathcal{H}_{v_a}$, $\mathcal{H}_{v_b}$ et $\mathcal{H}_{v_c}$ et les vecteurs v_{a_1}, v_{b_1} et v_{c_1} remplissent les conditions du théorème de propagation pour les VCSP. En particulier, on peut écrire

$$\prod_{v_{b_1}}(S) = R_{v_{b_1}} \circ R_{\left(v_{a_1}, v_{c_1}\right) \to v_{b_1}} \left(D_{v_{b_1}} \right) \text{ et } \prod_{v_{c_1}}(S) = R_{v_{c_1}} \circ R_{\left(v_{b_1}, v_{a_1}\right) \to v_{c_1}} \left(D_{v_{c_1}} \right). \blacksquare$$

E.2.2 Définitions sur les Tree VCSP

On rappelle qu'un TVCSP (Tree Vectorised Constraint Satisfaction Constraint) est un VCSP ayant une structure d'arbre. Dans la partie C de l'annexe, on a rappelé quelques définitions classiques sur les graphes. On rapporte ici, par souci de clarté, les quelques définitions classiques, réadaptées aux VCSP, dérivant de la théorie sur les arbres :

- On peut prendre l'origine d'un arbre, communément appelé racine, en choisissant un nœud quelconque. Pour un TVCSP $\mathcal{H}_{\mathcal{V}}$: (($\tilde{x} \in \mathrm{S}_{F_J}$) et ($\tilde{x}_{J_i} \in \mathrm{S}_{v_i} \forall\, i = 1 \ldots m$), $\tilde{x} \in [x]$), une racine correspondra à une restriction $\mathcal{H}_{v_i}$ sur un des vecteurs choisi dans $\mathcal{V} = \{v_1, \cdots v_m\}$.
- $(\mathcal{H}_{\mathcal{V}}, v_i)$ signifie que l'on choisit le vecteur v_i comme étant la racine du TVCSP $\mathcal{H}_{\mathcal{V}}$.
- On considérant un arbre $\mathcal{H}_{\mathcal{V}}$ de racine v_i et $v_1 \times \cdots \times v_q$ les variables reliées à v_i, en supprimant v_i, on obtient q nouveaux arbres. Chacun de ces arbres est par définition un sous-arbre de l'arbre $\mathcal{H}_{\mathcal{V}}$ que l'on note $\mathcal{H}_{\mathcal{V},k}$ avec k variant de *1* à q. En procédant de la sorte, pour chaque nœud correspond un arbre qui rassemble ses descendants.

E.2.3 Algorithmes FALL et CLIMB

On rappelle ici les deux algorithmes FALL et CLIMB, applicables à des TVCSP à connexité à une branche, ainsi que leurs preuves. FALL parcourt le TVCSP des feuilles aux racines en se basant sur le théorème de propagation. CLIMB, à son tour, parcourt le TVCSP de la racine aux feuilles en se basant sur le théorème de retro-propagation. On peut alors prouver qu'une exécution de FALL suivie d'une exécution de CLIMB suffit à atteindre la consistance optimale du TVCSP.

E.2.4 Algorithme FALL

Algorithme FALL : Input($\mathcal{H}_{\mathcal{V}}$, v_i), Output($\mathcal{H}_{\mathcal{V}}$, v_i)

DEBUT

SI $\mathcal{H}_{\mathcal{V}}$ est une feuille, ALORS retourner $\mathcal{H}_{\mathcal{V}}$ FIN

$v_{racine} = v_i$; $v_1, \cdots, v_q$ = vecteurs liés à v_{racine} par une contrainte binaire

$(v_{11}, v_{12}), \cdots, (v_{r1}, v_{r2})$ = couples de vecteurs liés à v_{racine} par une contrainte ternaire

% contraintes binaires

POUR $k = 1$ à q, FAIRE

FALL($\mathcal{H}_{\mathcal{V},k}$, v_k)

$D_{v_{racine}} = R_{v_k \to v_{racine}} \left(D_{v_{racine}} \right)$

FIN

% contraintes ternaires

POUR $k = 1$ à r, FAIRE

FALL($\mathcal{H}_{\mathcal{V},k}$, $v_{k,1}$)

FALL($\mathcal{H}_{\mathcal{V},k}$, $v_{k,2}$)

$D_{v_{racine}} = R_{(v_{k,1}, v_{k,2}) \to v_{racine}} \left(D_{v_{racine}} \right)$

FIN

$D_{v_{racine}} = R_{v_{racine}} \left(D_{v_{racine}} \right)$

FIN

Tableau E.7 : algorithme FALL écrit pour les TVCSP avec une connexité à une branche.

Considérons un TVCSP $(\mathcal{H}_{\mathcal{V}}, v_i)$ à connexité à une branche, et v_k un élément une composante dans $\mathcal{V}$. Définissons la notion de "haute-consistance" pour un vecteur de variable : v_k est dit haute-consistance s'il est consistant avec le VCSP $\mathcal{H}_{\mathcal{V},k}$ (sous arbre de $\mathcal{H}_{\mathcal{V}}$ construit à partir de $\mathcal{H}_{\mathcal{V},k}$). On dit alors de $\mathcal{H}_{\mathcal{V}}$ qu'il est haute-consistance si tous ses

nœuds sont sur consistants dans $\mathcal{H}_{\mathcal{V}}$. La notion de haute-consistance a été introduite pour permettre de justifier le niveau de consistance atteint par L'algorithme de FALL présenté dans le Tableau E.7. On peut ainsi annoncer le théorème suivant :

FALL appliqué à un TVSCP à connexité à une branche $\mathcal{H}_{\mathcal{V}}$ quelconque permet d'atteindre la haute-consistance.

Preuve

■ L'algorithme FALL est récursif et revient à propager la contraction des feuilles vers la racine. La preuve se fait par récurrence en partant des feuilles qui sont trivialement haute-consistances. Démontrer FALL revient alors à démontrer l'assertion suivante : En supposant que pour un nœud donné v_i, tous ses m vecteurs descendants v_k sont haute-consistants alors pour rendre le nœud v_i haute-consistants, il suffit de le contracter avec ses descendant en utilisant les sous résolveurs $R_{(v_{k,1},v_{k,2})\to v_{racine}}$ pour les contraintes binaires et les sous résolveurs $R_{(v_{k,1},v_{k,2})\to v_{racine}}$ pour les contraintes ternaires.

Démontrons alors cette assertion. Considérons d'abord les contraintes q binaires reliant v_i à ses q descendants $v_1,\cdots,v_q$. Posons pour tout k, de 1 à q, $\mathcal{H}_{\mathcal{V}}(k)$ le VCSP regroupant par v_i et les sous-arbres $\mathcal{H}_{\mathcal{V},1},\ldots,\mathcal{H}_{\mathcal{V},k}$ (généré à partir de $v_1,\cdots,v_k$). Pour k=1 et en appliquant le théorème de propagation, $D_{v_i} = R_{v_1\to v_i}\left(D_{v_i}\right)$ rend v_i est $\mathcal{H}_{\mathcal{V}}(1)$-consistant . En supposant que pour une valeur de $s < q$, v_i est $\mathcal{H}_{\mathcal{V}}(s)$-consistant et en considérant le CSP $\mathcal{H}_{\mathcal{V}}(s+1)$, on peut à nouveau appliquer le théorème de propagation pour obtenir que $D_{v_i} = R_{v_{r+1}\to v_i}\left(D_{v_i}\right)$ rend v_i $\mathcal{H}_{\mathcal{V}}(r+1)$-consistant. Ainsi donc par une récurrence sur les $\mathcal{H}_{\mathcal{V},k}$, on montre que v_i est $\mathcal{H}_{\mathcal{V}}(q)$-consistant.

Par un raisonnement analogue pour les contraintes ternaires en utilisant, cette fois ci, des sous résolveurs ayant la forme $R_{(v_{k,1},v_{k,2})\to v_{racine}}$, on montre que v_i est consistant avec les couples de descendants restants. Ceci prouve l'assertion et donc que FALL rend un VSCP à connexité à une branche Haut-consistant. ■

Algorithme CLIMB : Input($\mathcal{H}_{\mathcal{V}}$, v_i), Output([$\mathcal{H}_{\mathcal{V}}$, v_i])

DEBUT

SI $\mathcal{H}_{\mathcal{V}}$ est une feuille, ALORS retourner $\mathcal{H}_{\mathcal{V}}$ FIN

$v_{racine} = v_i$; $v_1, \cdots, v_q$ = vecteurs liés à v_{racine} par une contrainte binaire

$(v_{11}, v_{12}), \cdots, (v_{r1}, v_{r2})$ = couples de vecteurs liés à v_{racine} par une contrainte ternaire

% contraintes binaires

POUR $k = 1$ à q, FAIRE

$D_{v_k} = R_{v_k} \circ R_{v_{racine} \to v_k}\left(D_{v_k}\right)$

CLIMB($\mathcal{H}_{\mathcal{V},k}$, v_k)

FIN

% contraintes ternaires

POUR $k = 1$ à r, FAIRE

$D_{v_{k,1}} = R_{(v_{racine}, v_{k,2}) \to v_{k,1}}\left(D_{v_{k,1}}\right)$

$D_{v_{k,2}} = R_{(v_{racine}, v_{k,1}) \to v_{k,2}}\left(D_{v_{k,2}}\right)$

CLIMB($\mathcal{H}_{\mathcal{V},k}$, $v_{k,1}$)

CLIMB($\mathcal{H}_{\mathcal{V},k}$, $v_{k,2}$)

FIN

FIN

Tableau E.8 : algorithme CLIMB écrit pour les TVCSP avec une connexité à une branche.

L'algorithme de CLIMB fonctionne en contractant le CSP du vecteur racine vers les vecteurs feuilles. On peut annoncer le théorème suivant :

CLIMB appliqué à un TVSCP à connexité à une branche $\mathcal{H}_{\mathcal{V}}$ haut-consistant permet d'obtenir la consistance globale.

Preuve

■ La preuve se fait également par récurrence en appliquant le théorème de retro-propagation. L'arbre étant haut-consistant, par hypothèse, la racine est consistante avec $\mathcal{H}_{\mathcal{V}}$. Pour chacun des descendants v_k de la racine connecté par une contrainte binaire, on a v_k est haute-consistant avec l'arbre $\mathcal{H}_{\mathcal{V},k}$ qui lui est associé. En appliquant le théorème de retro-propagation, et en utilisant le sous-résolveur $R_{v_k} \circ R_{v_{racine} \to v_k}$, on prouve que chacun des descendants v_k de la racine est $\mathcal{H}_{\mathcal{V}}$-consistant. Pour chaque couple descendants connecté par une contrainte ternaire, on fait un raisonnement analogue pour conclure que à leur $\mathcal{H}_{\mathcal{V}}$-consistance après l'exécution des sous-résolveurs $R_{(v_{racine}, v_{k,1}) \to v_{k,2}}$. Ce raisonnement fait sur tout les nœud vis-à-vis de ses descendant prouve que CLIMB est optimal pour tout TVCSP à connexité à une branche qui est haut-consistant. ■

Résumé :

L'objectif de cette thèse est d'appliquer, aux problèmes de fusion de données, les techniques de résolution des CSP (problème de satisfaction de contraintes) sur des pavés réels avec comme application réelle la localisation dynamique d'un véhicule à l'aide des capteurs ABS, d'un gyromètre, et d'un GPS différentiel.

Après avoir mené cette étude et tiré des conclusions, deux contributions sont faites pour améliorer les techniques de satisfaction de contraintes utilisées. Dans un premier temps, la notion de "*domaine de consistance*" est introduite avec des propriétés et résultats variés qui nous ont permis de résoudre de manière formelle des exemples de cycles. Grâce à cet outil, on améliore la précision des pavés obtenus tout en ayant une information précieuse du temps de calcul a priori. Dans un second temps, l'algorithme connu de FALL/CLIMB permettant de résoudre de façon optimale un CSP ayant une représentation graphique sous forme de graphe a été généralisé aux CSP vectorisés "*VCSP*".

Mots clefs :

Estimation d'état, Fusion de donnée, Analyse par intervalles, Erreur bornée, Techniques de consistance, Automobile, Localisation garantie.

Abstract :

The objective of this PhD thesis is to apply, for data fusion problems, solving techniques using CSP (Constraint Satisfaction Problem) on real boxes, with application to guaranteed dynamic localisation of car-like vehicle thanks to the fusion of odometers, a gyro and a differential GPS receiver.

After this study and its conclusions, two main contributions have been proposed to ameliorate usual constraints satisfaction techniques thanks to the introduction of "VCSP" and the "consistency domains" concepts. The goal is to solve cycles thanks to consistency domains, to isolate and to organise cycles appearing on constraints graphs in order to obtain optimal box domains with an a priori known calculation time.

Key words :

State estimation, Data fusion, Interval analysis, Bounded error, Consistency techniques, Automobile, Guaranteed localisation.