Couplage de la configuration de produit et de projet de réalisation : exploitation des approches par contraintes et des algorithmes évolutionnaires

Mériem Djefel

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
Institut National Polytechnique de Toulouse (INP Toulouse)

Discipline ou spécialité :

Systèmes industriels

Présentée et soutenue par :
Mériem Djefel

le : mercredi 17 novembre 2010

Titre :

Couplage de la configuration de produit et
de projet de réalisation : exploitation des approches
par contraintes et des algorithmes évolutionnaires

JURY

Pr. Abdelaziz Bouras (président)
Pr. Samuel Gomez (membre)
Dr. Thierry Coudert (membre)
Dr. Elise Vareilles (membre)

Ecole doctorale :
Systèmes (EDSYS)

Unité de recherche :
École des mines d'Albi-Carmaux - Centre Génie Industriel

Directeur(s) de Thèse :

M. Michel Aldanondo et Mme Claude Baron

Rapporteurs :

Dr. Éric Bonjour et Dr. Pierre-Alain Yvars

Remerciements

En premier lieu, je tient à remercier mes encadrants de thèse ; ces travaux de thèse ne serait pas aussi aboutis sans leurs conseils et leur disponiblité : Michel Aldanondo directeur de these, Claude Baron co-directrice et sans oublier Elise Vareilles mon encadrante. Je vous remercie de m'avoir accompagné et enrichi de vos connaissances durant ces trois années.

Je remercie les rapporteurs de cette thèse Eric Bonjour et Pierre Alain Yvars pour leurs commentaires et leurs questions qui m'ont permis de pousser ma réflexion plus loin. Je remercie également Samuel Gomes, Thierry Coudert et Abdelaziz Bouras de m'avoir fait l'honneur d'accepter de juger ces travaux de thèse.

Je remercie tous les membres de Centre Génie Industriel pour leur soutient et leur sympathie et plus particulièrement Isabelle.

Merci à tous mes amis qui m'ont soutenu, aider et encourager.

Je tient aussi à remercier toute ma famille, qui m'a soutenu pendant tout ce temps. Enfin je ne remercierai jamais assez ma mère qui a toujours su trouver les mots justes pour me poussez à étudier et à ne jamais baisser les bras.

Thanimirth Yemma Azzizen.

Table des matières

Introduction

Contexte et problématique

Dans le contexte actuel de compétitivité des marchés, la maîtrise et l'optimisation des processus de conception et de planification sont nécessaires pour garantir, d'une part la fiabilité et la qualité des produits systèmes ou services conçus et, d'autre part, le cycle de développement et les coûts. Ce constat impose de développer et d'améliorer les méthodes, modèles, techniques et outils relatifs aux processus de conception et de gestion ou planification.

Les travaux très nombreux relatifs à l'amélioration des processus de conception ou de développement ont permis d'élaborer des méthodes, modèles, techniques et outils d'aide à la décision en conception très évolués. Simultanément, des travaux se sont intéressés à la maîtrise des cycles de développement ainsi qu'à la conduite de projet et ont abouti au développement de méthodes, modèles, techniques et outils de gestion et planification de projet. L'analyse de l'état de l'art montre que ces travaux ont été conduits, pour la plupart, de manière indépendante. Les méthodes, modèles, techniques et outils en découlant sont en conséquence très faiblement reliés.

Étant admis que de nombreux choix de conception impactent fortement le projet et, qu'inversement, de nombreux choix de gestion impactent également l'objet à concevoir, il en résulte que des prises de décision de conception et de planification non concertées ou séquentielles conduisent fréquemment à des incohérences, des itérations voire des erreurs. Ces dysfonctionnements génèrent alors systématiquement des délais supplémentaires et des surcoûts.

Afin de remédier à ce constat, il semble incontournable d'essayer de conduire simultanément, de mettre en relation ou encore de faire interagir les activités liées à la conception de produits, de systèmes ou de services et les activités liées à la gestion du projet. Cette problématique est le centre d'intérêt d'un groupe de scientifiques et d'industriels du *Pôle Aerospace Valley*. Ce groupe travaille dans le cadre d'un projet appelé *ATLAS*, pour Aides et assisTances pour la conception, la conduite de projet et leur coupLage par les connAissanceS, projet *ANR/RNTL 2007*. Ce projet vise le développement d'un environnement d'aide à la décision en ingénierie couplant des outils interactifs d'aide à la conception de produit avec des outils d'aide à la planification de projet.

Objectifs de nos recherches

Les travaux présentés dans cette thèse s'inscrivent dans les objectifs du projet *ATLAS*. L'objectif d'*ATLAS* est donc de faire coopérer la conception et la conduite de projet en exploitant des connaissances méthodologiques et métier. Suivant la disponibilité de différents types de connaissance, *ATLAS* assistera, aussi bien la conception innovante voire créative que la conception routinière.

Notre objectif de recherche se limite à une conception très routinière que l'on appelle configuration, caractérisée par le fait que l'espace de solutions peut être analysé dans sa globalité. Nous parlerons donc de configuration de produit que nous ferons interagir avec de la configuration de son projet de réalisation ou production.

Dans ce cadre nos objectifs de recherche sont :

- de définir le couplage de la configuration de produits, systèmes ou services avec la configuration ou planification du projet de sa production,
- d'identifier les éléments de la configuration de produits, systèmes ou services et de la planification de projet de sa production qui peuvent être mis en relation et supporter le couplage,
- d'adapter, mettre en œuvre et évaluer l'intérêt des approches par propagation de contraintes pour aider interactivement et simultanément la configuration de produit et la planification du projet de sa production,
- d'adapter, mettre en œuvre et évaluer l'intérêt des algorithmes évolutionnaires pour optimiser selon des critères antagonistes, cycle/coût, l'espace des solutions de configuration et de planification,
- d'évaluer la faisabilité et l'intérêt d'associer les approches de propagation de contraintes et d'optimisation évolutionnaire.

Plan du document

Le chapitre 1 page 5, dresse un état de l'art des différents domaines auxquels nous serons confrontés :

- la conception et plus particulièrement la configuration, section 1.1 page 5,
- la gestion de projet et plus particulièrement à la planification des tâches d'un projet, en section 1.2 17,
- les travaux ayant associés la conception de produit et la gestion de projet, en section 1.3 page 21,
- les approches par contraintes et plus particulièrement les méthodes de propagation, en section 1.4.1 page 24,
- le terrain d'application relatif à nos travaux, en section 1.5 page 33.

Le chapitre 2 page 37, s'intéresse à la mise en œuvre du couplage interactif de la configuration de produit et de la planification du projet de réalisation. Le chapitre aborde successivement :

- la configuration du produit, en section 2.1 page 40,
- la planification du projet de sa production, en section 2.2 page 55,
- le couplage de la configuration et de la planification, en section 2.3 page 72.

Pour chaque partie, trois types de problèmes seront définis, formalisés et expérimentés :

- un problème élémentaire, en sous-section 2.1.1.1 page 40 pour la partie configuration, en sous-section 2.2.1 page 55 pour la partie planification, en sous-section 2.3.1 page 75 pour la partie couplage,
- un problème à éléments optionnels, en sous-section 2.1.2 page 43 pour la partie configuration, en sous-section 2.2.2.1 page 58 pour la partie planification, en sous-section 2.3.2 page 80 pour la partie couplage,

- un problème à éléments en exclusion, en sous-section 2.1.3 page 49 pour la partie configuration, en sous-section 2.2.3 page 62 pour la partie planification, en sous-section 2.3.3 page 86 pour la partie couplage,

Le formalisme fera appel à différents types d'approches par contraintes :

- problème à contraintes discrètes, numériques, mixtes et temporelles,
- problème à structure statique ou dynamique.

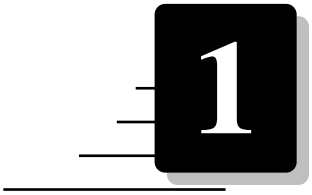
Un exemple fil rouge, correspondant à un avion de tourisme et d'affaires, illustre toutes nos propositions. Les expérimentations avec différents progiciels à base de contraintes sont menées et commentées, en sous-section 2.1.5 page 52 pour la partie configuration, en sous-section 2.2.5 page 67 pour la partie planification, en sous-section 2.3.4 page 92 pour la partie couplage.

Le chapitre 3 page 95, est consacré à l'optimisation du couplage configuration planification. Nous disposons alors d'un espace réduit de solutions cohérentes pour les problèmes de configuration et de planification. Nous proposons alors un moyen permettant à l'utilisateur de choisir parmi des solutions optimisant le compromis cycle/coût.

Ce chapitre aborde :

- un état de l'art succinct sur l'optimisation combinatoire et les approches évolutionnaires, en section 3.1 page 96,
- une proposition d'adaptation d'un algorithme évolutionnaire au problème contraint, en section 3.2 page 103,
- des expérimentations sur l'exemple fil rouge et une proposition relative à une utilisation itérative de la configuration par propagation de contraintes et de l'optimisation évolutionnaire, en section 3.3 page 107.

Une conclusion générale conclut ce mémoire de thèse, synthétise nos apports et propose des perspectives de recherche.



Cadre général des travaux et problématique

Introduction

Les travaux présentés dans ce mémoire s'intéressent donc aux problèmes découlant de l'association de la configuration de produit et de la planification de son processus de production. Ce chapitre pose le cadre général des travaux et notre problématique. Les deux premières sections dressent un état de l'art des domaines de la conception et de la configuration de produit, section 1.1, puis de la gestion et de la planification de projet, section 1.2. Ceci nous conduit à considérer leur association et/ou les relations entre eux pour déboucher sur la problématique de couplage, section 1.3. La section 1.4 introduit les approches par contraintes, qui serviront d'outil pour le couplage. La section 1.5 introduit la problématique industrielle support de nos travaux.

1.1 Conception et configuration de produit

La conception est une activité qui vise à transformer en objet réalisable un ensemble de besoins et d'exigences exprimés par un client. Chandrasekaran considère la conception comme une activité de résolution de problème dont les données d'entrée comprennent (Chandrasekaran, 1990) :

- la liste des fonctions que doit accomplir l'objet à concevoir ; ces fonctions sont celles exprimées par le client ainsi que celles qui sont implicitement définies par le domaine ;
- un ensemble de contraintes qui doivent être satisfaites par ce même objet ; les contraintes peuvent porter sur les paramètres de définition de l'objet ou sur son processus de réalisation ;
- un ensemble de composants prédéfinis et un ensemble de relations reliant ces différents composants.

Il définit la solution du problème de conception comme une spécification complète de l'ensemble des composants ainsi que les relations les reliant. Cette spécification décrit l'objet solution qui accomplit toutes les fonctions et vérifie toutes les contraintes. Le plus souvent, cette solution est choisie parmi un ensemble d'alternatives suivant différents critères d'évaluation (performance, coût, ...).

Suivant cette définition, l'activité de conception consiste à répondre à un ensemble de besoins et d'exigences décrites sous la forme de fonctions et de contraintes. La réponse est donnée sous la forme d'un ensemble de composants réalisant ces fonctions. La norme AFNOR (AFNOR, 1988) propose également une définition allant dans ce sens, mais fait apparaître la notion de connaissances nécessaires à la conception. La conception est définie comme une activité créative qui, partant des besoins exprimés et des connaissances existantes, aboutit à la définition d'un produit satisfaisant ces besoins.

Cette notion de connaissance en conception est importante, car elle permet de typer les problématiques de conception. Trois types de connaissances ont été identifiés par Brown et Chandrasekaran : la connaissance sur le domaine de l'objet à concevoir, la connaissance sur la démarche ou la manière de concevoir et la connaissance sur les besoins à l'origine du problème de conception (Brown et Chandrasekaran, 1985). Suivant la présence plus ou moins significative de ces trois types de connaissances, Brown et Chandrasekaran identifient la conception routinière, la conception innovante et la conception créative. Cette classification est acceptée par de nombreux auteurs (Kota et Ward, 1991), (Gero, 1989). La représentation des différents espaces de solutions est présentée en figure 1.1.

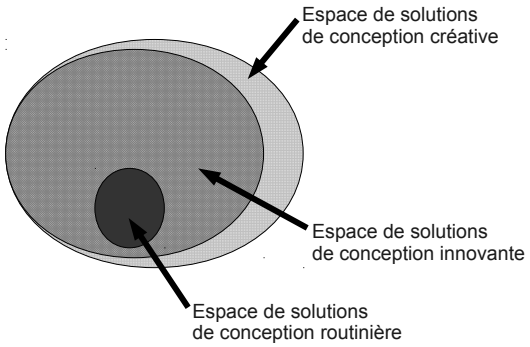


FIGURE 1.1: Espace de solutions selon le type de conception (Gero, 1990)

Dans le cas de la conception routinière dénommée également conception prédéfinie, les connaissances du domaine, de la démarche et des besoins sont disponibles. Le problème de conception, dans ce cas, revient à choisir une solution ou des principes de solutions existants et à les adapter ou les faire légèrement évoluer. La situation la plus extrême de ce type de conception (ou conception infiniment routinière) correspond au problème de configuration, encore appelé *customisation* ou personnalisation. La configuration fait l'hypothèse que l'espace de solutions a été totalement étudié et analysé. Il en résulte une connaissance exhaustive permettant de décrire toutes les solutions admissibles. Ce type de conception infiniment routinière se rencontre sur les sites web marchands, par exemple tous les constructeurs automobiles disposent sur leur site web d'un configurateur pour aider leurs clients à caractériser la voiture correspondant à leurs besoins.

Nos travaux se situent principalement dans le cadre des problèmes de conception routinière et plus spécifiquement en configuration. Les deux sections 1.1.1 et 1.1.2 vont documenter les approches et méthodes disponibles en conception et en configuration. La section 1.1.3 s'intéressera aux outils et techniques permettant d'aider à la résolution de ces deux types de problèmes.

1.1.1 Approches et méthodologies de conception

Oosterman, citant la revue bibliographique réalisée par Blessing (Blessing, 1996), conclut que toutes les méthodologies de conception font apparaître trois ou quatre phases allant de la définition du problème à la phase de conception détaillée (Oosterman, 2001). Chaque phase est associée à un processus à l'issue duquel un état particulier du produit est atteint. On retrouve également dans la thèse de Scaravetti une comparaison de plusieurs processus de conception issus de recherches académiques ou de processus industriels et normatifs (Scaravetti, 2004). En nous inspirant de ces travaux, nous passons maintenant en revue quelques modèles ou processus de conception significatifs.

1.1.1.1 Approche systématique

Pour Pahl et Beitz, la conception est un processus déterministe constitué d'activités ordonnées (Pahl et Beitz, 1996). Ils décrivent ce processus par l'enchaînement de quatre principales phases :

- phase de spécification et de définition du cahier des charges (*Planning clarifying the task*). Cette phase permet de définir précisément les exigences, les fonctions et les objectifs que doit vérifier l'objet à concevoir et d'aboutir à la définition du cahier des charges,
- phase de définition des concepts (*Conceptual design*). Cette phase permet de définir les principes de la solution qui répondra aux exigences définies précédemment. Cette phase est constituée par les activités suivantes : établir la structure fonctionnelle, rechercher et sélectionner les principes de solution et combiner les principes en concepts,
- Phase de définition structurelle (*Embodiment design*). Lors de cette phase, le concepteur doit déterminer et décrire la structure de l'objet : les éléments physiques de celui-ci (forme, matériau), mais aussi leurs arrangements. Il doit aussi vérifier et affiner les critères techniques et économiques,
- phase de définition détaillée (*Detail design*). Cette phase permet de définir, au niveau de détail le plus bas, les interactions entre les différents composants ainsi que leur forme et leur dimension. Les types de matériaux, les alternatives de fabrication et les coûts sont déterminés. Cette phase s'achève par la production d'une documentation détaillée décrivant complètement l'objet à concevoir.

Pahl et Beitz décrivent un processus séquentiel, présenté en figure 1.2, avec les quatre phases précédentes qui se succèdent de manière à ce que le résultat d'une phase permette à la phase qui suit de s'enclencher. Ils soulignent l'aspect itératif du processus de conception en insistant sur le fait que dans chacune des phases, plusieurs alternatives sont testées jusqu'à la sélection de la meilleure grâce aux activités de vérification et de validation qui jalonnent le processus.

1.1.1.2 Approche axiomatique

L'approche axiomatique (*Axiomatic Design*) décrite dans (Yoshikawa, 1989), (Suh, 1990) et (Albano et Suh, 1992) considère que l'activité de conception est une interaction continue entre le **Quoi**, qui exprime ce que l'on veut réaliser et le **Comment** on veut le réaliser.

Les concepts clés de cette approche (Suh, 2001) sont :

- le concept de domaine. Ce concept vise à systématiser le processus mental impliqué dans l'interaction du Quoi et du Comment et qui permet de créer une délimitation des différentes activités de conception fournies par cette approche. La conception est décrite par quatre domaines, tel que présenté en figure 1.3 :

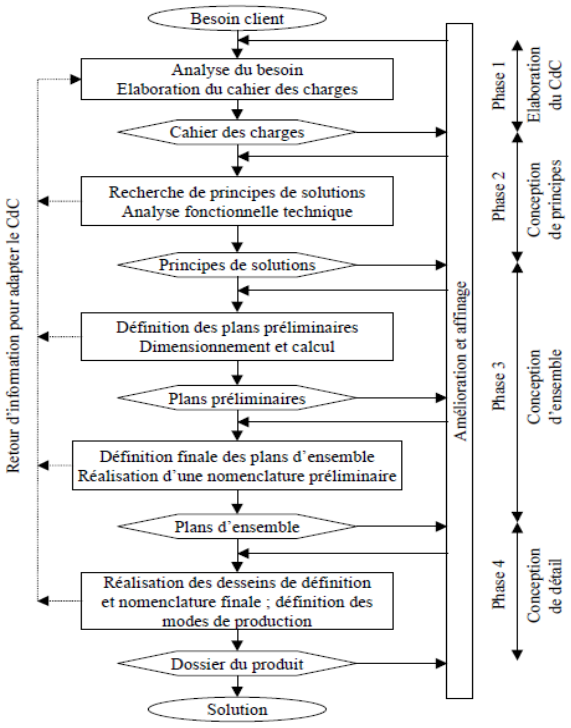


FIGURE 1.2: Processus de conception de Pahl et Beitz.

  - le domaine des exigences clients (*Customer Attributes*, noté *CA*),
  - le domaine des exigences fonctionnelles (*Functionnal Requirements*, noté *FR*),
  - le domaine des paramètres physiques de conception (*Design Parameter*, noté *DP*),
  - et le domaine des variables processus (*Process Parameter*, noté *PP*).

  Le domaine des exigences client *CA* caractérise les besoins que le client recherche dans le produit, le processus ou le système. Le domaine fonctionnel *FR* traduit le besoin du client en exigences fonctionnelles et en contraintes. La satisfaction des exigences fonctionnelles est assurée par la définition des paramètres de conception dans le domaine physique *DP*. Enfin, pour produire l'objet spécifié par les paramètres précédents, un processus caractérisé par des variables de processus *PP* est établi dans le domaine processus,
- le concept de hiérarchie. La conception progresse de manière descendante d'un niveau peu détaillé à des niveaux de plus en plus détaillés en procédant par décomposition successives,
- le concept de zigzag. La décomposition des entités des différents domaines est faite par une approche descendante passant d'un domaine à un autre (principalement entre les domaines des exigences fonctionnelles *FR* et des paramètres de conception *DP*),
- le concept d'axiomes de conception. Deux axiomes d'indépendance et d'information gouvernent le processus de conception. Ces deux axiomes permettent de distinguer les bonnes conceptions des mauvaises en raisonnant sur la forme des matrices reliant les exigences fonctionnelles *FR* aux paramètres de conception *DP*.

1.1.1.3 Ingénierie intégrée

Afin de maîtriser et de réduire le cycle de développement du produit, l'ingénierie intégrée, parfois dénommée conception intégrée ou ingénierie concourante (*Concurrent Engineering*, noté *CE*),

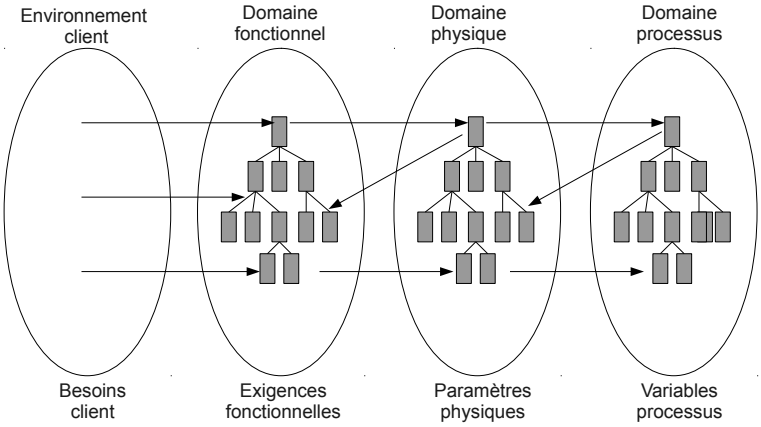


FIGURE 1.3: Les quatre domaines de l'approche axiomatique (Gumus, 2005).

est utilisée depuis plus de quinze ans par les entreprises (Prasad, 1997). Cette approche propose d'intégrer ou de mettre en étroite relation l'ensemble des activités de conception d'un produit dès l'initialisation de sa conception. Le *CE* (Kusiak et Wang, 1993), (Lawson et Karandikar, 1994), (Prasad, 1997) est apparu au milieu des années quatre-vingts comme une nouvelle forme d'organisation. Il est caractérisé par deux grands principes : la simultanéité et l'intégration. Le premier consiste à réaliser en parallèle différentes activités liées à la conception du produit et de son système de production. L'intégration consiste à établir des échanges entre les acteurs des différentes phases du projet, par la prise en compte, à chaque phase du développement, des considérations relatives à l'ensemble du développement du produit. Le *CE* est une façon différente d'organiser la conception, car il permet de rompre avec l'organisation séquentielle des activités.

1.1.1.4 Approche structure-comportement

Gero présente un modèle de conception basé sur trois éléments caractéristiques de l'objet à concevoir : les fonctions (*Functions*), le comportement (*Behavior*) et la structure (*Structure*) (Gero, 1990). Ce modèle permet une représentation explicite des fonctions de l'objet (problème), de sa structure (la solution) et de son comportement interne. Nous retrouvons une définition de ces trois concepts dans (Labrousse et Bernard, 2004) ou (Belu et Anghel, 2008) :

- *functions* : elles décrivent de manières abstraites les finalités d'un objet (processus, produit ou ressource) ;
- *behavior* : il décrit la dynamique de l'objet. Il peut comprendre un ensemble de lois et de règles (modèles continus), ainsi qu'une suite séquentielle d'états (modèles discrets) représentant l'évolution d'une structure ;
- *structure* : elle permet de spécifier les éléments qui composent l'objet modélisé ainsi que les attributs de ces éléments.

Selon Gero, l'activité de conception consiste à traduire un ensemble de fonctions F et de le transformer en une description de conception D d'un objet capable de réaliser ces fonctions.

D'après la définition des trois concepts *Functions*, *Behavior*, *Structure*, on constate qu'il n'existe pas de lien direct entre les fonctions F et la structure S. Pour pouvoir passer des fonctions à la structure, Gero suggère l'introduction de deux autres concepts dérivés du concept de *Behavior* : le comportement effectif de la structure solution ou *Actual Behavior* et le comportement souhaité ou *Expected Behavior*.

1.1.1.5 TRIZ

TRIZ, pour *Theory of Inventive Problem Solving*, prend comme origine des travaux de Altshuller (Altshuller, 1984). Altshuller indique que seulement 1% des solutions sont réellement des solutions innovantes, alors que les autres sont des adaptations de solutions existantes. La méthodologie TRIZ s'appuie sur des outils d'analyse du problème de conception et sur des bases de connaissances permettant de faire évoluer le problème de conception. Cette méthode est souvent l'objet de controverses du fait du peu de publications disponibles. Elle semble constituer néanmoins l'une des rares méthodes supportant les conceptions créatives et inventives. Elle bénéficie par ailleurs d'une panoplie d'outils aidant la créativité.

1.1.1.6 Processus normatif EIA-632

Le standard EIA-632 (ANSI, 1998) couvre une grande partie du cycle de vie des produits, qui, dans le cas de l'EIA-632, s'étend de la définition du système à son transfert à l'exploitation. Le rôle de l'EIA-632 est de fournir un ensemble de processus fondamentaux pour aider un développeur dans la conception ou l'amélioration d'un système (Martin, 1998). L'EIA-632 fournit cinq groupes de processus qui interagissent dans un processus général de développement : gestion (*Technical Management*), approvisionnement (*Acquisition Supply*), conception (*System Design*), production (*Product Realisation*) et évaluation (*Technical Evaluation*), tel qu'illustré en figure 1.4.

Le processus *System Design*, qui nous intéresse plus particulièrement, doit vérifier un certain nombre d'exigences permettant de décrire complètement l'objet à réaliser. Ce sont des exigences liées au processus de définition des exigences (*Requirement Definition Process*) et au processus de définition de la solution (*Solution Definition Process*), tel qu'illustré en figure 1.5. Les exigences peuvent être perçues en première définition comme la conceptualisation d'un besoin exprimé par un client ; les solutions seront déclinées en solutions logiques (représentation abstraite des principes de fonctionnement envisagés) menant à des solutions physiques (représentation informatique virtuelle des produits ou systèmes conçus). L'ensemble de ces concepts est fédéré au sein d'un bloc de construction (*Building Block*). C'est au niveau des solutions logiques des blocs de construction que s'effectue la décomposition du produit (ou système) en composants (et/ou sous-systèmes), faisant ainsi apparaître une décomposition hiérarchique du produit (ou système).

1.1.1.7 Synthèse entre les différentes approches

Après avoir passé en revue les approches de conception les plus étudiées, nous relevons quelques points qui nous semblent importants :

- **sur l'existence de domaines fondamentaux :** les quatre approches, approche systématique, approche axiomatique, approche structure-comportement et l'EIA-632, reposent fondamentalement sur des échanges ou associations de deux domaines : fonction/physique

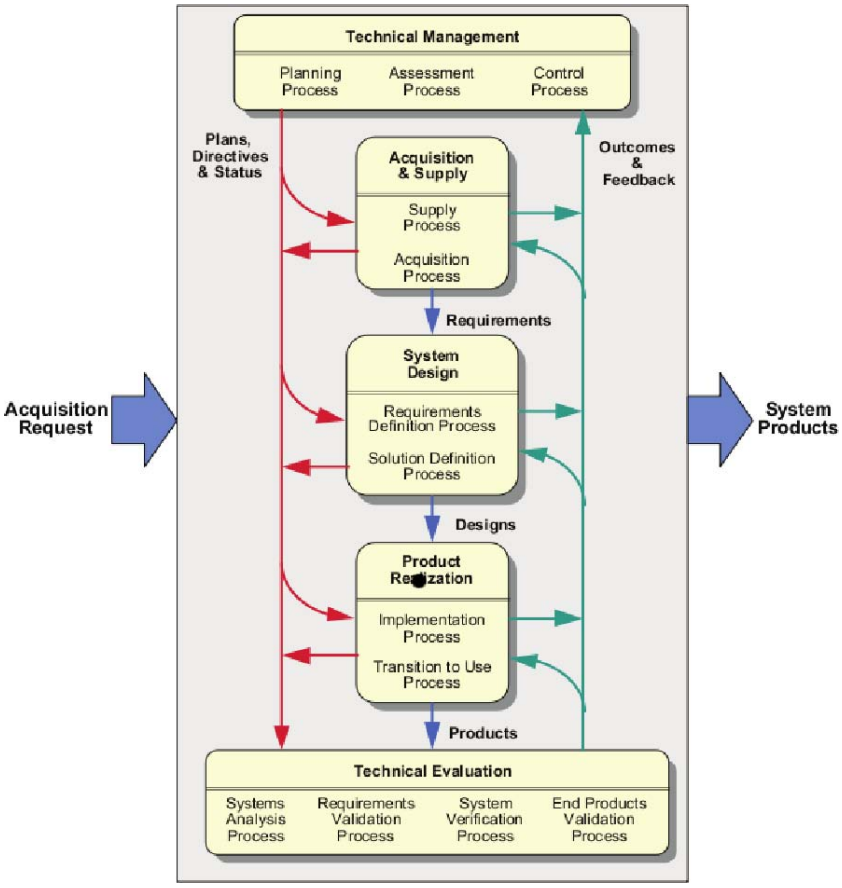


FIGURE 1.4: Les processus de EIA-632

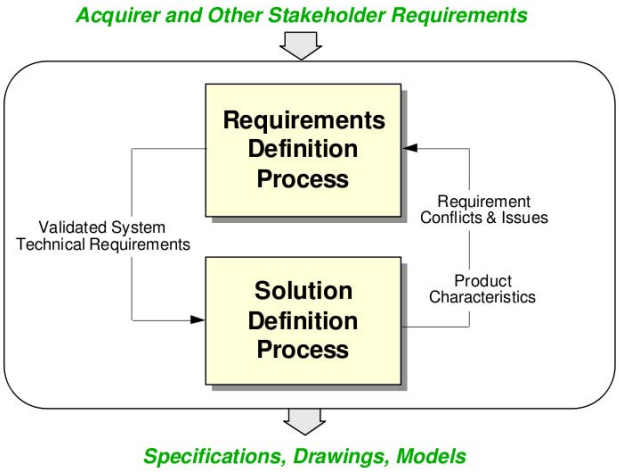


FIGURE 1.5: Le processus de conception de EIA-632

ou besoin/solution. Certaines de ces approches ajoutent la notion de processus de réalisation ou de dynamique de l'objet ou système à concevoir.

- **sur l'approche descendante :** deux approches, l'approche axiomatique et l'EIA-632, insistent fortement sur la décomposition, qui permet d'affiner progressivement la conception. L'EIA-632 exploite une structure de blocs de construction où chaque bloc associe à chaque niveau besoin et solution. L'approche descendante de l'approche axiomatique s'effectue en changeant de domaine. Cette notion de décomposition n'est par contre pas du tout explicite dans les approches systématiques et structure-comportement.
- **sur la globalité de l'approche ou le processus complet de conception :** trois approches, l'approche systématique, l'ingénierie intégrée et l'EIA-632, fournissent un cadre général ou un processus de conception. L'ingénierie intégrée est clairement une approche organisationnelle de la conception, qui vise à favoriser l'échange entre les activités et la simultanéité des activités de conception. L'EIA-632 s'apparente également à un ensemble d'activités complété par un recueil de bonnes pratiques de conception plutôt qu'à une description précise d'un processus de conception. Seule l'approche systématique fournit une succession d'étapes avec un minimum de formalisme.
- **sur la tâche fondamentale de conception :** seules l'approche axiomatique et la méthode TRIZ semblent s'intéresser à la tâche fondamentale de conception, qui est d'associer une solution à un besoin. L'approche axiomatique énonce des règles de bonne conception, alors que TRIZ formalise la conception comme une contradiction qui est à résoudre. En comparant TRIZ et l'approche axiomatique, nous remarquons que dans l'approche axiomatique, il n'existe pas de description claire de la manière avec laquelle on aboutit à une solution, alors que TRIZ s'attache à formuler les solutions en précisant le processus de résolution d'une contradiction.

En conclusion, nous retenons pour notre problématique de couplage entre la conception produit et la gestion de projet :

- certains domaines proposés par l'approche axiomatique : le domaine fonctionnel (*FR*), le domaine physique (*DP*) et le domaine des variables processus (*PP*) ;
- et l'approche descendante en zigzag (entre besoin et solution),

qui serviront de base à nos propositions.

Notre travail visant plutôt la mise au point d'outils d'assistance, nous n'aborderons pas les approches globales. Il en sera de même pour ce qui est lié à la tâche fondamentale de conception, car les problématiques de configuration considèrent que les associations possibles entre les besoins et les solutions constituent de la connaissance nécessaire et indispensable.

1.1.2 Configuration et méthodologies de configuration

L'activité de configuration peut donc être considérée comme une forme de conception infiniment routinière. La configuration fait l'hypothèse que l'espace des solutions possibles a été entièrement étudié, analysé et validé. Il doit être clair que cela n'empêche pas l'existence d'espaces de solutions très grands. Par exemple, il a été comptabilisé plus de mille quatre cent milliards de configurations possibles pour la gamme *Mégane* proposée par *Renault* (Amilhastre et al., 2002). Ce chapitre va en conséquence aborder une définition du problème, puis certaines de ses extensions correspondant aux domaines précédemment évoqués et au processus global de configuration.

1.1.2.1 Premières définitions

L'une des premières définitions du problème de configuration remonte aux travaux de Mittal et Frayman (Mittal et Frayman, 1989). De nombreux auteurs travaillant dans le domaine se réfèrent à cette définition.

Définition 1 : *configuration*

Étant donné :

- *un ensemble fixe et prédéfini de composants A, où un composant est décrit par un ensemble de propriétés, des « ports » pour le relier à d'autres composants, des contraintes sur chaque port pour décrire les composants qui peuvent s'y connecter et des contraintes structurelles,*
- *une description de la configuration désirée B,*
- *et éventuellement des critères pour faire une sélection optimale C,*

configurer consiste à trouver au moins une configuration qui réponde à toutes les exigences, où une configuration est un ensemble de composants et une description de leur connexion, sinon détecter des incohérences dans les conditions A, B et C.

Cette définition de la configuration est axée essentiellement sur les problèmes de choix de composants physiques et donc sur la constitution de la structure physique du produit. Reprise par de nombreux auteurs comme Sabin et Freuder (Sabin et Freuder, 1996), Sabin et Weigel (Sabin et Weigel, 1998) ou Soininen (Soininen, 1999), elle fait apparaître implicitement la notion de modèle générique (ensemble de composants, propriétés et connexions possibles) qui, lorsqu'il est superposé à une configuration partielle désirée (un ensemble de quelques composants clés souhaités par le client), permet d'obtenir le produit configuré spécifique au client.

Le modèle générique est un concept clé en configuration. Il représente l'ensemble de toutes les options et variantes d'un produit incluant leurs compatibilités et/ou incompatibilités. Ce modèle générique contient donc toute la connaissance caractérisant la diversité du produit. Cette définition ne considère pas explicitement les notions de besoin, fonction, exigence, processus de réalisation et ressource.

1.1.2.2 Extensions et évolutions du problème

Des auteurs comme Tiihonen et al. (Tiihonen et al., 1996), Veron (Veron, 2001) et Hvam et al. (Hvam et al., 2002) ont étendu cette définition en incluant les besoins et le processus de réalisation. Dans ce courant de travaux, Tiihonen et al. distinguent trois niveaux de configuration (Tiihonen et al., 1996) :

- le premier est une forme de configuration fonctionnelle du besoin client (*Sales Specification*),
- le second correspond à la configuration physique de la nomenclature (*Description of Product Instance*) respectant les fonctions et besoins du niveau précédent,
- le dernier permet la configuration du processus de production (*Component Manufacturing Specification*).

Il apparaît clairement que les domaines identifiés en conception (fonction ou besoin, physique ou solution et processus de production (voir section 1.1.1.7) apparaissent en configuration. Le principe de ces approches est de disposer d'un modèle générique par niveau de configuration. Ces

trois modèles représentent donc toutes les options et variantes pour configurer les besoins clients, les solutions physiques ou nomenclatures et les processus de production.

L'activité globale de configuration peut alors se scinder en trois étapes successives, mettant en œuvre des connaissances différentes et donc des acteurs différents : vendeur, concepteur, producteur par exemple. Par contre, la notion d'approche descendante mise en évidence en conception est beaucoup moins présente en configuration. Quelques travaux, (Sabin et Freuder, 1996) et (Veron, 2001) entre autres, se sont intéressés à cet aspect et ont montré qu'il ne soulevait pas de difficultés particulières. Ces extensions sont de première importance pour nous qui nous intéressons au couplage de la configuration de produit avec la planification de sa réalisation.

1.1.2.3 Configuration et configurateur

En complément des extensions précédentes, Hvam et al. ont proposé une approche visant à définir un processus global de configuration incluant les activités liées à l'obtention des modèles génériques nécessaires à la configuration (Hvam et al., 2002). Ces activités correspondent d'une part à l'identification, l'analyse et la validation des connaissances du domaine (c'est-à-dire la liste de toutes les options, variantes et interdépendances) et d'autre part à leur formalisation pour permettre leur exploitation lors de la configuration et leur maintenance ultérieure. Il est important de noter que l'élaboration des modèles génériques pour la configuration est un travail critique, car il engage et définit la diversité proposée au client. Cela nécessite de définir des notions de familles de produits et de faire converger les préoccupations des vendeurs, des concepteurs et des producteurs. De plus, cette activité de modélisation est récurrente car une fois modélisée, il est important de noter que la diversité (liées aux besoins, solutions, procédés) évolue sans cesse du fait de nouvelles attentes et de nouvelles technologies.

Il est maintenant possible de compléter ces éléments avec la caractérisation de ce qu'est un configurateur ou progiciel de configuration. La plupart des auteurs, comme (Tiihonen et Soininen, 1997) ou (Moynard, 2003), définissent le configurateur comme un progiciel, qui aide la fonction de configuration. Sans préjuger de la technique informatique employée, il est toujours composé :

- d'une base de connaissance où sont stockés les modèles génériques,
- éventuellement d'un module aidant la saisie du modèle générique,
- d'un module aidant la tâche de configuration, c'est-à-dire l'instanciation du modèle générique conformément aux besoins client.

Il est possible de se référer à (Sabin et Weigel, 1998) pour un aperçu des progiciels existants il y a quelques années.

En ce qui concerne nos travaux, nous ne nous intéresserons pas aux aspects extractions de connaissances d'un domaine particulier, par contre nous serons amenés à considérer des typologies de paramètres descriptifs suivant les différents domaines considérés (besoins, solution, procédés, ressources) et bien sûr à détailler les outils permettant d'aider la tâche de configuration.

1.1.2.4 Quelques applications significatives

Les terrains d'application des techniques de configuration sont nombreux et variés. Il est cependant intéressant de noter que les premières applications se sont intéressées à la configuration d'ordinateur. F. Frayman travaillait chez *Hewlett-Packard* lorsqu'il a proposé la définition de la configuration (Mittal et Frayman, 1989) ; à la même époque, *Digital Equipement Corp.* proposait également un système expert pour configurer ses solutions (Barker et al., 1989). Plus

récemment, l'entreprise *Dell* a basé son offre produit sur les techniques de configuration. Les applications concernent aussi le domaine automobile : tous les constructeurs disposent d'un configurateur sur leur site web et de très nombreuses publications scientifiques sont illustrées sur des problématiques issues de l'automobile (Amilhastre et al., 2002), (Sabin et Freuder, 1996), (Soininen et Gelle, 1999) et (Pargamin, 2002). Par la suite, d'autres applications comme, par exemple, les centraux téléphoniques (Fleischanderl et al., 1998), les avions (Mulyanto, 2002), les aménagements de placards (Aldanondo et al., 2003), les mixers industriels (Gelle, 1998) ou encore les opérations de traitement thermiques (Vareilles et al., 2007), montrent la diversité du champ d'exploitation de ces techniques.

1.1.3 Outils et techniques d'aide à la décision en conception ou en configuration

Cette section s'intéresse aux outils d'aide à la décision existant pour la conception ou la configuration. De ce fait, nous n'évoquerons pas volontairement les outils dits de conception assistée par ordinateur, qui sont spécifiques à une discipline (par exemple : CAO ou XAO mécanique, électronique, automatique, chimique...), ceci pour s'affranchir de tout champ disciplinaire. Nous évoquerons succinctement les outils mettant en relation les domaines considérés, les outils exploitant de la connaissance contextualisée et les outils exploitant de la connaissance formalisée dans des modèles.

1.1.3.1 Outils et techniques reliant des domaines

Dans le courant des premières techniques d'analyse fonctionnelle et d'analyse de la valeur, l'outil *QFD* (*Quality Function Deployment*) a été développé en 1966 par Akao (Akao, 1997). Le *QFD* est un outil issu du monde de la qualité, qui permet d'aider à traduire les besoins du client en un nouveau produit qui répond exactement aux attentes du client. Cohen définit le *QFD* comme une méthode pour la conception et la planification de produit structuré, qui permet à une équipe de développement de spécifier clairement les désirs et les besoins des clients, d'en dériver une solution et de l'évaluer (Cohen, 1995). Le *QFD* est basé sur la construction et l'exploitation de matrices (besoin, solution) plus connues sous la terminologie « maison de la qualité ». Le *QFD* met explicitement en relation besoins et solutions et plus précisément fonctions et composants.

Au début des années quatre-vingts, les domaines besoin, solution, processus et ressource ont été associés et complétés pour déboucher sur tout un courant de travaux dénommé *DSM* pour *Design Structure Matrix*[1]. Les *DSM* sont un ensemble d'outils et de techniques analytiques introduit par (Steward, 1981) dont un état de l'art peut être consulté dans (Bonjour, 2008). Ces matrices décrivent les liens pouvant exister entre les éléments : d'un même domaine (*Design Structure Matrix* ou DSM), de deux domaines différents (*Domain Maping Matrix* ou *DMM*) et les deux à la fois (*Multiple-Domain Matrix* ou *MDM*), comme illustré en figure 1.6. Les domaines peuvent caractériser des besoins, des fonctions, des composants, des tâches du processus et des ressources. Dans le cas d'un produit, les matrices sont généralement utilisées par les concepteurs pour modéliser les relations binaires entre les sous-systèmes d'un produit complexe. Dans le cas d'un projet, elles sont utilisées pour représenter les liens de dépendance et/ou de précédence entre les tâches. Enfin, dans le cas des organisations, elles sont utilisées pour représenter de nouvelles organisations basées sur les regroupements, qui minimisent les interactions entre les équipes (Eppinger, 2001).

1. http://www.dsmweb.org/

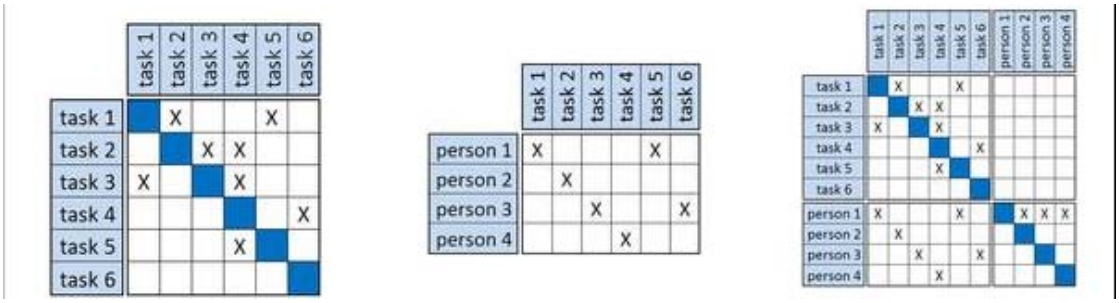


FIGURE 1.6: Matrice *DSM - DMM- MDM*

1.1.3.2 Outils et techniques à base de connaissance contextualisée

Le raisonnement à partir de cas, inspiré du raisonnement humain, constitue la technique la plus significative exploitant de la connaissance contextualisée. L'être humain, afin de résoudre un problème, fait le plus souvent appel à son expérience dans la résolution de problèmes semblables rencontrés dans le passé (Gentner, 1983). On parle souvent de raisonnement par analogie.

Dans le raisonnement à partir de cas, l'expérience passée est sauvegardée sous la forme d'un ensemble de cas dans une bibliothèque, ou base de cas. Un cas représente la description du problème et la description d'une ou de plusieurs solutions de ce problème. Pour résoudre un problème (Aamodt et Plaza, 1994), il est alors nécessaire de comparer le problème actuel aux problèmes décrits dans les cas sauvegardés. Les cas similaires ou proches sont alors retenus et fourniront des solutions de base à adapter, de manière plus ou moins conséquente, pour proposer une solution au problème courant. La description du nouveau problème et de sa solution une fois validée pourra enrichir la bibliothèque de cas.

Cette approche présente l'avantage d'éviter l'extraction de connaissance et de faire appel à un cogniticien pour formaliser la connaissance explicite du problème (Watson et Marir, 1994). La connaissance est effectivement « cachée » dans la description des cas ; on sait que la solution est bonne, mais l'on ne sait pas exactement pourquoi. On parle de connaissance implicite ou contextualisée, dans la mesure où elle est intimement liée au problème décrit dans le cas. De très nombreux travaux concernant la modélisation des cas et la définition de mécanismes de recherche de cas similaires ont été conduits, un état de l'art récent peut être consulté dans (Renaud et al., 2008).

1.1.3.3 Outils et techniques à base de modèle de connaissance

Lorsque la conception est routinière, la récurrence des problèmes et l'exploitation éventuelle de cas issus d'un mécanisme de raisonnement à partir de cas permettent d'identifier des bonnes pratiques, des manières de faire intéressantes, des solutions typiques ou encore des règles que doit vérifier toute bonne solution. Tous ces éléments constituent de la connaissance dite explicite ou non contextualisée, dans la mesure où elle est indépendante de l'occurrence du problème. L'idée de ces outils est de capitaliser, formaliser et stocker cette connaissance dans une base de connaissance et de confronter ou superposer le problème de conception à étudier. On retrouve en filigrane le problème de configuration et la notion de modèle générique qui lui est associé (section 1.1.2.1). Deux courants de travaux liés à deux manières de modéliser la connaissance élémentaire se sont succédés : les systèmes experts et les problèmes de satisfaction de contraintes.

Dans les systèmes experts, la connaissance élémentaire est formalisée sous forme de règles de

type « si-alors ». Un système expert (noté *SE*) est un outil capable de reproduire les mécanismes cognitifs d'un être humain. Un *SE* est composé d'une base de faits, d'une base de règles et d'un moteur d'inférence. Le moteur d'inférence utilise les faits et les règles disponibles pour produire des nouveaux faits, jusqu'à parvenir à répondre aux questions posées (Lévine et Pomerol, 1989). Un exemple typique de *SE* dans le domaine de l'aide à la conception est le système d'aide à la configuration d'ordinateurs, évoqué en section 1.1.2.4, développé par *Digital Equipement Corp.*, appelé XCON (McDermott, 1982). Après avoir suscité de très grands espoirs, cette technique a été plus ou moins abandonnée. Le problème de la maintenance des bases de connaissances et la difficulté de séparer les connaissances du domaine de la méthode de résolution peuvent partiellement expliquer cet abandon.

À l'opposé, les travaux sur les problèmes de satisfaction de contraintes ou *Constraint Satisfaction Problem*, noté *CSP* sont actuellement extrêmement nombreux et variés. Dans ce type de problème, la connaissance élémentaire est formalisée sous la forme de contraintes qui restreignent le produit cartésien d'un ensemble de variables, c'est-à-dire les combinaisons de valeurs possibles. Les *CSP* ont été introduits par (Montanari, 1974) pour réaliser un étiquetage cohérent des arêtes dans des images en deux dimensions. Cette approche est basée sur la modélisation d'un problème avec deux concepts fondamentaux : les variables et les contraintes. Ces approches ont été utilisées dans les travaux de Vargas pour représenter les contraintes produit et processus d'une culasse d'automobile (Vargas, 1995), dans ceux de Sam-Haroud pour la préconception de pont (Sam-Haroud, 1995) et plus récemment dans les travaux de Vernat pour la conception de batteries électriques (Vernat, 2004). De très nombreux travaux déjà évoqués ont utilisé cette approche pour aborder différents aspects de la configuration d'automobile (Amilhastre et al., 2002), (Sabin et Freuder, 1996), (Soininen et Gelle, 1999) et (Pargamin, 2002). Bien que cette approche nécessite un travail conséquent d'extraction et de formalisation des connaissances sous la forme d'un *CSP*, elle est fréquemment utilisée pour aider à la configuration dans l'industrie.

1.1.4 Conclusion sur conception et configuration de produit

Nous avons ainsi fait un tour d'horizon des outils d'aide à la décision en conception et en configuration. Nous retenons pour nos travaux et développements futurs :

- la notion de domaines, qui recouvre, suivant les auteurs, les besoins, les fonctions, les solutions, les composants physiques et le processus de réalisation,
- la notion d'approche descendante en zigzag, identifiée dans l'état de l'art sur la conception,
- la notion de modèle générique de produit et d'instanciation de modèle générique pour aider à la décision,
- le concept de matrice *DSM* pour identifier les relations entre différents domaines,
- les approches par contraintes, qui permettent de supporter la notion de modèle générique.

1.2 Gestion et planification de projet

Avant de détailler la gestion et la planification de projet, il est important de revenir sur le terme projet. Définir le projet n'est pas chose simple, car tout peut être vu comme étant un projet. Selon Meredith et Mantel (Meredith et Mantel, 1989) le projet se définit comme une tâche spécifique et unique à accomplir afin d'atteindre un ensemble d'objectifs. Cette tâche peut être subdivisée en sous-tâches qui doivent d'être accomplies afin d'atteindre les objectifs du projet. Le projet nécessite alors une coordination et un contrôle minutieux en termes de délai, coût et

performance. De même, Rosenau et Githens confirment le caractère temporaire et unique du projet, ainsi que son élaboration progressive (Rosenau et Githens, 2005). Pour l'AFNOR, le projet est une démarche spécifique, qui permet de structurer méthodiquement et progressivement une réalité à venir (norme AFNOR X50-10). Il correspond également à un processus unique regroupant un ensemble d'activités coordonnées et maîtrisées comportant des dates de début et de fin, entrepris dans le but d'atteindre un objectif conforme à des exigences spécifiques et respectant des contraintes de délais, de coûts et de ressources (norme AFNOR FD X50-118). Ces éléments de caractérisation se retrouvent également dans le *PMI handbook*, auquel nous nous référerons par la suite. De ces définitions, nous retenons les caractéristiques principales d'un projet qui comprennent : un objectif ou résultat unique à atteindre (produit ou service), des délais, des dates de début et de fin et enfin une décomposition en tâches nécessitant des ressources. Ces éléments étant posés, ce chapitre va documenter et distinguer les notions de gestion et de planification de projet et aborder les méthodes et outils associés.

1.2.1 Processus de gestion et de planification de projet

Afin de permettre la réalisation de projet, il est nécessaire de coordonner et de contrôler les différentes sous-tâches du projet en termes de délais, de coût et de performance. Ces deux activités composent la gestion ou management de projet.

Selon Meredith et Mantel, le management de projet a pour objectif d'intégrer tous les aspects du projet, en termes de disponibilité de ressources et de connaissances et particulièrement de réaliser le projet dans le respect des délais et coûts initialement prévus (Meredith et Mantel, 1989). Pour Rosenau et Githens, le management de projet correspond à la mise en application de connaissances, d'outils et de techniques sur un projet (Rosenau et Githens, 2005). L'association francophone de management de projet AFITEP (Marciniak, 1998), considère que la gestion de projet est constituée d'un ensemble d'activités, qui apportent au chef de projet tous les éléments lui permettant de respecter les termes du contrat passé avec les clients, en contenu, coût, délais et qualité. Le standard d'ingénierie système EIA-632 se réfère, quant à lui, à trois processus de planification, d'évaluation et de supervision. L'ouvrage du *Project Management Institute* (*PMI*) (Institute, 2004) propose une définition, qui se rapproche de celle de (Rosenau et Githens, 2005) : « le management de projet met en œuvre des connaissances, des compétences, des outils et des méthodes, en vue d'atteindre ou de dépasser les besoins et les attentes des parties prenantes du projet ». Ces connaissances sont organisées en neuf domaines, qui vont représenter les neuf activités du management de projet :

- le management de la coordination (*project integration management*),
- le management du contenu du projet (*project scope management*),
- le management des délais (*project time management*),
- le management des coûts (*project cost management*),
- le management de la qualité (*project quality management*),
- le management des ressources humaines (*project human resource management*),
- le management de la communication (*project comunications management*),
- le management des risques (*project risk management*),
- le management des approvisionnements (*project procurement management*).

Nous nous inscrivons dans cette dernière définition comprenant les neuf activités identifiées et positionnons nos travaux dans l'activité de management des délais (*project time management*),

que nous détaillons maintenant. La planification de projet, ou management des délais, comprend principalement des activités en rapport avec le temps, s'articulant autour de la prévision, de la réalisation et du suivi des activités ou tâches du projet. Le PMI détaille cette planification en six activités :

1. identification des activités (*Activity Definition*) : identification des activités qui doivent être accomplies pour produire les différents livrables du projet,
2. Séquencement des activités (*Activity Sequencing*) : identification et mise en évidence des relations de précédence entre les activités,
3. estimation des ressources des activités (*Activity Ressource Estimating*) : définition du type et de la quantité de ressources nécessaires à chaque activité,
4. estimation des durées des activités (*Activity Duration Estimating*) : définition des temps nécessaires pour réaliser chacune des activités,
5. élaboration de l'échéancier (*Schedule Development*) : élaboration du planning de réalisation des activités en fonction des durées et des besoins en ressources,
6. maîtrise de l'échéancier (*Schedule Control*) : contrôler le déroulement et suivre les modifications du planning de réalisation.

Les cinq premières activités correspondent pleinement à ce que nous entendons par planification lorsque nous évoquons notre problématique de couplage de configuration de produit avec la planification de sa réalisation.

1.2.2 Méthodes et outils pour la planification de projet

Nous distinguons dans un premier temps les méthodes visant avant tout à décrire le problème de planification de projet pour aborder ensuite les approches visant à aider sa résolution.

1.2.2.1 Méthodes orientées description du problème de planification du projet

Ces méthodes et outils permettent de représenter le résultat des quatre premières activités : les tâches, les antériorités entre tâches, les durées et ressources nécessaires.

Un projet est le plus souvent modélisé par un réseau d'activités associé à un graphe $G = (N, A)$, avec N l'ensemble des nœuds et A l'ensemble des arcs. Il existe deux modes de représentation d'un réseau de projet : le réseau *AoA* pour *activity-on-arc* et *AoN* pour *activity-on-node*. Le premier utilise l'ensemble A des arcs pour représenter les activités et l'ensemble N de nœuds pour représenter les évènements, tandis que le second associe les nœuds aux arcs et les arcs aux précédences.

Le réseau *AoA* est à la base des deux outils très connus en gestion de projet : le *PERT* (*Projet Evaluation and Review Technic*) et le *CPM* (*Critical Path Method*). Les travaux ont ensuite débouché sur, d'une part, les réseaux à activités déterministes et, d'autre part, sur les réseaux à activités généralisées. Le réseau à activités déterministes mis en œuvre dans le *PERT* et le *CPM* fait l'hypothèse que toutes les activités seront programmées et effectuées. Dans un réseau à activités généralisées, des tâches peuvent avoir leur programmation et/ou leur existence conditionnée, comme proposé voilà plus de trente ans dans l'approche *GAN* (*Generalised Activity Network*) (Elmaghraby, 1977), ou plus récemment dans *RAIH* (Réseau d'Activités Incertaines et Hiérarchisées) (Ramat, 1997).

En complément de ces approches basées avant tout sur les graphes, tout un courant de travaux s'est consacré à la représentation de la décomposition du projet en tâches et a débouché sur la notion de *Work Breakdown Structure* (*WBS*). Le *WBS* est un schéma qui représente l'ensemble des travaux à réaliser au cours d'un projet.

1.2.2.2 Approches pour aider à résoudre le problème de planification de projet

Une fois le problème de planification de projet défini par les quatre premières activités, il faut jalonner les tâches ou réaliser un échéancier des tâches. C'est-à-dire déterminer les dates de début et de fin des tâches du projet. Une très grande quantité de travaux s'est intéressée à cette problématique, que nous ne détaillerons pas. Par contre, nous allons mentionner deux caractéristiques clés du problème permettant de situer nos travaux. Nous conclurons sur le point de vue de la communauté de la programmation par contraintes en matière de planification.

La première caractéristique du projet est liée à la prise en compte de la capacité limitée des ressources lors de la réalisation de l'échéancier des tâches. La prise en compte simultanée des contraintes de délais et de capacité de ressources conduit au problème difficile d'ordonnancement à capacité finie (Esquirol et Lopez, 1999). La prise en compte séquentielle de ces contraintes conduit à effectuer une planification respectant les délais dans un premier temps, qui permet ensuite d'établir un plan de charge, dans un deuxième temps. En cas de surcharge, l'acteur chargé de la planification peut alors, soit augmenter la capacité de la ressource, soit modifier les contraintes de date du projet et relancer une nouvelle planification. Ce second fonctionnement est rencontré dans le processus de planification des besoins en composant en gestion de production pour les systèmes manufacturiers.

Nous nous orienterons vers cette seconde solution pour deux raisons. D'une part, la pratique industrielle en planification de projet s'est bien accommodée de cette démarche en deux temps, qui a tendance à privilégier avant tout le respect des délais. D'autre part, nous constaterons ultérieurement que notre objectif d'aide à la décision exploitant le filtrage de contraintes interactif est peu compatible avec la prise en compte simultanée des deux contraintes de délais et de ressources.

La seconde caractéristique de la planification de projet est liée à l'aspect dit « glouton » que peut avoir la recherche de solutions en planification. En cas de conflits portant sur l'utilisation des ressources, une méthode gloutonne proposera un ordre pour traiter les conflits et une règle pour arbitrer chaque conflit avec l'hypothèse qu'un conflit résolu n'est jamais remis en cause. Toutes les méthodes de placement au plus tôt, au plus tard, toutes les heuristiques basées sur des priorités minimisant par exemple les retards ou les marges sont des méthodes gloutonnes. A l'opposé, une méthode non gloutonne interdira l'emploi de toute règle arbitraire pour résoudre un conflit et essaiera de réduire progressivement l'espace de solution. Les approches par contraintes constituent dans cette situation une approche incontournable (Cavaillé et Aldanondo, 2001).

Dans un objectif d'aide à la décision interactive où la planification interagira avec la configuration via des approches par contraintes, nous considérons les approches non gloutonnes pour la planification.

Étant donné les éléments précédents, il est maintenant possible de considérer le point de vue de la communauté de la programmation par contraintes. Dans un article de référence récent, Bartak et al. synthétisent de manière très détaillée ce que recouvre le problème de planification avec les approches par contraintes (Bartak et al., 2010). Selon ces travaux, deux problèmes clés sont à considérer : la définition du projet ou *planning problem* et l'ordonnancement ou

scheduling problem. Le premier vise à définir le graphe des tâches permettant d'évoluer d'une situation initiale actuelle à une situation finale correspondant à l'objectif cible et qui correspond aux quatre premières tâches du processus *time management* du PMI. Le second vise à allouer les ressources et à fournir un échéancier des tâches et correspond à la tâche cinq du processus *time management* du PMI. Après avoir clairement distingué ces deux problèmes, ces auteurs indiquent que, très souvent, ceux-ci sont souvent partiellement mélangés dans les problématiques appliquées. Nous nous inscrivons entièrement dans cette dernière remarque pour introduire la notion de configuration de projet.

La section 1.1.2 de ce chapitre a présenté la conception et la configuration en indiquant comment la configuration était une forme d'activité routinière où un modèle générique de produit pouvait être instancié en un produit personnalisé répondant à des besoins spécifiques. Dans la lignée des travaux de (Aldanondo et Vareilles, 2008), nous considérons de même que le processus décrivant la réalisation ou la production des produits personnalisés peut être configuré ou dérivé d'un processus générique, c'est-à-dire identifier les opérations de production, leur séquencement, leur besoins en ressources, avec durée, qualité et coût. Nous considérons également que l'ensemble de ces opérations de production peut être ordonnancé.

1.2.3 Conclusions sur gestion et planification de projet

En bilan de ce regard sur gestion et planification de projet, nous retenons pour nos travaux et développements futurs :

- les caractéristiques principales du projet : objectif, résultat unique à atteindre, occurrence et déroulement unique, prise en compte de délais, de date de début et de fin, de coût et enfin décomposition du projet en tâches nécessitant des ressources,
- la notion d'approche multi-niveau : si elle est claire en conception, elle est par contre peu soulignée par les travaux en gestion de projet, mis à part la notion de *Work Breakdown Structure*,
- la vision proposée par le PMI découpant la gestion de projet en neuf chapitres dont le management des délais (*project time management*), qui recouvre ce que nous entendons par planification de projet,
- les approches par contraintes, qui en parfaite cohérence avec le management des délais du PMI, outillent la définition du projet et son ordonnancement.

Tous ces éléments nous ont permis d'introduire la notion de configuration de projet, qui de manière analogue à la configuration de produit, visera à instancier et ordonnancer un modèle générique de projet conformément à un produit configuré et des besoins projets.

1.3 Couplage de la conception de produit et planification de projet

Les principaux éléments bibliographiques relatifs, d'une part, aux problématiques de conception et configuration de produit et, d'autre part, à la planification de projet avec une ouverture sur une notion de configuration de projet, ont été rappelés. L'objet de cette section est de poser une première définition de ce que nous entendons par couplage et de faire le point sur les travaux l'ayant abordé.

1.3.1 Éléments de définition relatifs au couplage conception-planification

Si, comme nous l'avons vu, de très nombreux travaux ont été menés en aide à la conception et en aide à la planification, en débouchant sur la mise au point d'un très grand nombre de méthodes et d'outils, très peu de travaux se sont intéressés aux interactions pouvant exister entre ces deux processus et leurs outils respectifs. Pourtant, les décisions prises en conception de produit ou de système peuvent avoir un impact important sur la planification de sa réalisation. Par exemple, une technologie choisie pour une fonction peut nécessiter un délai supplémentaire de transport. Réciproquement, des décisions prises ou des contraintes de planification de projet peuvent fortement influencer les choix de conception. Par exemple, une date de mise à disposition peut interdire l'adaptation d'une fonctionnalité.

Il apparaît clairement un besoin d'interaction entre ces deux processus. En conséquence, le sens que nous mettons derrière le terme de couplage correspond à une mise en relation ou encore une mise en interaction des deux domaines, considérés généralement de façons disjointes, le domaine de la conception de produit et le domaine de la planification de projet. L'objectif principal de cette mise en relation est de permettre de propager les conséquences de décisions de la conception vers la planification, mais également de la planification vers la conception. Cette forme de propagation a pour but de réduire les incohérences entre la conception et l'organisation de la réalisation et d'améliorer ainsi la maîtrise des délais de réalisation.

Deux types de travaux sont détaillés, ceux qui proposent des approches générales et qui insistent sur l'identification de liens causaux, puis ceux qui présentent des méthodes et outils visant la résolution de problèmes spécifiques, où apparaît une forme de couplage.

1.3.2 Travaux généraux reliant les deux domaines

La conception axiomatique (*axiomatic design*), évoquée en section 1.1.1.2, mettait avant tout en relation le besoin (exigences fonctionnelles) et la solution (paramètres de conception). Certains auteurs ont étendu cette approche aux processus de réalisation. Les travaux rapportés dans (Goncalves-Coelho et al., 2005) ou encore dans (Goncalves-Coelho, 2004) ont montré qu'il était tout à fait possible d'ajouter un domaine correspondant aux processus de production et d'étendre les interactions besoins/solutions aux processus de fabrication.

Les travaux de Oosterman, également basés sur la conception axiomatique, se sont intéressés à l'influence du choix de l'architecture produit sur la coordination et l'organisation du projet (Oosterman, 2001). Ils ont montré que certaines interactions permettent d'organiser les équipes travaillant sur le projet.

De manière similaire à la conception axiomatique, les travaux reposant sur les matrices *DSM* (voir section 1.1.3.1) ont été également étendus pour considérer les processus de réalisation. Le travail de Lindemann exploitant les matrices multi-domaines (Lindemann, 2007) fait clairement apparaître les liens causaux pouvant exister entre les quatre domaines : fonctions, composants, processus et ressources. Danilovic et Browning s'inscrivent également dans ce courant de travaux et montrent que ces matrices peuvent constituer un bon support pour tracer et aider à la synchronisation des décisions entre domaines (Danilovic et Browning, 2007).

Évoqué également en section 1.1.1.6, la norme d'ingénierie système EIA-632 fait aussi apparaître les processus de conception de système et de gestion du développement, ainsi que les échanges pouvant exister entre ces processus. Des travaux à finalité appliquée ont également étudié les interactions et leur outillage pouvant exister dans l'entreprise (Abeille et al., 2009). Il en est

ressorti que ces interactions sont indispensables, mais le plus souvent peu formalisées et peu outillées et font appel avant tout à la mémoire des hommes et à la bonne entente entre concepteur et planificateur.

En bilan, il apparaît clairement que les principaux travaux en conception ont vu des tentatives d'extensions d'une part vers la définition de processus de réalisation et d'autre part vers la planification de projet, principalement sous l'angle de l'organisation de ressources. Les approches proposées visent principalement à définir, caractériser ou encore valider des liens causaux pouvant exister entre les domaines. Aucun travail ne met explicitement en rapport l'aspect temporel ou l'ordonnancement de projet avec des caractéristiques du problème de conception.

1.3.3 Méthodes et outils couplant les domaines sur des problèmes spécifiques

Dans le cadre d'un problème visant à faire interagir les décisions en conception logicielle et la gestion du développement de logiciel, Steward et Tate ont proposé de coupler la conception axiomatique et la planification de projet (Steward et Tate, 2000). Leur idée a été d'associer aux variables de conception, correspondant aux résultats des décisions de conception, les tâches du processus de développement modélisées sous la forme d'un graphe de tâches. La démarche a été outillée avec un développement spécifique pour la partie aide à la conception couplé au progiciel Microsoft Project. L'intérêt de ce travail est de clairement montrer qu'il est possible de définir un couplage entre la conception et la planification et de l'outiller au sens de la propagation des conséquence des décisions d'un domaine vers un autre.

Des travaux visant l'optimisation simultanée du produit conçu et du projet de réalisation associé ont été conduits par (Baron et Rochet, 2005) ou encore (Pitiot, 2009). Leur idée est d'inclure les degrés de liberté du produit conçu dans le graphe d'activités du projet et de l'optimiser. Les solutions obtenues permettent alors de définir d'une part l'échéancier final de réalisation et d'autre part, de propager les choix de projet sur les degrés de liberté de conception.

Un courant de travaux très conséquent s'est également intéressé à l'association de famille de produit et de graphe générique de projet. L'ensemble des travaux de Jiao et al. (Jiao et al., 2007) ou (Jiao et al., 2008) a proposé des règles permettant de mettre en cohérence des familles de produits et des graphes génériques de projet. Ces travaux sont très intéressants pour notre problématique, car ils définissent et mettent explicitement en relation des notions de structure générique de produit avec des structures génériques de processus de réalisation. Cependant, ils ne documentent et n'abordent pas la partie ordonnancement du processus de réalisation. De manière similaire, les travaux de (Aldanondo et Vareilles, 2008) ont proposé et relié des modèles génériques de produit et de processus de réalisation sans aborder l'aspect ordonnancement des processus.

1.3.4 Conclusions sur couplage

L'ensemble des travaux mentionnés montrent clairement que l'association ou la mise en relation des processus de conception de produit et de planification de la réalisation est une problématique actuelle. Même si la plus grande partie des travaux s'intéresse avant tout à l'identification de liens entre entités des deux domaines afin de caractériser ou formaliser des dépendance entre structure produit et structure projet, il apparaît cependant clairement que la définition et l'outillage de la propagation des conséquences des décisions entre les deux processus constituent une problématique qui mérite d'être étudiée.

Dans ce dernier but de définition et d'outillage du couplage, nous retenons :

– les quatre domaines besoins, solutions, processus et ressources proposés par Lindemann (Lindemann, 2007) lors de l'exploitation des matrices de conception *DSM* et *MDM*,
– le résultat de conception comme étant une association entre les deux premiers domaines besoin et solution dans le prolongement des travaux sur la conception axiomatique,
– la planification de projet comme le processus *project time management* du PMI regroupant l'élaboration du graphe de tâche et l'ordonnancement des tâches qu'il est tout à fait possible d'associer aux deux derniers domaines processus et ressources.

Nos travaux s'intéressant plus particulièrement à la configuration ou conception infiniment routinière et à l'assistance interactive et non à la résolution, nous retenons également :

– les principes de configuration mettant en œuvre un modèle générique et une instanciation de ce modèle générique, aussi bien pour la configuration de produit que pour la planification de projet,
– l'aptitude *a priori* des approches par contraintes à formaliser et à aider interactivement, aussi bien la configuration de produit que la planification de projet.

Dans le but de réduire l'étendue des investigations :

– nous nous concentrerons avant tout sur les interactions entre la conception et la planification et nous limiterons nos propositions sur le couplage se déroulant sur un même niveau d'abstraction. Les problèmes mutli-niveaux ne seront donc pas considérés,
– nous retiendrons également pour la planification l'approche courante en gestion de projet considérant les contraintes de délais dans une première étape, suivie d'une étape prenant en compte les ressources. Ceci permettra alors d'exploiter les capacités de filtrage des approches par contraintes et de proposer une aide interactive.

1.4 Problèmes de satisfaction de contraintes

Nous présenterons dans cette section les raisonnements à base de contraintes ou problèmes de satisfaction de contraintes. Dans un premier temps (section 1.4.1), nous définirons formellement les problèmes de satisfaction de contraintes et dans un deuxième temps (sections 1.4.2 et 1.4.3), nous passerons en revue les différents types de problèmes de satisfaction de contraintes : nous les classifierons en fonction du type de leurs variables, du type de leurs contraintes et de la nature de celles-ci. Puis, nous exposerons succinctement les méthodes permettant la recherche de solutions de ces problèmes et quels sont leurs apports en aide à la décision (section 1.4.4). La première méthode indique si des choix réalisés conduisent effectivement à une solution, nous parlerons alors de filtrage ou propagation ; la seconde fournit un ensemble de solutions, nous parlerons alors de résolution.

1.4.1 Définition d'un problème de satisfaction de contraintes

Les problèmes de satisfaction de contraintes ou *CSP (Constraint Satisfaction Problems)* permettent de modéliser de la connaissance et de raisonner sur celle-ci afin de trouver l'ensemble des solutions compatibles avec un problème courant. Les premiers problèmes de satisfaction de contraintes ont été définis par (Montanari, 1974) il y a une quarantaine d'années.

Définition 2 : ***problèmes de satisfaction de contraintes***

Les problèmes de satisfaction de contraintes sont définis comme un triplet $(\mathbb{V}, \mathbb{D}, \mathbb{C})$ *où :*

- $\mathbb{V} = \{v_1, v_2, ..., v_k\}$ *est un ensemble fini de variables,*
- $\mathbb{D} = \{d_1, d_2, ..., d_k\}$ *est un ensemble fini de domaines de définition des variables,*
- $\mathbb{C} = \{c_1, c_2, ..., c_m\}$ *est un ensemble fini de contraintes portant sur les variables où une contrainte décrit les combinaisons autorisées ou exclues des valeurs des variables.*

Réaliser un outil d'aide à la décision à partir des concepts de *CSP* revient à traduire les connaissances soit sous forme d'élément unique de $\mathbb{C}$, soit sous forme de plusieurs éléments répartis sur le triplet $(\mathbb{V}, \mathbb{D}, \mathbb{C})$ (Vernat, 2004). C'est le niveau d'abstraction de la connaissance qui va déterminer si celle-ci est directement utilisable sous la forme d'une contrainte, ou si elle doit être décomposée.

Le modèle de connaissances est alors confondu avec le problème de satisfaction de contraintes. Trouver les solutions d'un problème donné revient à résoudre le problème de satisfaction de contraintes.

Définition 3 : *solution d'un* CSP

Une solution d'un problème de satisfaction de contraintes est une instanciation de toutes les variables respectant toutes les contraintes.

Les problèmes de satisfaction de contraintes peuvent être classés selon plusieurs critères (Gelle et Faltings, 2003), (Vareilles, 2005) :

- le type de variables sur lesquels ils portent : symbolique, continue, temporelle...
- le type de contraintes présentes : liste de valeurs autorisées, fonction mathématique et relation temporelle,
- et sur la nature des contraintes mises en jeu : contrainte de compatibilité permettant de réduire l'espace de solution et contrainte d'activation, permettant de modifier la structure du problème.

Nous distinguons donc plusieurs types de *CSP* en fonction du type de variables manipulées et en fonction de leur structure statique ou dynamique.

1.4.2 *CSP* à structure statique

Les *CSP* à structure statique sont des *CSP* dans lesquels la totalité des variables de $\mathbb{V}$ et la totalité des contraintes de $\mathbb{C}$ caractérisent l'ensemble des solutions. Aucune variable, ni contrainte n'est rajoutée au fil de la résolution du problème. Nous pouvons distinguer quatre types de *CSP* statiques en fonction du type de variables qu'ils manipulent, tel qu'illustré en figure 1.7.

1.4.2.1 CSP *discrets*

Les *CSP* discrets ont été le premier type de *CSP* défini par (Montanari, 1974). Ils se caractérisent par le fait que :

- leurs variables sont discrètes : elles peuvent être soit décrites par des listes de symboles, soit par des intervalles et des listes d'entiers, soit par des listes de réels.

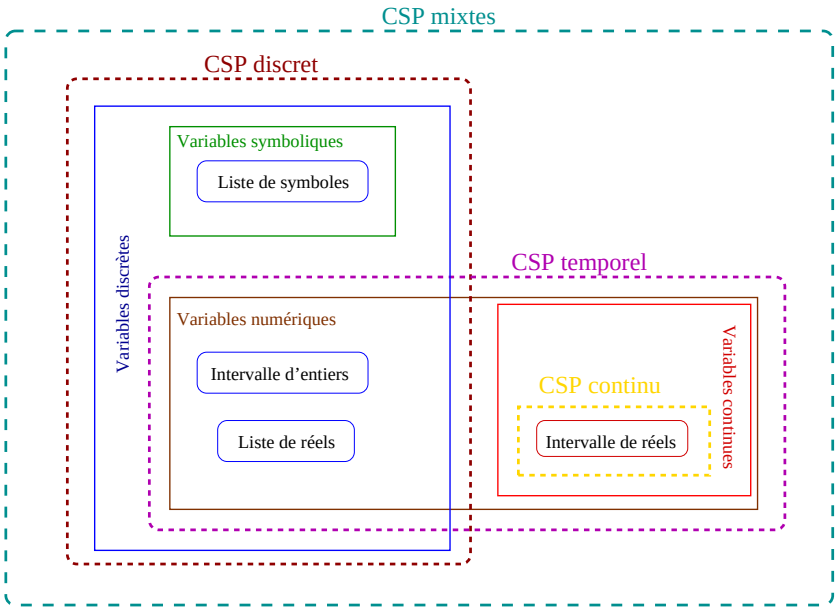


FIGURE 1.7: Types de *CSP* statiques

– leurs contraintes sont des contraintes de compatibilité qui permettent de définir les combinaisons de valeurs autorisées pour un ensemble de variables. Ces contraintes peuvent être décrites aussi bien en extension qu'en intension. Les contraintes de compatibilité exprimées en extension sont représentées par des listes de n-uplets indiquant quelles sont les valeurs compatibles entre elles. Les contraintes de compatibilité exprimées en intension sont représentées soit par des listes de combinaisons factorisées de valeurs autorisées (intervalle d'entiers ou de réels), soit par des expressions mathématiques discrètes. Certaines contraintes de compatibilité peuvent aussi exprimer une liste de n-uplets interdits (Tsang, 1993).

Par exemple, soit le *CSP* discret composé de deux variables X et Y, de domaine respectif $D_x = \{a, b\}$ et $D_y = \{[1,5]\}$ et d'une contrainte cc_1 de compatibilité réduisant l'espace des solutions possibles. Cette contrainte peut être exprimée soit en extension, soit en intension, soit exprimée *en exclusion* $\overline{cc_1}$ par la liste des combinaisons interdites, comme illustré par le tableau 1.1.

cc_1 en extension

X	Y
a	1
a	2
a	3
b	4
b	5

cc_1 en intension

X	Y
a	[1, 3]
b	[4, 5]

cc_1 en exclusion : $\overline{cc_1}$

X	Y
a	$\neq$ [4, 5]
b	$\neq$ [1, 3]

TABLE 1.1: Différentes représentations de contraintes de compatibilités discrètes

1.4.2.2 CSP *continus*

Les *CSP* continus se caractérisent par le fait que :

- leurs variables sont numériques continues : elles ne peuvent être décrites que par des intervalles de réels,
- leurs contraintes sont des contraintes de compatibilité, qui permettent de définir les combinaisons de valeurs autorisées pour un ensemble de variables. Ces contraintes sont décrites soit de manière générale comme des fonctions mathématiques, soit plus rarement par des contraintes de compatibilité continues.

Par exemple, soit le *CSP* continu composé de deux variables réelles Z et R, de domaine respectif $D_Z = \{[0,5]\}$ et $D_R = \{[0,10]\}$ et deux contraintes cc_1 et cc_2 de compatibilité réduisant l'espace des solutions possibles. Ces contraintes peuvent être exprimées soit en intension sous forme de fonction mathématique, soit en extension relative sous forme de liste d'intervalles autorisés, comme illustré par le tableau 1.2.

cc_1 et cc_2 en intension

cc_1 : $Z \leq R$

cc_2 : $R \leq 4$

cc_1 et cc_2 en extension relative

Z	R
$[0,4]$	4
$[0,4]$	$[2,4]$
$[0,2]$	$[0,2]$

TABLE 1.2: Différentes représentations de contraintes de compatibilités numériques

1.4.2.3 CSP *temporels*

Il existe deux courants de pensée pour gérer le temps en *CSP* :

- soit les variables temporelles représentent des intervalles I de temps décrits par un début, noté i^+ et d'une fin, notée i^- : nous parlons dans ce cas-là de *CSP* temporel *qualitatif*. Les contraintes temporelles représentent des relations symboliques entre intervalles pour lesquelles Allen a proposé treize primitives telles que *avant*, *après* ou *pendant* (Allen, 1983). Ces relations temporelles permettent de positionner les intervalles de manière relative les uns par rapport aux autres.
- soit les variables temporelles représentent des instants et des événements : nous parlons alors de *CSP* temporel *quantitatif* (Dechter et Pearl, 1991). Les contraintes temporelles représentent des distances temporelles entre des instants tel que l'instant j peut arriver entre 0 et 3 unités de temps après l'instant i : $0 < i - j < 3$, encore noté $\{[0,3]\}_{i,j}$. Ces relations temporelles permettent de positionner les instants de manière absolue les uns par rapport aux autres.

Il est à noter que (Meiri, 1996) a proposé de combiner ces deux types de *CSP* temporels pour augmenter l'expressivité des contraintes temporelles.

Les *CSP* temporels *quantitatifs* sont le plus utilisé pour la planification sous contraintes pour leur pouvoir expressif, leur gestion numérique et continue du temps et le positionnement relatif des tâches d'un projet de manière absolue par rapport à une date de début t_0 (Bartak et al., 2010).

1.4.2.4 CSP *mixtes*

Nous étendons la définition des *CSP* proposée par (Gelle et Weigel, 1995) par le fait que :

- leurs variables sont discrètes, numériques ou *temporelles*,

– leurs contraintes regroupent des contraintes discrètes, continues, temporelles et mixtes en faisant intervenir des variables de tous types.

Par exemple, soit le *CSP* mixte composé de deux variables, l'une réelle D et l'une symbolique Q, de domaine respectif $D_D = \{[0,5]\}$ et $D_Q = \{a,b\}$ et d'une contrainte de compatibilité cm_1 réduisant l'espace des solutions possibles. Cette contrainte faisant intervenir deux variables de types différents est décrite en intension 1.3.

TABLE 1.3: Exemple de contrainte de compatibilité mixte

cm_1

D	Q
$[0,2]$	a
$[0,4]$	b

1.4.3 *CSP* à structure dynamique

Les *CSP* à structure dynamique sont des *CSP* dans lesquels un sous-ensemble des variables de $\mathbb{V}$ et un sous-ensemble des contraintes de $\mathbb{C}$ caractérisent un ensemble maximal de solutions. La structure du *CSP* peut être modifiée par l'ajout de variables et l'espace de solutions réduit par l'ajout de contraintes supplémentaires au fil de la résolution du problème. Les *CSP* à structure dynamique permettent donc de gérer la *pertinence* de leurs variables et de leurs contraintes. Nous entendons par *pertinence* la possibilité d'ajouter au fil de la résolution des éléments au *CSP* par l'activation implicite ou non de ceux-ci (Van Oudenhove de Saint Géry, 2006). Nous pouvons distinguer quatre approches de *CSP* à structure dynamique en fonction du mécanisme mis en œuvre pour la gestion de la pertinence de leurs éléments.

1.4.3.1 CSP *

Les *CSP* * permettent de gérer la pertinence d'une variable *via* l'ajout d'une valeur spécifique au domaine [2] (Amilhastre, 1999). Cette approche n'ajoute pas d'éléments à proprement parlé au *CSP* en cours de traitement, mais permet d'indiquer quelles sont les variables qui n'appartiennent pas à la solution.

Les *CSP** se caractérisent par le fait que :

– leurs variables sont discrètes et possèdent, pour certaines, la valeur $\star$ dans leur domaine,

– leurs contraintes sont des contraintes de compatibilité qui permettent de définir les combinaisons de valeurs autorisées pour un ensemble de variables, incluant la valeur spécifique $\star$.

Par exemple, soit le *CSP* * composé de deux variables A et B, de domaine respectif $D_A = \{a_1, a_2, \star\}$ et $D_B = \{b_1, b_2\}$ et une contrainte de compatibilité réduisant l'espace des solutions possibles $cc_\star$. Cette contrainte est décrite en intension 1.4.

2. Le symbole spécifique choisi est $\star$ dans le cas des *CSP* * et *DC* pour « *don't care* » dans le cas de (McDonald et Prosser, 2002).

TABLE 1.4: Exemple de contrainte de compatibilité des *CSP* *

$cc_\star$

A	B
a_1	b_1
a_2	b_1
$\star$	b_2

1.4.3.2 CSP *conditionnels*

Les *CSP* conditionnels ou dynamiques, notés *DCSP*, ont été introduits par (Mittal et Falkenhainer, 1990) pour gérer de manière explicite la pertinence de variables, en autorisant ou en interdisant explicitement leur activation.

Définition 4 : CSP *conditionnels*

Les CSP *conditionnels sont définis comme un triplet* $(\mathbb{V},\mathbb{D},\mathbb{C})$ *où :*

- $\mathbb{V} = \{v_1, v_2, ..., v_l\}$ *est un ensemble fini de variables discrètes composé de deux sous-ensembles disjoints :*
  - *un ensemble* $\mathbb{V}_a$ *non vide de variables initialement actives,*
  - *un ensemble* $\mathbb{V}_i$ *de variables initialement inactives.*
- $\mathbb{D} = \{d_1, d_2, ..., d_l\}$ *est un ensemble fini de domaines de définition des variables de* $\mathbb{V}$*,*
- $\mathbb{C}$ *est un ensemble fini de contraintes de deux natures possibles :*
  - *un ensemble* $\mathbb{C}_C = \{c_1^c, c_2^c, \ldots, c_n^c\}$ *de contraintes de compatibilité,*
  - *un ensemble* $\mathbb{C}_A = \{c_1^a, c_2^a, \ldots, c_k^a\}$ *de contraintes d'activation. Les contraintes d'activation sont composées d'une prémisse* P *et d'un conséquent* $\mathbb{V}$ *et servent à gérer la pertinence de variables dont l'existence est conditionnée. Elles sont notées* $C_A : P \xrightarrow{ACT} \mathbb{V}$ *ou* $P \xrightarrow{\overline{ACT}} \mathbb{V}$*, où* $\mathbb{V}$ *représente un ensemble de variables appartenant à* X_I*.*

Mittal et Falkenhainer ont ainsi été les premiers à définir la notion d'état de variable et à proposer et définir quatre types de contraintes d'activation permettant de modifier ces états explicitement. Il est à noter que les contraintes de compatibilité sont pertinentes uniquement lorsque toutes les variables sur lesquelles elles portent sont actives : elles deviennent donc pertinentes implicitement.

L'expressivité des prémisses et des conséquents des contraintes d'activation fut étendue par la prise en compte d'autres conditions logiques portant sur des variables numériques (Gelle et Weigel, 1995), sur leur type *objet* (Gelle, 1998), sur l'activation de sous-ensembles de variables des conséquents (Soininen et Gelle, 1999), ainsi qu'aux *CSP* temporels (Tsamardinos et al., 2003), (Vilim et al., 2004), (Mouhoub et Sukpan, 2005a).

Par exemple, soit le *DCSP* composé de deux variables : B initialement active et A initialement inactive, de domaine respectif $D_A = \{a_1, a_2\}$ et $D_B = \{b_1, b_2\}$et B et d'une contraint d'activation ca permettant d'ajouter la variable A au problème courant lorsque B vaut b_1. Cette contrainte d'activation s'écrit donc $ca : B = b_1 \xrightarrow{ACT} A$.

1.4.3.3 CSP *à état*

Les *CSP* à états, notés *CSPe*, ont été introduits par (Veron, 2001) et reprennent les notions présentées dans les *CSP* conditionnels de (Mittal et Falkenhainer, 1990), mais en associant à chacune des variables de base v_b du problème une variable booléenne d'état v_e. La valeur 0 (resp. 1) d'une variable d'état v_e indique que la variable de base associée v_b ne participe pas (resp. participe) à la solution. Les variables de base dont les variables d'état ne sont pas réduites à un singleton, sont dans un état indéterminé.

Définition 5 : *CSP à états*

Les CSP *à états sont définis comme un triplet* $(\mathbb{V},\mathbb{D},\mathbb{C})$ *où :*

- $\mathbb{V} = \{(v_{b1},v_{e1}),(v_{b2},v_{e2}),...,(v_{bl},v_{el})\}$ *est un ensemble fini de couples de variables composé :*
  - *d'un ensemble* $\mathbb{V}_b = \{v_{b1},v_{b2},\ldots,v_{bl}\}$ *de variables de base discrètes ou continues,*
  - *d'un ensemble* $\mathbb{V}_e = \{v_{e1},v_{e2},\ldots,v_{el}\}$ *de variables booléennes d'état.*
- $\mathbb{D} = \{D_b,D_e\}$ *est un ensemble fini de domaines de définition des variables de* $\mathbb{V}$ *où :*
  - D_b *est l'ensemble des domaines des variables* $\mathbb{V}_b$,
  - D_e *est l'ensemble des domaines booléens des variables* $\mathbb{V}_e$.
- $\mathbb{C}$ *est un ensemble fini de contraintes portant sur l'ensemble des variables* $\mathbb{V}$ *de deux natures possibles :*
  - *un ensemble* $\mathbb{C}_C = \{c_1^c,c_2^c,\ldots,c_n^c\}$ *de contraintes de compatibilité,*
  - *un ensemble* $\mathbb{C}_A = \{c_1^a,c_2^a,\ldots,c_k^a\}$ *de contraintes d'activation.*

Les prémisses des contraintes d'activation des *CSP* à états reprennent celles définies dans les *CSP* conditionnels en utilisant les variables de base et les variables d'état. Elles s'assurent, de plus, que les domaines des variables de base activées contiennent encore une valeur cohérente avec le problème courant.

1.4.3.4 CSP *composites*

Les *CSP* composites ou *CCSP* ont été introduits par (Sabin et Freuder, 1996) pour modéliser la structure hiérarchique des problèmes. Nous devons noter que les *CSP* à état permettent aussi de représenter cette structure hiérarchique des problèmes en associant un état non plus à une seule variable, mais en le factorisant pour un groupe de variables.

Dans les *CSP* composites, une variable peut prendre pour valeur un sous-problème ; ces variables sont appelées méta-variables. Dès qu'elles sont valuées, elles activent l'ensemble du sous-problème comprenant un ensemble de variables et de contraintes. Les valeurs des variables des *CSP* composites peuvent être substituées par des sous-problèmes : la structure du modèle n'est plus modifiée localement (comme dans les *CSP* conditionnels), mais globalement.

Définition 6 : CSP *composites*

Un CCSP *est défini par un triplet* $(\mathbb{V},\mathbb{D},\mathbb{C})$, *de la même façon qu'un* CSP *discret. Cependant, la différence entre* CCSP *et* CSP *est que les valeurs des variables peuvent être des sous-problèmes entiers :* $P' = (\mathbb{V}',\mathbb{D}',\mathbb{C}')$. *Lorsqu'une variable* V_i *prend pour valeur* P', *le problème à résoudre est modifié et devient :* $P" = (\mathbb{V}'',\mathbb{D}'',\mathbb{C}'')$, *avec :*

- $\mathbb{V}'' = \mathbb{V} \cup \mathbb{V}' \backslash V_i$;
- $\mathbb{D}'' = \mathbb{D} \cup \mathbb{D}' \backslash D_i$;
- $\mathbb{C}'' = \mathbb{C} \cup \mathbb{C}' \backslash C(V_i)$, *où* $C(V_i)$ *est l'ensemble de contraintes portant sur* V_i.

Les *CSP* composites ont été étendus aux *CSP* temporels par (Mouhoub et Sukpan, 2005a).

1.4.4 Filtrage et résolution

Il existe plusieurs manières de traiter un *CSP*. Nous pouvons vouloir :

- réduire les domaines des variables en supprimant les valeurs ne menant pas à une solution,
- trouver juste une solution,
- trouver toutes les solutions,
- trouver la solution optimale, ou du moins une bonne solution étant donné un objectif portant sur quelques variables ou toutes les variables.

1.4.4.1 Filtrage

Les techniques de filtrage utilisent les contraintes pour effectuer des déductions sur le problème en détectant les affectations partielles localement ou totalement incohérentes [3]. L'une des techniques les plus utilisées est la technique de renforcement de la cohérence locale ou cohérence d'arc (Montanari, 1974). La cohérence d'arc vérifie que toute valeur du domaine d'une variable est compatible avec chaque contrainte prise séparément. Il existe des méthodes de filtrage basées sur la cohérence d'arc spécifiques aux différents types de *CSP* présentés en section 1.4.1. Nous pouvons citer :

- les techniques de cohérence d'arc (arc-cohérence, chemin-cohérence) principalement utilisées dans la cas des *CSP* discrets ou mixtes (Mackworth, 1977), (Bessière et Cordier, 1993), (Bessière et Régin, 1994), (Faltings, 1994),
- les techniques de 2*B*-cohérence (Lhomme, 1993) ou *Box*-cohérence (Benhamou et al., 1994), (Benhamou, 1996) pour les *CSP* continus et temporels quantitatifs (Rossi et al., 2006).

Il existe plusieurs degrés de filtrage qui permettent de vérifier la cohérence de n-uplets de valeurs de variables. Le degré de filtrage correspond au nombre de variables participant à la vérification de la cohérence locale. Plus le degré est grand, meilleure sera la détection des combinaisons incohérentes, mais plus il faudra de temps pour les détecter.

Ces méthodes de filtrage utilisées permettent de répercuter des choix sur le problème courant en éliminant les valeurs devenues incohérentes. La recherche de solutions est alors interactive : c'est la séquence de choix cohérents qui conduit à une ou plusieurs solutions.

1.4.4.2 Résolution

Nous présentons ici deux méthodes de résolution : l'une basée sur des algorithmes de résolution, l'autre basée sur la compilation d'un *CSP* en un automate à états finis.

3. Une affectation est partielle lorsque seul un sous-ensemble des variables est instancié.

1.4.4.2.1 Algorithme de résolution. Les méthodes complètes de résolution de *CSP* explorent de manière systématique l'espace de recherche et sont capables de fournir toutes les solutions d'un problème. L'algorithme de recherche de base le plus souvent utilisé est l'algorithme de retour arrière ou *backtrack* (Golomb et Baumert, 1965). Cet algorithme met en place une stratégie de *profondeur d'abord*, avec un mécanisme de retour arrière sur la situation précédente lorsqu'il détecte que l'affectation partielle courante n'est pas cohérente. Cet algorithme est souvent amélioré par des heuristiques déterminant, par exemple, l'ordre des variables à instancier et l'ordre des valeurs à tester pour minimiser le nombre de branches à explorer.

Les méthodes de résolution dites incomplètes n'explorent pas de façon systématique l'espace de recherche. Elles sont basées sur une exploration opportuniste de l'ensemble des affectations complètes [4] et ne fournissent qu'un sous-ensemble de solutions. Elles nécessitent une fonction d'évaluation et de comparaison d'affectations. Nous pouvons citer la méthode de recherche tabou (Glover et Laguna, 1993) ou le recuit simulé (Kirkpatrick et al., 1983). Ces méthodes incomplètes sont généralement utilisées pour résoudre des problèmes de taille élevée.

Les algorithmes de résolution sont souvent couplés à des méthodes de filtrage afin d'améliorer la résolution en supprimant les valeurs incohérentes des domaines des variables. Par exemple, le *Forward checking* (Haralick et Elliot, 1980) combine le filtrage par arc-cohérence avec un algorithme de retour arrière. Une comparaison des différents couplages peut être trouvée dans (Lobjois et Lemaitre, 1997).

Ces méthodes de résolution complète ou incomplètes permettent de résoudre indifféremment des *CSP* discrets, continus après discrétisation des domaines de variables et mixtes. Une revue des différentes méthodes de résolution actuelle peut être trouvée dans le *Handbook of Constraint Programming* (Rossi et al., 2006).

1.4.4.2.2 Diversité compilée Certaines approches par contraintes sont basées sur la transformation d'un *CSP* par compilation en un automate à états finis pour générer toutes les successions d'instanciations possibles (Vempaty, 1992). Ce type de méthode a l'avantage d'éviter les retours arrière une fois l'automate compilé et ne pose pas de problèmes de qualité de filtrage. Cependant, cette approche a trois inconvénients par rapport à nos besoins :

- les *CSP* continus ou mixtes ne peuvent être pris en compte. En effet, un *CSP* continu ou mixte ne peut fondamentalement pas être modélisé sous forme d'un automate à états fini, étant donné que certaines de ses variables ont des domaines infinis ;
- le temps de compilation peut constituer un problème suivant la taille du modèle de connaissances ;
- il n'existe pas de mécanismes de gestion de la pertinence (de variables, de sous-ensembles, de contraintes) : un automate compilé est une structure statique.

Du fait de ces inconvénients, nous ne prendrons pas en compte les systèmes basés sur des automates à états finis dans la suite de ce mémoire.

1.4.5 Conclusion sur les *CSP*

Dans la suite de ce mémoire, nous utiliserons des *CSP*à structure statique et dynamique pour modéliser le couplage entre la configuration de produit et la configuration du processus de réalisation. Plus particulièrement, nous assemblerons les *CSP*discrets, les *CSP*continus, les *CSP**

4. Une affectation est complète lorsque l'ensemble de toutes les variables est instancié.

et les *DCSP* pour modéliser notre problème de couplage et illustrer nos propositions. Étant en aide à la décision interactive, nous nous baserons uniquement sur les méthodes de filtrage des différents types *CSP*.

1.5 Terrain d'application

Nous présentons, dans cette section, la problématique industrielle à l'origine de nos travaux, à savoir le couplage de la conception de systèmes et de la gestion de projet de réalisation. Ces travaux s'inscrivent dans un projet ANR RNTL [5] nommé *ATLAS* pour *Aides et assisTances pour la conception, la conduite et leur coupLage par les connAissanceS*. Dans un premier temps, nous définirons ce que nous entendons par couplage entre conception de systèmes et planification de projet et nous évoquerons les différents principes de couplage identifiés. Dans un deuxième temps, nous présenterons l'organisation du projet en lots de travail et nous positionnerons nos travaux par rapport à ceux-ci.

1.5.1 Problématique industrielle

Le projet *ATLAS* vise à rapprocher la conception de systèmes et la planification de projet afin d'aider les entreprises à mettre sur le marché des systèmes de plus en plus compétitifs en prenant en compte au plus tôt les contraintes de conception et celles de projet. Une étude menée auprès d'une quinzaine d'entreprises du pôle *Aerospace Valley* a permis de conforter le besoin d'outils permettant de rapprocher ou de coupler ces deux domaines de manière interactive (Abeille et al., 2009). Nous posons la définition du couplage suivante :

Définition 7 : *Couplage*

Nous entendons par couplage la possibilité de propager, sur un même niveau de décomposition, les décisions prises en conception sur le projet et inversement, du projet vers la conception. Cette notion de couplage repose sur une hypothèse forte de bijection entre la conception de systèmes et la gestion de projet, tel qu'illustré en figure 1.8.

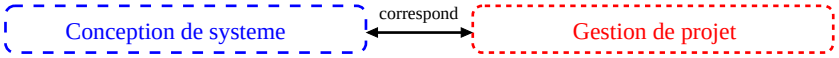


FIGURE 1.8: Hypothèse de bijection entre la conception de système et la gestion de projet.

Cinq types de couplage ont été identifiés et reposent sur la disponibilité et la source des connaissances mises en jeu pour aider à la décision de manière interactive (Aldanondo et al., 2010) :

- trois types exploitent exclusivement des connaissances méthodologiques :
  - le couplage structurel : ce type de couplage permet d'assurer la bijection entre la conception et la gestion de projet, quelque soit le niveau de décomposition. En effet, la décomposition d'un système complexe en n sous-systèmes élémentaires aura pour conséquence la décomposition du projet associé en n sous-projets (et inversement). Chaque sous-système sera donc associé à son propre sous-projet.

5. projet n° ANR-07-TLOG-002.

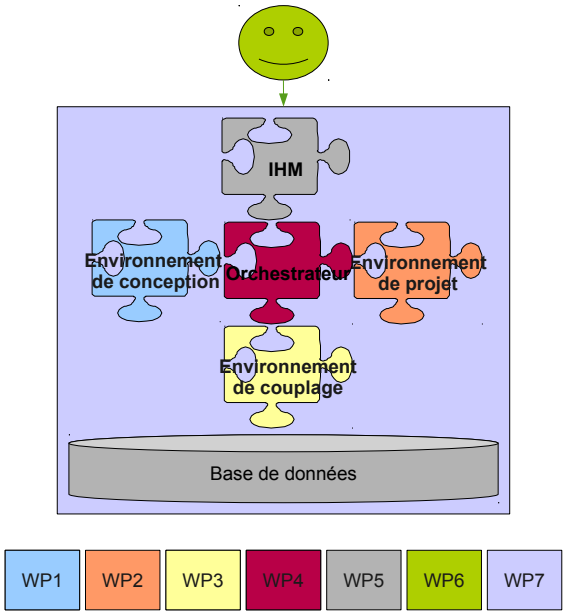


FIGURE 1.9: Architecture de la plate-forme *ATLAS* et positionnement des lots de travail.

  - le couplage informationnel : ce type de couplage permet d'alerter l'un des deux environnements qu'un événement majeur (choix de composant, modification de délais) est survenu dans l'autre environnement et que celui-ci peut impacter son propre déroulement.
  - le couplage décisionnel : ce type de couplage permet de proposer aux décideurs un tableau de bord regroupant des indicateurs de conception (indicateurs de performance, d'avancement) et des indicateurs de projet (coût engagé, risque) afin de les aider à prendre la meilleure décision.
- deux types exploitent à la fois des connaissances méthodologiques et des connaissances métier formalisées :
  - soit sous forme de cas passés regroupés dans une base de cas. Un cas sera donc composé de deux parties : une partie permettant de décrire la conception et une seconde permettant de décrire le projet associé. Une recherche de cas passé pourra ainsi permettre une réutilisation partielle ou complète d'un couple (conception, projet).
  - soit sous forme de contraintes regroupées dans un *CSP* mixte décrivant à la fois des connaissances de conception, de projet et de couplage et exploitant des techniques de filtrage.

Le projet *ATLAS* doit aboutir à la mise au point d'une plate-forme logicielle permettant de supporter le couplage entre conception et projet.

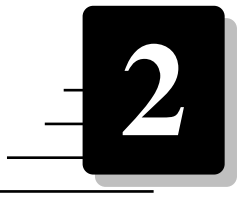
1.5.2 Organisation du projet ATLAS

Le projet *ATLAS* est organisé en sept lots de travail, nommés *WP* pour *Work Package* et reprend l'architecture logicielle de la plate-forme, présentée en figure 1.9 :

- Le *WP1* se concentre sur la mise au point de l'environnement support de conception de systèmes et sur des mécanismes de capitalisation et de réutilisation de l'expérience.

- Le *WP2* se concentre sur la mise au point de l'environnement support de la gestion de projet et sur des mécanismes de planification et de réutilisation de projets.
- Le *WP3* se concentre sur la définition du couplage entre la conception de systèmes et la gestion de projet et sur la formalisation des échanges de données entre ces deux modules logiciels.
- Le *WP4* a pour but de spécifier le système d'information complet, la base de données, ainsi que l'orchestrateur, en se basant sur les livrables et les maquettes logicielles fournis par les *WP1*, *WP2* et *WP3*.
- Le *WP5* intègre les différents modules logiciels dans un environnement global d'aide à la décision et développe l'orchestrateur, la base de données, ainsi que l'interface homme-machine de la plate-forme.
- Le *WP6* a pour but de fournir des exemples industriels de validation de la maquette globale et des principes de couplage évoqué.
- Le *WP7* correspond à l'organisation générale du projet.

Nos travaux se concentrent sur l'étude de la faisabilité du couplage entre la configuration de produit et la planification de projet sur un seul niveau de décomposition, couplage basée sur des connaissances méthodologiques et métier représentées comme un *CSP* dans le cadre du lot de travail *WP3*.



Couplage et propagation de contraintes

Introduction

Dans ce chapitre, nous étudions le couplage de la configuration de produit avec la planification du projet associé à sa production. Nous commençons, dans cette introduction, par établir une typologie des problèmes identifiés tant en configuration qu'en planification. Nous présentons, par la suite, de manière synthétique, les outils logiciels utilisés pour implémenter ces problèmes. Chacune de nos propositions est illustrée sur un exemple fil rouge d'avion de tourisme et d'affaire. Le plan de ce chapitre, composé de trois sections homogènes mettant en évidence les éléments communs, conclut cette introduction.

Typologie de problèmes : Les travaux présentés dans ce chapitre reposent sur des types de problèmes liés à différents aspects que peut recouvrir la diversité des produits et des processus de production. Cette diversité repose sur des problèmes génériques mettant en œuvre des entités devant être valuées (c'est-à-dire entités pour lesquelles une valeur est attribuée) et éventuellement ajoutées au problème lors de son traitement. Les différentes possibilités d'ajout conduisent à quatre types de problèmes :

- Sélection de valeur d'entité. Dans ce cas, toutes les entités décrivant le problème sont présentes dans toutes les solutions des problèmes de configuration et de planification. Il faut donc leur attribuer une valeur unique afin d'aboutir à une solution. Ce problème est dénommé élémentaire. Ceci peut correspondre par exemple :
  - en configuration, à la sélection d'un moteur dans une liste de moteurs ou à la sélection d'un nombre de sièges entre plusieurs valeurs possibles,
  - en planification, à la sélection d'une machine pour réaliser une opération ou à la sélection d'un nombre d'opérateurs pour réaliser un assemblage.
- Ajout d'entité. Dans ce cas, certaines entités décrivant le problème ne sont pas présentes dans toutes les solutions des problèmes de configuration et de planification : elles sont dites optionnelles. Ceci peut correspondre par exemple :
  - en configuration, à l'ajout d'une entité *réservoir supplémentaire* ou à l'ajout d'une entité *propriété taille du réservoir*,

  - en planification, à l'ajout d'une entité *tâche de montage de réservoir supplémentaire*.
- Sélection d'entité à ajouter. Dans ce cas, il est obligatoire d'ajouter une entité décrivant le problème, mais plusieurs entités sont candidates et s'excluent mutuellement. Ceci peut correspondre par exemple :
  - en configuration, à la sélection d'une entité *finition* parmi les deux entités : *finition catalogue* et *finition personnalisée*,
  - en planification, à la sélection de l'entité *tâche de finition* parmi les deux entités : *tâche de finition catalogue* et *tâche finition personnalisée*.
- Le quatrième type de problème est lié à l'ajout de sous-ensembles d'entités. Il met en œuvre des notions de sous-ensembles de produits en configuration et de sous-projets en planification. Ce dernier aspect arborescent ou structuré des problèmes sera mentionné mais non détaillé dans les travaux, car notre objectif est avant tout d'étudier le couplage entre les entités de configuration et de planification se situant à un même niveau d'abstraction.

Privilégiant un couplage entre la configuration et la planification sur un seul niveau de décomposition, nous concentrerons donc nos travaux sur les trois premiers types de problèmes.

Implémentation logicielle : Certains des problèmes identifiés verront leurs modèles de contraintes implémentés et testés avec des outils de programmation par contraintes. Étant donné que nous sommes en aide à la décision interactive, nous utiliserons l'outil CoFiADe, développé au sein de notre centre de recherche et qui est dédié au filtrage des problèmes de contraintes. Nous compléterons ces expérimentations avec deux outils de résolution plus conventionnels : ILOG CP 5.5 et ECLiPSe. L'expérimentation avec ces trois outils vise :

- lors de l'implémentation : à évaluer la faisabilité et la difficulté de l'implémentation des différents problèmes,
- lors de l'utilisation : à évaluer la possibilité du filtrage des problèmes et, si c'est possible, la qualité de ce filtrage.

Nous entendons par difficulté d'implémentation le fait que l'implémentation nécessite l'écriture d'un code informatique spécifique, une traduction dans un langage particulier ou une reformulation du problème. Nous entendons par qualité de filtrage une forme d'aptitude à identifier et supprimer les valeurs de variables conduisant à une absence de solution. Les trois outils utilisés lors de nos expérimentations sont :

- CoFiADe : *Constraint Filtering for Aiding Design* (http://cofiade.enstimac.fr/) est un outil de filtrage développé par le centre Génie Industriel de l'école des mines d'Albi-Carmaux dans le cadre de plusieurs thèses (Vareilles, 2005), (Van Oudenhove de Saint Géry, 2006). Cet outil permet la modélisation de variables symboliques, entières et réelles. Les contraintes peuvent mettre en œuvre des tables de compatibilité et des formules mathématiques simples respectant les hypothèses de la 2B-cohérence (Lhomme, 1993). Il est également possible de définir des contraintes d'activation permettant d'ajouter explicitement des variables et des contraintes au problème durant le filtrage. CoFiADe est un outil dédié à la propagation de contraintes discrètes, mixtes et numériques.
- ILOG CP 5.5 : est un outil commercial de résolution de problèmes combinatoires comprenant plusieurs librairies, dont la librairie de programmation par contraintes *CP*. Il permet la modélisation de problèmes à variables discrètes, qui sont entières ou réelles. Il autorise différents types de contraintes : les tables de compatibilité, les formules mathématiques et les contraintes exprimées sous la forme de formules logiques. Il permet également l'écriture de contraintes d'activation permettant d'ajouter explicitement des contraintes au problème durant le traitement. Bien que l'outil ILOG CP 5.5 soit un outil de résolution, il exploite des

méthodes de filtrage pour améliorer sa recherche de solution. Lors de nos expérimentations, nous utiliserons la méthode PROPAGATE(). ILOG CP 5.5 est fréquemment rencontré dans les travaux scientifiques tant en planification (Bidot, 2005) qu'en configuration (Junker et Mailharro, 2003) et très souvent en optimisation linéaire (Belmokhtar et al., 2007) (Lamothe et al., 2006).

– ECLiPSe : est un outil libre de programmation par contraintes basé sur Prolog. Il comprend des librairies spécifiques à chaque type de variables comme LIB(IC) pour les entiers et les réels, la LIB(SD) pour les symboliques et la LIB(FD) pour les variables à domaine fini. Il permet d'écrire des contraintes sous la forme de tables de compatibilité, de formules mathématiques et de formules logiques. Des contraintes permettant d'ajouter explicitement des contraintes lors du traitement peuvent être également définies. Lors de nos expérimentations, nous utiliserons principalement le traitement de filtrage qu'effectue ECLiPSe lors de la vérification d'existence de solution avant résolution. Plusieurs travaux de thèse, aussi bien en configuration (Mulyanto, 2002) (Giaccobi, 2009) qu'en planification (Despontin Monsarrat, 2004), (Lizarralde, 2007) ont utilisé ECLiPSe.

Exemple fil rouge d'avion de tourisme et d'affaire : L'ensemble de ce chapitre est illustré sur un exemple correspondant à un avion de tourisme et d'affaire. L'objet de cette section est de présenter succinctement, le produit qui sera l'objet de configuration et son processus de production, qui fera l'objet de planification. Le problème de configuration de l'avion met en œuvre huit entités :

– cinq entités toujours présentes décrivent le produit :
  – *nombre de sièges* (SN) avec six valeurs possibles,
  – *vitesse de croisière* (CS) avec six valeurs possibles,
  – *distance de vol* (FR) avec six valeurs possibles,
  – *nombre de moteurs* (NEN) avec deux valeurs possibles,
  – *type de motorisation* (EN) avec six valeurs possibles,
– une entité optionnelle peut être ajoutée : *réservoir supplémentaire* (ST) avec trois valeurs possibles,
– deux entités en exclusion mutuelle correspondent à des propriétés :
  – *finition catalogue* (FCA) avec trois valeurs,
  – *finition customisée* (FCU) avec deux valeurs,

Le problème de planification de la production de cet avion met en œuvre six entités tâches :

– trois entités tâches toujours présentes décrivent :
  – *fabrication* (T_1) de durée $[10,90]$,
  – *approvisionnement* (T_2) de durée $[10,170]$,
  – *assemblage* (T_3) de durée $[10,180]$,
– une entité tâche optionnelle peut être ajoutée : *assemblage réservoir supplémentaire* (T_4) de durée $[5,40]$,
– deux entités tâches en exclusion mutuelle correspondent à :
  – *finition catalogue* (T_5) de durée $[5,40]$,
  – *finition customisée* (T_6) de durée $[60,100]$,

Ce modèle servira de base à l'illustration des différents problèmes identifiés et sera complété par des contraintes suivant les besoins spécifiques de chaque section.

Plan du chapitre : Ce chapitre se compose de trois sections homogènes portant respectivement sur la configuration de produit, section 2.1, la planification du projet associé à la production du produit configuré, section 2.2 et le couplage de ces deux domaines, section 2.3. Pour chacune de ces sections, nous retrouverons un plan similaire. En premier lieu, nous rappellerons les types de problèmes étudiés. Dans un deuxième temps, nous proposerons des éléments de modélisation sous la forme de problèmes de satisfaction de contraintes : essentiellement discret (*CSP*) pour la partie configuration, temporels qualitatifs (*TCSP*) pour la partie planification et mixte, car mêlant des variables de configuration (variables discrètes) à celles de planification (variables temporelles continues) pour la partie couplage. Dans un troisième temps, les possibilités de filtrage des modèles *CSP* sont discutées. Chacune de nos propositions sera illustrée par l'exemple de l'avion de tourisme et d'affaire. Enfin, des expérimentations logicielles termineront chacune des trois sections.

2.1 Configuration et *CSP*

Dans cette section, nous nous intéressons au problème de configuration uniquement. L'étude est réalisée en quatre étapes documentant successivement : le problème élémentaire, sous-section 2.1.1, la prise en compte des éléments optionnels, sous-section 2.1.2, la sélection d'éléments en exclusion, sous-section 2.1.3 et enfin, la prise en compte de sous-ensembles optionnels, sous-section 2.1.4. Pour chaque étape et une fois le problème défini, nous nous intéresserons à sa modélisation en *CSP*, à ses possibilités de filtrage et illustrerons la modélisation par l'exemple d'avion de tourisme. L'emploi des outils de propagation de contraintes sur ces problèmes terminera cette section.

2.1.1 Configuration : problème élémentaire

Le problème de configuration considère que le produit générique est constitué d'une part d'un ensemble de groupes de composants complété éventuellement par un ensemble de propriétés. Nous mélangeons donc l'aspect physique (composants) et descriptif (propriétés) du produit, c'est-à-dire les entités des deux domaines besoin et solutions de l'approche axiomatique 1.1.1.2.

Tout produit configuré comprend un composant choisi dans chaque groupe et une valeur sélectionnée pour chaque propriété. Le problème élémentaire considère donc que tous les groupes de composants et toutes les propriétés sont présents dans tous les produits configurés possibles. Nous allons nous attacher dans ce qui suit, à définir le problème élémentaire de manière plus formelle et à montrer comment nous pouvons le modéliser en utilisant des *CSP*. Une fois la modélisation posée, nous illustrons nos propos sur la configuration de l'avion de tourisme et discuterons les possibilités de filtrage associées.

2.1.1.1 Définition du problème élémentaire

Dans le cadre du problème élémentaire de configuration et comme évoqué précédemment, nous considérons le produit à configurer comme un ensemble de composants et de propriétés valuées. Nous mélangeons donc l'aspect physique (composants) et descriptif (propriétés) du produit. Nous définissons notre problème élémentaire de la manière suivante :

a) Un produit configuré est un ensemble de composants et de propriétés valuées.
b) Les composants ayant les mêmes caractéristiques sont regroupés dans un groupe de composants.
c) Un produit générique met en œuvre des groupes de composants et des propriétés.
d) Les groupes de composants et les propriétés sont présents dans toutes les solutions du problème de configuration.
e) Des relations de compatibilité peuvent exister
   – entre les groupes de composants,
   – entre les propriétés,
   – entre les groupes de composants et les propriétés.
f) Une exigence du client se traduit soit par la sélection d'un composant dans un groupe soit par la sélection d'une valeur pour une propriété.
g) Une solution ou produit configuré est une instanciation du produit générique, elle comprend un composant sélectionné dans chaque groupe de composants et une valeur choisie pour chaque propriété.

Résoudre ce problème de configuration consiste soit à trouver une solution, soit toutes les solutions du problème ou à prouver que le problème n'admet pas de solutions.

2.1.1.2 Modélisation du problème élémentaire

Ce problème élémentaire de configuration peut se formaliser comme un problème de satisfaction de contraintes. La modélisation se fait en associant chaque groupe de composants et chaque propriété du produit à une variable dite de configuration. Les domaines de ces variables correspondent aux différentes références des composants et aux différentes valeurs des propriétés. Bien que certaines propriétés puissent être numériques, nous considérons néanmoins des domaines discrets pour toutes les variables (valeurs symboliques ou entières). Les relations de compatibilité sont alors associées à des contraintes de configuration et peuvent exister :

– entre les groupes de composants,
– entre les propriétés,
– entre les groupes de composants et les propriétés.

Ces contraintes expriment les compatibilités ou les incompatibilités entre valeurs de variables et sont représentées très souvent sous la forme de tables listant les combinaisons de valeurs autorisées ou interdites. Les domaines des variables étant discrets, le *CSP* associé est un *CSP* discret.

2.1.1.3 Exemple de modèle du problème élémentaire

L'avion de tourisme évoqué en introduction comprend un seul groupe de composants et quatre propriétés associés à cinq variables de configuration. Le groupe de composants correspond à la motorisation (associé à la variable EN). Les propriétés sont la vitesse de croisière (associée à la variable CS), la distance de vol (associée à la variable FR), le nombre de sièges (associé à la variable SN) et le nombre de moteurs (associé à la variable NEN). Toutes les variables, décrites par un ensemble fini de valeurs discrètes, sont soit des variables entières (distance de vol, vitesse de croisière, nombre de sièges, nombre de moteurs), soit des variables symboliques (type de motorisation).

Leurs domaines de définition sont les suivants :

– nombre de sièges (SN) : $D_{SN} = \{3, 4, 6, 8, 10, 12\}$,
– type de motorisation (EN) : $D_{EN} = \{1LP, 2LP, 1HP, 2HP, 1ECO, 2ECO\}$ où LP signifie *Low Power*, HP pour *High Power*, ECO pour *Economic*, alors que le chiffre correspond à la quantité de moteur (un pour les monomoteurs, deux pour les bimoteurs),
– vitesse de croisière (CS) : $D_{CS} = \{350, 400, 450, 500, 550, 600\}$,
– distance de vol (FR) : $D_{FR} = \{600, 800, 1000, 1200, 1400, 1600\}$,
– nombre de moteurs (NEN) : $D_{NEN} = \{1, 2\}$.

Les variables de configuration du problème sont liées par des contraintes traduisant des relations de compatibilité entre les groupes de composants et les propriétés. Dans notre exemple, ces contraintes correspondent aux tables de compatibilité suivantes :

– la contrainte C_{C1}, qui lie le nombre de sièges disponibles dans l'avion (SN) avec le type de motorisation nécessaire (EN). Afin de supporter le nombre de passagers, six combinaisons extrêmes correspondant à certaines associations (faible nombre de siège, grande puissance) et (grand nombre de sièges, faible puissance) sont exclues et représentées dans le tableau 2.1.
– La contrainte C_{C2}, qui lie la variable vitesse de croisière (CS) et le type de motorisation (EN). Cette contrainte exprime le fait qu'une vitesse élevée nécessite une motorisation performante. Cette contrainte est représentée par les combinaisons exclues dans le tableau 2.1.
– La contrainte C_{C3},qui lie la variable distance de vol (FR) à la vitesse de croisière (CS). Cette contrainte stipule que l'on ne peut pas aller loin à de grandes vitesses. Cette contrainte est décrite par les combinaisons exclues dans le tableau 2.1.
– Une contrainte lie la variable type de motorisation (EN) à la variable nombre de moteurs (NEN). Cette contrainte C_{C4}est décrite par les combinaisons possibles dans le tableau 2.1.

Le modèle du problème complet de l'exemple du problème élémentaire est représenté en figure 2.1 et comprend donc 5 variables discrètes et toujours présentes dans toutes les solutions (SN, EN, CS, FR, NEN) contraintes par quatre relations de compatibilité (C_{C1}, C_{C2}, C_{C3} et C_{C4}).

$\overline{C_{c1}}$

SN	EN
3	1HP
3	2HP
4	2HP
10	1LP
12	1LP
12	2LP

$\overline{C_{c2}}$

CS	EN
550	1LP
600	1LP
600	2LP
600	1HP
600	1ECO

$\overline{C_{c3}}$

FR	CS
1600	600
1600	550
1600	500
1400	600
1400	550
1400	500
1200	600
1200	550
1200	500

C_{c4}

EN	NEN
1LP	1
2LP	2
1HP	1
2HP	2
1ECO	1
2ECO	2

TABLE 2.1: Tables de compatibilité et d'incompatibilité du problème élémentaire

2.1.1.4 Filtrage du problème élémentaire

Pour ce problème élémentaire se formalisant comme un *CSP* discret, le filtrage fait appel à des traitements d'arc cohérence bien éprouvés, qui ne soulèvent pas de problème particulier.

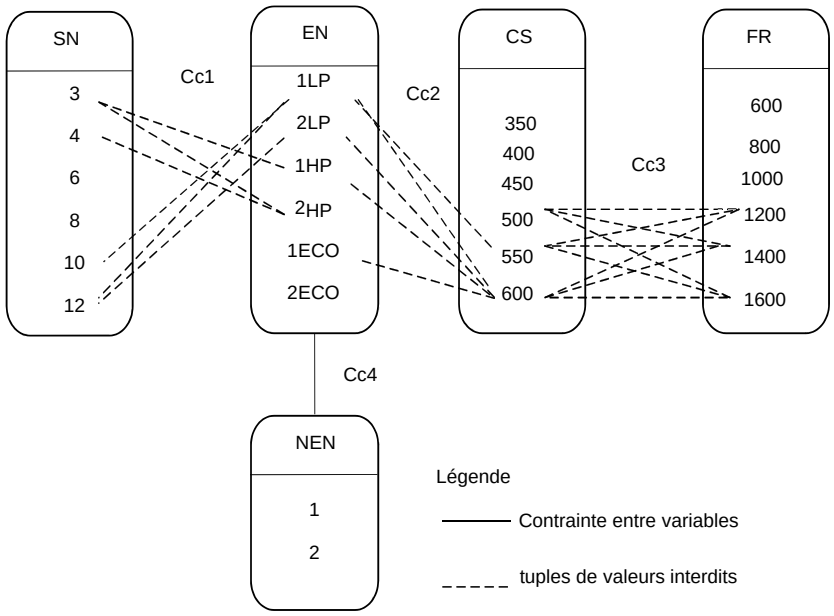


FIGURE 2.1: Configuration de l'avion de tourisme dans le cas du problème élémentaire

La restriction du domaine de n'importe quelle variable est propagée aux autres variables par l'intermédiaire des contraintes : les valeurs qui ne sont plus compatibles avec le problème courant sont donc retirées des domaines des variables.

2.1.2 Configuration : problème avec des éléments optionnels

Le problème précédent est rarement rencontré en situations réelles. Le plus souvent, les problèmes de configuration font apparaître des groupes de composants ou des propriétés du produit qui ne sont pas présents dans tous les produits configurés. Ces problèmes sont dénommés problème avec éléments optionnels. Dans l'exemple considéré, un groupe de composants *réservoir supplémentaire* doit pouvoir être ajouté pour certaines configurations de l'avion de tourisme.

2.1.2.1 Définition du problème avec éléments optionnels

La définition du problème de configuration élémentaire, présenté en section 2.1.1.1, est modifiée pour définir le problème avec éléments optionnels de la manière suivante :

a) Reste inchangé : un produit est un ensemble de composants et de propriétés valuées.

b) Reste inchangé : les composants ayant les mêmes caractéristiques sont regroupés dans un groupe de composants.

c) Reste inchangé : le produit générique met en œuvre des groupes de composants et des propriétés du produit.

d) Devient : les groupes de composants et les propriétés sont de deux types :
   - ceux qui sont présents dans toutes les solutions,
   - ceux dont la présence est conditionnée par le choix d'autres composants et/ou de valeurs de propriétés. Ils correspondent aux éléments optionnels du problème.

e) Devient : le problème de configuration comprend :
   - d'une part des relations de compatibilité entre groupes de composants et propriétés,
   - et d'autre part des relations permettant de décrire les conditions d'existence des groupes de composants et des propriétés optionnels du produit générique.

f) Reste inchangé : une exigence du client se traduit soit par la sélection d'un composant dans un groupe, soit par la sélection d'une valeur pour une propriété.

g) Devient : une solution de configuration est une instanciation du produit générique, elle comprend un composant sélectionné dans chaque groupe existant et une valeur choisie pour chaque propriété existante.

Résoudre ce problème de configuration consiste soit à trouver une solution, soit toutes les solutions du problème ou à prouver que le problème n'admet pas de solutions.

2.1.2.2 Différents modèles du problème avec éléments optionnels et exemple

Le problème de configuration avec éléments optionnels peut être formalisé à l'aide de différents types de *CSP* à structure dynamique. L'existence de l'élément peut se modéliser soit par l'ajout de valeurs particulières dans le domaine des variables ($CSP^\star$), ou par l'ajout de nouveaux éléments au problème (*DCSP*). Dans les sections qui suivent, nous allons détailler chaque modèle et les illustrer par l'exemple de l'avion de tourisme.

Dans l'exemple de l'avion de tourisme, les variables FR, CS, SN, EN et NEN, ainsi que l'ensemble des contraintes de compatibilité C_{C1}, C_{C2} , C_{C3}, C_{C4} définies en section précédente, restent inchangées et correspondent à des éléments qui sont présents dans toutes les solutions du problème de configuration. Un groupe de composants optionnel, *réservoir supplémentaire* (associé à une variable ST), est ajouté au problème suivant une condition d'ajout C_{CA} dépendant d'une combinatoire entre la vitesse de croisière (CS) et la distance de vol (FR).

- Réservoir supplémentaire (ST) : D_{ST}= $\{ST_{100}, ST_{200}, ST_{300}\}$ où le nombre correspond à la capacité du réservoir.
- Condition d'ajout de ST : $(CS > 380 \wedge FR > 1100) \vee (CS > 480 \wedge FR > 700)$.

De plus, lorsque le réservoir supplémentaire est présent dans la solution, une relation de compatibilité permettant de déduire le type de réservoir (ST) en fonction de la vitesse et de la distance de vol (CS et FR) est aussi à prendre en compte.

- relation de compatibilité (CS, FR, ST) :

$$\{(600, \{800, 1000\}, ST_{300}), (\{400, 450\}, 1600, ST_{300}), (550, \{800, 1000\}, ST_{200}),$$
$$(\{400, 450\}, 1400, ST_{200}), (500, \{800, 1000\}, ST_{100}), (\{400, 450\}, 1200, ST_{100})\}$$

2.1.2.3 Modélisation par ajout de valeur (CSP *) et exemple

Dans ce cas, l'élément optionnel est associé à une variable de configuration, toujours présente dans le problème. Une valeur particulière, ajoutée au domaine de cette variable, émule l'absence de l'élément optionnel dans le problème (voir la section 1.4.3.1 sur les $CSP^\star$).

Nous ajouterons la valeur $\star$ dans le cas des variables symboliques et 0 dans le cas des variables numériques entières. Lorsque la variable vaut 0 ou $\star$, cela signifie que l'élément optionnel n'est pas présent dans le produit configuré. Cet ajout de valeur nécessite ensuite de compléter toutes

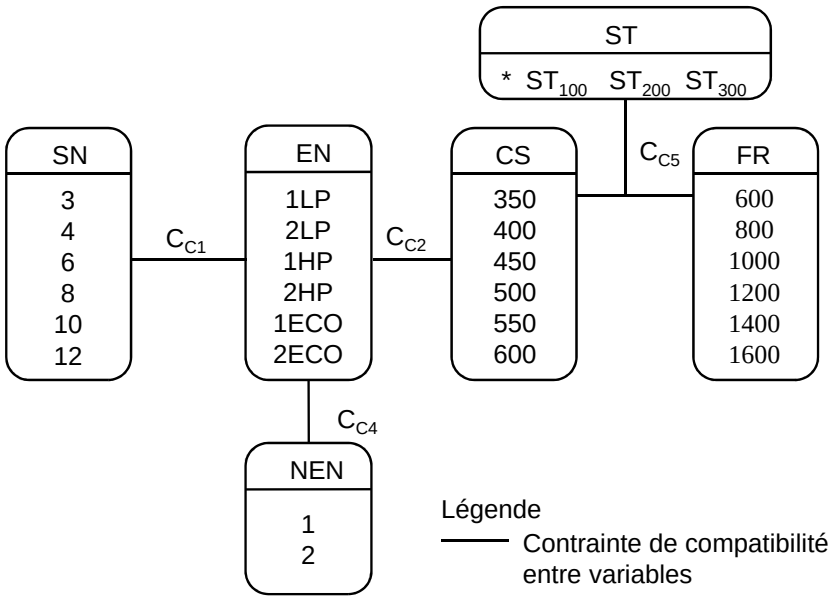


FIGURE 2.2: Configuration de l'avion de tourisme représenté en *CSP**

les contraintes où la variable correspondant à l'élément optionnel apparaît, pour prendre en compte son caractère facultatif. Ceci peut conduire à fusionner des contraintes lors de la phase de modélisation du problème sous forme de *CSP*.

Pour modéliser notre exemple et le problème par un ajout de valeur, nous ajoutons la valeur $\star$ dans le domaine de définition de la variable ST :

$$D_{ST} = \{\star, ST_{100}, ST_{200}, ST_{300}\}$$

Cette variable apparaît dans :

- la condition d'ajout de la variable ST fonction de CS et FR,
- la contrainte de compatibilité $C_{C5}(CS, FR, ST)$.

Ces deux contraintes portant sur les mêmes éléments, nous fusionnons les deux contraintes précédentes en une contrainte $C_{C5}(CS, FR, ST)$ présenté par le tableau 2.2. Cette contrainte exprime à la fois la condition d'existence du réservoir et la contrainte permettant de déterminer le type de réservoir nécessaire. Les neuf combinaisons manquantes sont celles qui sont exclues par la contrainte $C_{C3}(CS, FR)$ du problème élémentaire.

Le modèle du problème élémentaire de la figure 2.1 peut être alors complété par l'ajout de la variable ST et de la contrainte C_{C5}, comme cela est représenté en figure 2.2. Ce modèle comprend donc cinq variables discrètes et toujours présentes dans toutes les solutions (SN, EN, CS, FR, NEN) contraintes par quatre relations de compatibilité (C_{C1}, C_{C2}, C_{C3} et C_{C4}), une variable optionnelle (ST) et une contrainte (C_{C5}) permettant de déterminer si la variable optionnelle est présente dans la solution de configuration.

2.1.2.4 Modélisation par ajout de valeur (DCSP) et exemple

La modélisation précédente a exploité l'ajout d'une valeur particulière dans le domaine de la variable pour émuler l'existence d'un élément (composant ou propriété). Les *CSP* conditionnels (ou « dynamiques » dans leur première proposition, voir section 1.4.3.2) permettent d'ajouter

TABLE 2.2: Contrainte C_{C5} entre CS, FR et ST

C_{C5}

CS	FR	ST	CS	FR	ST	CS	FR	ST
350	600	$\star$	400	1200	ST_{100}	500	600	$\star$
350	800	$\star$	400	1400	ST_{200}	500	800	ST_{100}
350	1000	$\star$	400	1600	ST_{300}	500	1000	ST_{100}
350	1200	$\star$	450	600	$\star$	550	600	$\star$
350	1400	$\star$	450	800	$\star$	550	800	ST_{200}
350	1600	$\star$	450	1000	$\star$	550	1000	ST_{200}
400	600	$\star$	450	1200	ST_{100}	600	600	$\star$
400	800	$\star$	450	1400	ST_{200}	600	800	ST_{300}
400	1000	$\star$	450	1600	ST_{300}	600	1000	ST_{300}

ou activer des variables au cours de la résolution ou du filtrage du problème de configuration. Les variables présentes dans toutes les solutions du problème sont dites initialement *active* et le restent. Les variables associées à des éléments optionnels sont initialement inactives et objets d'une contrainte permettant de les rendre actives. Une contrainte d'activation est composée d'une prémisse (en fait, une contrainte de compatibilité) qui, si elle est vérifiée, active la variable associée. Il est également à noter que le formalisme *DCSP* prend en compte et filtre les contraintes uniquement si toutes les variables intervenant dans la contrainte sont actives.

Dans le cas de l'exemple d'avion de tourisme, pour modéliser l'ajout de la variable réservoir supplémentaire (ST), nous utilisons la contrainte d'activation C_{A1}, qui dépend des valeurs prises par les variables vitesse de croisière (CS) et distance de vol (FR) représentée par l'équation suivante :

$$C_{A1} : (CS > 380 \vee FR > 1100) \wedge (CS > 480 \vee FR > 700) \xrightarrow{ACT} ST$$

Dès que la variable ST est activée, toutes les variables de la contrainte de compatibilité permettant de déterminer le type de réservoir $C_{C6}(CS,FR,ST)$ sont actives. Cette contrainte C_{C6}, présentée dans le tableau 2.3, est alors prise en compte et filtrée.

Le modèle du problème élémentaire de la figure 2.1 peut être alors complété avec la variable optionnelle ST, la contrainte d'activation C_{A1} et la contrainte C_{C6} comme cela est représenté en figure 2.3. Ce modèle comporte donc cinq variables discrètes et toujours présentes dans toutes les solutions (SN, EN, CS, FR, NEN) contraintes par quatre relations d'incompatibilité (C_{C1}, C_{C2}, C_{C3} et C_{C4}), une variable optionnelle (ST), une contrainte (C_{A1}) permettant de déterminer si la variable optionnelle est présente dans la solution de configuration, ainsi qu'une contrainte (C_{C6}) prise en compte uniquement lorsque la variable optionnelle est présente dans la solution.

2.1.2.5 Filtrage du problème avec éléments optionnels

Lorsque les éléments optionnels sont modélisés avec l'ajout de valeur ($\star$ ou 0) proposé par le formalisme des $CSP^\star$, le problème revient à un *CSP* discret. Les traitements d'arc-cohérence évoqués pour le problème élémentaire sont exploitables sans modification. Le seul besoin est d'interpréter, dans la solution de configuration, les variables dont le domaine de définition est réduit à $\star$ ou 0.

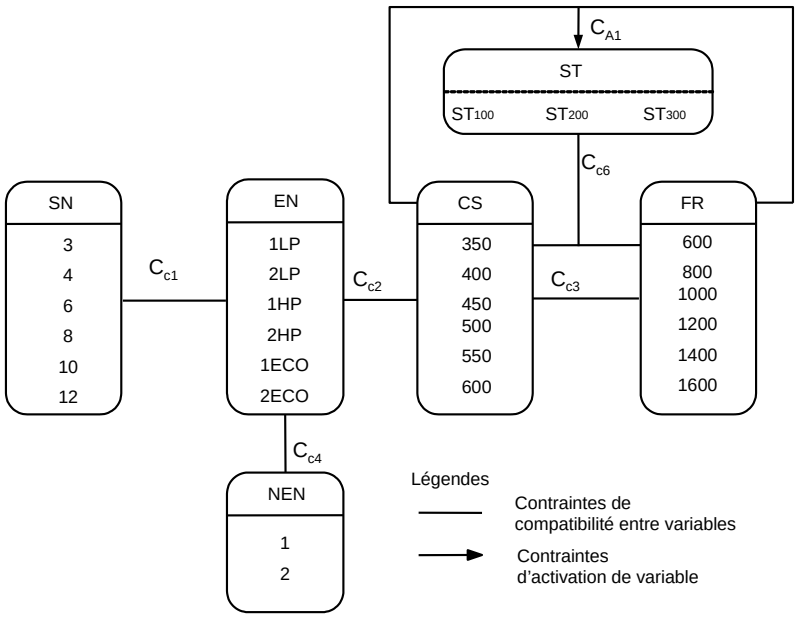


FIGURE 2.3: Configuration de l'avion de tourisme représenté en *DCSP*

TABLE 2.3: Contrainte C_{C6} entre CS, FR et ST

C_{C6}

CS	FR	ST
500	800	ST_{100}
500	1000	ST_{100}
400	1200	ST_{100}
400	1400	ST_{200}
400	1600	ST_{300}
450	1200	ST_{100}
450	1400	ST_{200}
550	800	ST_{200}
550	1000	ST_{200}
600	800	ST_{300}
600	1000	ST_{300}
450	1600	ST_{300}

Lorsque le formalisme *DCSP* est employé, le caractère interactif de la configuration et le filtrage associé peuvent conduire à des incohérences. Van Oudenhove de Saint Géry montre que ces incohérences proviennent de l'hypothèse stipulant qu'une contrainte de compatibilité est prise en compte si et seulement si toutes les variables de la contrainte sont actives. Considérons l'exemple suivant mettant en œuvre deux propriétés toujours présentes et une propriété optionnelle :

- deux propriétés toujours présentes : Vitesse (V) deux valeurs $\{haute, faible\}$ et Rayon d'action (R) deux valeurs $\{grand, petit\}$,
- une propriété optionnelle activée par une contrainte non détaillée Finition (F) deux valeurs $\{performance, standard\}$,
- deux contraintes de compatibilité :

$$\begin{aligned} & C_{C1}(V,F) = \{(haute, performance), (faible, standard)\} \\ et \quad & C_{C2}(R,F) = \{(grand, performance), (petit, standard)\} \end{aligned}$$

Les variables V et R étant initialement actives, considérons le processus de configuration interactif suivant :

- l'utilisateur sélectionne la valeur *haute* pour V,
- l'utilisateur sélectionne la valeur *petit* pour R,
- l'utilisateur fait une sélection quelconque conduisant à l'ajout de la variable F dans le problème,
- C_{C1} et C_{C2} sont alors prises en compte et filtrées :
  - le filtrage de C_{C1} supprime la valeur *standard* pour F,
  - le filtrage de C_{C2} supprime la valeur *performance* pour F,
  - le domaine de valuation de F devient vide, le problème n'a plus de solution et devient donc incohérent.

Il est clair qu'une modélisation différente avec des contraintes supplémentaires, permettrait de contourner ce problème, mais il n'est pas toujours évident dans les problèmes réels d'isoler ce cas de figure afin de le traiter.

Le même modèle avec le formalisme *CSP** fait apparaître les modifications suivantes :

- une propriété optionnelle activée par une contrainte non détaillée, Finition (F) deux valeurs $\{performance, standard\}$ et la valeur particulière $\star$,
- deux contraintes de compatibilité,

$$\begin{aligned} C_{C1}(V,F) &= \{(haute, performance), (haute, \star), (faible, standard), (faible, \star)\} \\ C_{C2}(R,F) &= \{(grand, performance), (grand, \star), (petit, standard), (petit, \star)\} \end{aligned}$$

La propagation des choix des valeurs des variables $V = haute$ et $R = petit$ conduit à la restriction de la variable F à la seule valeur $\star$ sans incohérence : la propriété F ne fait donc pas partie de la solution de configuration.

L'intérêt du formalisme *CSP** est donc de ne pas soulever de problème de cohérence lors du filtrage. Par contre, il est nécessaire de propager systématiquement toutes les contraintes sur toutes les variables associées à tous les éléments optionnels. À l'opposé du formalisme *CSP**, *DCSP* permet d'ajouter les variables au besoin durant le processus de configuration et de manipuler en conséquence des modèles de taille plus réduite, mais il peut soulever des problèmes de cohérence lors de l'ajout d'entités optionnelles au problème en cours.

2.1.3 Configuration : problème avec sélection d'éléments en exclusion

Le problème précédent faisait apparaître des éléments optionnels qui pouvaient être absents ou présents dans le produit configuré. Cette section s'intéresse au cas où il est obligatoire de sélectionner un élément optionnel parmi plusieurs éléments optionnels. Cela correspond à une forme de sélection d'alternatives en exclusion. Pour l'exemple considéré, nous ajoutons deux propriétés en exclusion correspondant à deux finitions différentes : *finition catalogue* (FCA) et *finition customisée* (FCU). Chacune de ces finitions ayant pour valeurs possibles :

- finition catalogue (FCA) : $D_{FCA} = \{standard, confort, luxe\}$,
- finition customisée (FCU) : $D_{FCU} = \{work, leisure\}$.

Ce besoin n'est pas critique en configuration de produit, car le plus souvent les éléments optionnels en exclusion sont fusionnés en un unique élément sans condition d'existence, par exemple finition avion (FA) avec $D_{FA} = D_{FCA} \cup D_{FCU}$. Mais étant donné que la prise en compte de tâches en exclusion est primordiale en planification pour connaître, par exemple, la durée maximale ou le coût maximal du projet générique (voir section 2.2), nous le considérons dès maintenant avant tout pour faciliter l'étude du couplage de ces deux domaines (en section 2.3).

2.1.3.1 Définition du problème avec sélection d'éléments en exclusion

Le problème de configuration dans ce cas, peut être décrit par :

a) Reste inchangé : un produit est un ensemble de composants et de propriétés valuées.

b) Reste inchangé : les composants ayant les mêmes caractéristiques sont regroupés dans un groupe de composants.

c) Reste inchangé : le produit générique met en œuvre des groupes de composants et des propriétés du produit.

d) Devient : les groupes de composants et les propriétés sont de trois types :
   - ceux qui sont présents dans toutes les solutions,
   - ceux dont la présence est conditionnée par le choix d'autres composants et/ ou de valeurs de propriétés et correspondent aux éléments optionnels du problème,
   - ceux qui sont optionnels et qui de plus, doivent être sélectionnés parmi un ensemble d'éléments optionnels en exclusion.

e) Devient : le problème de configuration comprend :
   - des relations de compatibilité entre groupes de composants et propriétés,
   - des relations permettant de décrire les conditions d'existence des groupes de composants et les propriétés optionnels du produit générique,
   - des relations permettant de décrire l'exclusion mutuelle d'éléments optionnels.

f) Reste inchangé : les exigences du client se traduisent par la sélection d'un composant dans un groupe et/ou d'une valeur pour une propriété ou par la sélection d'un élément parmi un ensemble d'éléments disjoints.

g) Reste inchangé : une solution de configuration est une instanciation du produit générique, elle comprend un composant sélectionné dans chaque groupe existant et une valeur choisie pour chaque propriété existante.

Résoudre ce problème de configuration consiste soit à trouver une solution, soit toutes les solutions du problème ou à prouver que le problème n'admet pas de solutions.

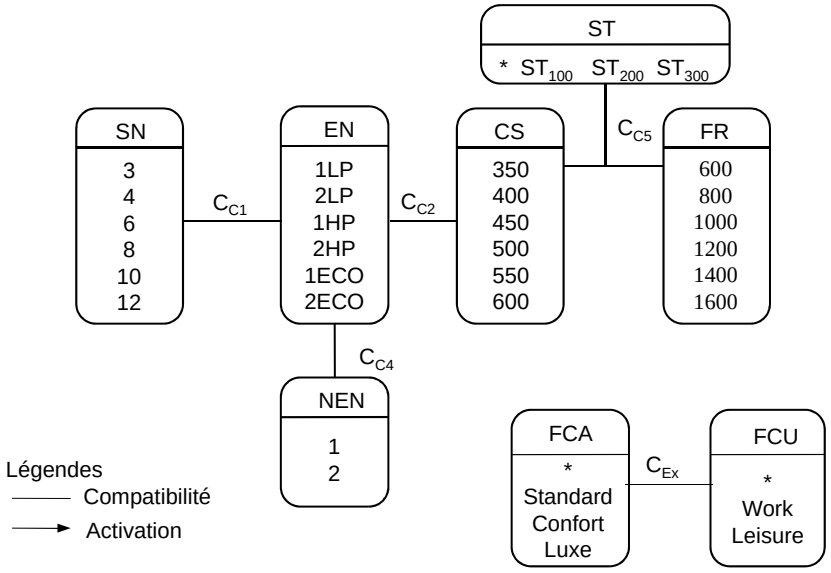


FIGURE 2.4: Configuration de l'avion de tourisme avec exclusion représentée par un CSP*

2.1.3.2 Différents modèles du problème avec sélection d'éléments en exclusion et exemple

Ce problème revient à compléter la formalisation des groupes de composants et propriétés optionnels avec des exclusions. Les deux formalismes *CSP** et *DCSP* proposés en section précédente sont donc considérés à nouveau et complétés.

L'exemple précédent est complété avec les propriétés associées à des variables de configuration :

- finition catalogue (FCA) : D_{FCA} = $\{standard, confort, luxe\}$
- finition customisée (FCU) : D_{FCU} = $\{work, leisure\}$

et la condition d'exclusion de ces deux propriétés : FCA active ou FCU active.

*2.1.3.3 Modélisation par ajout de valeur (*CSP**) et exemple*

Dans ce cas, le caractère optionnel de l'élément est obtenu en ajoutant la valeur $\star$ dans le cas des variables symboliques et 0 dans le cas des variables numériques entières. L'exclusion est formalisée par une contrainte de compatibilité entre les variables associées à chaque couple d'éléments en exclusion. Cette contrainte force l'existence d'un des deux éléments (pas de tuple ($\star$,$\star$)) et interdit l'existence simultanée des deux éléments (pas de tuple $(\neq \star, \neq \star)$).

Par exemple, pour deux variables symboliques en exclusion (X,Y) de domaines respectifs $\{\star, x1, x2, \ldots, xn\}$ et $\{\star, y1, y2, \ldots, ym\}$, la contrainte d'exclusion se formalise par :

$$C_{Cex}(X,Y) = \{(\star, y1), (\star, y2), \ldots, (\star, ym), (x1, \star), (x2, \star), \ldots, (xn, \star)\}$$

Pour notre exemple, nous ajoutons la valeur $\star$ dans le domaine de définition des variables :

- finition catalogue (FCA) : $D_{FCA} = \{\star, standard, confort, luxe\}$
- finition customisée (FCU) : $D_{FCU} = \{\star, work, leisure\}$
- et la contrainte d'exclusion :

$$C_{Cex}(FCA, FCU) = \{(\star, work), (\star, leisure), (standard, \star), (confort, \star), (luxe, \star)\}$$

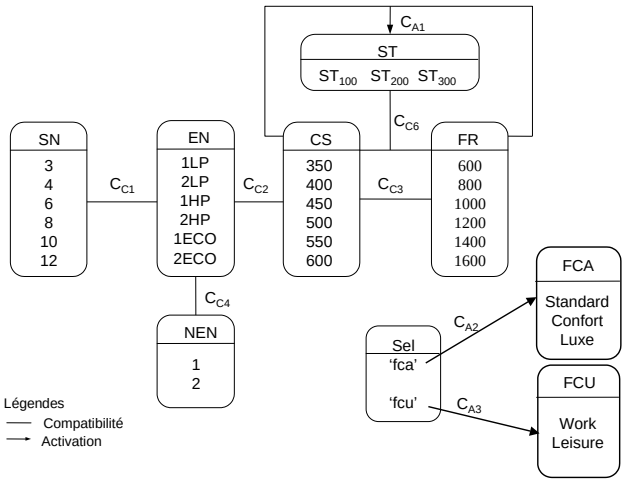


FIGURE 2.5: Configuration de l'avion de tourisme avec exclusion représentée par un *DCSP*

Le modèle du problème à éléments optionnels de la section précédente (figure 2.2) peut être alors complété par les deux variables (FCA et FCU) et la contrainte d'exclusion (C_{Cex}) comme cela est représenté en figure 2.4. Ce modèle comprend donc cinq variables discrètes et toujours présentes dans toutes les solutions (SN, EN, CS, FR, NEN) contraintes par quatre relations d'incompatibilité (C_{C1}, C_{C2}, C_{C3} et C_{C4}), une variable optionnelle (ST), une contrainte (C_{C5}) permettant de déterminer si la variable optionnelle est présente dans la solution de configuration, deux variables en exclusion mutuelle (FCA, FCU), ainsi que la contrainte d'exclusion (C_{Cex}). Il peut être remarqué sur cette figure que les deux morceaux du modèle sont indépendants.

2.1.3.4 *Modélisation par ajout d'éléments (*DCSP*) et exemple*

Le formalisme *DCSP* nécessite de formaliser une contrainte d'activation déclenchant l'ajout ou l'activation de la variable associée à chaque élément optionnel. L'exclusion de variable n'entre pas dans ce cas, car c'est l'existence d'une des deux variables qui empêche l'ajout de l'autre variable. Il est donc nécessaire d'exclure les deux conditions d'activation portant sur chaque variable en exclusion. Le moyen proposé est de recourir à une variable auxiliaire dénommée sélecteur dont les valeurs serviront à conditionner les contraintes d'activation des variables.

Pour deux variables symboliques en exclusion (X, Y) de domaine de respectif $\{x_1, x_2, \ldots, x_n\}$ et $\{y_1, y_2, \ldots, y_m\}$ l'exclusion se formalise donc :

- une variable sélecteur $(Sel - X - Y) : D_{Sel-X-Y} = \{x, y\}$
- deux contraintes d'activation :
  - $C_{Ax} : Sel - X - Y = x \xrightarrow{ACT} X$
  - $C_{Ay} : Sel - X - Y = y \xrightarrow{ACT} Y$

Étant donnés que :

- la configuration est terminée lorsque les domaines de toutes les variables actives sont réduits à un singleton, la sélection d'un des éléments optionnels est une obligation.

– la prémisse se déclenche uniquement lorsque le domaine de définition du sélecteur est réduit à un singleton, le fonctionnement exclusif est validé.

Pour notre exemple, nous ajoutons en conséquence :

– la variable sélecteur Sel de domaine $D_{Sel} = \{fca, fcu\}$,
– les deux contraintes d'activation :
  – $C_{A2} : Sel = fca \xrightarrow{ACT} FCA$
  – $C_{A3} : Sel = fcu \xrightarrow{ACT} FCU$

Le modèle du problème à éléments optionnels de la section précédente (figure 2.3) peut être alors complété par les deux variables (FCA et FCU), les contraintes d'activation (C_{A2}, C_{A3}) et le sélecteur (Sel) comme cela est représenté en figure 2.5. Ce modèle comprend donc cinq variables discrètes et toujours présentes dans toutes les solutions (SN, EN, CS, FR, NEN) contraintes par quatre relations de compatibilité (C_{C1}, C_{C2}, C_{C3} et C_{C4}), une variable optionnelle (ST), une contrainte (C_{A1}) permettant de déterminer si la variable optionnelle est présente dans la solution de configuration, une contrainte de compatibilité portant sur variable optionnelle (C_{C6}), deux variables en exclusion mutuelle (FCA, FCU), ainsi qu'un sélecteur (Sel) et deux contraintes d'activation et d'exclusion (C_{A2}, C_{A3}). Il peut être également remarqué sur cette figure que les deux morceaux du modèle sont indépendants.

2.1.3.5 Filtrage du problème avec sélection d'éléments en exclusion

La formalisation des éléments optionnels en exclusion, que ce soit avec les $CSP^{\star}$ ou les $DCSP$, exploite simplement et sans modification les éléments proposés pour les éléments optionnels en section 2.1.2.5. En conséquence, les remarques relatives au filtrage de la section précédente s'appliquent pleinement au problème mettant en œuvre l'exclusion d'éléments optionnels.

2.1.4 Configuration : problème avec des sous-ensembles optionnels

Dans le cas de produits comportant beaucoup de composants ou faisant apparaître des sous-ensembles, il est nécessaire de pouvoir représenter de manière structurée plusieurs niveaux de décomposition du produit.

Dans ce cas, quelques formalisations ont été proposées. Veron considère une définition récursive du produit où le produit peut être constitué de produits et/ou de composant et/ou de propriétés valuées. Sabin et Freuder (Sabin et Freuder, 1996) ont de même proposé une extension structurée des CSP, les composites CSP ($CCSP$) où les valeurs de variables peuvent correspondre à un sous problème.

Étant donné que nous nous intéressons au couplage de la configuration de produit et de la planification de projet, nous ne considérons dans ce mémoire qu'un seul niveau d'abstraction. En conséquence, nous ne traiterons pas l'aspect structuré ou multi-niveaux du problème de configuration et ultérieurement du couplage. Par contre, nous avons gardé à l'esprit dans l'ensemble de nos propositions ce caractère hiérarchique des produits afin de pouvoir l'intégrer dans de futurs travaux de recherche.

2.1.5 Implémentation des modèles sur les outils logiciels

Certains des modèles proposés précédemment ont été implémentés sur trois outils de programmation par contraintes. L'objet de cette section est d'évoquer les problèmes ou difficultés rencontrés

lors de l'implémentation et lors de l'exploitation de CoFiADe, d'ILOG CP 5.5 et d'ECLiPSe. Le problème élémentaire (formalisme *CSP*) et le problème avec éléments optionnels avec et sans exclusion (avec les deux formalismes $CSP^\star$ et *DCSP*) seront considérés.

2.1.5.1 Expérimentation avec CoFiADe

Implémentation CoFiADe supportant les variables symboliques, entières et réelles, leur implémentation s'effectue sans problème. Pour ce qui est des contraintes du problème élémentaire discret, il est possible de décrire uniquement les tables de compatibilité en extension en listant les combinaisons autorisées ou exclues. Des raccourcis de type « différent d'une valeur » sont cependant possibles et permettent de réduire la quantité de tuples à écrire.

CoFiADe ne gère pas explicitement l'implémentation du $CSP^\star$. Il est donc nécessaire d'ajouter manuellement les valeurs particulières simulant la présence ou l'absence de la variable ($\star$ ou 0) et les contraintes associées, ou bien d'effectuer un pré-traitement du modèle. CoFiADe gère par contre explicitement l'implémentation du *DCSP*. Il n'est donc pas nécessaire d'effectuer de pré-traitement du modèle pour décrire des contraintes d'activation de variables et de contraintes. L'annexe 3.3.3.4 montre le code source du modèle utilisé par CoFiADe correspondant à certaines de ces implantations.

Exploitation En ce qui concerne le problème élémentaire, le filtrage des *CSP* discrets que CoFiADe effectue suit les principes de la cohérence d'arc. En conséquence, une incohérence nécessitant un filtrage plus fort (trois-cohérence par exemple) ne sera pas détectée. Par exemple, si l'on considère les variables $V1$, $V2$ et $V3$ de même domaine $D = \{1, 2\}$ et les contraintes $V1{=}V2$, $V2{=}V3$ et $V1 \neq V3$, le filtrage effectué par CoFiADe ne détectera pas l'incohérence.

En ce qui concerne le filtrage des problèmes avec éléments optionnels, mis à part l'incohérence liée au formalisme *DCSP* vue en section 2.1.2.5, CoFiADe propage sans problème les contraintes. Il est par contre nécessaire d'interpréter les résultats d'une implémentation en $CSP^\star$ dans les solutions de configuration.

2.1.5.2 Expérimentation avec ILOG CP 5.5

Implémentation ILOG CP 5.5 supporte uniquement les variables numériques entières. Il est en conséquence nécessaire d'encoder les valeurs symboliques en valeurs entières et de traduire l'ensemble du problème en un problème à variables entières. En ce qui concerne les contraintes tables, ILOG CP 5.5 propose deux façons différentes pour implémenter une table de compatibilité :

- énumérer chaque tuple compatible en utilisant la méthode ALLOWEDASSIGNMENTS(), ce qui revient à exprimer la contrainte en extension,
- exprimer la table de compatibilité sous la forme d'une relation logique, c'est-à-dire utiliser des opérateurs logiques $\vee$ et $\wedge$, ce qui revient à l'exprimer en intension.

Comme CoFiADe, ILOG CP 5.5 ne gère pas explicitement les $CSP^\star$, il est donc nécessaire de pré-traiter les modèles. En ce qui concerne l'implémentation des *DCSP*, ILOG CP 5.5 ne sait ajouter que des contraintes (pas de variables) au problème durant la résolution ou le filtrage. Il est en conséquence nécessaire de transformer l'activation de chaque variable en une activation de contraintes et de l'interpréter. Un encodage ou un pré-traitement est donc également nécessaire. L'annexe .2 montre le code source du modèle utilisé par ILOG CP 5.5 correspondant à certaines de ces implantations.

Exploitation En ce qui concerne le problème élémentaire, le filtrage effectué par ILOG CP 5.5 via la méthode PROPAGATE() est plus performant que celui de CoFiADe et l'incohérence du problème à trois variables est détectée. Par contre le filtrage est effectué uniquement si les contraintes sont exprimées en extension avec la méthode ALLOWEDASSIGNMENTS(). Ceci limite l'intérêt d'exprimer la combinatoire avec des formules logiques dans le cas du filtrage seul, bien entendu. En ce qui concerne le filtrage des problèmes avec éléments optionnels, l'activation de contraintes est filtrée de manière satisfaisante et permet d'émuler le fonctionnement d'un *DCSP*. L'ensemble des encodages et des pré-traitements nécessite dans tous les cas une interprétation de l'ensemble les résultats.

2.1.5.3 Expérimentation avec ECLiPSe

Implémentation Comme CoFiADe, ECLiPSe supporte les variables symboliques, entières et réelles, par l'intermédiaire de librairies spécifiques. De manière similaire à ILOG CP 5.5, ECLiPSe permet d'implémenter les contraintes tables de compatibilité en extension et en intension. Comme CoFiADe et ILOG CP 5.5, ECLiPSe ne gère pas explicitement les *CSP** et nécessite un pré-traitement des modèles. En ce qui concerne l'implémentation des *DCSP*, ECLiPSe s'apparente à ILOG CP 5.5 et ne sait ajouter que des contraintes à un problème durant la résolution. Un encodage ou un pré-traitement de modèle est donc également nécessaire. L'annexe .3 montre le code source du modèle utilisé par ECLiPSe correspondant à certaines de ces implantations.

Exploitation En ce qui concerne le problème élémentaire, le filtrage effectué par ECLiPSe apparaît de même niveau que celui d'ILOG CP 5.5 et l'incohérence du problème à trois variables est également détectée. Par contre, le filtrage est effectué de la même manière sur les contraintes exprimées en extension et en intension. Le filtrage des problèmes avec éléments optionnels par le biais de l'activation de contraintes donne des résultats similaires à ceux proposés par ILOG CP 5.5. Une interprétation des résultats est alors également nécessaire.

2.1.5.4 Conclusions

En ce qui concerne la modélisation, les trois outils utilisés répondent globalement au besoin, avec les remarques suivantes. Il est nécessaire avec ILOG CP 5.5 de traduire le problème en ne considérant que des variables numériques entières. Avec ILOG CP 5.5 et ECLiPSe, il faut également émuler l'activation de variable des *DCSP* par l'activation de contraintes. Pour les trois outils, il est nécessaire de modéliser et de pré-traiter les *CSP**. En ce qui concerne le filtrage nécessaire à l'aide interactive à la configuration, les trois outils filtrent correctement l'ensemble des problèmes proposés avec un degré de filtrage moins abouti pour CoFiADe. Les trois outils nécessitent une interprétation des résultats lorsque les modèles exploitent le formalismes *CSP**.

2.1.6 Conclusion sur la configuration

Cette section a proposé une synthèse du problème de configuration et de sa formalisation comme un problème de satisfaction de contraintes. Trois types de problèmes ont été identifiés et étudiés de manière détaillée : le problème élémentaire, où tous les éléments sont toujours présents, le problème à éléments optionnels et le problème à éléments en exclusion. Il a ensuite été proposé pour chaque problème : une définition, des éléments de modélisation, une illustration sur un exemple et une discussion sur les possibilités de filtrage. Il ressort logiquement que le formalisme des *CSP* discrets et les algorithmes de filtrage associés constituent une approche tout à fait

intéressante pour aider la configuration interactive. L'implémentation des modèles proposés sur des outils logiciels a conclu sur une bonne faisabilité opérationnelle. La deuxième section de ce chapitre va s'efforcer de suivre ce cheminement pour le problème de planification afin de conduire à l'étude du problème de couplage en troisième section.

2.2 Planification et CSP

Dans cette section, nous allons nous intéresser au problème de planification et à sa représentation sous la forme d'un problème de satisfaction de contraintes. L'organisation de ce chapitre, calquée sur le précédant, considère les quatre mêmes problèmes : élémentaire, section 2.2.1, avec des tâches optionnelles, section 2.2.2, avec des sélections de tâches en exclusion, section 2.2.3, avec des sous-ensembles de tâches, section 2.2.4. Pour chaque étape et une fois le problème défini, nous nous intéresserons à sa modélisation en *CSP*, ses possibilités de filtrage et illustrerons la modélisation sur l'exemple d'avion de tourisme. Dans un but de lisibilité, tous ces développements ne prendront pas en compte les ressources. Celles-ci seront par contre considérées dans une dernière sous-section spécifique.

2.2.1 Planification : problème élémentaire

Nous allons, dans cette section, nous intéresser à la définition du problème élémentaire de planification en correspondance avec la définition du problème de configuration élémentaire. Nous considérons qu'un projet générique est un ensemble de tâches génériques reliées par des contraintes de précédence. Étant donné que les ressources ne sont pas considérées, le caractère générique de la tâche est uniquement lié à sa durée et toutes les autres caractéristiques de la tâches sont figées, non modifiables et uniquement descriptives. Ces caractéristiques descriptives représentent qualitativement le mode opératoire en indiquant les ressources techniques et/ou humaines et ne font l'objet d'aucun traitement lié à la planification. De même les contraintes de précédence sont figées et non modifiables. En conséquence, l'aspect configuration de projet correspond à la sélection d'une durée pour chaque tâche entre deux durées (minimum et maximum) tandis que l'aspect planification correspond à la détermination des dates de début et de fin au plus tôt et au plus tard. Cette section définit le problème de planification élémentaire, le formalise comme un *CSP*, illustre nos propositions sur l'exemple de l'avion de tourisme et discute des possibilités de filtrage associées.

2.2.1.1 Définition du problème élémentaire

Compte tenu des éléments précédemment évoqués, nous définissons le problème de planification élémentaire de la manière suivante :

a) Un projet est caractérisé par une date de début et une date de fin. Il est composé d'un ensemble de tâches liées par des relations de précédence.

b) Un projet générique met en œuvre un ensemble de tâches génériques et deux tâches fictives de début-projet et de fin-projet de durée nulle. Une tâche générique est caractérisée par :
   - une durée définie sur un intervalle : [durée minimum, durée maximum]
   - des dates de début et de fin résultant du traitement de planification et définies sur des intervalles : [début plus tôt, début plus tard], [fin plus tôt, fin plus tard]
   - des caractéristiques descriptives figées,

– une relation entre la date de début, la date de fin et la durée d'une tâche stipulant que la date de fin d'une tâche est égale à sa date de début additionnée à sa durée.

c) Toutes les tâches génériques sont présentes dans toutes les solutions de la planification.

d) Des relations temporelles lient :

– les dates de fin et de début des tâches en succession et expriment les antériorités entre les tâches du projet,

– la date de la tâche de début-projet et la date de début des tâches sans antécédent, la date de fin des tâches sans successeur et la date de la tâche de fin-projet.

e) Une exigence utilisateur se traduit par la restriction de l'intervalle de définition (ou la sélection d'une valeur unique) : d'une durée de tâche, d'une date de début au plus tôt de tâche ou d'une date de fin au plus tard de tâche, des dates des tâches de début-projet et de fin-projet.

f) Une solution de planification est une instanciation du projet générique où la date de début, la date de fin et la durée de chaque tâche générique sont instanciées avec un domaine de valeurs réduit à un singleton.

Nous considérons que la résolution de ce problème élémentaire de planification consiste soit à trouver une solution, soit l'ensemble des solutions du problème ou encore de démontrer que le problème n'admet pas de solution.

2.2.1.2 Modélisation du problème élémentaire

Le modèle définit du problème de planification peut être décrit par un *CSP* de type temporel quantitatif (*TCSP*) (Dechter et Pearl, 1991). Pour la modélisation du problème de planification, chaque tâche à planifier est décrite par trois variables continues de type temporel décrivant la date de début (*Possible Starting Time*, noté *pst*), la date de fin (*Possible Finish Time*, noté *pft*) et la durée de la tâche (*Possible Duration Time pdt*, noté).

Les relations temporelles sont alors associées à deux types de contraintes numériques de planification qui correspondent :

– à la contrainte de durée de tâche, notée C_D, qui stipule que la date de fin d'une tâche T est égale à sa date de début additionnée avec sa durée,

– à la contrainte de précédence, notée C_P, qui stipule que si une tâche T_j précède une tâche T_k, le début de T_k est postérieur à la fin de T_j.

L'ensemble se formalise avec les contraintes suivantes :

$$C_D : T \bullet pft = T \bullet pst + T \bullet pdt \text{ et } C_P : T_k \bullet pst \geq T_j \bullet pft$$

Ce modèle très simple de projet permet néanmoins de représenter des tâches pouvant se dérouler de manière simultanée.

2.2.1.3 Exemple de modèle du problème élémentaire

Les trois premières tâches du problème de planification de la production de l'avion de tourisme (T_1, T_2, T_3) sont maintenant considérées. Les tâches T_1 et T_2 peuvent se dérouler de manière simultanée, par contre ces deux tâches précèdent la tâche T_3. Le modèle de planification comprend les trois tâches génériques suivantes :

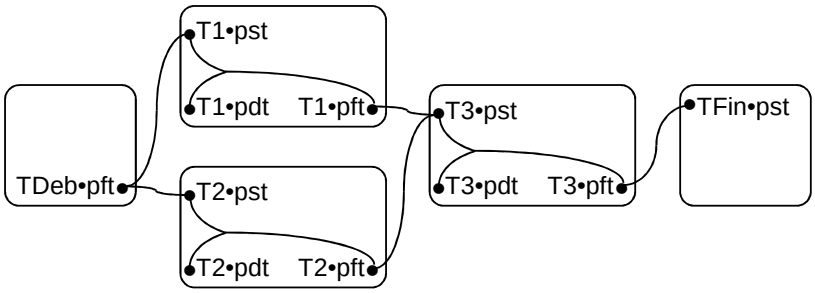


FIGURE 2.6: Modélisation du problème élémentaire de planification de l'exemple de l'avion de tourisme.

- la tâche fabrication (T_1) : cette tâche comprend la fabrication de tous les éléments de l'avion qui ne sont pas sous-traités. Elle est définie par :
  - une durée :$T_1 \bullet pdt$ de domaine $D_{T_1 \bullet pdt} = [10, 90]$
  - des dates de début et de fin $T_1 \bullet pst$, $T_1 \bullet pft$ de domaine $D_{T_1 \bullet pst} = D_{T_1 \bullet pft} = [0, \infty[$
  - la contrainte de durée : $C_{D1} : T_1 \bullet pft = T_1 \bullet pst + T_1 \bullet pdt$
- La tâche d'approvisionnement (T_2) : cette tâche décrit l'approvisionnement de tous les composants nécessaires et qui ne sont pas fabriqués en interne. Elle est définie par :
  - une durée : $T_2 \bullet pdt$ de domaine $D_{T_2 \bullet pdt} = [20, 40]$
  - des dates de début et de fin $T_2 \bullet pst$, $T_2 \bullet pft$ de domaine $D_{T_2 \bullet pst} = D_{T_2 \bullet pft} = [0, \infty[$
  - la contrainte de durée : $C_{D2} : T_2 \bullet pft = T_2 \bullet pst + T_2 \bullet pdt$
- La tâche d'assemblage (T_3) : cette tâche réalise l'assemblage de tous les composants fabriqués en interne et approvisionnés. Elle est définie par :
  - une durée : $T_3 \bullet pdt$ de domaine $D_{T_3 \bullet pdt} = [10, 180]$
  - des dates de début et de fin $T_3 \bullet pst$, $T_3 \bullet pft$ de domaine$D_{T_3 \bullet pst} = D_{T_3 \bullet pft} = [0, \infty[$
  - la contrainte de durée :$C_{D3} : T_3 \bullet pft = T_3 \bullet pst + T_3 \bullet pdt$

Les relations de précédence décrivant le réseau d'activités sont :

- $C_{PD} : T_1 \bullet pst \geq T_{Deb} \bullet pft$
- $C_{P1} : T_2 \bullet pst \geq T_{Deb} \bullet pft$
- $C_{P2} : T_3 \bullet pst \geq T_1 \bullet pft$
- $C_{P3} : T_3 \bullet pst \geq T_2 \bullet pft$
- $C_{PF} : T_{Fin} \bullet pst \geq T_3 \bullet pft$

Le modèle du problème élémentaire est décrit dans la figure 2.6 et comprend cinq tâches toujours présentes dans tous les projets ($T_{Deb}, T_1, T_2, T_3, T_{Fin}$) contraintes par cinq relations de précédence.

2.2.1.4 Filtrage du problème élémentaire

Le filtrage de ce problème élémentaire de planification formalisé comme un *CSP* de type temporel quantitatif (*TCSP*) s'effectue avec un traitement de cohérence aux bornes des intervalles (2*B*-cohérence évoqué en section 1.4.4.1) exploitant l'arithmétique des intervalles. Il est cependant à remarquer qu'avec un tel modèle :

– la réduction de la durée maximum, l'augmentation de la date de fin au plus tôt, ou encore la réduction de la date de début au plus tard d'une tâche n'a pas d'impact sur les autres variables du problème,
– par contre l'augmentation de la durée minimum (tout en respectant la durée maximum), la réduction de la date de fin au plus tard, ou encore l'augmentation de la date de début au plus tôt d'une tâche impactent toutes les autres variables du problème.

2.2.2 Planification : problème avec des tâches optionnelles

Dans la mesure où le problème de configuration admet des éléments optionnels, nous considérons qu'il peut en être logiquement de même pour le problème de planification. Nous allons donc considérer maintenant la possibilité d'ajouter des tâches au planning pour répondre à ce besoin spécifique. Dans l'exemple de l'avion de tourisme, cela correspondra à la tâche de montage du réservoir supplémentaire optionnel.

2.2.2.1 Définition du problème avec tâches optionnelles

Nous définissons le problème de planification avec tâches optionnelles de la manière suivante :

a) reste inchangé : un projet est caractérisé par une date de début, une date de fin et est composé d'un ensemble de tâches liées par des relations de précédence.
b) reste inchangé : un projet générique met en œuvre un ensemble de tâches génériques et deux tâches fictives de début-projet et de fin-projet de durée nulle. Une tâche générique est caractérisée par :
   – une durée définie sur un intervalle : [durée minimum, durée maximum]
   – des dates de début et de fin résultant du traitement de planification et définies sur des intervalles : [début plus tôt, début plus tard], [fin plus tôt, fin plus tard]
   – des caractéristiques descriptives figées,
   – une relation entre la date de début, la date de fin et la durée d'une tâche stipulant que la date de fin d'une tâche est égale à sa date de début additionnée à sa durée.
c) devient : les tâches génériques sont de deux types :
   – celles qui sont toujours présentes dans toutes les solutions de planification,
   – celles dont la présence est conditionnée par des contraintes externes (choix utilisateur ou autres variables n'appartenant pas au problème de planification) ou par le filtrage des contraintes temporelles. Ces tâches sont dites optionnelles.
d) devient : les relations temporelles du problème comprennent les relations temporelles du problème élémentaire et d'autre part des relations permettant de décrire les conditions d'existence des tâches optionnelles du projet générique.
e) devient : une exigence utilisateur correspond soit à une exigence du problème élémentaire ou soit à un choix utilisateur stipulant la présence d'une tâche optionnelle dans le projet.
f) devient : une solution de planification est une instanciation du projet générique, elle comprend pour chaque tâche générique existant dans le projet : une date de début, une date de fin et une durée avec un domaine de valeurs réduit à un singleton.

Résoudre ce problème de planification consiste soit à trouver une solution, soit toutes les solutions du problème ou de démontrer que le problème n'admet pas de solution.

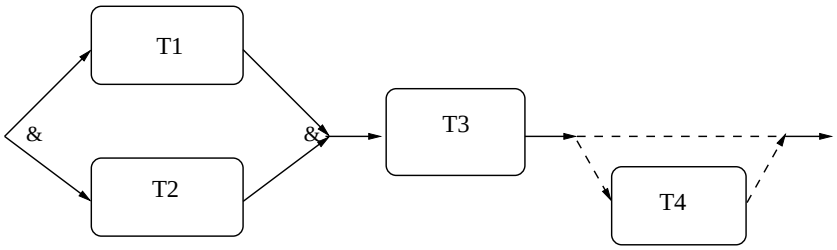


FIGURE 2.7: Graphe de tâche du problème avec ajout de tâche

2.2.2.2 Différents modèles du problème avec tâches optionnelles et exemple

En s'inspirant des modèles proposés en configuration, nous modélisons le problème de planification avec tâches optionnelles sous la forme des mêmes *CSP* à structures dynamique. L'ajout de valeur dans le domaine des variables ($CSP^{\star}$) est traité dans un premier temps suivi par l'ajout de nouveaux éléments au problème (*DCSP*).

L'exemple de l'avion de tourisme comprend maintenant une tâche optionnelle, il s'agit de la tâche d'assemblage du réservoir supplémentaire (T_4) qui n'est ajoutée au problème que si le condition C_4 (réservoir supplémentaire présent) est vérifiée. La tâche T_4 est décrite par sa durée pdt appartenant à l'intervalle $[5,40]$. La tâche d'assemblage du réservoir supplémentaire T_4 lorsqu'elle existe, est liée à la tâche d'assemblage T_3 et la tâche de fin-projet T_{Fin} par une contrainte de précédence. Ceci nécessite :

- les variables définissant la tâche optionnelle : $T_4 \bullet pdt$ de domaine $D_{T_4 \bullet pdt} = [5,40]$ et $T_4 \bullet pst$, $T_4 \bullet pft$ de domaine $D_{T_4 \bullet pst} = D_{T_4 \bullet pft} = [0,+\infty[$
- la condition d'ajout de la tâche (T_4) : $C_4 = vrai$
- les contraintes temporelles :
  - $C_{P4} : T_4 \bullet pst \geq T_3 \bullet pft$
  - $C_{PF} : T_{Fin} \bullet pst \geq T_4 \bullet pft$
  - $C_{PF} : T_{Fin} \bullet pst \geq T_3 \bullet pft$

Le réseau de tâches est décrit dans la figure 2.7 et comprend cinq tâches toujours présentes dans tous les projets ($T_{Deb}, T_1, T_2, T_3, T_{Fin}$) contraintes par cinq relations de précédence ainsi qu'une tâche optionnelle (T_4) liée à ce réseau par deux relations de précédence conditionnées par sa présence dans la solution de planification.

2.2.2.3 Modélisation par ajout de valeur (CSP*) et exemple

Dans le cadre des *CSP** (voir 1.4.3.1), la modélisation d'une tâche optionnelle est réalisée en permettant à la durée de la tâche de prendre la valeur nulle 0. Une durée nulle permet de simuler l'absence et la non exécution d'une tâche.

Soit T une tâche optionnelle définie par une date de début, une date de fin et une durée. Le domaine de définition de cette tâche T vaut alors : $D_{T \bullet pdt} = \{0\} \cup D_{T \bullet pdt}$.

La condition d'existence de la tâche se traduit par une contrainte de compatibilité entre la variable associée à la condition d'existence (C) et la durée de la tâche ($T \bullet pdt$) dont les tuples autorisés sont : $(C, T \bullet pdt) = \{(faux, 0), (vrai, \neq 0)\}$

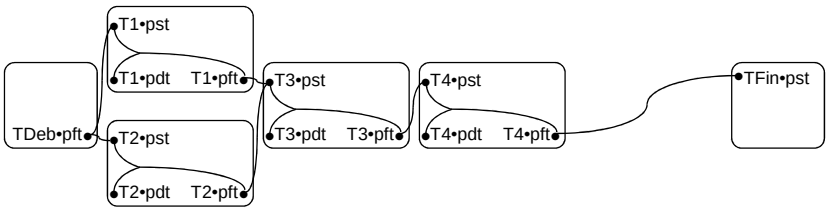


FIGURE 2.8: Modélisation du problème avec ajout de valeur

Pour l'exemple de la tâche d'assemblage du réservoir supplémentaire (T_4), le domaine de définition de la durée est complété avec la valeur 0 et son activation conditionnée par la condition C_4. Le domaine de la durée de la tâche T_4 devient : $D_{T_4 \bullet pdt} = \{0 \cup [5,40]\}$ avec la contrainte $(C_4, T_4 \bullet pdt) = \{(faux, 0), (vrai, \neq 0)\}$.

Les contraintes de durée entre $(T_4 \bullet pst, T_4 \bullet pft, T_4 \bullet pdt)$ et de précédence entre (T_3, T_4) et (T_4, T_{Fin}) restent inchangées. Par contre il n'est pas nécessaire de conserver la contrainte précédence (T_3, T_{Fin}) car elle est redondante et peut être supprimée. Le modèle avec ajout de valeur au problème avec des tâches optionnelles est représenté en figure 2.8 et comprend cinq tâches toujours présentes dans tous les projets $(T_{Deb}, T_1, T_2, T_3, T_{Fin})$ contraintes par quatre relations de précédence ainsi qu'une tâche optionnelle (T_4) liée à ce réseau par deux relations de précédence.

*2.2.2.4 Modélisation par ajout d'éléments (*DCSP*) et exemple*

La modélisation par ajout d'éléments consiste à ajouter au problème au cours de la résolution ou du filtrage des variables et/ou des contraintes supplémentaires conformément au formalisme des *CSP* dynamiques (voir 1.4.3.2). Lorsque des conditions d'activation, qui constituent les prémisses des contraintes d'activation, sont vérifiées, les variables associées sont ajoutées au problème. L'hypothèse liée à ce formalisme, stipulant qu'une contrainte n'est considérée et filtrée que si toutes ses variables sont actives, permet de piloter l'activation de contraintes. Soit T une tâche optionnelle définie par une date de début, une date de fin et une durée. Chacune de ces trois variables sera donc considérée comme une variable non initialement active et comme le conséquent d'une contrainte d'activation C_A.

$$C = vrai \xrightarrow{ACT} \{T \bullet pst, T \bullet pft, T \bullet pdt\}$$

Dans le cas de l'exemple d'avion de tourisme, l'activation de la tâche T_4 est soumise à la condition d'activation C_4. Dès que la condition C_4 est vérifiée, les variables de la tâche T_4 sont ajoutées au problème et toutes les contraintes qui portent sur elles sont alors prises en compte :

$$C_4 = vrai \xrightarrow{ACT} \{T_4 \bullet pst, T_4 \bullet pft, T_4 \bullet pdt\}$$

Le modèle avec ajout de tâches optionnelles au problème est représenté en figure 2.9 et comprend cinq tâches toujours présentes dans tous les projets $(T_{Deb}, T_1, T_2, T_3, T_{Fin})$ contraintes par cinq relations de précédence ainsi qu'une tâche optionnelle (T_4) liée à ce réseau par deux relations de précédence conditionnées par sa présence dans la solution de planification.

2.2.2.5 Filtrage du problème avec tâches optionnelles

Lorsque les tâches optionnelles sont modélisées avec l'ajout de valeur 0 proposé par le formalisme des $CSP^\star$, le filtrage par $2B$-cohérence n'impacte pas de fonctionnement particulier. Par

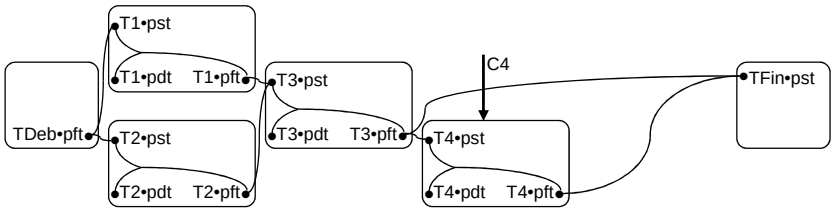


FIGURE 2.9: Modélisation du problème avec ajout de variables

contre, le fait que la tâche soit toujours présente dans le problème peut entraîner des contraintes d'enchaînement de tâches non souhaités. Considérons,

- quatre tâches toujours présentes (T_1, T_2, T_3, T_4), et les six antériorités suivantes :
  - $T_{Deb} \rightarrow T_1$, $T_1 \rightarrow T2$, $T_2 \rightarrow T_{Fin}$
  - $T_{Deb} \rightarrow T_3$, $T_3 \rightarrow T_4$, $T_4 \rightarrow T_{Fin}$
- une tâche optionnelle (T_O) et les quatre antériorités suivantes :
  - $T_1 \rightarrow T_o$, $T_o \rightarrow T_2$,
  - $T_3 \rightarrow T_o$, $T_o \rightarrow T_4$

Lorsque la condition d'activation de T_o n'est pas satisfaite, la durée de la tâche T_o est nulle mais le filtrage s'opère en respectant toutes les antériorités et forçant un déroulement séquentiel des couples de tâches (T_1, T_3) et (T_2, T_4) non prévu initialement dans le problème lorsque la tâche optionnelle n'existe pas. Nous n'avons pas trouvé d'artifice de modélisation permettant de contourner ce problème qui peut être critique. Si sur notre exemple, les durées des tâches T_1, T_2, T_3 et T_4 sont respectivement de $10, 20, 20, 10$, le filtrage indiquera une durée minimum du projet de 40 alors qu'en fait une solution de durée 30 existe.

Lorsque le formalisme *DCSP* est employé, le problème précédent n'existe pas car tant que l'existence de la tâche optionnelle n'est pas validée, les variables et les contraintes associées à la tâche ne sont pas considérées. En conséquence, le filtrage sur le dernier exemple propose bien une durée minimum de 30. Par contre le problème de perte de cohérence décrit en section 2.1.2.5 lié au formalisme *DCSP* est présent dans le problème avec tâches optionnelles. Il peut même être qualifié de critique si l'utilisateur exprime des besoins sur la date de fin du projet ou sur des dates de début ou de fin de tâche, ce qui semble un besoin incontournable en planification interactive. Dans l'exemple précédent comportant les cinq tâches, en considérant que la durée de la tâche T_o est de 20 et que l'utilisateur émet successivement les besoins :

- date fin de projet à date 45 : le filtrage ne détectera aucune incohérence,
- condition C vraie impactant l'activation de la tâche T_o : les variables et les contraintes associées à T_o sont ajoutées au problème, le problème est filtré et devient incohérent car la tâche T_o ne peut être insérée dans le réseau.

Il doit être remarqué que cette séquence de besoin sur le problème modélisé sous forme de *CSP** n'aurait pas été possible car le filtrage du premier besoin (fin projet $= 45$) force la durée de T_o à la valeur nulle et interdit la considération de la tâche optionnelle.

Le formalisme *CSP** a tendance à sur-contraindre le problème d'origine, il rejette donc des solutions admissibles, mais il est par contre moins sensible à la perte de cohérence résultant du filtrage des contraintes temporelles. Le formalisme *DCSP* s'inscrit plutôt dans l'autre tendance et soulève des problèmes de cohérence difficilement contournables en modélisation mais éventuellement gérable lors de la planification interactive si l'utilisateur comprend l'impact de ses choix sur les tâches optionnelles.

2.2.3 Planification : problème avec sélection de tâches en exclusion

Lors de l'étude de la sélection d'éléments en exclusion dans les problèmes de configuration (voir section 2.1.3), il a été mentionné que le problème constituait un besoin fort en planification. Effectivement, la possibilité de pouvoir considérer des chemins alternatifs dans les processuss de production permet de considérer des tâches employant des ressources différentes optimisant des critères variés (coût, qualité, risque ...). Nous considérons donc, dans cette section, le problème de la sélection de tâches en exclusion, c'est-à-dire qu'il est obligatoire de sélectionner une unique tâche parmi plusieurs tâches. La question que soulève cette possibilité est de choisir entre une représentation de la diversité au sein d'une même tâche générique ou entre plusieurs tâches en exclusion.

Pour l'exemple considéré, nous ajoutons deux tâches T_5 et T_6 en exclusion correspondant à deux travaux de finition exclusifs : *tâche de finition catalogue* et *tâche de finition customisée*.

2.2.3.1 Définition du problème de planification avec sélection de tâches en exclusion

Nous définissons le problème avec sélection de tâches en exclusion de la manière suivante :

a) reste inchangé : un projet est caractérisé par une date de début et une date de fin. Il est composé d'un ensemble de tâches liées par des relations de précédence.

b) Reste inchangé : un projet générique met en œuvre un ensemble de tâches génériques et deux tâches fictives de début-projet et de fin-projet de durée nulle. Une tâche générique est caractérisée par :
   - une durée définie sur un intervalle : [durée minimum, durée maximum],
   - des dates de début et de fin résultant du traitement de planification et définies sur des intervalles : [début plus tôt, début plus tard] , [fin plus tôt, fin plus tard],
   - des caractéristiques descriptives figées,
   - une relation entre la date de début, la date de fin et la durée d'une tâche stipulant que la date de fin d'une tâche est égale à sa date de début additionnée à sa durée.

c) Devient : les tâches génériques sont de trois types :
   - celles qui sont toujours présentes dans toutes les solutions de planification,
   - celles dont la présence est conditionnée par des contraintes externes (choix utilisateur ou autres variables n'appartenant pas au problème de planification) ou par le filtrage des contraintes temporelles. Ces tâches sont dites optionnelles.
   - celles qui sont optionnelles et qui de plus doivent être sélectionnées parmi un ensemble de tâches optionnelles en exclusion.

d) Devient : les relations temporelles du problème comprennent d'une part les relations temporelles du problème élémentaire et d'autre part des relations permettant de décrire les conditions d'existence et d'exclusion des tâches optionnelles du projet générique.

e) Devient : une exigence utilisateur correspond soit à une exigence du problème élémentaire ou soit à un choix utilisateur stipulant la présence d'une tâche optionnelle ou en exclusion dans le projet.

f) Reste inchangé : une solution de planification est une instanciation du projet générique, elle comprend pour chaque tâche générique existant dans le projet : une date de début, une date de fin et une durée avec un domaine de valeurs réduit à un singleton.

Résoudre ce problème de planification consiste soit à trouver une solution, soit toutes les solutions du problème ou à démontrer que le problème n'admet pas de solution.

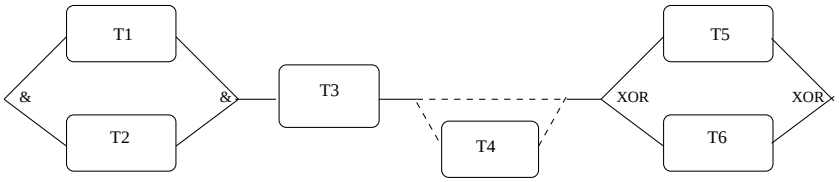


FIGURE 2.10: Graphe de tâches dans l'exemple d'avion de tourisme

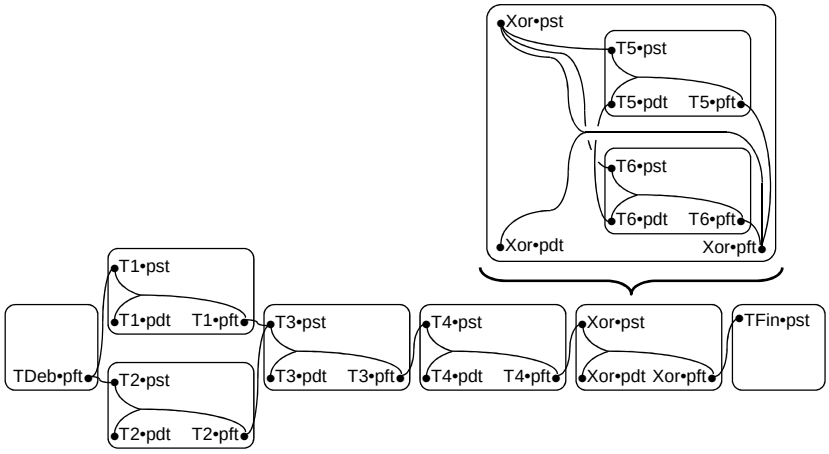


FIGURE 2.11: Modélisation d'un choix d'alternative par ajout de valeur

2.2.3.2 Différents modèles du problème avec sélection de tâches en exclusion et exemple

De manière similaire à la configuration, la formalisation du problème avec des tâches optionnelles est étendue au problème avec des tâches en exclusion. Les deux formalismes *CSP** et *DCSP* sont considérés successivement.

Dans le cas de l'avion de tourisme, l'exemple précédent (section 2.2.2.2), est complété avec deux tâches de finition, qui sont réalisées de différentes manières selon l'exigence du client. Nous avons le choix entre une finition catalogue représentée par la tâche T_5 relativement brève et une finition customisée beaucoup plus longue représentée par la tâche T_6. Les tâches T_5 et T_6 sont définies par :

- $T_5 \bullet pdt$ de domaine $D_{T_5 \bullet pdt} = [5,40]$ et $T_5 \bullet pst$, $T_5 \bullet pft$ de domaine $D_{T_5 \bullet pst} = D_{T_5 \bullet pft} = [0,+\infty[$
- et $T_6 \bullet pdt$ de domaine $D_{T_6 \bullet pdt} = [60,100]$ et $T_6 \bullet pst$, $T_6 \bullet pft$ de domaine $D_{T_6 \bullet pst} = D_{T_6 \bullet pft} = [0,+\infty[$.

Le réseau de tâches correspondant est décrit dans la figure 2.10 et comprend cinq tâches toujours présentes dans tous les projets $(T_{Deb}, T_1, T_2, T_3, T_{Fin})$ contraintes par quatre relations de précédence, ainsi qu'une tâche optionnelle (T_4) liée à ce réseau par deux relations de précédence conditionnées par sa présence dans la solution de planification et deux tâches en exclusion (T_5, T_6) liées au réseau par deux couples de relation de précédence.

2.2.3.3 Modélisation par ajout de valeur (CSP*) et exemple

Dans cette modélisation, il faut représenter le fait qu'une seule des tâches en exclusion doit être prise en compte dans le projet configuré. Pour cela nous considérons que ces tâches sont

optionnelles et introduisons une condition ainsi qu'une durée nulle dans le domaine de définition des tâches. Étant donné n tâches en exclusion $(T_1, T_2 \ldots, T_i, \ldots T_n)$, le modèle est alors :

- $D_{T_i \bullet pdt} = \{0\} \cup D_{T_i \bullet pdt}$
- $(C_i, D_{T_i \bullet pdt}) = \{(faux, 0), (vrai, \neq 0\}$

L'exclusion de tâches est alors obtenue en ajoutant une contrainte de compatibilité entre les durées des tâches en exclusion, qui interdit la présence de deux valeurs non nulles. La contrainte d'exclusion est alors de la forme :

$$C_{Ex}(D_{T_1 \bullet pdt}, D_{T_2 \bullet pdt}, \ldots, D_{T_n \bullet pdt}) = \{(\neq 0, 0, \ldots, 0), (0, \neq, \ldots, 0), \ldots (0, 0, \ldots, \neq 0\}$$

L'ajout du zéro 0 dans le domaine de définition de la durée des tâches génère des résultats non conformes aux attentes lors de la planification : le filtrage considère, pour toutes les tâches en exclusion, une durée minimum de zéro. Ceci ne correspond pas au fonctionnement attendu, où au moins une des tâches doit être exécutée. En effet, la durée à prendre en compte lors de la planification doit être supérieure ou égale à la durée minimum de toutes les tâches en exclusion.

Pour résoudre ce problème, nous introduisons une méta-tâche XOR, qui va encadrer les tâches en exclusion. Cette méta-tâche XOR est une tâche décrite par les trois variables temporelles $(XOR \bullet pst, XOR \bullet pft, XOR \bullet pdt)$, dont la durée est initialisée à l'union des durées non nulles des tâches en exclusion. Par conséquent, lors du filtrage et de la planification de cette méta-tâche, il est bien pris en compte le fait qu'une et une seule des tâches en exclusion doit être obligatoirement sélectionnée : dans le meilleur des cas, la plus rapide (minimum de l'ensemble des durées non-nulles) et dans le pire des cas la plus lente (maximum de l'ensemble des durées).

Ceci conduit au modèle suivant si l'on considère toujours n tâches (T_1, T_2,..., T_i, ..., T_n) :

- définition de méta-tâche XOR :
  - $XOR \bullet pdt$ de domaine $D_{XOR} \bullet pdt = \bigcup \{D_{T_1} \bullet pdt, D_{T_2} \bullet pdt, \ldots, D_{T_n} \bullet pdt\}/0$
  - $XOR \bullet pft = XOR \bullet pst + XOR \bullet pdt$
- contraintes entre méta-tâche et tâches en exclusion :
  - $T_i \bullet pst \geq XOR \bullet pst$
  - $XOR \bullet pft \geq Ti \bullet pft$

Pour notre exemple d'avion de tourisme, les tâches T_5 et T_6 en exclusion conduisent au modèle suivant :

- pour la tâche T_5 :
  - durée de la tâche : $D_{T_5} \bullet pdt = \{0, [5, 40]\}$
  - début de la tâche et fin de la tâche : $D_{T_5} \bullet pst = D_{T_5} \bullet pst = [0, +\infty[$
- pour la tâche T_6 :
  - durée de la tâche : $D_{T_6} \bullet pdt = \{0, [60, 100]\}$
  - début de la tâche et fin de la tâche : $D_{T_6} \bullet pst = D_{T_6} \bullet pst = [0, +\infty[$
- condition d'existence :
  - pour la tâche T_5 : $C_5, T_5 \bullet pdt = \{(faux, 0), (vrai, \neq 0)\}$
  - pour la tâche T_6 : $C_6, T_6 \bullet pdt = \{(faux, 0), (vrai, \neq 0)\}$
- condition d'exclusion : $C_{ex}(T_5 \bullet pdt, T_6 \bullet pdt) = \{([5, 40], 0), (0, [60, 100]\}$
- pour la méta-tâche XOR :

  - durée de la tâche : $D_{XOR} \bullet pdt = \{[5,40],[60,100]\}$
  - début de la tâche et fin de la tâche : $D_{XOR} \bullet pst = D_{XOR} \bullet pst = [0,+\infty[$
- encadrement des tâches en exclusion :
  - $T_5 \bullet pst \geq XOR \bullet pst$
  - $T_6 \bullet pst \geq XOR \bullet pst$
  - $T_5 \bullet pft \leq XOR \bullet pft$
  - $T_6 \bullet pst \leq XOR \bullet pft$
- précédences amont et aval de la méta-tâche XOR :
  - tâche amont : $XOR \bullet pst \geq T_4 \bullet pft$
  - tâche aval : $XOR \bullet pft \leq T_{Fin} \bullet pst$

Ce modèle avec ajout de valeur du problème avec sélection de tâches en exclusion est représenté en figure 2.11 et comprend 5 tâches toujours présentes dans tous les projets (T_{Deb}, $T1$, $T2$, $T3$, T_{Fin}) contraintes par quatre relations de précédence ainsi qu'une tâche optionnelle ($T4$) liée à ce réseau par une relation de précédence et une méta-tâche XOR agrégeant les deux tâches en exclusion ($T5$, $T6$) et liée au réseau par deux contraintes de précédence.

2.2.3.4 Modélisation par ajout d'éléments (DCSP) et exemple

Ce principe de modélisation considère également que les tâches en exclusion sont optionnelles et qu'une et une seule ne peut être ajoutée au problème. Dans le formalisme *DCSP*, la tâche étant ajoutée sur déclenchement d'une contrainte d'activation, il est nécessaire de rendre exclusive les prémisses de ces contraintes. De manière similaire à la configuration (voir 2.1.3.4), nous utilisons également une variable auxiliaire ou sélecteur, dont les valeurs sont les prémisses des contraintes d'activation des tâches.

Étant donné n tâches en exclusion $(T_1, T_2, \ldots T_i, \ldots T_n)$ et la variable sélecteur $Sel - T$, le modèle est alors :

- une variable sélecteur $(Sel - T)$: $D_{Sel-T} = T_1, T_2, \ldots, T_n$,
- une contrainte C_{Ai} d'activation pour chaque tâche T_i

$$C_{Ai} = vrai \Leftrightarrow Sel - T = T_i \xrightarrow{ACT} \{T_i \bullet pst, T_i \bullet pft, T_i \bullet pdt\}$$

Le problème de filtrage évoqué en section précédente est présent. Tant qu'aucune des tâches en exclusion n'est activée, le filtrage ne prend pas en compte ces tâches et considère en conséquence une durée nulle, car en fait inexistante. Le résultat du traitement ne correspond pas au fonctionnement attendu. Il est donc de même nécessaire d'avoir recours au même artifice de modélisation s'appuyant sur une méta-tâche simulant l'existence des tâches en exclusion.

Ceci conduit au modèle suivant, si l'on considère toujours n tâches $(T_1, T_2, \ldots T_i, \ldots T_n)$:

- définition de méta-tâche XOR : $XOR \bullet pdt$ de domaine

$$D_{XOR \bullet pdt} = \bigcup_{i=1}^{n} D_{T_i \bullet pdt}$$

  avec $XOR \bullet pft = XOR \bullet pst + XOR \bullet pdt$
- contraintes entre méta-tâche et tâches en exclusion : $T_i \bullet pst \geq XOR \bullet pst$ et $XOR \bullet pft \geq Ti \bullet pft$

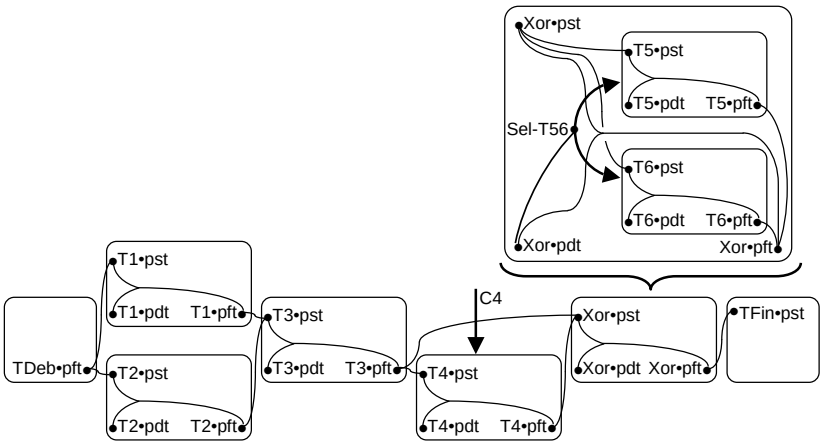


FIGURE 2.12: Modélisation d'un choix d'alternatives par ajout d'éléments

Tant qu'aucune des tâches en exclusion n'est activée, le traitement de filtrage s'effectue sur la méta-tâche XOR et la durée minimum des tâches est exploitée conformément au fonctionnement attendu. Dès que le sélecteur est réduit à un singleton, une unique tâche est ajoutée au problème en suivant exactement le mécanisme décrit pour l'ajout de la tâche optionnelle, décrit en section 2.1.3.4. Seule particularité de ce modèle, lorsqu'aucune des tâches en exclusion n'est activée, le filtrage peut réduire la durée de la méta-tâche (dont le domaine de définition initial est l'union des durées des tâches en exclusion). Cette réduction doit alors pouvoir interdire l'ajout des tâches dans l'ordre des durées décroissantes pour éviter les incohérences. Ceci nécessite une contrainte de compatibilité supplémentaire entre le sélecteur de tâches et la durée de la méta-tâche indiquant les tâches en exclusion compatibles avec la durée de la méta-tâche. Cette contrainte s'écrit :

$$C_C(Sel-T, XOR \bullet pdt) = \{(T_1, D_{T_1 \bullet pdt}), (T_2, D_{T_2 \bullet pdt}) \ldots, (T_i, D_{T_i \bullet pdt}) \ldots, (T_n, D_{T_n \bullet pdt})\}$$

Ce modèle est décrit, dans le cas de l'avion de tourisme, de la manière suivante :

- pour les tâches T_5 :
  - durée de la tâche : $T_5 \bullet pdt = [5, 40]$
  - début de la tâche et fin de la tâche : $D_{T_5 \bullet pst} = D_{T_5 \bullet pft} = [0, +\infty[$
- pour la tâche T_6 :
  - durée de la tâche : $T_6 \bullet pdt = [60, 100]$
  - début de la tâche et fin de la tâche : $D_{T_6 \bullet pst} = D_{T_6 \bullet pft} = [0, +\infty[$
- pour le sélecteur $Sel - T_{56}$
  - de domaine $D_{Sel-T_{56}} = \{T_5, T_6\}$
  - permettant d'activer soit la tâche T_5 : $Sel - T_{56} = T_5 \xrightarrow{ACT} \{T_5 \bullet pst, T_5 \bullet pft, T_5 \bullet pdt\}$
  - permettant d'activer soit la tâche T_6 : $Sel - T_{56} = T_6 \xrightarrow{ACT} \{T_6 \bullet pst, T_6 \bullet pft, T_6 \bullet pdt\}$
- pour la méta-tâche XOR :
  - durée du XOR : $XOR \bullet pdt = [5, 40] \cup [60, 100]$
  - début de la tâche et fin de la tâche : $D_{XOR \bullet pst} = D_{XOR \bullet pft} = [0, +\infty[$
  - encadrement des tâches en exclusion :
    - $T_5 \bullet pst \geq XOR \bullet pst, XOR \bullet pft \geq T_5 \bullet pft$
    - $T_6 \bullet pst \geq XOR \bullet pst, XOR \bullet pft \geq T_6 \bullet pft$

- précédences amont et aval de la méta-tâche XOR :
  - tâches amont T_3 et T_4 : $XOR \bullet pst \geq T_3 \bullet pft, XOR \bullet pst \geq T_4 \bullet pft$
  - tâche aval T_{fin} : $T_{Fin} \bullet pst \geq XOR \bullet pft$
- contrainte sélecteur et durée de méta-tâche :

$$C_c(Sel - T_{56}, XOR \bullet pdt) = \{(T_5, [5,40]), (T_6, [60,100])\}$$

Ce modèle avec ajout de valeur du problème avec sélection de tâches en exclusion est représenté en figure 2.12 et comprend cinq tâches toujours présentes dans tous les projets $(T_{Deb}, T_1, T_2, T_3, T_{Fin})$ contraintes par quatre relations de précédence, ainsi qu'une tâche optionnelle (T_4) liée à ce réseau par deux relations de précédence conditionnées par sa présence dans la solution de planification et une méta-tâche XOR agrégeant les deux tâches en exclusion (T_5, T_6) et liée au réseau par deux contraintes de précédence, une variable sélecteur $(Sel - T_{56})$ liée à la fois à la méta-tâche et aux tâches en exclusion par une contrainte de compatibilité et deux contraintes d'activation.

2.2.3.5 Filtrage du problème avec sélection de tâches en exclusion et exemple

La formalisation des tâches en exclusion, que ce soit avec les $CSP^\star$ ou les *DCSP*, exploite simplement et sans modification les éléments proposés pour les tâches optionnelles en section 2.2.2.5. En conséquence, les remarques relatives au filtrage de la section précédente s'appliquent pleinement au problème mettant en œuvre l'exclusion de tâches.

2.2.4 Planification : problème avec des sous-projets optionnels

Dès qu'un projet comporte un grand nombre de tâches ou devient complexe, il est le plus souvent décomposé en sous-projets. Un sous-projet est, dans ce cas, représenté par une tâche au niveau supérieur et un ensemble de tâches liées par des contraintes de précédence au niveau inférieur. Très peu de travaux se sont intéressés à ce type de problème visant à associer les trois aspects : temporel, conditionnel ou dynamique et composite des *CSP*. Les travaux de (Mouhoub et Sukpan, 2005b) sont néanmoins à mentionner et peuvent constituer une base pour de futurs travaux de recherche.

Comme nous l'avons précisé dans la partie configuration, nous nous intéressons au couplage de la configuration de produit et de la planification de projet en ne considérant qu'un seul niveau d'abstraction. En conséquence, nous ne traiterons pas cet aspect structuré ou multi-niveaux du problème de planification. Par contre, nous avons gardé à l'esprit dans l'ensemble de nos propositions, ce caractère hiérarchique des projets, afin de pouvoir l'intégrer dans de futurs travaux de recherche.

2.2.5 Implémentation des modèles sur les outils logiciels

Les modèles pour la planification de projet sont donc formalisés par un *CSP* temporel quantitatif. Les variables sont les différentes durées et dates des tâches ; les contraintes sont des contraintes numériques représentées par des relations arithmétiques simples. Nous qualifions de « simples » les relations respectant les hypothèses nécessaires à l'emploi des techniques de filtrage de cohérence aux bornes des intervalles (Lhomme, 1993) :

- la fonction doit être monotone,

– la fonction doit être projetable,

– chaque variable de la fonction ne doit apparaître qu'une seule fois dans la contrainte.

Ces hypothèses étant vérifiées par les modèles que nous proposons, l'emploi des outils CoFiADe, ILOG CP 5.5 et ECLiPSe est maintenant discuté pour la planification de manière similaire à la configuration (section 2.1.5).

2.2.5.1 Expérimentation avec CoFiADe

Implémentation

Le nouvel élément à considérer concerne l'implémentation des formules mathématiques. Pour que CoFiADe puisse filtrer correctement les formules ou relations mathématiques, il est nécessaire d'exprimer toutes les projections de chaque relation sur chacune des variables qui sont liées par la formule (Lebbah et al., 2002). Par exemple, la contrainte qui lie la date de fin, la date de début et la durée de la tâche s'exprime par : $T \bullet pft = T \bullet pst + T \bullet pdt$. Cette contrainte doit être projetée sur les trois variables $T \bullet pst, T \bullet pft$ et $T \bullet pdt$. Ce qui donne les trois contraintes suivantes :

– $T \bullet pft = T \bullet pst + T \bullet pdt$,

– $T \bullet pst = T \bullet pft - T \bullet pdt$,

– $T \bullet pdt = T \bullet pft - T \bullet pst$.

Les tâches en exclusion nécessitent également un codage spécifique de la méta-tâche XOR, que le modèle fasse appel au formalisme $CSP^\star$ ou $DCSP$. L'annexe 3.3.3.4 montre le code source du modèle utilisé par CoFiADe correspondant à certaines de ces implantations.

Exploitation Le résultat, en ce qui concerne le filtrage des contraintes numériques, est conforme au principe de l'algorithme d'arc-cohérence aux bornes. Lorsque la contrainte ne respecte pas les hypothèses entre autres de monotonie ou d'unicité d'occurrence de variable, les résultats se dégradent logiquement. Nous avons testé l'exemple de la contrainte $Y = X^2 + X$ avec les domaines de définition $D_X = [-1,2]$ et $D_Y = [-5,10]$. CoFiADe fournit $D_X = [-1,2]$ et $D_Y = [-3,6]$, alors que le résultat théorique est $D_X = [-1,2]$ et $D_Y = [-0.25,6]$. De manière similaire à la configuration, il est nécessaire d'interpréter les modèles exploitant le formalisme $CSP^\star$.

2.2.5.2 Expérimentation avec ILOG CP 5.5

Implémentation Comme il a été indiqué en section 2.1.5.2, ILOG CP 5.5 n'accepte que les variables numériques entières. Les modélisations des problèmes proposés reposant entièrement sur des variables et des contraintes continues, il est nécessaire de recoder et de discrétiser tous les problèmes de planification. Le problème élémentaire de planification a, en conséquence, été discrétisé et implémenté. L'écriture des contraintes numériques est plus simple qu'avec CoFiADe, dans la mesure où il n'est pas nécessaire d'écrire toutes les projections. L'annexe .2 montre des morceaux de code source du modèle utilisé par ILOG CP 5.5 correspondant à cette implémentation.

Exploitation Le filtrage obtenu pour le problème élémentaire équivaut exactement au filtrage obtenu par CoFiADe. Le problème $Y = X^2 + X$ sur $D_X = [-1,2]$ et $D_Y = [-5,10]$ a de même été testé et a fourni le résultat $D_X = [-1,2]$ et $D_Y = [0,6]$ montrant la limite de la manipulation des entiers. Le besoin de ré-écriture des modèles nous a conduit à abandonner les tests systématiques avec ILOG CP 5.5 et à nous concentrer par la suite sur les deux autres outils.

2.2.5.3 Expérimentation avec ECLiPSe

Implémentation Le codage des contraintes sous forme de relations mathématiques s'effectue également plus simplement qu'avec CoFiADe sans le besoin d'énumérer toutes les projections. ECLiPSe, autorisant par contre l'emploi des variables numériques continues, a accepté sans problème l'implémentation des différents problèmes de planification. Outre les modèles de contraintes décrits, une implémentation remplaçant l'écriture de la méta-tâche XOR par un traitement explorant systématiquement la combinatoire des chemins possibles résultant des tâches en exclusion et agrégeant les résultats a été effectuée. Des problèmes comportant une succession de cinq exclusions de cinq tâches (3125 chemins) ont été implémentés. L'annexe .3 montre le code source du modèle utilisé par ECLiPSe correspondant à certaines de ces implantations.

Exploitation Le filtrage des différents problèmes fournit des résultats identiques à ceux de CoFiADe. De même, le problème $Y = X^2 + X$ sur $D_X = [-1,2]$ et $D_Y = [-5,10]$ a été testé et a fourni un troisième résultat $D_X = [-1,2]$ et $D_Y = [-1,6]$ légèrement meilleur que l'approximation réalisée par CoFiADe. L'exploration et l'agrégation de la combinatoire cinq exclusions / cinq tâches sans recourir aux cinq méta-tâches a nécessité quelques minutes de traitement difficilement conciliable avec des besoins de fonctionnement interactif.

2.2.5.4 Conclusions

En ce qui concerne la modélisation, seuls les deux outils CoFiADe et ECLiPSe nous semblent bien adaptés à notre problème de modélisation étant donné que nous considérons fondamentalement du temps continu. Le modèle du problème élémentaire a été discrétisé, implémenté et testé avec ILOG CP 5.5 montrant que son emploi est envisageable, mais l'écriture de traducteurs ou de pré-traitements nous conduisent à nous limiter pour la suite de nos travaux aux expérimentations avec CoFiADe et ECLiPSe. Pour ces deux outils, les remarques sur l'implémentation avec les formalismes *DCSP* et *CSP** sont toujours d'actualité. La qualité du filtrage déjà notée légèrement meilleure avec ECLiPSe en configuration est confirmée lorsque l'on quitte les hypothèses de la cohérence aux bornes.

2.2.6 Problème de planification avec prise en compte des ressources

2.2.6.1 Planification à capacité infinie de ressource

Depuis le début de cette section 2.2, nous nous sommes intéressés uniquement à l'aspect temporel du problème de planification. Nous allons maintenant prendre en compte les ressources. La prise en compte des ressources peut s'effectuer à deux niveaux :

- un premier niveau que l'on peut qualifier de descriptif, où le choix de ressource et la quantité de ressources employées permettent uniquement de moduler la durée des tâches et donc du projet. Cette approche considère, lors de la planification des tâches, que la disponibilité ou la capacité des ressources est infinie,
- le second niveau complète le premier en considérant lors de la planification que la capacité des ressources est finie et sujette à un calendrier de disponibilité.

Étant donné que nous nous intéressons dans ces travaux à des problèmes de configuration de produit et de planification de production répondant à une demande unique spécifiquement définie par un client, nous nous plaçons dans une optique de gestion de délais. Dans cette optique orientée client, la contrainte de délais client doit être privilégiée lors de la planification plutôt que les

capacités des ressources qui peuvent, une fois la planification terminée, être adaptées par un second traitement de vérification de charge. Ce fonctionnement s'apparente à la philosophie *MRP-2* (*Manufacturing Resource Planing*), où seules les contraintes de délais sont prises en compte pour la planification des besoins en composants dans un premier temps et suivies par une vérification de charge, dans un deuxième temps.

Étant donné que nous exploitons les approches par contraintes pour aider interactivement la décision, nous recherchons des techniques de filtrage ou de réduction d'espace de solutions relativement rapides. Comme l'indique très clairement (Bartak et al., 2010), la prise en compte de capacité finie de ressources en planification à capacité finie nécessite l'ajout de multiples contraintes de types « avant ou après », dont le filtrage est peu performant dans le sens où l'espace de solution est très faiblement réduit. Par contre, ce même auteur indique clairement que la propagation sans contraintes de capacité s'effectue de manière performante avec les procédures de filtrage de $2B$-Cohérence. Il est cependant à noter qu'il n'en est pas de même en résolution où la modélisation avec des contraintes dites globales permet de prendre en compte la capacité finie des ressources. Ces deux constats nous conduisent donc à confirmer l'orientation de nos travaux sur la problématique du premier niveau et de considérer les ressources à capacité infinie lorsque nous parlons de planification.

2.2.6.2 Modification du problème avec ressource à capacité infinie

Dans ce cadre, les seules modifications apportées au problème de planification résident dans un complément de définition des tâches à l'aide de besoins en ressource. Pour chaque tâche générique, chaque besoin en ressource définit un groupe de ressources requises et des quantités possibles de ressources. Par défaut, la quantité possible de ressource est unitaire. Une tâche instanciée comprendra pour chaque besoin en ressource, une unique ressource choisie dans le groupe de ressources et une unique quantité.

Un groupe de ressources peut rassembler des ressources différentes pouvant produire :

- des produits ayant les mêmes propriétés, par exemple le poids, la longueur, la puissance ... mais de valeurs différentes, par exemple de 2 à 10 kgs, de 2 à 5 m ou 2 à 10 kws,
- un même produit avec des performances différentes, par exemple une qualité, un coût ou encore une durée différente.

Il apparaît clairement que des relations peuvent relier au sein de la tâche : la ressource choisie dans le groupe, le nombre de ressources requises et la durée de réalisation de la tâche.

La définition du problème de planification élémentaire est en conséquence modifiée en ce qui concerne :

- b) Un projet générique met en œuvre un ensemble de tâches génériques et deux tâches fictives de début-projet et de fin-projet de durée nulle. Une tâche générique est caractérisée par :
  - une durée définie sur un intervalle : [durée minimum, durée maximum],
  - des dates de début et de fin résultant du traitement de planification et définies sur des intervalles : [début plus tôt, début plus tard], [fin plus tôt, fin plus tard],
  - des caractéristiques descriptives figées,
  - des besoins en ressources définis par des couples (groupe de ressources requises, quantité possible de ressources),

  - des relations ternaires peuvent relier : la durée de la tâche, les groupes de ressources requises et les quantités possibles de ressources.
- e) Une exigence utilisateur se traduit par la restriction de l'intervalle de définition (ou la sélection d'une valeur unique) :
  - d'une durée de tâche, d'une date de début au plus tôt de tâche ou d'une date de fin au plus tard de tâche, des dates de tâches de début-projet et de fin-projet,
  - d'un groupe de ressources requises ou de la quantité possible de ressources.
- f) Une solution de planification est une instanciation du projet générique où :
  - pour chaque tâche générique, la date de début, la date de fin et la durée,
  - pour chaque besoin, le groupe de ressources requises et la quantité possible de ressources, ont un domaine de valeurs réduit à un singleton.

2.2.6.3 Modélisation du problème avec ressource à capacité infinie et exemple

Il apparaît que la modification du problème est interne à chaque tâche en ajoutant les notions de ressource requise, de quantité possible et de relation avec la durée de la tâche. La variable ressource requise est une variable définie sur un domaine symbolique, tandis que la variable quantité de ressource est une variable numérique discrète ou continue. Chaque tâche T_i est donc maintenant décrite avec :

- trois variables temporelles
  - durée de tâche : $T_i \bullet pdt$
  - début de tâche : $T_i \bullet pst$
  - fin de tâche : $T_i \bullet pft$
- un couple de variables pour chaque besoin B_j
  - ressource requise : $T_i \bullet rrj$
  - quantité possible : $T_i \bullet qpj$
- une contrainte temporelle : $T_i \bullet pft = T_i \bullet pst + T_i \bullet pdt$
- une contrainte ternaire de ressource associée à chaque besoin, décrivant les combinaisons possibles (durée, ressource requise, quantité de ressources) :

$$C_{T_iB_j}(T_i \bullet pdt, T_i \bullet rrj, T_i \bullet qpj)$$

Cette dernière contrainte peut se restreindre à une contrainte binaire faisant intervenir uniquement des ressources requises et une durée $(T_i \bullet pdt, T_i \bullet rrj)$ ou une quantité de ressources et une durée $(T_i \bullet pdt, T_i \bullet qpj)$. Des contraintes entre les différents besoins d'une même tâche peuvent de même être ajoutées. Lorsque les variables ressources et quantités caractérisent des tâches optionnelles ou en exclusion, les modèles *CSP** et *DCSP* sont complétés en conséquence.

Ces compléments conduisent à un problème mixte (voir section 1.4.2.4) mélangeant des variables discrètes et des continues. Les contraintes de ressources sont mixtes, car elles regroupent variables discrètes (ressource requise et éventuellement quantité) et continues (durée de la tâche et éventuellement quantité). Ces contraintes peuvent néanmoins s'exprimer comme des tableaux de combinaisons de valeurs où les valeurs des variables numériques continues sont des intervalles. Dans ce cas, les algorithmes de filtrage des *CSP* discrets étendus aux contraintes mixtes (Gelle et Faltings, 2003) s'emploient alors sans problème pour filtrer ces problèmes.

En ce qui concerne l'implémentation et l'exploitation de ces compléments dans les outils logiciels CoFiADe et ECLiPSe :

– CoFiADe permet l'implémentation de variables continues et discrètes et la définition de contraintes mixtes sous forme de tableau de valeurs faisant apparaître des intervalles. Le filtrage par arc cohérence des *CSP* discrets a été étendu à ces contraintes mixtes,
– ECLiPSe, de même, permet l'implémentation de variables continues et discrètes et les contraintes mixtes sous une forme similaire. Une librairie spécifique permet le filtrage de ces contraintes mixtes.

2.2.7 Conclusion sur la planification

Cette section a proposé une approche du problème de planification à capacité infinie de ressources et une formalisation sous forme de problème de contraintes en cohérence avec les éléments proposés en configuration. Comme en configuration, trois types de problèmes de planification sans toutefois prendre en compte les ressources (élémentaire, avec tâche optionnelle et tâches en exclusion) ont été abordés. Chaque problème a été défini, formalisé comme un *CSP* et illustré sur un exemple. Les *CSP* continus et le filtrage par cohérence aux bornes associé se sont montrés tout à fait satisfaisants pour aider interactivement la planification. Lorsque les ressources sont prises en compte (tout en considérant toujours leurs capacités comme infinies), il est alors nécessaire de recourir aux formalismes des *CSP* mixtes et à l'extension de l'arc-cohérence des *CSP* discrets aux contraintes mixtes. Les possibilités et les problèmes de filtrage, d'implémentation et d'exploitation des outils logiciels ont de même été discutés. Ces deux premières sections se sont efforcées de suivre une structuration et une terminologie proche pour pouvoir aborder simplement la mise en relation ou couplage, objet de la section suivante.

2.3 Couplage configuration/planification et *CSP*

Les éléments concernant la configuration de système et la planification de sa production étant documentés, nous allons nous intéresser à leur association ou couplage en ayant comme objectif de considérer l'ensemble comme un problème de satisfaction de contraintes. Après une introduction cadrant l'ensemble du propos, nous étudierons le couplage des trois problèmes, certaines de leurs extensions et discuterons l'emploi des outils de propagation de contraintes.

Éléments de définition du couplage configuration/ planification Les deux sous-sections précédentes ont rappelé que les problèmes de configuration et de planification pouvaient être considérés comme des *CSP*. Nous définissons en conséquence, le couplage de ces deux problèmes par leur mise en relation à l'aide de contraintes liant des variables de configuration et de planification et éventuellement complétées par des variables intermédiaires. Étant donné que nous ciblons une aide interactive, notre objectif fondamental est de pouvoir propager les conséquences de décisions prises en configuration vers la planification et inversement propager les conséquences de décisions prises en planification vers la configuration.

Les contraintes reliant ces deux modèles sont issues de l'expérience et de la connaissance des utilisateurs et sont spécifiques à un domaine. Considérant notre exemple d'avion de tourisme, ces contraintes reliant la configuration de produit et la planification de production peuvent exprimer par exemple le fait que :

– l'utilisation de cette technologie nécessite une étape de formation supplémentaire,
– un rotor de plus de 15 kg nécessite deux opérateurs en phase de montage final,
– les avions de plus de quatre tonnes sont systématiquement déplacés avec un pont roulant,

– un délai de réalisation de moins de deux mois interdit les solutions à deux moteurs.

Nous allons en conséquence considérer deux ensembles de variables, les variables du problème de configuration (V_c) et les variables du problème de planification (V_p). Les contraintes de couplage comprendront au minimum une variable de chaque ensemble.

Couplage des problèmes L'étude des problèmes de configuration et de planification s'est déroulée de manière similaire suivant trois problèmes :

– en configuration : le problème élémentaire, le problème avec des éléments optionnels et le problème avec sélection d'éléments en exclusion,

– en planification : le problème élémentaire, le problème avec des tâches optionnelles et le problème avec sélection de tâches en exclusion.

Présenté de cette manière, il semblerait logique d'associer les problèmes de même nature, c'est-à-dire les deux problèmes élémentaires, les deux problèmes avec option et les deux problèmes avec sélection. Mais en fait, rien n'impose cette limite ou cette forme de correspondance, et il est possible de s'interroger sur les neufs associations possibles, présentées dans le tableau 2.4, qui résultent fondamentalement des besoins de modélisation et des choix de description ou de modélisation des deux problèmes. Nous verrons en effet qu'il est possible d'associer le problème élémentaire de configuration avec les trois problèmes de planification et inversement le problème de planification élémentaire avec les trois problèmes de configuration. Par contre, les associations de problèmes avec entités optionnelles et sélection d'entités en exclusion ne seront pas considérées, car l'association des deux problèmes n'a pas d'intérêt opérationnel.

TABLE 2.4: Problèmes de couplage à documenter

Planif. / Config.	Élémentaire	Tâches optionnelles	Sélection de tâches
Élémentaire	Pb1	Pb2.1	Pb3.1
Éléments optionnels	Pb2.2	Pb2.0	rien
Sélection d'éléments	Pb3.2	rien	Pb3.0

En conséquence, nous aborderons sept des neuf problèmes de la manière suivante :

– la section 2.3.1 traitera l'association des problèmes élémentaires de configuration et de planification (Pb1 du tableau 2.4),

– la section 2.3.2 traitera l'association des problèmes avec des entités optionnelles de configuration et de planification et les associations non homogènes entre les problèmes élémentaires et les problèmes à entités optionnelles (Pb2.x du tableau 2.4),

– la section 2.3.3 traitera l'association des problèmes avec sélection d'entités en exclusion de configuration et de planification et les associations non homogènes entre les problèmes élémentaires et les problèmes avec sélection d'entités en exclusion (Pb3.x du tableau 2.4).

Couplage des modèles de problèmes Les travaux sur la configuration ont exploité les formalismes des *CSP* discrets. La modélisation de l'existence des éléments optionnels ou à sélectionner s'est effectuée avec des *CSP** et des *DCSP*. L'aide interactive associée est issue d'algorithmes de filtrage discret basés sur l'arc-cohérence.

Les travaux sur la planification ont fait appel pour la partie temporelle au formalisme des *CSP* temporels quantitatifs et pour la partie ressources à des *CSP* mixtes. La modélisation de l'existence des tâches optionnelles ou à sélectionner s'est effectuée de même avec des *CSP** et des *DCSP*.

L'aide interactive associée est issue d'une part, d'algorithmes de filtrage de $2B$-cohérence pour la partie temporelle du problème et d'autre part, d'algorithmes étendant l'arc-cohérence des *CSP* discrets aux *CSP* mixtes pour la partie mixte du problème.

Les trois sections exploiteront en conséquence :

- pour l'association des problèmes élémentaires, un modèle de couplage associant un *CSP* discret (partie configuration) et un *CSP* mixte (partie planification),
- pour l'association des problèmes avec entités optionnelles, le modèle précédant complété soit avec des $CSP^\star$ avec ajout de valeur émulant l'existence de l'entité, soit avec des *DCSP* avec ajout de variables associées à l'entité.
- pour l'association des problèmes avec sélection d'entités en exclusion, le même type de modèle nécessitant l'emploi des $CSP^\star$ ou des *DCSP*.

Dans ce qui suit, nous considérons trois sous-ensembles de variables en configuration et trois sous-ensembles de variables décrivant les tâches en planification. Les sous-ensembles sont :

- pour la configuration :

  V_{CA} : le sous-ensemble des variables de configuration présentes dans toute solution,

  V_{CO} : le sous-ensemble des variables de configuration optionnelles,

  V_{CS} : le sous-ensemble des variables de configuration en disjonction.
- et pour la planification :

  V_{PA} : le sous-ensemble des variables décrivant les tâches présentes dans toute solution,

  V_{PO} : le sous-ensemble des variables décrivant les tâches optionnelles,

  V_{PS} : le sous-ensemble des variables décrivant les tâches en disjonction.

Présentation de l'exemple illustrant les sections suivantes L'exemple de l'avion de tourisme est toujours considéré et les variables sont distribuées dans les sous-ensembles de la manière suivante. Le problème de configuration comprend huit variables regroupées dans les ensembles suivants :

- $V_{CA} = \{CS, FR, EN, NEN, SN\}$: cinq variables présentes dans toute solution de configuration respectivement : vitesse de croisière, distance de vol, type de moteurs, nombre de moteurs et nombre de sièges,
- $V_{CO} = \{ST\}$: une variable optionnelle, le réservoir supplémentaire,
- $V_{CS} = \{FCA, FCU\}$: deux variables en exclusion respectivement finition catalogue et finition customisée.

Le problème de planification comprend six tâches. Toutes les tâches sont décrites avec trois variables temporelles ($T_i \bullet pst, T_i \bullet pdt, T_i \bullet pft$) et éventuellement des besoins en ressources (ressource requise $T_i \bullet rrj$ et quantité possible $T_i \bullet qpj$) regroupées dans les ensembles suivants :

- $V_{PA} = \{T_1 \bullet pst, T_1 \bullet pdt, T_1 \bullet pft, T_1 \bullet qp1, T_2 \bullet pst, T_2 \bullet pdt, T_2 \bullet pft, T_3 \bullet pst, T_3 \bullet pdt, T_3 \bullet pft, T_3 \bullet qp1\}$

  Les trois tâches (T_1, T_2, T_3) respectivement fabrication, approvisionnement et assemblage sont présentes dans toute solution. La tâche T_2 approvisionnement n'a pas de ressource associée. La tâche T_1 de fabrication a une durée modulée par une quantité de ressource disponible ($T_1 \bullet qp1$) et le nombre de sièges de l'avion (SN). La tâche T_3 d'assemblage a une durée modulée par une quantité de ressource disponible ($T_3 \bullet qp1$), le nombre de sièges de l'avion (SN) et le nombre de moteurs (NEN).

- $V_{PO} = \{T_4 \bullet pst, T_4 \bullet pdt, T_4 \bullet pft, T_4 \bullet qp1\}$
  La tâche (T_4) montage réservoir supplémentaire est une tâche optionnelle, dont la durée est modulée par une quantité de ressource disponible ($T_4 \bullet qp1$) et le type de réservoir à monter (ST).
- $V_{PS} = \{T_5 \bullet pst, T_5 \bullet pdt, T_5 \bullet pft, T_5 \bullet qp1, T_6 \bullet pst, T_6 \bullet pdt, T_6 \bullet pft, T_6 \bullet rr1\}$
  Les tâches (T_5 et T_6) respectivement finition catalogue et customisée sont deux tâches en exclusion. La durée de T_5 est modulée par une quantité de ressources ($T_5 \bullet qp1$) et le type de finition catalogue (FCA). La durée de T_6 est modulée par une ressource requise ($T_6 \bullet rr1$) et le type de finition customisée (FCU).

2.3.1 Couplage : problèmes élémentaires

2.3.1.1 Définition du problème couplant configuration et planification élémentaires

Ce problème comprend l'ensemble des variables présentes dans toutes les solutions du problème de configuration (V_{CA}) et dans toutes les solutions du problème de planification (V_{PA}). Conformément à ce qui a été posé dans l'introduction précédente, les contraintes de couplage associent, par définition, au moins une variable de chacun de ces deux ensembles. Chaque variable du problème de configuration (V_{CA}) représente soit un groupe de composants, soit une propriété du produit. Les variables décrivant chaque tâche du problème de planification (V_{PA}) comprennent systématiquement trois variables temporelles numériques (date début, date de fin, durée : définies par des intervalles de valeur) et éventuellement deux variables ressources pour chaque besoin (ressource requise, quantité requise : respectivement symbolique et numérique).

Bien que ce soit les composants qui soient l'objet des processus de production et/ou d'assemblage, les contraintes de couplage ne se limitent pas à ces seuls composants et aux variables groupes de composants. Il est en effet courant de vouloir relier des propriétés décrivant le produit à des caractéristiques de son processus de réalisation. Considérons, par exemple, « les avions de plus de quatre tonnes sont systématiquement déplacés avec un pont roulant ». En conséquence, nous considérons que les contraintes de couplage peuvent agir aussi bien sur les groupes de composants que sur les propriétés du produit. Toutes les variables de l'ensemble (V_{CA}) sont donc susceptibles d'être des variables de couplage. Nous ne ferons donc plus de distinction entre les variables associées à des groupes de composants et celles associées à des propriétés.

En ce qui concerne les variables de planification, il nous semble raisonnable de faire l'hypothèse que les deux variables associées à toutes les tâches (dates de début et date de fin) n'entrent pas dans les contraintes de couplage. En effet, il nous semble qu'une connaissance reliant une variable de configuration produit à une date de début ou de fin de tâche (par exemple : avion de quatre places en janvier, six places en février) correspond à des phénomènes saisonniers particuliers qu'il est plus judicieux de prendre en compte par le biais d'un besoin utilisateur exprimé en planification. Par contre, les trois types de variables restantes, durée, ressource requise et quantité de ressource, entrent pleinement dans la définition des contraintes de couplage. Il est clair qu'une ou plusieurs variables de configuration de produit peuvent moduler :

- la durée d'une tâche, par exemple, « un délai de réalisation de moins de deux mois interdit les solutions à deux moteurs »,
- une ressource requise, par exemple, « les avions de plus de quatre tonnes sont systématiquement déplacés avec un pont roulant »,
- une quantité de ressources, par exemple, « un rotor de plus de 15 kg nécessite deux opérateurs en phase de montage final ».

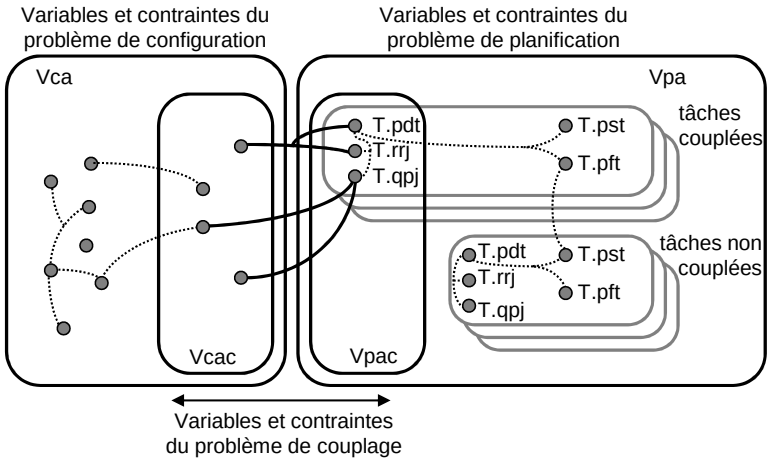


FIGURE 2.13: Modèle du couplage des problèmes élémentaires

Il est également à noter qu'une même contrainte de couplage peut combiner les trois types de variables durée, ressource requise et quantité de ressource. En conséquence, nous considérons que seuls les types de variables durée ($T_i \bullet pdt$), ressource requise ($T_i \bullet rrj$) et quantité de ressources ($T_i \bullet qpj$) de l'ensemble (V_{PA}) sont susceptibles d'être des variables de couplage.

2.3.1.2 Modèle du problème couplant configuration et planification élémentaires

Le modèle couplé final fait apparaître deux ensembles de variables de couplage. Une variable de configuration associée à une contrainte de couplage appartient à l'ensemble des variables de configuration de couplage (V_{CAC}) inclus dans l'ensemble des variables de configuration (V_{CA}). Lorsqu'une tâche de planification est associée à une contrainte de couplage, certaines de ces variables ($T_i \bullet pdt, T_i \bullet rrj, T_i \bullet qpj$) appartiennent à l'ensemble des variables de planification de couplage (V_{PAC}) inclus dans l'ensemble des variables de planification (V_{PA}). La figure 2.13 représente les éléments de ce modèle avec :

- les cercles pleins en gris représentant les variables appartenant soit à V_{CA}, soit à V_{PA},
- la partie gauche du modèle de configuration faisant apparaître le sous-ensemble des variables de couplage V_{CAC},
- la partie droite du modèle de planification faisant apparaître le sous-ensemble des variables de couplage V_{PAC}, les tâches objet de couplage sont dans la partie haute de la figure et les tâches non couplées dans la partie basse,
- les arcs reliant :
  - les variables de V_{CA} sont des contraintes de configuration (pointillés),
  - les variables de V_{PA} sont des contraintes de planification (pointillés),
  - les variables de V_{CAC} et V_{PAC} sont des contraintes de couplage (traits pleins).

Les contraintes de couplage sont discrètes ou mixtes du fait qu'elles relient les variables discrètes du problème de configuration avec des variables continues ($T_i \bullet pdt$ ou $T_i \bullet qpj$) ou discrètes ($T_i \bullet rrj$ ou $T_i \bullet qpj$) du problème de planification. Le modèle du problème couplant configuration et planification élémentaire est donc un *CSP* mixte.

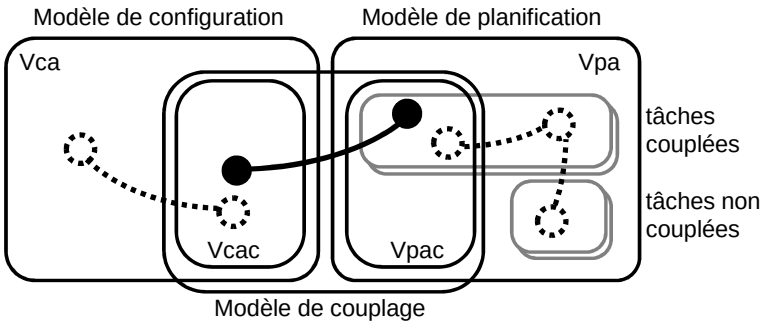


FIGURE 2.14: Trois modèles se complétant

2.3.1.3 Filtrage du problème couplant configuration et planification élémentaires

Le filtrage du problème couplant configuration et planification élémentaire fait appel au filtrage de trois modèles (figure 2.14) se complétant mutuellement :

- filtrage du modèle de configuration (variables et contraintes définies sur V_{CA}),
- filtrage du modèle de planification (variables et contraintes définies sur V_{PA}),
- filtrage du modèle de couplage (variables et contraintes définies sur $V_{CAC} \cup V_{PAC}$).

L'idée proposée est de filtrer successivement chaque modèle de la manière suivante. Dès qu'une variable d'un des deux modèles (configuration ou planification) a son domaine de définition réduit par l'utilisateur, le modèle est filtré. Si ce filtrage impacte une variable du sous-ensemble des variables de couplage (V_{CAC} ou V_{PAC}), le modèle de couplage est filtré. Si ce filtrage impacte une variable de couplage de l'autre modèle (V_{PAC} ou V_{CAC}), ce modèle est alors filtré. Ce processus se reproduit tant que le filtrage génère des réductions de domaine.

Le couplage s'effectue dans les deux sens. Une réduction de domaine d'une variable de configuration impactera les variables de planification et dans le sens opposé, une réduction de domaine d'une variable de planification impactera les variables de configuration. Un tel filtrage peut engendrer l'apparition de discontinuité dans les domaines de variables numériques, qu'elles soient discrètes ou continues. Ce problème aurait pu être déjà mentionné lors de la définition des durées des méta-tâches XOR en section 2.2.

En effet, la durée de ces méta-tâches est définie comme l'union des durées des tâches (union pouvant être un ensemble disjoint d'intervalles de durée). Dans le cas du couplage, le problème peut apparaître dès que l'on associe une variable de configuration symbolique à une durée de tâche numérique. Considérons :

- une variable de configuration symbolique $vc1$ de domaine $\{t_1, t_2, t_3\}$,
- une variable de planification numérique continue durée $T \bullet pdt$ de domaine $[20, 400]$,
- une contrainte de couplage les reliant : $C_{Cpl}(vc1, T \bullet pdt)$ avec la combinatoire : $C_{Cpl} = \{(t_1, [20, 120]), (t_2, [80, 250]), (t_1, [200, 400])\}$

Si le filtrage ou l'utilisateur supprime la valeur 't_2' du domaine de $vc1$, le domaine de définition de la durée $T \bullet pdt$ vaut alors $\{[20, 80] \cup [200, 400]\}$.

Ceci est directement lié à la nature mixte des *CSP* et à l'utilisation interactive qui en est faite. En résolution, les méthodes de filtrage (employées pour réduire l'espace de recherche) conservent le plus souvent l'intervalle englobant et donc des valeurs incohérentes (intervalle $[20, 400]$ dans l'exemple précédent). Ces dernières sont par contre supprimées ultérieurement lorsque le processus d'affectation teste, par dichotomie, les bornes des intervalles.

En filtrage et dans le cadre de l'aide interactive à la décision visée par ce travail, il faut au maximum éviter de proposer des plages de valeurs incohérentes à l'utilisateur. Il est donc nécessaire de pouvoir considérer les domaines numériques multi-intervalles et d'étendre les principes de la $2B$-cohérence aux multi-intervalles.

2.3.1.4 Exemple de modèle couplant configuration et planification élémentaires

Le modèle complet de l'exemple de l'avion de tourisme peut être maintenant obtenu. Il est représenté en figure 2.15 et rassemble les trois modèles :

1. le modèle de configuration avec cinq variables $V_{CA} = \{CS, FR, EN, NEN, SN\}$ correspondant respectivement : vitesse de croisière, distance de vol, type de moteurs, nombre de moteurs et nombre de sièges, et quatre contraintes de configuration définies en section 2.1.1.3 : $C_{C1}(SN, EN), C_{C2}(EN, CS), C_{C3}(FR, CS)$ et $C_{C4}(EN, NEN)$,
2. le modèle de planification avec onze variables : $V_{PA} = \{T_1 \bullet pst, T_1 \bullet pdt, T_1 \bullet pft, T_1 \bullet qp1, T_2 \bullet pst, T_2 \bullet pdt, T_2 \bullet pft, T_3 \bullet pst, T_3 \bullet pdt, T_3 \bullet pft, T_3 \bullet qp1\}$ associées aux trois tâches (T_1, T_2, T_3) respectivement fabrication avec une quantité de ressources, approvisionnement sans variable de ressource ou de quantité et assemblage avec une quantité de ressources, auxquelles il faut ajouter les tâches Tdébut-projet et Tfin-projet.
   Cinq contraintes de précédence entre tâches, trois contraintes de durée définies en 2.2.1.3 et des contraintes de ressources reliant durée et quantité de ressources pour T_1 et T_3,
3. le modèle de couplage avec six variables : $V_{CAC} = \{SN, NEN\}$ et $V_{PAC} = \{T_1 \bullet pdt, T_1 \bullet qp1, T_3 \bullet pdt, T_3 \bullet qp1\}$,
   - une contrainte ternaire modulant la durée de la fabrication des sous-ensembles en fonction du nombre de sièges de l'avion et du nombre de ressources employées $C_{Cc1}(SN, T_1 \bullet pdt, T_1 \bullet qp1)$, présentée dans le tableau 2.5,
   - une contrainte d'arité quatre modulant la durée de l'assemblage de l'avion en fonction du nombre de sièges, du nombre de moteurs de l'avion et du nombre de ressources employées $C_{Cc2}(SN, NEN, T_3 \bullet pdt, T_3 \bullet qp1)$, présentée dans le tableau 2.6.
     Il est à noter que la tâche d'approvisionnement est indépendante de la configuration.

C_{Cc1}

SN	$T_1 \bullet pdt$	$T_1 \bullet qp1$
[3, 4]	[22, 30]	1
[3, 4]	[12, 22]	2
[3, 4]	[10, 12]	3
[6, 8]	[45, 60]	1
[6, 8]	[35, 45]	2
[6, 8]	[30 35]	3
[10 12]	[77, 90]	1
[10,12]	[37, 77]	2
[10 12]	[60, 67]	3

TABLE 2.5: Contrainte de couplage C_{Cc1}

Test 1 : Avec une bonne connaissance du besoin technique lié à un avion particulier, il est possible d'évaluer différentes manières de le réaliser et d'estimer les délais associés. Par exemple, avec un besoin correspondant à un avion performant :

C_{Cc2}

SN	NEN	$T_3 \bullet pdt$	$T_3 \bullet qp1$
[3, 4]	1	[22, 30]	1
[3, 4]	1	[12, 22]	2
[3, 4]	1	[10, 12]	3
[3, 4]	2	[45, 60]	1
[3, 4]	2	[35, 45]	2
[3, 4]	2	[30, 35]	3
[6, 8]	1	[45, 60]	1
[6, 8]	1	[35, 45]	2
[6, 8]	1	[30, 35]	3
[6, 8]	2	[90, 120]	1
[6, 8]	2	[70, 90]	2
[6, 8]	2	[60, 73]	3
[10, 12]	1	[77, 90]	1
[10, 12]	1	[67, 77]	2
[10, 12]	1	[60, 67]	3
[10 12]	2	[135, 180]	1
[10 12]	2	[105, 135]	2
[10, 12]	2	[90, 105]	3

TABLE 2.6: Contrainte de couplage C_{Cc2}

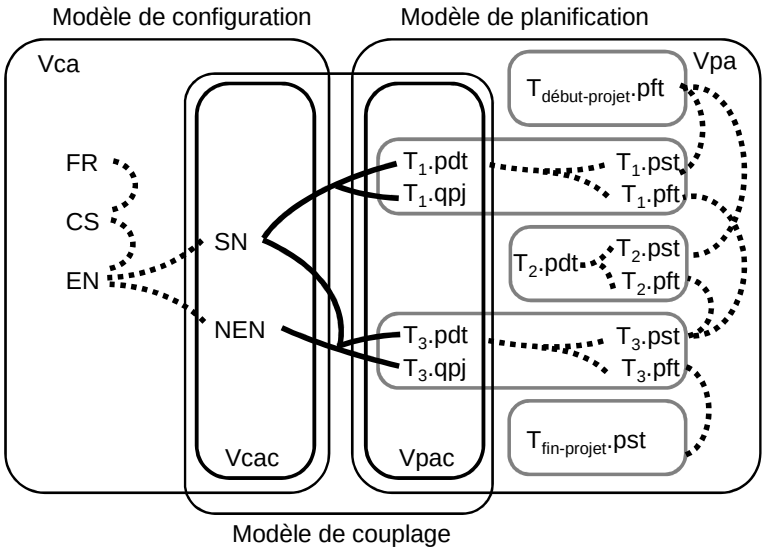


FIGURE 2.15: Modèle complet de l'avion couplant les problèmes élémentaires

– nombre de siège $SN = 10$,

– vitesse $CS = 600$,

– rayon d'action $FR = 1000$.

Le filtrage du modèle de configuration déduira $EN = \{2HP, 2ECO\}$ et $NEN = \{2\}$. Le filtrage du modèle de couplage déduira (pour mémoire la durée $T_2 \bullet pdt$ vaut $[20, 40]$) :

– pour la fabrication $T_1 \bullet pdt = [60, 90]$ et $T_1 \bullet qp1 = \{1, 2, 3\}$

– pour l'assemblage $T_3 \bullet pdt = [90, 180]$ et $T_3 \bullet qp1 = \{1, 2, 3\}$.

Avec une date de début au plus tôt à 0 et une date de fin au plus tard à 250, le filtrage du modèle de planification fournira une date de fin pour la tâche $T_3 \bullet pft = [150, 250]$. Si le nombre minimum de ressources est affecté aux tâches T_1 et T_3, la date de fin devient $T_3 \bullet pft = [207, 250]$.

Test 2 : Avec une connaissance partielle du besoin technique, il est possible de rechercher l'avion qui pourrait être produit le plus rapidement possible. Par exemple, avec un seul besoin correspondant au nombre de siège $SN = 6$, et les mêmes dates de début (0) et de fin (250), le filtrage déduira :

– pour la fabrication $T_1 \bullet pdt = [30, 60]$ et $T_1 \bullet qp1 = \{1, 2, 3\}$,

– pour l'assemblage $T_3 \bullet pdt = [30, 120]$ et $T_3 \bullet qp1 = \{1, 2, 3\}$,

– date de fin de l'assemblage $T_3 \bullet pft = [60, 250]$.

Recherchant le délai minimum, il est possible d'essayer de contraindre la date de fin du projet à la valeur 60. Le filtrage déduira : $T_1 \bullet pdt = T_3 \bullet pdt = \{30\}$, $T_1 \bullet qp1 = T_3 \bullet qp1 = \{3\}$, un seul moteur $NEN = \{1\}$, trois motorisations $EN = \{1LP, 1HP, 1ECO\}$, toutes les vitesses sauf la plus élevée $CS = \{350, 400, 450, 500, 550\}$ et tous les rayons d'action $FR = \{400, 600, 800, 1000, 1200, 1400, 1600\}$.

Test 3 : Dans la mesure où l'utilisateur indique qu'il ne souhaite pas un avion de six ou huit sièges, le filtrage déduira des intervalles multiples pour la durée de fabrication : $T_1 \bullet pdt = \{[10, 30] \cup [60, 90]\}$.

2.3.2 Couplage : problèmes avec entités optionnelles

2.3.2.1 Définition du problème couplé avec entités optionnelles

Nous considérons maintenant la configuration avec éléments optionnels associée à la planification avec tâches optionnelles. Ce problème exploite, comme illustré en figure 2.16, deux nouveaux ensembles de variables :

V_{CO} : ensemble des variables de configuration optionnelles,

V_{PO} : ensemble des variables décrivant les tâches optionnelles

et deux nouveaux sous-ensembles de variables de couplage :

V_{COC} : ensemble des variables de configuration optionnelles de couplage,

V_{POC} : ensemble des variables décrivant les tâches optionnelles de couplage.

Nous considérons que le couplage s'effectue soit entre entités toujours présentes décrites par des variables de (V_{CAC}) et (V_{PAC}), soit entre entités optionnelles représentées par les variables de (V_{COC}) et (V_{POC}). Ce problème correspond au problème Pb2.0 du tableau 2.4, les problèmes non homogènes couplant (V_{CO}) et (V_{PA}) ou (V_{CA}) et (V_{PO}) seront discutés en section 2.3.2.4.

Les variables optionnelles susceptibles d'appartenir aux contraintes de couplage sont rigoureusement de même type que celles du problème élémentaire :

– groupe de composants ou propriétés pour la configuration,
– durée de tâche, ressource requise ou quantité de ressources pour la planification.

Nous aboutissons en conséquence au problème de la figure 2.16, qui n'est pas un modèle car la formalisation des conditions d'existence (double flèche pleine noire sur la figure) dépend du type de formalisme employé (*CSP** ou *DCSP*). Ce problème reprend le modèle de la figue 2.14 et le duplique pour la partie optionnelle.

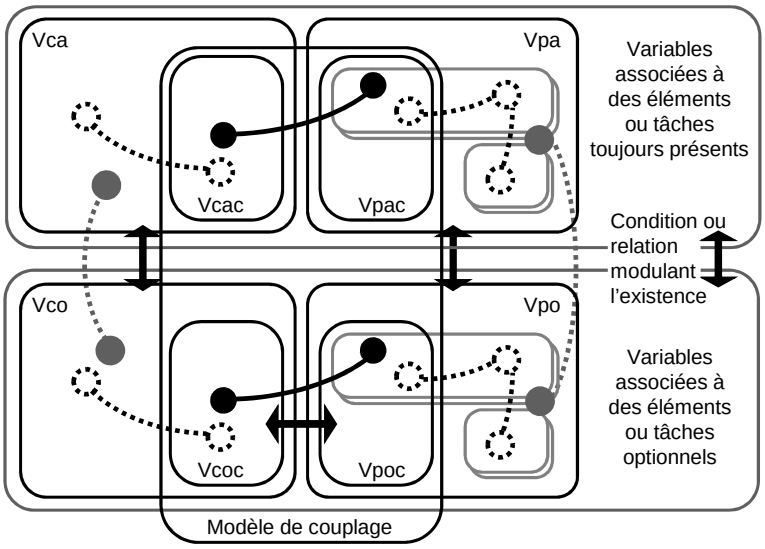


FIGURE 2.16: Problème couplé avec entités optionnelles

Trois types de conditions d'existence peuvent apparaître entre :

– V_{CA} et V_{CO} : ce sont les conditions d'existence du problème de configuration,
– V_{PA} et V_{PO} : ce sont les conditions d'existence du problème de planification,
– V_{COC} et V_{PAC} : ce sont les conditions d'existence qui couplent les deux problèmes.

Si les premier et troisième types de condition d'existence semblent logiques, le second type (double flèche noire entre V_{PA} et V_{PO}) issu du problème de planification est moins évident.

En fait, la modulation d'existence de tâche dans le modèle proposé en section 2.2.2 est uniquement liée au filtrage du problème de planification et de ses contraintes de précédence entre tâches. En conséquence, les conditions d'existence (double flèche noire entre V_{PA} et V_{PO}) sont assimilables à ces contraintes de précédence entre les tâches optionnelles et les tâches toujours présentes (pointillé grisé entre V_{PA} et V_{PO}).

Les arcs en pointillés représentent les contraintes de compatibilité de configuration (à gauche) et de planification (à droite). Ceux qui sont grisés représentent les contraintes entre les variables décrivant des entités toujours présentes et les entités optionnelles. Les arcs en trait plein noir sont les contraintes de couplage.

2.3.2.2 Modèle et filtrage du problème couplé avec entités optionnelles

Les modèles diffèrent légèrement suivant le formalisme *CSP** ou *DCSP* employé. Dans les deux cas, le modèle du problème élémentaire est entièrement conservé avec ses trois sous-modèles :

– modèle de configuration (variables et contraintes définies sur V_{CA}),

– modèle de planification (variables et contraintes définies sur V_{PA}),
– modèle de couplage (variables et contraintes définies sur $V_{CAC} \cup V_{PAC}$).

Les entités optionnelles sont de même représentées avec trois sous-modèles :

– modèle de configuration (variables et contraintes définies sur V_{CO}),
– modèle de planification (variables et contraintes définies sur V_{PO}),
– modèle de couplage (variables et contraintes définies sur $V_{COC} \cup V_{POC}$).

Suivant le formalisme utilisé, la modélisation des conditions d'existence des entités optionnelles diffèrent. Le domaine de définition des variables associées aux entités optionnelles, ainsi que les contraintes de compatibilité, peuvent également différer légèrement.

Formalisme *CSP** Ce formalisme nécessite l'ajout d'une valeur spécifique ($\star$ ou 0) dans le domaine des variables émulant l'absence de l'élément ainsi que l'ajout de contraintes de compatibilité. Pour la partie configuration et les variables appartenant à V_{CO}, la valeur $\star$ est ajoutée dans le domaine de toutes les variables optionnelles. Pour la partie planification et les variables appartenant à V_{PO}, la valeur 0 est ajoutée dans le domaine des variables durée ($T_i \bullet pdt$) et quantité de ressource ($T_i \bullet qpj$) et la valeur $\star$ est ajoutée pour les variables ressource requise ($T_i \bullet rrj$). Avec le formalisme *CSP**, les tâches sont toujours présentes dans le modèle (même avec une durée nulle) et leurs dates de début et de fin résultant du filtrage sont toujours présentes dans le problème.

En ce qui concerne les contraintes de compatibilité, celles qui sont ajoutées concernent :

– les variables de V_{CA} et V_{CO}, ceci pour moduler l'existence des variables de configuration en fonction d'autres variables de configuration,
– les variables de V_{COC} et V_{POC}, ceci pour moduler l'existence de variables attachées à des tâches de planification optionnelles (durée, ressource requise et quantité de ressources) en fonction de variables de configuration optionnelles.

Formalisme *DCSP* Ce formalisme nécessite uniquement l'ajout de contraintes d'activation : $prémisse \xrightarrow{ACT} conséquent$. Il est à noter que la prémisse d'une telle contrainte peut porter aussi bien sur des variables toujours présentes (V_{CA} ou V_{PA}) que sur des variables optionnelles (V_{CO} ou V_{PO}). Par contre, le conséquent d'une telle contrainte sera toujours lié à une variable optionnelle (V_{CO} ou V_{PO}). Il est rappelé également que dans ce modèle, une contrainte est prise en compte et filtrée uniquement si toutes ses variables sont actives.

Les variables des modèles sont donc inchangées et les contraintes d'activation à ajouter permettent de manière similaire au formalisme *CSP** :

– l'activation des variables de V_{CO} en fonction des variables V_{CA}, ceci pour moduler l'existence des variables de configuration en fonction d'autres variables de configuration,
– l'activation des variables de V_{POC} en fonction des variables V_{COC}, ceci pour moduler l'existence de variables attachées à des tâches de planification optionnelles (durée, ressource requise et quantité de ressources) en fonction de variables de configuration optionnelles.

Filtrage des modèles du problème couplé avec entités optionnelles Le principe de filtrage proposé pour le problème élémentaire (basé sur les trois filtrages : configuration, planification et couplage) est rigoureusement identique. Les remarques concernant le rejet de solutions admissibles par le formalisme *CSP** et la perte de cohérence lors de l'activation de variables soulevé par le formalisme *DCSP* sont toujours à considérer dans le problème avec entités optionnelles.

Ce dernier problème de perte de cohérence avec le formalisme *DCSP* est particulièrement présent avec des tâches optionnelles. En présence d'une planification relativement contrainte, toute

modification de la configuration générant une tâche optionnelle peut alors rendre le problème de planification incohérent. Le seul moyen de limiter l'occurrence de ces incohérences est de traiter le problème de configuration dans un premier temps, pour ensuite prendre en compte les contraintes de planification.

2.3.2.3 Exemple de modèle du problème couplé avec entités optionnelles

Dans le cas de l'avion, représenté en figure 2.17, nous considérons maintenant :

- un groupe de composants optionnels, le réservoir supplémentaire associé à la variable ST, seule variable de l'ensemble V_{CO},
- une tâche optionnelle correspondant au montage de ce réservoir supplémentaire T_4 décrite par les variables de l'ensemble V_{PO} : durée ($T_4 \bullet pdt$), date de début et de fin ($T_4 \bullet pst, T_4 \bullet pft$), ainsi que par une quantité de ressources possible ($T_4 \bullet qp1$).

L'existence des ces entités optionnelles est modulée :

- pour ce qui est du réservoir supplémentaire ST, par les deux variables distance de vol (FR) et vitesse de croisière (CS) de la contrainte $(CS > 380 \wedge FR > 1100) \vee (CS > 48 \wedge FR > 700)$ décrite en section 2.1.2.2,
- pour la tâche de montage réservoir, par la variable réservoir supplémentaire ST de la contrainte de couplage, qui conditionne logiquement l'existence de la tâche montage du réservoir et des variables qui la décrivent.

Ces contraintes sont représentées par les doubles flèches en figure 2.17 et seront ensuite particularisées pour chaque formalisme $CSP^\star$ et $DCSP$.

De plus, une contrainte de couplage module la durée de montage du réservoir $T4 \bullet pdt$ en fonction du type de réservoir ST et de la quantité de ressources possible $T4 \bullet qp1$.

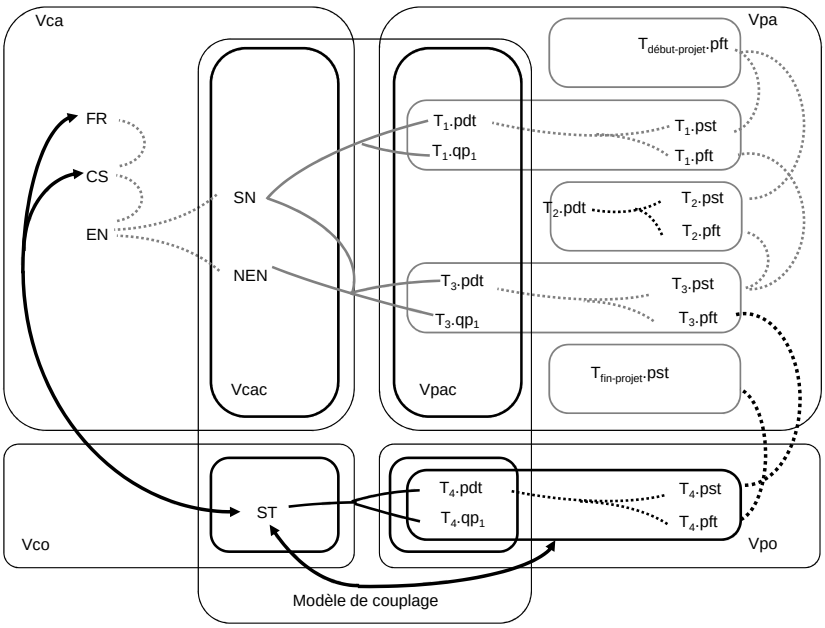


FIGURE 2.17: Modèle complet couplant les problèmes avec éléments optionnels

Ces éléments formalisés avec les $CSP^\star$ débouchent sur le modèle détaillé en partie droite de la figure 2.18. Ce modèle comprend les variables optionnelles :

– $ST : D_{ST} = \{\star, ST_{100}, ST_{200}, ST_{300}\}$
– $T_4 \bullet pdt : D_{T_4 \bullet pdt} = \{0 \cup [5,40]\}$
– $T_4 \bullet qp1 : D_{T_4 \bullet qp1} = \{0,1,2,4\}$
– $T_4 \bullet pst$ et $T_4 \bullet pft : D_{T_4 \bullet pst} = D_{T_4 \bullet pst} = \{[0,+\infty[\}$

et les contraintes modulant l'existence des variables optionnelles :

– du modèle de configuration : $C_{C5}(CS, FR, ST)$ définie en extension dans la table 2.2 page 46, qui fusionne existence et compatibilité,
– du modèle de couplage : $C_{Cpl4}(ST, T_4 \bullet pdt, T_4 \bullet qp1)$ définie en extension dans la table de la figure 2.18 et qui fusionne également existence et compatibilité.

ST	T_4•pdt	T_4•qp$_1$
*	0	0
ST_{100}	[15, 20]	1
ST_{100}	[8, 15]	2
ST_{100}	[5,8]	4
ST_{200}	[22, 30]	1
ST_{200}	[11,22]	2
ST_{200}	[8, 11]	4
ST_{300}	[30,40]	1
ST_{300}	[15,30]	2
ST_{300}	[10,15]	4

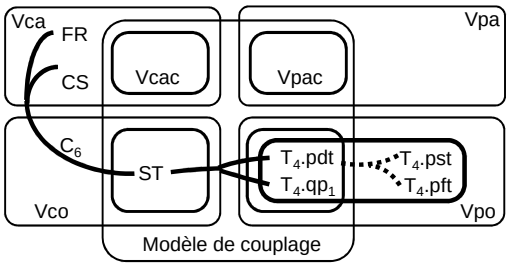


FIGURE 2.18: détail du modèle des contraintes d'existence avec les $CSP^\star$

Ces éléments formalisés avec les *DCSP* débouchent sur le modèle détaillé en partie droite de la figure 2.19. Ce modèle comprend les variables optionnelles :

– $ST : D_{ST} = \{ST_{100}, ST_{200}, ST_{300}\}$
– $T_4 \bullet pdt : D_{T_4 \bullet pdt} = \{[5,40]\}$
– $T_4 \bullet qp1 : D_{T_4 \bullet qp1} = \{1,2,4\}$
– $T_4 \bullet pst \wedge T_4 \bullet pft : D_{T_4 \bullet pst} = D_{T_4 \bullet pst} = \{[0,+\infty[\}$.

Les contraintes modulant l'existence des variables optionnelles :

– $(CS > 380 \wedge FR > 1100) \vee (CS > 480 \wedge FR > 700) \xrightarrow{ACT} ST$, définie en section 2.1.2.2,
– $STactive \xrightarrow{ACT} \{T_4 \bullet pst, T_4 \bullet pft, T_4 \bullet pdt, T_4 \bullet qp1\}$

Les contraintes de compatibilité entre ces mêmes variables, qui sont dissociées des contraintes modulant l'existence :

– (ST, FR, CS) définie en extension dans la section 2.1.2.2,
– $(ST, T_4 \bullet pdt, T_4 \bullet qp1)$ définie en extension dans la table de la figure 2.19

ST	T_4• pdt	T_4• qrs
ST_{100}	[15, 20]	1
ST_{100}	[8, 15]	2
ST_{100}	[5,8]	4
ST_{200}	[22, 30]	1
ST_{200}	[11,22]	2
ST_{200}	[8, 11]	4
ST_{300}	[30,40]	1
ST_{300}	[15,30]	2
ST_{300}	[10,15]	4

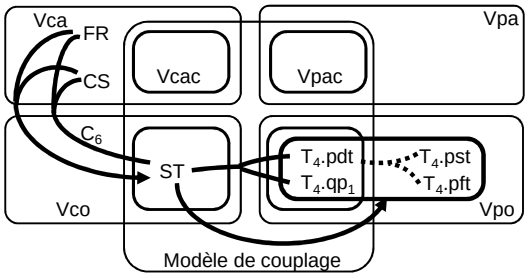


FIGURE 2.19: Détail du modèle des contraintes d'existence avec les *DCSP*

En considérant un exemple avec les besoins initiaux suivants : $CS = \{600\}$, $SN = \{10\}$, correspondant partiellement au besoin étudié en section précédente, le filtrage réduit la distance de vol FR à $\{400, 600, 800, 1000\}$ et fournit les mêmes durées de tâche :

- de fabrication $T_1 \bullet pdt = [60, 90]$,
- d'approvisionnement $T_2 \bullet pdt = [20, 40]$
- d'assemblage $T_3 \bullet pdt = [90, 180]$,

Dans le cas des $CSP^\star$, le réservoir supplémentaire est réduit $ST = \{\star, ST_{300}\}$ et la durée de la tâche du montage de réservoir est réduite : $T_4 \bullet pdt = \{0 \cup [10, 40]\}$.

Dans le cas des *DCSP*, la condition d'activation ne se déclenche pas, car la distance de vol FR n'est pas réduite. En conséquence, la variable ST et la tâche T_4 ne sont pas prises en compte dans le problème.

Avec une date de fin de projet réduite de 250 à 155, la date de début étant inchangée à 0, le filtrage fournit alors entre autres :

- date de fin de fabrication $T_3 \bullet pft = [60, 65]$,
- date de fin d'approvisionnement $T_3 \bullet pft = [20, 65]$,
- date de fin d'assemblage $T_3 \bullet pft = [150, 155]$.

Dans le cas des $CSP^\star$, la durée du montage de réservoir est réduite, $T_4 \bullet pdt = \{0\}$, la variable réservoir supplémentaire est également réduite, $ST = \{\star\}$, ainsi que la distance de vol, $FR = \{400, 600\}$. L'utilisateur ne peut pas fournir un besoin sur FR de 800 ou 1000.

Dans le cas des *DCSP*, aucun autre filtrage ne se produit. L'utilisateur peut fournir un besoin sur FR de $\{800\}$ ou 1000. Celui-ci va déclencher la contrainte d'activation du réservoir et de la tâche et rendre le problème incohérent. Si le besoin de $FR = 800$ avait été exprimé avant la contrainte de délai, l'ajout de la tâche aurait conduit à une date de fin au plus tôt de 160, rendant alors impossible la saisie d'une date de fin égale à 155.

Cet exemple montre l'occurrence d'une incohérence avec l'emploi du formalisme *DCSP*. Il illustre également comment la prise en compte des besoins en configuration avant ceux de planification permet de réduire celles-ci.

2.3.2.4 Couplage non homogène avec entités optionnelles

Les modèles proposés précédemment ont considéré des contraintes de couplage entre :

- les variables de V_{CAC} et V_{PAC},
- les variables de V_{COC} et V_{POC}.

Les problèmes non homogènes identifiés en introduction dans le tableau 2.4 :

- configuration élémentaire et planification avec tâches optionnelles (Pb2.1),
- configuration avec entités optionnelles et planification élémentaire (Pb2.2).

correspondent respectivement à des contraintes de couplage reliant :

- les variables de V_{CAC} et V_{POC},
- les variables de V_{COC} et V_{PAC}.

Le problème Pb2.1 signifie simplement que des variables toujours présentes dans le problème de configuration peuvent piloter l'existence d'une tâche. Le problème Pb2.2 signifie que des variables optionnelles dans le problème de configuration peuvent moduler des caractéristiques d'une tâche sans altérer son existence.

D'un point de vue modélisation, les deux formalismes *CSP** et *DCSP* s'utilisent sans complément particulier. Ces trois problèmes Pb2.0, Pb2.1 et Pb2.2 peuvent se combiner et leur emploi est uniquement lié à la manière de modéliser la réalité et au niveau de granularité des modèles. Un modèle détaillé (resp. agrégé) de configuration et plus agrégé (resp. détaillé) de planification conduira probablement à un Pb2.2 (resp. un Pb2.1).

Sur l'exemple de l'avion, un exemple de modèle suivant le Pb2.1 :

- ne considérerait pas la variable réservoir supplémentaire ST,
- relierait directement la condition d'existence de la tâche de montage du réservoir T_4 décrite par les variables de V_{POC} aux variables vitesse CS et distance de vol FR éléments de V_{CAC}.

A l'opposé, le problème Pb2.2 :

- ne considérerait pas la tâche de montage du réservoir T_4,
- considérerait la variable réservoir supplémentaire ST et l'associerait par une contrainte de couplage à la tâche T_3 pour moduler sa durée.

2.3.3 Couplage : problèmes avec sélection d'entités en exclusion

2.3.3.1 Définition du problème couplé avec sélection d'entités en exclusion

Cette section s'intéresse aux problèmes couplant la configuration avec sélection d'éléments associée à la planification avec sélection de tâches. Ce problème fait appel à deux nouveaux ensembles de variables :

V_{CS} : ensemble des variables de configuration à sélectionner en exclusion,

V_{PS} : ensemble des variables décrivant les tâches à sélectionner en exclusion.

Et deux nouveaux sous-ensembles de variables de couplage :

V_{CSC} : ensemble des variables de configuration de couplage à sélectionner en exclusion,

V_{PSC} : ensemble des variables décrivant les tâches de couplage à sélectionner en exclusion.

De même, nous ne considérons que le problème homogène couplant les variables V_{CAC} avec V_{PAC} et V_{CSC} avec V_{PSC} correspondant au problème Pb3.0 du tableau 2.4. Les problèmes non homogènes seront discutés en section 2.3.3.4.

Les variables à sélectionner en exclusion susceptibles d'appartenir aux contraintes de couplage sont, comme pour les problèmes avec entités optionnelles, de même type que celles du problème élémentaire :

- groupe de composants ou propriétés pour la configuration,
- durée de tâche, ressource requise ou quantité de ressources pour la planification.

Quatre types de condition d'exclusion (double flèche noire) peuvent apparaître au sein des ensembles :

- V_{CS} et V_{PS} : exclusions indépendantes de tout couplage configuration-planification,
- V_{CSC} : exclusions en configuration et objet de couplage,

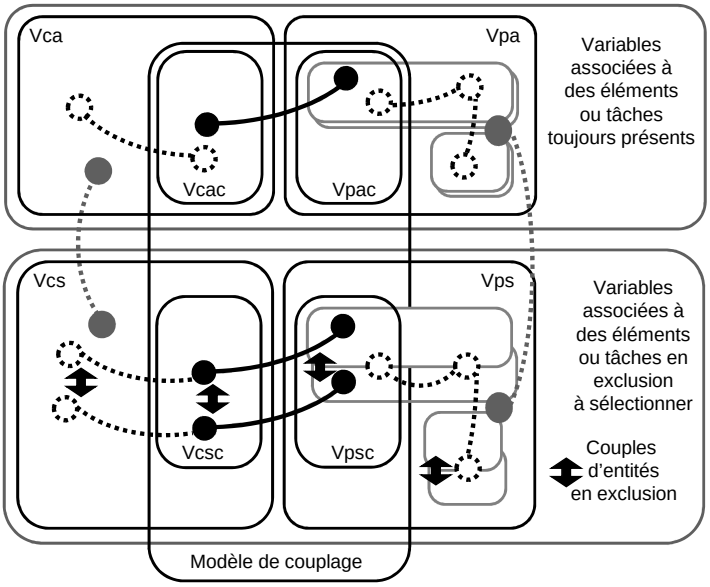


FIGURE 2.20: Problème couplé avec entités à sélectionner en exclusion

– V_{PSC} : exclusions en planification et objet de couplage.

Les arcs en pointillés, de la figure 2.20, représentent les contraintes de compatibilité de configuration (à gauche) et de planification (à droite). Ceux qui sont grisés représentent les contraintes entre les variables décrivant des entités toujours présentes et celles qui sont à sélectionner en exclusion. Les arcs en trait plein noir sont les contraintes de couplage.

2.3.3.2 Modèle et filtrage du problème couplé avec sélection d'entités en exclusion

Les modèles diffèrent légèrement suivant le formalisme *CSP** ou *DCSP* employé. Dans les deux cas, le modèle du problème élémentaire est entièrement conservé avec ces trois sous-modèles :

– modèle de configuration (variables et contraintes définies sur V_{CA}),
– modèle de planification (variables et contraintes définies surV_{PA}),
– modèle de couplage (variables et contraintes définies sur $V_{CAC} \cup V_{PAC}$).

Les entités à sélectionner sont de même représentées avec trois sous-modèles :

– modèle de configuration (variables et contraintes définies sur V_{CS}),
– modèle de planification (variables et contraintes définies sur V_{PS}),
– modèle de couplage (variables et contraintes définies sur $V_{CSC} \cup V_{PSC}$).

Suivant le formalisme, la modélisation des conditions de sélection des entités en exclusion diffèrent.

Formalisme *CSP** Pour la partie configuration, la valeur * est ajoutée dans le domaine de toutes les variables appartenant à V_{CS}. Pour la partie planification et les variables appartenant à V_{PS}, la valeur 0 est ajoutée dans le domaine des variables durée ($T_i \bullet pdt$) et quantité de ressources ($T_i \bullet qpj$) et la valeur $\star$ est ajoutée pour les variables ressource requise ($T_i \bullet rrj$).

Les contraintes de compatibilité formalisant les exclusions sont ajoutées indépendamment pour les parties configuration et planification :

– en configuration pour les variables de (V_{CS}), une contrainte excluant le tuple $(\neq \star, \neq \star)$ interdit la présence simultanée de deux éléments en exclusion (voir sous-section 2.1.3.3),

– en planification pour les variables de (V_{PS}), ce sont les durées des tâches en exclusion qui sont objet du même type de contrainte excluant le tuple $(\neq 0, \neq 0)$ forçant une unique durée non nulle. Une méta-tâche XOR complète la modélisation pour éviter la durée nulle lors du filtrage (voir sous-section 2.2.3.3).

Les contraintes reliant les variables des deux ensembles précédents (V_{CSC}) et (V_{PSC}), permettent alors de coupler des éléments en exclusion du modèle de configuration avec des tâches en exclusion du modèle de configuration. Ce type de contrainte autorise les associations de valeurs $(\star, 0)$ (resp. $(\neq \star, \neq 0)$) dans le but d'imposer simultanément l'absence (resp. l'existence) des entités dans chaque modèle de configuration et de planification. Cette contrainte émulant l'existence simultanée des entités peut être complétée par une contrainte de compatibilité pilotant la durée, le type de ressource requise et la quantité de ressource requise par la tâche.

Formalisme *DCSP* Comme pour les entités optionnelles, ce modèle ne nécessite pas de modification du domaine de définition des variables, mais l'ajout de contraintes d'activation. Par contre, comme cela a été montré dans les sections 2.1.3.4 et 2.2.3.4, la modélisation des éléments en exclusion doit s'effectuer à l'aide d'une variable de type sélecteur qu'il faut ajouter pour chaque ensemble d'entités en exclusion. Ces variables sélecteur n'appartiennent ni à V_{CS} ou V_{PS}, mais à V_{CA} ou V_{PA}, car ces sélecteurs, s'ils permettent la sélection, ne sont pas eux mêmes sélectionnés. Il est rappelé que la prémisse de chaque contrainte d'activation correspondra alors à un test de valeur du sélecteur.

Le conséquent de cette contrainte correspondra à l'élément devant être activé (variable de configuration ou variables décrivant la tâche de planification). De même, une méta-tâche XOR complète la modélisation de la partie planification pour éviter la prise en compte d'une durée nulle en filtrage.

Le couplage des éléments en exclusion des deux modèles s'effectue dans le cas des *DCSP* par la mise en relation des deux sélecteurs avec une contrainte de compatibilité. Les sélecteurs sont en conséquence dans les ensembles V_{CAC} ou V_{PAC}. Le couplage peut être alors finalisé par une contrainte de compatibilité définie sur $V_{CSC} \cup V_{PSC}$ modulant la durée, le type de ressource requise et la quantité de ressource requise par la tâche.

2.3.3.3 Exemple de modèle du problème couplé avec sélection d'entités en exclusion

Nous considérons l'exemple du problème couplant configuration et planification élémentaire étudiée en section 2.3.1, auquel nous ajoutons des entités à sélectionner. Dans l'exemple de l'avion, les entités à sélectionner en exclusion correspondent à deux types de finition (catalogue (FCA) ou customisée (FCU)) pour la partie configuration, que l'on va coupler aux deux tâches de finition en planification (finition catalogue (T_5) ou finition customisée (T_6)). La durée de la tâche (T_5) dépend du type de finition et de la quantité de ressource ($T_5 \bullet qp1 = \{1,2,4\}$) affectée à la tâche. La durée de la tâche (T_6) dépend du type de finition et du type de ressource ($T_6 \bullet rr1 = \{t1,t2,t3\}$) affectée à la tâche.

En ce qui concerne la configuration, les deux finitions sont complètement indépendantes des autres variables du problème (voir section 2.1.3.2). Par contre, les deux tâches de finition sont reliées aux autres tâches par des contraintes de précédence (voir section 2.2.3.2). En conséquence, nous reprenons l'intégralité du modèle du problème élémentaire détaillé en figure 2.15, que nous allons compléter suivant les deux formalismes étudiés.

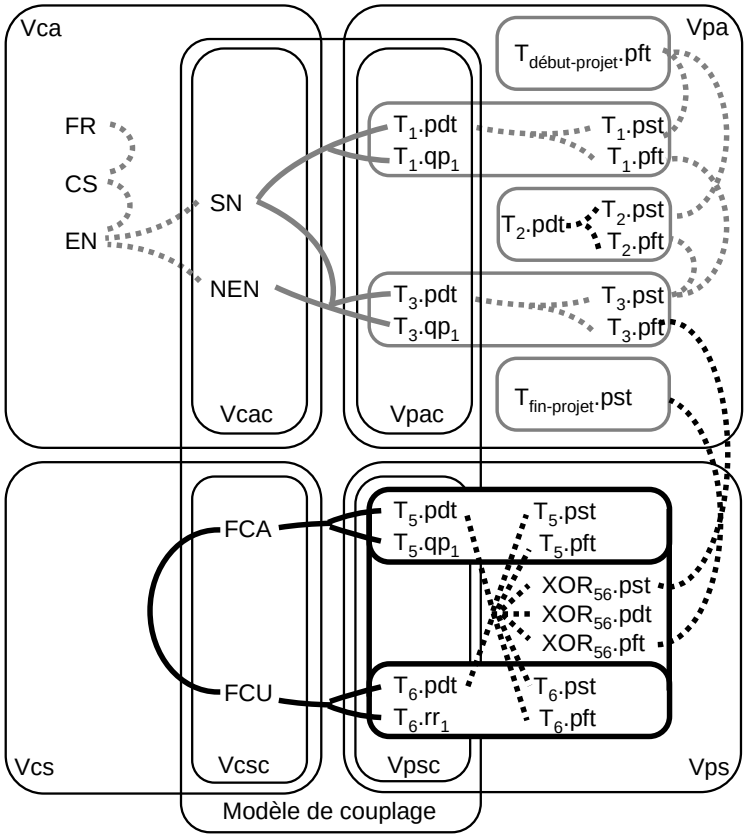
Formalisme *CSP** Le modèle associé est représenté en figure 2.21. Pour la partie configuration, les deux variables FCA et FCU sont ajoutées dans l'ensemble des variables de configuration à sélectionner de couplage (V_{CSC}). Leur domaine de définition est complété avec la valeur $\star$. La contrainte d'exclusion entre ces deux finitions (définie en section 2.1.3.3) est ajoutée :

$$C_{Ex}(FCA, FCU) = \{(\star, work), (\star, leisure), (standard, \star), (confort, \star), (luxe, \star)\}$$

Pour la partie planification, les variables décrivant les deux tâches (T_5) et (T_6) sont ajoutées dans l'ensemble des variables de planification à sélectionner de couplage (V_{PSC}). Le domaine de définition des variables : durée, ressource requise et quantité de ressources est de même complété avec $\star$ ou 0. La méta-tâche XOR_{56} incluant T_5 et T_6 est de même ajoutée, mais non représentée en détail. Elle inclut la contrainte d'exclusion entre ces deux tâches (définie en section 2.2.3.3)

En ce qui concerne le couplage, chaque type de finition (FCA et FCU) est associé à une tâche de finition (T_5 et T_6) par une contrainte de compatibilité (dont les tables sont données en partie droite de la figure 2.21) permettant de garantir la cohérence de présence simultanée des éléments adéquats dans les deux modèles. Chacune de ces deux contraintes est complétée avec une contrainte modulant la durée de la tâche de finition en fonction du type de finition de la ressource requise ou de la quantité de ressource :

- $C_{Cpl}(FCA, T_5 \bullet pdt, T_5 \bullet qp1)$,
- $C_{Cpl}(FCU, T_6 \bullet pdt, T_6 \bullet rr1)$.



FCA	T5•pdt	T5•qp1
*	0	0
Standard	[15,20]	1
Standard	[8,15]	2
Standard	[5,8]	4
Confort	[22,30]	1
Confort	[11,22]	2
Confort	[8,11]	4
Luxe	[30,40]	1
Luxe	[15,30]	2
Luxe	[10,15]	4

FCU	T6•pdt	T6•rr1
*	0	0
Work	[50,70]	t1
Leisure	[50,70]	t1
Work	[70,90]	t2
Leisure	[70,90]	t2
Work	[90,110]	t3
Leisure	[90,110]	t3

Contraintes de couplage

FIGURE 2.21: Modèle couplé avec sélection d'entités en exclusion avec les *CSP**

Formalisme *DCSP* Le modèle associé est représenté en figure 2.22. Pour la partie configuration, les deux variables FCA et FCU sont également ajoutées à l'ensemble des variables de configuration à sélectionner de couplage (V_{CSC}). Par contre, l'existence de chacune de ces variables est le conséquent d'une contrainte d'activation dont la prémisse est le sélecteur de finition SF (de domaine $D_{SF} = \{fca, fcu\}$ définie en section 2.1.3.4) :

- $SF = fca \xrightarrow{ACT} FCA$
- $SF = fcu \xrightarrow{ACT} FCU$

Pour la partie planification, les variables décrivant les deux tâches (T_5) et (T_6) sont également ajoutées à l'ensemble des variables de planification à sélectionner de couplage (V_{PSC}). L'existence des variables décrivant ces deux tâches est de même conditionnée par une contrainte d'activation, dont la prémisse est un sélecteur ST_{56} (de domaine $D_{ST_{56}} = \{T_5, T_6\}$ définie en section 2.2.3.4) :

- $ST_{56} = T_5 \xrightarrow{ACT} \{T_5 \bullet pst, T_5 \bullet pft, T_5 \bullet pdt, T_5 \bullet qp1\}$
- $ST_{56} = T_6 \xrightarrow{ACT} \{T_6 \bullet pst, T_6 \bullet pft, T_6 \bullet pdt, T_6 \bullet rr1\}$

La méta-tâche XOR_{56} incluant T_5 et T_6 est de même ajoutée, mais non représentée en détails. Par contre, elle n'inclut pas la contrainte d'exclusion entre ces deux tâches, car c'est le sélecteur de tâche qui s'en charge. Ce même sélecteur est associé à la durée du XOR_{56} par une contrainte pour permettre à la planification de pouvoir forcer le sélecteur sur l'une des tâches :

$$C_{Cpl}(ST_{56}, XOR_{56} \bullet pdt) = \{(T_5, [5,40]), (T_6, [60,100])\}$$

En ce qui concerne le couplage, les deux sélecteurs sont reliés par une contrainte de compatibilité permettant de garantir la cohérence de présence simultanée des éléments adéquats dans les deux modèles :

$$C_{Cpl}(SF, ST_{56}) = \{(fca, T_5), (fcu, T_6)\}$$

Comme pour le modèle en $CSP^\star$, deux contraintes (dont les tables sont données en partie droite de la figure 2.22) modulent la durée de la tâche de finition en fonction du type de finition de la ressource requise ou de la quantité de ressource requise :

- $C_{Cpl}(FCA, T_5 \bullet pdt, T_5 \bullet qp1)$
- $C_{Cpl}(FCU, T_6 \bullet pdt, T_6 \bullet rr1)$

2.3.3.4 Couplage non homogène avec sélection d'entités en exclusion

Les modèles proposés précédemment ont considéré des contraintes de couplage entre :

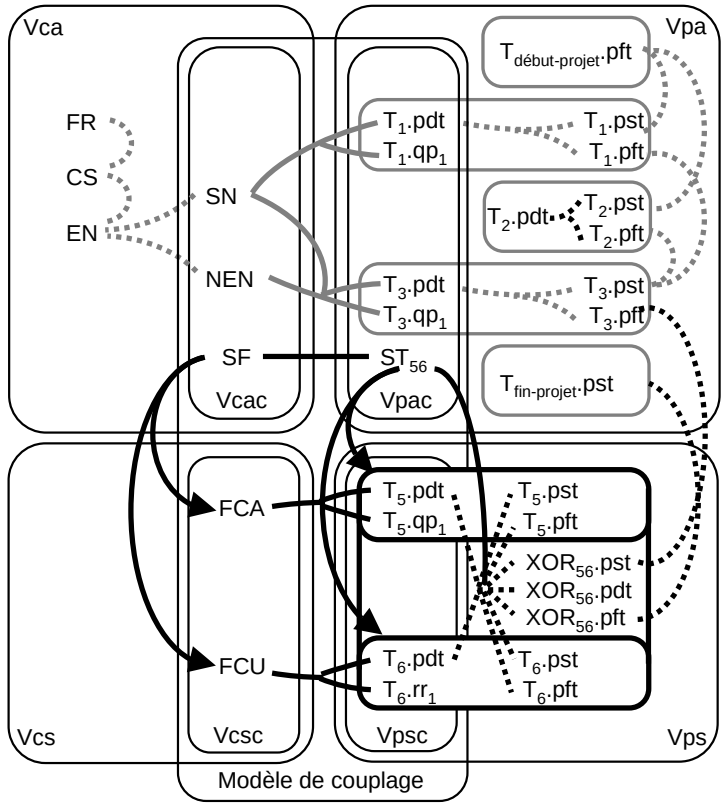
- les variables de V_{CAC} et V_{PAC},
- les variables de V_{CSC} et V_{PSC}.

Les problèmes non homogènes identifiés en introduction dans le tableau 2.4 :

- configuration élémentaire et planification avec sélection de tâches en exclusion (Pb3.1),
- configuration avec sélection d'entités en exclusion et planification élémentaire (Pb3.2),

Correspondent respectivement à des contraintes de couplage reliant :

- les variables de V_{CAC} et V_{PSC},
- les variables de V_{CSC} et V_{PAC}.



SF	ST56
fca	T5
fcu	T6

FCA	T5•pdt	T5•qp1
Standard	[15,20]	1
Standard	[8,15]	2
Standard	[5,8]	4
Confort	[22,30]	1
Confort	[11,22]	2
Confort	[8,11]	4
Luxe	[30,40]	1
Luxe	[15,30]	2
Luxe	[10,15]	4

FCU	T6•pdt	T6•rr1
Work	[50,70]	t1
Leisure	[50,70]	t1
Work	[70,90]	t2
Leisure	[70,90]	t2
Work	[90,110]	t3
Leisure	[90,110]	t3

Contraintes de couplage

FIGURE 2.22: Modèle couplé avec sélection d'entités en exclusion avec les *DCSP*

Le problème Pb3.1 signifie simplement que des variables toujours présentes dans le problème de configuration peuvent être associées à des tâches à sélectionner. Le problème dual Pb3.2 signifie que des variables à sélectionner dans le problème de configuration peuvent moduler des caractéristiques d'une tâche sans altérer son existence.

De même, les formalismes *CSP** et *DCSP* s'emploient sans modification. Les trois problèmes Pb3.0, Pb3.2 et Pb3.2 peuvent se combiner suivant la granularité ou le niveau de détail de modélisation requis.

Sur l'exemple de l'avion, un exemple de modèle suivant le problème Pb3.1 :

- ne considérerait qu'une seule variable finition (FC), dont les valeurs regrouperaient toutes les finitions possibles (union des domaines des variables FCA et FCU) ,
- associerait directement l'exclusion des deux tâches de finition à l'unique variable finition du modèle de configuration.

À l'opposé, un modèle suivant le problème pb3.2 :

- ne considérerait qu'une seule tâche de finition agrégeant toute la diversité des deux tâches (T_5) et (T_6),
- conserverait par contre les deux variables de configuration : finition catalogue (FCA) et finition customisée (FCU).

2.3.4 Implémentation des modèles sur les outils logiciels

Les modèles proposés associant configuration et planification font donc appel au formalisme des *CSP* mixtes. Le filtrage requis doit donc pouvoir propager des contraintes discrètes, numériques et mixtes. Les expérimentations de cette dernière section ont été conduites avec CoFiADe et ECLiPSe.

2.3.4.1 Expérimentation avec CoFiADe

Implémentation Ayant abordé l'implémentation des problèmes discrets (section 2.1.5) et numériques (section 2.2.5), le nouvel élément à implémenter correspond d'une part aux contraintes mixtes et d'autre part, à l'opérateur XOR. En ce qui concerne les contraintes de compatibilité mixtes, CoFiADe autorise l'écriture de table de compatibilité mélangeant variable numérique et variable discrète. De même, les prémisses des contraintes d'activation peuvent correspondre à de telles tables mixtes. Dernière particularité à souligner, CoFiADe autorise la définition de domaines de définition des variables numériques comme des ensembles d'intervalles disjoints et permet ainsi la prise en compte de « trou » dans les domaines.

Exploitation Le filtrage des différents problèmes modélisés avec des *CSP* mixtes n'appelle pas de restriction particulière. CoFiADe propage les contraintes mixtes et fait apparaître si nécessaire des domaines multi-intervalles. Par contre, CoFiADe étant basé de manière native sur un formalisme *DCSP* ajoutant effectivement des variables au problème, les pertes de cohérence liées à ce formalisme sont fréquentes avec les problèmes couplés lorsque l'utilisateur exprime des besoins de configuration après avoir fourni des besoins en planification.

2.3.4.2 Expérimentation avec ECLiPSe

Implémentation De manière similaire à CoFiADe, le nouvel élément à considérer correspond aux contraintes mixtes. Lorsque les librairies symboliques et numériques sont employées simultanément, ECLiPSe autorise sans problème l'écriture de table de compatibilité mixte. Il permet de plus l'écriture de contraintes logiques mélangeant des variables symboliques et numériques. ECLiPSe ne permet pas, par contre, de définir les domaines des variables numériques comme des unions d'intervalles disjoints.

Exploitation ECLiPSe, n'autorisant pas les domaines multi-intervalles, conserve lors du filtrage un intervalle englobant toutes les valeurs compatibles. Il conserve, en conséquence, des valeurs incohérentes. Les filtrages successifs liés à l'exploitation interactive de cet outil peuvent alors conduire à une perte de cohérence du problème.

Par contre, ECLiPSe, ne permettant pas d'implémenter l'ajout de variable du formalisme *DCSP* et nécessitant de l'émuler avec l'ajout de contraintes, est beaucoup moins sensible que CoFiADe à la perte de cohérence lorsque l'utilisateur exprime des besoins de configuration après des besoins de planification.

2.3.4.3 Conclusions

Les deux outils utilisés répondent globalement bien aux différents besoins. Le seul point en retrait concerne les domaines multi-intervalles des variables numériques, qui ne peuvent être supportés par ECLiPSe. Comme cela a déjà été évoqué, ceci n'est pas gênant en résolution, car

les incohérences sont détectées ultérieurement dans le processus de recherche de solution. Par contre, en exploitation interactive et sur des problèmes de couplage configuration/ planification où les contraintes mixtes sont présentes par définition, la conservation de valeurs incompatibles constitue un réel problème.

2.3.5 Conclusion sur le couplage configuration planification avec des *CSP*

L'objectif de ce second chapitre était d'étudier comment les approches par contraintes pouvaient être exploitées pour supporter le couplage de la configuration de produit avec la planification de sa production sur un seul niveau de modélisation.

En ce qui concerne la notion de couplage ou d'interaction entre configuration et planification, les différents modèles proposés en regard des trois types de problèmes ont clairement montré la bonne propagation des décisions de configuration vers la planification et réciproquement des décisions de planification vers la configuration.

Cependant, la manière de définir les deux problèmes de configuration et de planification comporte une différence importante. La configuration permet simplement de définir un produit comme un ensemble de composants et de propriétés valués. Par contre, la planification définit le processus de fabrication comme un ensemble de tâches nécessitant des ressources, mais fournit également les dates de toutes les tâches, s'apparentant à une forme d'évaluation d'un indicateur temporel du planning résultat.

Cette différence est directement dépendante des attentes envers les deux processus. L'attente majeure envers le processus de configuration est avant tout de personnaliser un produit, tandis que l'attente majeure envers la planification est avant tout une date de disponibilité. Il en résulte un produit configuré sans indicateur et un planning de réalisation avec un indicateur temporel. Il est à noter qu'un indicateur de coût peut être ajouté aux deux processus de configuration et de planification, cet indicateur sera à la base du problème d'optimisation de la section suivante.

Les trois types de problèmes proposés (élémentaire, avec entités optionnelles, avec entités à sélectionner en exclusion), ont permis de caractériser différentes formes de diversités produit et projet. Les différents couplages homogènes et non homogènes ont par ailleurs montré que la manière de réaliser l'abstraction du problème vers le modèle de contraintes est liée à la granularité ou au détail de modélisation souhaité. Le problème avec les sous-ensembles, mentionné mais non abordé, reste à documenter afin de permettre de considérer différents niveaux d'abstraction.

En ce qui concerne les formalismes des problèmes de contraintes, la configuration fait appel le plus souvent à des *CSP* discrets. L'utilisation des *CSP* numériques en planification permet de facilement prendre en compte l'imprécision sur les durées et les dates en raisonnant sur des intervalles. En conséquence, le couplage débouche sur des *CSP* Mixtes.

La modélisation de la diversité, induisant systématiquement le besoin de moduler l'existence d'entités aussi bien en configuration qu'en planification, nous a conduit à exploiter les deux extensions *CSP** et *DCSP* du formalisme *CSP*. Si le premier formalisme a tendance à réduire l'occurrence d'incohérence, il peut rejeter des solutions intéressantes. Le second, à l'opposé, conserve plus de solutions mais peut conduire à des incohérences.

Les expérimentations sur trois progiciels permettant le filtrage des problèmes de contraintes a montré les possibilités de rendre opérationnelles nos propositions. ILOG CP 5.5 s'étant montré performant mais nécessitant le recours à des pré-traitements, nous avons limité nos expérimentations sur CoFiADe et ECLiPSe. CoFiADe, développé pour des besoins de filtrage, a fait

pratiquement jeu égal avec les modules de filtrage de la suite ECLiPSe, davantage orienté résolution. Les exemples illustrant les situations sont de petites tailles et le problème de passage à l'échelle mériterait un travail complémentaire.

3

Optimisation du couplage configuration planification

Introduction

Dans le chapitre précédent, nous nous somme intéressés à la mise en œuvre du couplage interactif de la configuration de produit et de la planification du projet de réalisation. Ces deux problèmes étant considérés comme des problèmes de satisfaction de contraintes, le couplage interactif est réalisé en utilisant le filtrage des contraintes de couplage reliant ces deux problèmes. Cette approche permet de réduire l'espace des solutions en supprimant certaines zones d'infaisabilité, c'est-à-dire les zones qui ne contiennent pas de solutions. Il est en conséquence tout à fait possible de progressivement et interactivement configurer le produit et planifier le projet associé afin de déboucher sur une solution associant le produit configuré et le projet planifié spécifique au besoin du client ou de l'utilisateur.

Cependant, les variables à renseigner, pour décrire les besoins et définir complètement la solution, peuvent d'une part être nombreuses et d'autre part ne pas toutes intéresser le client ou l'utilisateur. Nous proposons en conséquence de définir deux sous-ensembles de variables associées aux besoins :

- les variables **non négociables** ou critiques, celles-ci seront renseignées par l'utilisateur car elles l'intéressent fondamentalement (nombre de places de l'avion ou date de livraison, par exemple) ,
- les variables **négociables** ou non critiques, celles-ci peuvent ne être pas renseignées par l'utilisateur car elles ne correspondent pas à des caractéristiques qui l'intéressent (taille du réservoir supplémentaire ou choix de ressource, par exemple).

Étant donné ce cadre, le processus de configuration/planification étudié dans ce chapitre se décompose en deux étapes :

- une première étape de configuration/planification interactive traitant uniquement les besoins non négociables. L'espace des solutions est alors progressivement réduit en respectant les besoins du client,

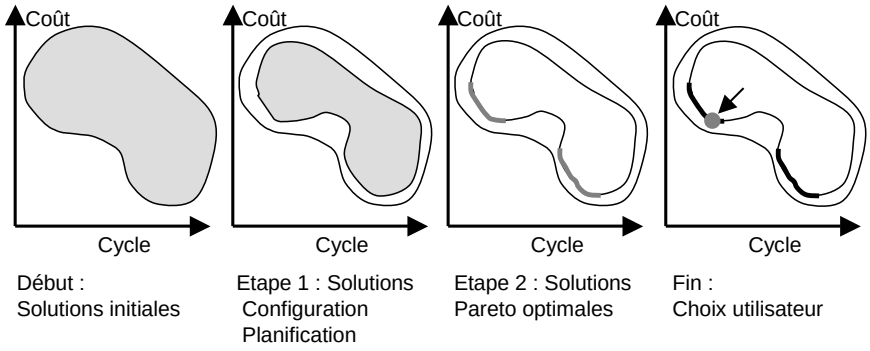


FIGURE 3.1: Situation du problème d'optimisation

– une seconde étape de recherche de solution optimale dans cet espace réduit. Pour les problèmes de configuration/planification les deux critères coût et cycle sont au minimum à prendre en compte.

L'objet de ce chapitre est donc l'étude de l'optimisation multicritères coût/cycle du problème de configuration/planification. L'idée proposée est de rechercher les solutions Pareto optimales et de les proposer à l'utilisateur pour qu'il puisse prendre la meilleure décision au regard des deux critères suscités, comme cela est représenté en figure 3.1.

Ce troisième chapitre est organisé de la manière suivante. Dans une première section, le problème d'optimisation est défini et situé par rapport à l'état de l'art pour conduire à la justification de l'emploi des approches évolutionnaires. Dans une seconde section nous proposons un algorithme évolutionnaire, basé sur l'algorithme SPEA2. Afin de maintenir la cohérence du problème d'optimisation lors de la recherche des solutions Pareto optimales avec les contraintes, celui-ci est modifié par l'introduction du filtrage au niveau des opérateurs spécifiques de l'algorithme. Une troisième section sera consacrée à des expérimentations sur le problème de l'avion de tourisme exploité présenté au fil du chapitre 2.

3.1 Problème d'optimisation

Nous nous situons dans le cadre des problèmes d'optimisation combinatoire. C'est-à-dire lorsque les éléments sur lesquels est défini le problème sont discrets, finis et dénombrables.

3.1.1 Méthodes d'optimisation multi-objectifs

Dans sa forme la plus générale, un problème d'optimisation combinatoire consiste à trouver la meilleure solution dans un ensemble de solutions potentielles ou individus, la notion de meilleure solution étant définie par une fonction objectif.

Définition 8 : ***Problème d'optimisation combinatoire***

Étant donnés :

– *un ensemble discret et fini D de solutions potentielles, également appelé individus x,*

– *une solution potentielle x est un vecteur regroupant une instanciation de chacune des k variables de décision :* $x(x_1, x_2, \ldots, x_k)$,

– *une fonction objectif* $f(x) : D \rightarrow R$.

Un problème d'optimisation combinatoire consiste à déterminer :
$Max f(x)$ *ou* $Min f(x)$ *avec* $x \in D$.

Pour un tel problème, l'ensemble des solutions potentielles ne saurait être décrit par une liste exhaustive car la difficulté réside ici précisément dans le fait que le nombre des solutions réalisables rend son énumération impossible (Ghedira, 2007). Il existe deux types de problèmes d'optimisation combinatoire : les mono-objectifs et les multi-objectifs.

Les exemples de problèmes d'optimisation combinatoire mono-objectifs sont divers : nous pouvons citer le problème du voyageur de commerce, le problème du plus court chemin, le problème de coloration minimale, le problème du sac à dos …. Le problème d'optimisation de configuration/planification, objet de ce chapitre, est bien un problème d'optimisation combinatoire. Par contre, visant à rechercher un compromis coût/cycle, il est en fait mutli-objectifs.

Un problème d'optimisation multi-objectifs est un problème avec un nombre n supérieur ou égal à deux de fonctions objectifs à maximiser ou à minimiser. La solution d'un problème multi-objectifs consiste en un ensemble de solutions contrairement à un problème mono-objectif où la solution est unique. Un problème d'optimisation multi-objectifs peut être défini ainsi :

Définition 9 : ***Problème d'optimisation combinatoire multi-objectifs***

Étant donnés :

– *un ensemble discret et fini D de solutions potentielles ou individus x,*

– *une solution potentielle x est un vecteur regroupant une instanciation de chacune des k variables de décision :* $x(x_1, x_2, \ldots, x_k)$,

– n *(*$n \geq 2$*) est le nombre de fonctions objectifs* $f_i(x) : D \rightarrow R$ *avec* $i = 1, \ldots, n$

Un problème d'optimisation combinatoire multi-objectifs consiste à déterminer :
$MaxF(x)$ *ou* $MinF(x)$ *avec* $F(x) = (f_1(x), f_2(x), \ldots, f_n(x))$ *avec* $x \in D$.

L'optimisation consiste donc à trouver plusieurs solutions optimisées par rapport à plusieurs fonctions objectifs parmi un ensemble de solutions potentielles au problème. Cet aspect multi-critères introduit des notions importantes de dominance et d'ensemble de solutions Pareto optimales.

Les fonctions objectifs sont le plus souvent antagonistes, par exemple, la performance et le poids en conception, le coût et le cycle en configuration/planification. Ceci nous amène à rechercher un ensemble de solutions satisfaisant au mieux l'ensemble de ces objectifs.

Si nous ne souhaitons pas favoriser un objectif par rapport à un autre (pas d'ordre sur les objectifs) ou agréger les objectifs pour rendre le problème mono-objectif, la notion de dominance doit être introduite. Cette notion de dominance permet de comparer les solutions du problème d'optimisation relativement entre elles dans l'espace des objectifs.

Définition 10 : ***Dominance (Deb, 2002)***

Une solution x_1 *domine une autre solution* x_2*, si les deux conditions 1 et 2 sont vérifiées :*

– *La solution* x_1 *n'est pas plus mauvaise que la solution* x_2 *pour tous les objectifs, ou* $f_j(x_1) \not\prec f_j(x_2)$ *pour tout* $j = 1, 2, \ldots, M$ *où* $\prec$*une relation d'ordre qui peut être la relation* $>$ *ou la relation* $<$,

– *la solution x_1 est strictement meilleure que x_2 sur au moins un des objectifs, où $f_{\bar{j}}(x_1) \prec f_{\bar{j}}(x2)$ pour au moins un $\bar{j} \in 1,2,\ldots,M$.*

Notons que pour toutes paires de solutions x_1 et x_2, un et un seul des cas suivants peut se présenter :

– x_1 domine $x_2 : x_1 \prec x_2$,

– x_1 est dominé par $x_2 : x_1 \succ x_2$,

– x_1 et x_2 sont équivalentes au sens de la dominance : $x_1 \precsim x_2$.

Définition 11 : ***Optimalité de Pareto***

Soit P un ensemble de solutions potentielles d'un problème d'optimisation multi-objectifs. L'ensemble P'⊆ P, composé de tous les éléments de P qui ne sont dominés par aucun élément de P est dit sous-ensemble non dominé de l'ensemble. Lorsque P correspond à l'espace de recherche l'ensemble P' est appelé l'ensemble Pareto optimal.

Définition 12 : ***Optimalité globale au sens de Pareto***

Une solution x_1 est optimale globalement au sens de Pareto, ou optimale au sens de Pareto, ou encore Pareto-optimale s'il n'existe aucun point de l'espace faisable D qui la domine. L'ensemble des solutions Pareto optimales est appelé l'ensemble de Pareto ou également l'ensemble des compromis optimaux. L'image de l'ensemble de Pareto dans l'espace des objectifs est appelée la surface de Pareto (ou le front de Pareto pour le cas bi-objectif) ou encore la surface de compromis optimale.

Définition 13 : ***Force (strength) d'une solution au sein d'une population***

La force d'une solution est égale au nombre de solutions de cette population dominées par cette solution.

Définition 14 : ***Performance préliminaire (raw fitness) d'une solution au sein d'une population***

La performance préliminaire d'une solution est égale à la somme des forces des solutions qui domine cette solution. Elle vaut zéro si la solution est sur le front de Pareto.

Il existe deux finalités dans l'optimisation multi-objectifs :

– trouver un ensemble de solutions aussi proche que possible de la surface Pareto optimale,

– trouver un ensemble des meilleures solutions aussi diverses que possible pour couvrir l'ensemble de la surface Pareto optimale.

Il apparaît clairement que pour notre problématique d'optimisation de configuration/planification, nous privilégierons le second problème afin de proposer à l'utilisateur un éventail de solutions le plus étendu possible.

3.1.2 Méthodes d'optimisation multi-objectifs

Nous pouvons classer les méthodes d'optimisation en deux classes : les méthodes complètes ou exactes et les méthodes heuristiques ou approchées (Talbi, 2002).

Les méthodes exactes ont pour but de calculer la ou toutes les solutions optimales. Elles garantissent que tout l'espace de solution a été analysé sans l'utilisation d'un processus aléatoire. Elles sont préconisées, la plupart du temps, pour la résolution de problèmes de petite taille ou de complexité limitée avec pas plus de deux objectifs du fait du temps de calcul qui peut devenir rapidement conséquent. Parmi les méthodes exactes, nous pouvons citer le *Branch and Bound*, les méthodes de programmation dynamique (Carraway et al., 1990), (Duharcourt, 1969) et l'algorithme $A^{\star}$ (Stewart, 1991), (Hart, 1968). Ces méthodes exacte ont été mise au point pour des méthodes mono-objectifs et leur adaptation éventuelle aux problèmes multi-objectifs est spécifique à chaque cas.

Les méthodes heuristiques sont des méthodes qui partent d'une ou plusieurs solutions initiales qu'elles améliorent de proche en proche afin d'atteindre de meilleures solutions.

Pour cela, elles utilisent de la connaissance partielle le plus souvent spécifique au problème traité (heuristique d'ordonnancement par exemple). Les méta-heuristiques suivent le même fonctionnement mais essaient d'employer une connaissance non spécifique au problème. Le plus souvent, les méta-heuristiques s'inspirent de phénomènes naturels ou physiques (évolution de gènes, température de recuit, vie des fourmis ...).

Il existe un grand nombre de méta-heuristiques d'optimisation. Elles se distinguent classiquement en deux groupes : les méthodes locales et les méthodes globales. Les méthodes locales recherchent autour d'une solution initiale et ne peuvent donc pas quitter facilement un optimum local. Les méthodes globales permettent par contre de quitter un optimum local et de couvrir l'espace de recherche. L'aspect fortement combinatoire du problème de configuration/planification et le comportement relativement chaotique des critères coût/cycle font que nous considérons pour nos travaux les méthodes globales. Nous nous intéressons donc aux méthodes de type : Tabou (Golver et Laguna, 1997), recuit simulé (Weisbuch, 1989), essaim particulaire (Kennedy et Eberhart, 1995), ou algorithme évolutionnaires *AE* (Goldberg, 1989).

Étant donné que l'obtention d'un front de Pareto constitue notre objectif d'une part et que, d'autre part, les travaux de Pitiot ont montré leur bonne adéquation au problème d'optimisation du couplage, nous choisissons d'exploiter les approches évolutionnaires (Pitiot, 2009).

3.1.3 Algorithmes évolutionnaires

Historiquement, le terme d'algorithme évolutionnaire regroupe un ensemble de techniques diverses : les stratégies d'évolution ((Schwefel, 1981),(Rechenberg, 1965)), la programmation évolutionnaire (Fogel et al., 1966) et les algorithmes évolutionnaires développés par Holland (Holland, 1992). Les algorithmes évolutionnaires (*AE*) sont des techniques de résolution de problèmes inspirées des processus naturels d'évolution et la survie des meilleurs individus ((Goldberg, 1989),(Michalewicz, 1996), (Bäck, 1996) et (Beyer et Schwefel, 2002)).

Il s'agit de méthodes basées sur une population d'individus où chaque individu de la population représente une solution potentielle au problème. Chaque solution ou individu est représenté sous forme de chromosome (vecteur de valeurs discrètes). Il existe différents encodages possibles pour les individus selon le problème : encodage binaire, réel, mixte ou symbolique. Chaque

individu est évaluée par une performance (ou fonction de *fitness*) et la population évolue par l'intermédiaire de trois opérateurs évolutionnaires :

- la sélection : permettant de choisir de bons individus, ce qui permet l'amélioration progressive de la population de façon globale,
- le croisement et la mutation : permettant de perturber les solutions afin de parcourir aléatoirement l'espace de recherche.

Les algorithmes évolutionnaires diffèrent des autres algorithmes d'optimisation en quatre principaux points :

- les algorithmes évolutionnaires utilisent un codage des éléments de l'espace de recherche et non les éléments eux-mêmes,
- les algorithmes évolutionnaires recherchent une solution à partir d'une population de points et non à partir d'un seul point,
- Les algorithmes évolutionnaires n'imposent aucune régularité des fonctions étudiées (continuité, dérivabilité, convexité, ...)
- Les algorithmes évolutionnaires ne sont pas déterministes, ils utilisent des règles de transition probabiliste.

De nombreux algorithmes évolutionnaires existent, les deux reconnus comme les plus performants sont SPEA2 pour *Strength Pareto Evolutionary Algorithm* Zitzler et al. (2001) et NSGA2 pour *Non Sorted Genetic Algorithm* Srinivas et Deb (1994) . Étant de performance équivalente et se basant sur les travaux de Pitiot, nous baserons nos propositions ultérieures sur SPEA2.

3.1.3.1 Principe général d'un algorithme évolutionnaire

Le fonctionnement général d'un algorithme évolutionnaire se compose de quatre étapes clés, tel qu'illustré en figure 3.2 :

1. Génération d'une population initiale de façon aléatoire.
2. Évaluation d'une performance aux individus par calcul de la fonction de *fitness*.
3. Sélection d'un certain nombre d'individus selon la valeur de performance afin de générer une population intermédiaire (*mating pool*).
4. Engendrement de la nouvelle génération à partir de la population intermédiaire (*mating pool*). L'opérateur de croisement et l'opérateur de mutation sont appliqués à la population intermédiaire selon une probabilité de croisement *pc* et une probabilité de mutation *pm*. Les opérations de sélection, de croisement et de mutation sont réitérées afin d'élaborer une nouvelle population. Ceci termine l'engendrement de la nouvelle génération.

On réitère les opérations précédentes à partir de 2 jusqu'à ce qu'un critère d'arrêt soit satisfait (nombre de générations atteint, par exemple).

3.1.3.2 Cinq constituants importants des algorithmes évolutionnaires

Principe de codage du chromosome : cet élément associe à chacun des points de l'espace de décision une structure de données qui synthétise toutes les informations liées à ce dernier. Dans notre cas, il est associé un gène à chaque variable de décision du problème (nombre de siège de l'avion, par exemple).

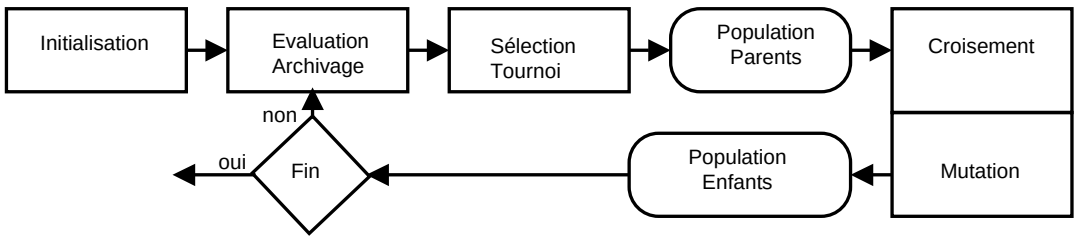


FIGURE 3.2: Fonctionnement général d'un algorithme évolutionnaire

Mécanisme de génération de la population initiale : ce mécanisme permet de générer une population initiale d'individus uniformément répartie.

Critères de performance : ces critères permettent d'évaluer les individus (*fitness*) et de les classer (notion de dominance).

Deux opérateurs permettant de diversifier la population : opérateur de croisement et opérateur de mutation. L'opérateur de croisement s'attache à brasser les gènes des individus dans la population tandis que la mutation a pour but d'introduire de la diversité (ceci permet de sortir des optimaux locaux).

Des paramètres généraux : taille de population, nombre de génération, probabilité de croisement, probabilité de mutation.

3.1.3.3 Trois opérateurs principaux évolutionnaires

Opérateur de reproduction ou sélection : le premier objectif de l'opérateur de reproduction est de faire des copies des bonnes solutions et d'éliminer les mauvaises de la population tout en gardant une taille de population fixe. L'opérateur accomplit les tâches suivantes :

- identifier les bonnes solutions dans la population g (*fitness* et dominance),
- faire des copies des bonnes solutions dans l'archive,
- sélectionner des individus de l'archive pour remplir la *mating pool*.

Opérateur de croisement : l'opérateur de croisement est appliqué à la population intermédiaire (*mating pool*). Le but du croisement est d'enrichir la diversité de la population en manipulant la structure des chromosomes (Michalewicz, 1996), (Beyer, 2001). Le croisement fait intervenir deux parents pour aboutir à deux enfants. Il consiste à échanger les gènes des parents afin de donner des enfants qui portent à la fois les propriétés des deux parents. Bien qu'il soit aléatoire, cet échange d'information permet de remplacer les mauvais gènes d'un parent avec les bons gènes de l'autre.

Opérateur de mutation : l'opérateur de croisement est essentiellement utilisé pour l'aspect recherche de meilleures solutions. L'opérateur de mutation peut aussi remplir cet objectif mais il est plutôt nécessaire pour conserver une diversité dans la population et couvrir au maximum l'espace des solutions. L'opérateur de mutation apporte aux algorithmes évolutionnaires la propriété d'ergodicité de parcours de l'espace de recherche. Cette propriété stipule que l'algorithme peut atteindre tous les points de l'espace de recherche. La convergence des algorithmes évolutionnaires dépend donc fortement de cet opérateur. L'opérateur de mutation consiste à tirer aléatoirement un gène dans le chromosome d'un individu pris dans la *mating pool* et à remplacer la valeur de ce dernier par une valeur tirée aléatoirement dans l'ensemble des valeurs potentielles du gène.

3.1.3.4 Prise en compte des contraintes dans les algorithmes évolutionnaires

Les éléments de description des algorithmes évolutionnaire précédents répondent au problème d'optimisation multi-critères de base, présenté en définition 8 et rappelé ci dessous :

Étant donnés :

- un ensemble discret et fini D de solutions potentielles ou individus x, une solution potentielle x est un vecteur regroupant une instanciation de chacune des k variables de décision : $x(x_1, x_2, \ldots, x_k)$
- n ($n \geq 2$) est le nombre de fonctions objectifs $f_i(x) : D \rightarrow R$ avec $i = 1, \ldots, n$

Un problème d'optimisation combinatoire consiste à déterminer :
$MaxF(x)$ ou $MinF(x)$ avec $F(x) = (f_1(x), f_2(x), \ldots, f_n(x))$ avec $x \in D$.

Cette définition du problème d'optimisation multi-critères n'est pas suffisante pour le traitement de notre problème de configuration/planification. En effet, la particularité de ce problème est de faire apparaître des contraintes multiples entre les valeurs des variables de décision définissant les solutions potentielles. Une opération de mutation ou de croisement peut générer des solutions incompatibles avec les contraintes du modèle. Pour nos travaux, il est donc indispensable de prendre en compte ces contraintes et d'ajouter une hypothèse à la définition précédente :

- un ensemble de m contraintes $\{C_1, C_2, \ldots C_m\}$ reliant les variables de décision et devant être vérifiées par toutes les solutions potentielles $C_m(x_1, x_2, \ldots, x_k)$.

De nombreux travaux essayent d'intégrer les contraintes et les *EA*. Coello Coello maintient un large état de l'art de ces méthodes [1]. Il semblerait que la performance des *EA* diminue lorsque le nombre de contraintes augmente. Les contraintes délimitent, en effet, des espaces de solutions faisables et infaisables dans l'espace de recherche. Plus le problème initial contient de contraintes, plus l'espace des solutions faisables est réduit et plus il sera difficile aux *EA* de générer aléatoirement des individus respectant toutes les contraintes.

Quatre types de méthode prenant en compte les contraintes existent :

Les fonctions de pénalité (Richardson et al., 2009) : l'idée principale de cette méthode est de transformer un problème d'optimisation contraint en un problème non contraint et d'inclure dans la fonction d'évaluation des solutions (*fitness*) la notion de cohérence de cette solution aux contraintes. La fonction d'évaluation contient donc un terme basé sur le nombre de contraintes violées par la solution évaluée, nommé pénalité. Idéalement, la pénalité doit être gardée aussi petite que possible, juste au dessus de la limite où les solutions infaisables deviennent optimales (ceci est appelé la règle de la pénalité minimum). La règle de la pénalité minimum est conceptuellement simple mais difficile à implémenter car la frontière entre les solutions faisables et infaisables est difficilement identifiable.

La recherche des solutions faisables : il existe essentiellement deux méthodes dans cette catégorie : la réparation des individus infaisables et l'échantillonnage de l'espace faisable :

- les méthodes de réparation (Salcedo-Sanz, 2009) essayent de ne manipuler que des individus faisables. Dès qu'un individu infaisable est détecté, il est aussitôt redirigé vers l'espace des faisables par un opérateur spécifique. Cette méthode se base sur l'idée suggérée dans (Michalewicz et Nazhiyath, 1995) qui fait co-évoluer deux populations : une contient les individus ne satisfaisant pas toutes les contraintes et l'autre est constituée d'individus faisables. Ces derniers sont évalués directement en utilisant la fonction objectif du problème

1. www.cs.cinevestav.mx/~constraint/

d'optimisation ; les premiers, quant à eux, sont réparés (rendus faisables par une procédure spéciale) avant d'être évalués.

– La méthode d'échantillonnage de l'espace faisable, proposé dans (Schoenauer et Xanthakis, 1993), s'appelle *Behavioral memory*. L'algorithme consiste à échantillonner l'espace des faisables en traitant les contraintes du problème une par une dans un ordre particulier.

Les méthodes hybrides : la caractéristique commune des méthodes hybrides est qu'elles ont tendance à séparer la fonction objectif des contraintes du problème. On retrouve dans la littérature deux approches possibles pour réaliser cette séparation. La première traite les contraintes avec des procédures d'optimisation déterministes (en utilisant le Simplex, par exemple), alors que la fonction objectif est toujours optimisée par *AE* (Myung et Fogel, 1995), (Waagen et al., 1992). Alors que dans la deuxième approche, c'est l'*AE* lui-même qui réalise ces deux optimisations en utilisant les techniques multi-objectifs (Parmee et Purchase, 1994),(Surry et Boyd, 1995).

La préservation de la faisabilité des solutions : toutes les méthodes de cette catégorie utilisent des opérateurs de reproduction spécifiques qui permettent de générer à partir des individus faisables d'autres individus qui sont faisables eux aussi. Pour plus de détails, nous renvoyons aux travaux de (Michalewicz et Janikow, 1991) (première version de Genocop), (M. et Michalewicz, 1998), (recherche sur les frontières de la région faisable) et (Koziel et Michalewicz, 1999) (*Homomorphous mapping* des représentations spécifiques ou des opérateurs spécifiques). Les méthodes proposées dans (Kowalczyk, 1997) et (Michalewicz et Nazhiyath, 1995) ont pour objectif de préserver les individus faisables durant leur construction. Kowalczyk a proposé l'utilisation de la consistance des contraintes durant l'instanciation des gènes des individus.

C'est dans cette dernière catégorie de méthode que nous allons inscrire nos propositions. Durant la génération des individus, le filtrage des contraintes sera employé pour guider l'élaboration, la mutation et le croisement des individus.

3.2 Approche d'optimisation proposée

L'algorithme SPEA2 est rappelé dans un premier temps pour être modifié par la suite afin de prendre en compte les contraintes lors des phases d'élaboration, de mutation et de croisement des individus.

3.2.1 Algorithme SPEA2

Zitzler et Thiele ont proposé un algorithme évolutionnaire mettant en œuvre la notion d'élitisme : SPEA (Zitzler et Thiele, 1989). L'algorithme SPEA2 introduit cette notion en maintenant explicitement une population externe ou archive qui sauvegarde un nombre fixe de solutions non-dominées trouvées depuis le début de la simulation. À chaque génération, les nouvelles solutions trouvées sont comparées aux solutions de l'archive et les solutions non-dominées sont préservées et placées dans l'archive. L'algorithme SPEA2 fait plus que simplement préserver les élites, il les utilise pour participer aux opérations évolutionnaires dans le but d'influencer la recherche vers les régions intéressantes de l'espace de recherche. L'algorithme SPEA2, que nous proposons d'utiliser dans nos travaux, proposé par Zitzler et al., a été développé comme une amélioration de SPEA.

SPEA2 diffère de son prédécesseur sur plusieurs aspects :

– la taille de l'archive E est fixe, si il n'y a pas assez d'individus non-dominés elle est complétée par des individus dominés.

– le calcul de la performance (*fitness*) est plus raffiné que celui de SPEA car il tient compte de la densité des solutions. La *fitness* des individus est calculé par la règle suivante : $fitness(p) = R(p) + D(p)$ avec :
  – $R(p)$ la performance préliminaire (*raw fitness*) d'un individu p.
  – $D(p)$ est la densité d'un individu p. Elle est fonction de la distance euclidienne au k_{im} plus proche voisin de la manière suivante :
    – K est une valeur fixe égale à la racine carré de la somme de la taille de la population avec la taille de l'archive,
    – La distance euclidienne de l'individu p au k_{im} plus proche voisin $\sigma_{p,k}$ est alors calculée
    – La densité vaut alors $D(p) = \frac{1}{2+\sigma_{p,k}}$
– la procédure qui contrôle la taille de l'archive dans SPEA est remplacée par une méthode de troncature. Cette méthode préserve les individus situés aux extrémités du front de Pareto.
– de plus, dans SPEA2, seuls les membres de l'archive participent au processus de reproduction.

Nous basons nos modifications sur la version préconisée dans (Coello Coello, 2002) qui suit :

1. Initialiser population P
2. Créer une population externe vide E
3. Pour $i = 1$ to *Genmax* Faire
   – Calculer la fitness de chaque individu de P et E
   – Copier tous les individus non-dominés de P et de E dans E
   – Utiliser l'opérateur de troncature pour supprimer des éléments de E lorsque la capacité de E est dépassée
   – Si la capacité de E n'est pas atteinte alors E est complété par des individus dominés de P.
   – Utiliser une sélection par tournoi binaire avec remplacement pour remplir population intermédiaire
   – Appliquer le croisement et la mutation à la population intermédiaire
4. FinPour

3.2.2 Proposition d'un algorithme SPEA2 prenant en compte les contraintes

Dans cette section, nous allons nous intéresser à décrire les différents opérateurs évolutionnaires de l'algorithme SPEA2 modifié par l'utilisation du filtrage. Comme le montre la figure 3.3, le filtrage intervient dans les étapes d'initialisation, de croisement et de mutation.

Le processus d'optimisation va en conséquence interagir avec un processus de propagation de contraintes ou filtrage. Les données du problème d'optimisation définies en section 3.1.3.4 se répartissent de la manière suivante :

– données communes aux deux processus :
  – un ensemble discret et fini D de solutions potentielles x,
  – une solution potentielle x est un vecteur regroupant les k variables de décision : $x(x_1, x_2, \ldots, x_k)$

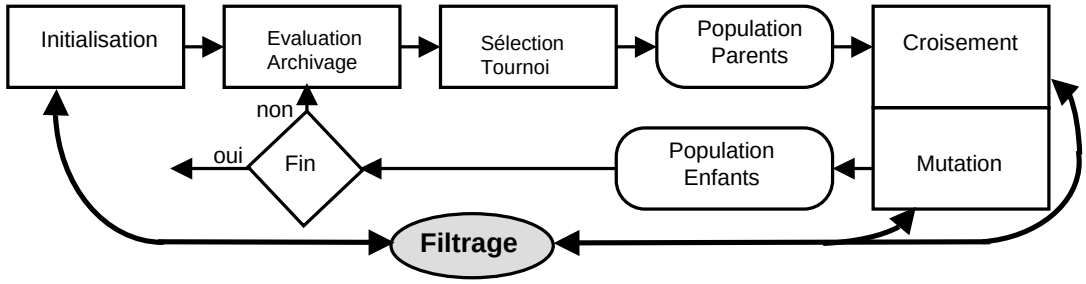


FIGURE 3.3: Schéma de l'algorithme avec filtrages complémentaires

- données spécifiques au processus d'optimisation
  - n est le nombre de fonctions objectifs ($n \geq 2$),
  - n fonctions objectif $f_i(x) : D \rightarrow R$ avec $i = 1, \ldots, n$.
- données spécifiques au processus de propagation de contraintes : un ensemble de m contraintes $\{C_1, C_2, \ldots C_m\}$ reliant les variables de décisions $C_m(x_1, x_2, \ldots, x_k)$. Les contraintes doivent être vérifiées par toutes les solutions potentielles

Nous allons détailler chaque étape dans ce qui suit.

3.2.2.1 Initialisation

Cet opérateur permet d'initialiser la population de départ de l'algorithme évolutionnaire. Il s'agit de choisir un ensemble diversifié d'individus représentant l'ensemble de l'espace de recherche. Pour cela, les gènes de chaque individu sont instanciés au hasard. Afin d'éviter d'avoir des individus infaisables ne vérifiant pas les contraintes du problème, à chaque instanciation de gène, l'individu est filtré et les gènes non instanciés ont leur domaine de définition réduit du fait de la propagation de contraintes.

Pour cela, le moteur de filtrage est invoqué afin de mettre à jour les valeurs de chaque variable non instanciée. Si il s'avère que l'individu est incohérent, les choix déjà réalisés sont remis en cause (par un mécanisme de *backtracking* permettant la désinstanciation d'une variable sélectionnée aléatoirement). Dans le cas contraire, le filtrage retourne la liste des domaines mis à jour de toutes les variables de l'individu. L'ensemble des domaines mis à jour est enregistré au niveau de l'individu et on réitère jusqu'à ce que toutes les variables soient instanciées.

1. Pour i de 1 à u
2. Liste chaînée inst-cour $\rightarrow$ var ((xi, Dxi), ...)
3. Individu ind
4. Tant que inst-cour non vide
   - gene = get-var(inst-cour, rand (1, nb-elem(Inst-cour))
   - Assign-gen (ind, getnum(gene), rand (1, nb-elem(Domaine(gene)))
   - Filtrage(ind, inst-cour)
   - Si Var-domaine-vide(Inst-cour) alors backtrack(ind, inst-cour, var)
5. Fin Tant que
6. Ajouter(Population, ind)
7. Fin Pour

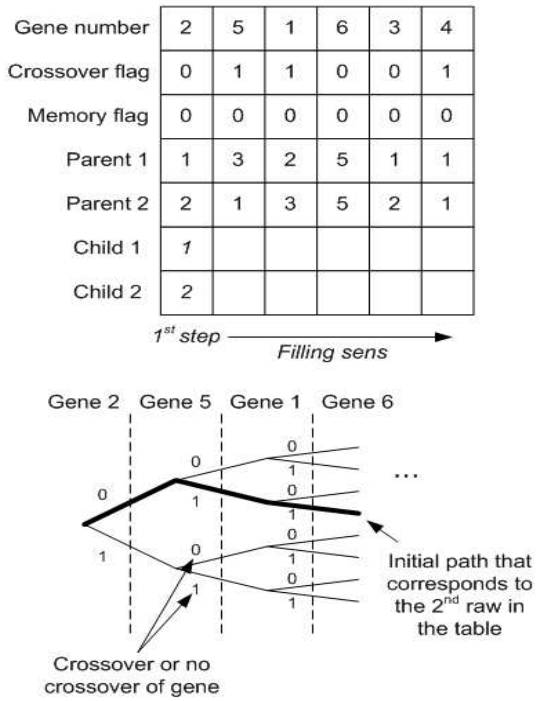


FIGURE 3.4: Table de croisement et arbre binaire correspondant

3.2.2.2 Croisement : opérateur de croisement uniforme

Il s'agit de créer deux nouveaux individus à partir d'un couple d'individus existants et cohérents (parents) tout en utilisant le filtrage durant le croisement afin de vérifier la cohérence des individus fils crées. Cet opérateur permet de mélanger aléatoirement et uniformément les valeurs des gènes issus de deux individus sélectionnés selon une probabilité de croisement. Afin d'accomplir l'opération de croisement, une table de croisement est utilisée. Elle permet de sélectionner aléatoirement les gènes qui verront leurs valeurs échangées entre les deux parents. En effet, le croisement correspond à la sélection d'un chemin dans un arbre binaire où chaque branche est liée au croisement d'un gène particulier (figure 3.4). L'instanciation de la table de croisement est équivalente à la sélection d'un chemin dans un arbre de croisement. Les gènes sont positionnés aléatoirement dans la table afin d'éviter la dominance des gènes par leur position dans les chromosomes.

Durant l'opération de croisement, à chaque instanciation d'un gène, le filtrage est appelé et met à jour les domaines des autres gènes. Comme pour l'initialisation, si un individu ne respecte pas les contraintes, un *backtraking* est réalisé. De plus, un compteur de *backtrak* est ajouté, lorsque la limite de *backtrack* est atteinte, l'enfant correspondant est abandonné. Enfin, chaque enfant viable est ajouté à la population de la génération suivante.

1. inst-cour $\rightarrow var((xi, Dxi), \dots)$
2. Tant que inst-cour non vide faire
   - gene = get-var(inst-cour, rand (1, nb-elem(Inst-cour))
   - Si (rand (0,1) < $Pmut$) alors
   - Mutation-gene(gene, var)

- Filtrage(inst-cour)
- Si Var-domaine-vide(Inst-cour) alors backtrack(ind, inst-cour, var)
- Sinon Enlever(gene, inst-cour)

3. Fin Tant que

3.2.2.3 Mutation : opérateur de mutation uniforme

Cet opérateur permet d'introduire une variation aléatoire dans la population courante en créant de nouveaux individus à partir des individus existants et en utilisant le filtrage après la mutation uniforme pour maintenir la cohérence des individus créés. Il s'agit de modifier aléatoirement les gènes d'individus sélectionnés aléatoirement dans la *mating pool* selon une probabilité de mutation.

Une première étape permet de sélectionner, selon le taux de mutation, les individus pour lesquels la mutation va avoir lieu parmi la *mating pool*. Puis, pour chaque individu, certains gènes sont sélectionnés, toujours selon la probabilité de mutation et désinstanciés (leur valeur est supprimée). Le filtrage est alors appelé pour mettre à jour les domaines des autres gènes.

Une fois l'initialisation des mutations réalisée pour tous les individus sélectionnés, l'opération de mutation est réalisée. Pour chaque individu et pour chaque gène devant muter, une valeur est aléatoirement sélectionnée dans le domaine filtré et assigné au gène traité. Le filtrage est appelé pour mettre à jour les domaines des autres gènes. Comme précédemment un mécanisme de *backtrack* avec compteur est déclenché si les contraintes ne sont pas vérifiées. Lorsque tous les gènes sont instanciés, la mutation est terminée.

3.2.3 Conclusion sur l'algorithme proposée

L'algorithme proposé permet donc de générer, de croiser et de faire muter des individus tout en respectant des contraintes exprimées sur les variables de décision. À notre connaissance, il n'existe pas de tels algorithmes évolutionnaires dans la littérature du domaine. Les travaux les plus proches de notre proposition sont ceux de (Kowalczyk, 1997), qui semble s'être heurté à l'époque à des problèmes de temps de calcul coûteux, auxquels aucune suite n'a été donnée.

3.3 Mise en œuvre et expérimentations de l'algorithme sur l'exemple

Le principe de l'optimisation étant défini, cette section s'intéresse à sa mise en œuvre et aux expérimentations menées sur le problème de l'avion de tourisme. Dans une première sous-section, le problème de configuration/planification est abordé dans sa globalité. Le modèle de l'avion est ensuite complété et présenté dans son ensemble. Une troisième sous-section fournit les résultats expérimentaux.

3.3.1 Optimisation évolutionnaire du problème de configuration planification

Les différents modèles de couplage du chapitre 2 ont montré que deux extensions des *CSP* pouvaient être utilisées : les *CSP** et les *DCSP*. L'algorithme évolutionnaire qui a été proposé ne travaille que sur des problèmes dont la structure ou nombre de variable n'est pas modifiée durant

le traitement. Nous ne considérerons en conséquence, dans cette section expérimentale, que des modèles de type $CSP^{\star}$. Nous tenons cependant à préciser qu'un travail concernant la mise au point d'algorithme évolutionnaire pouvant travailler avec des modèles à structure évolutive de type *DCSP* mériterait d'être conduit.

Le modèle de contrainte de type $CSP^{\star}$ étant retenu, il est nécessaire de le compléter avec le critère coût. Pour chacun des modèles de configuration et de planification, un ensemble de variables de coût, noté $Vc€$ pour le produit configuré et $Vp€$ pour le processus configuré est ajouté. Ces variables de coût sont reliées respectivement aux variables de configuration Vc et de planification Vp par des contraintes.

En ce qui concerne les contraintes reliant Vc avec $Vc€$, les contraintes ont tendance à davantage concerner des variables associées à des composants physiques qui ont un coût plutôt que des propriétés qui décrivent le produit. Pour les contraintes entre Vp et $Vp€$, les variables concernées sont les durées de tâche et les variables de ressources (ressource requise et quantité de ressource). Le critère associé à la variable coût global peut être alors défini comme la somme de toutes les variables coût :

Coût-global = $\sum Vc$ € + $\sum Vp€$.

Le critère de cycle de production correspond à la date de fin du projet ou date de fin de la dernière tâche du projet. L'ensemble de ces éléments sont représentés sur la figure 3.5 décrivant l'architecture du modèle du problème à optimiser.

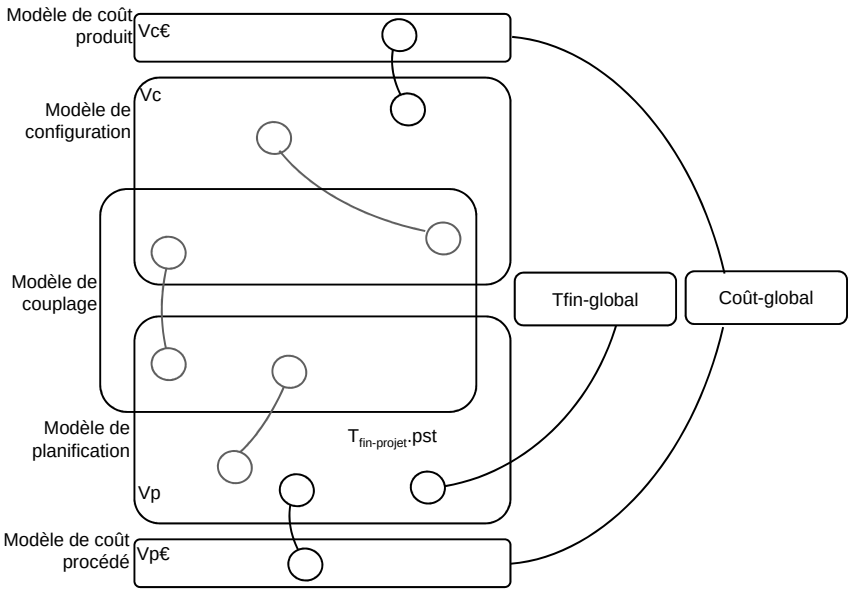


FIGURE 3.5: Modèle du problème de configuration/planification à optimiser

3.3.2 Présentation de l'exemple de l'avion de tourisme

L'exemple considéré pour les expérimentations est similaire à celui qui a été utilisé tout au long du chapitre 2. Les seules modifications correspondent :

- au fait qu'une seule variable finition et une seule tâche de finition agrègent les deux finitions catalogue et customisée,
- à l'ajout d'un réglage moteur, noté par la variable ES,
- à l'ajout de cinq variables coût associées aux variables de configuration : nombre de siège (SN) associé à SNC, motorisation (EN) associé à ENC, réglage moteur (ES) associé à ESC, réservoir supplémentaire (ST) associé à STC et finition (FI) associé à FIC,
- à l'ajout de cinq variables coût associées aux tâches de planification : approvisionnement (T_1) associé à T_1C, fabrication (T_2) associé à T_2C, assemblage (T_3) associé à T_3C, montage réservoir (T_4) associé à T_4C et finition (T_5) associé à T_5C.

Les critères globaux valent en conséquence :

- Coût-global = $SNC + ENC + ESC + STC + FIC + T_1C + T_2C + T_3C + T_4C + T_5C$
- Tfin-global = $T_5 \bullet pft$

Une particularité du problème de configuration/planification est que l'évaluation de ces deux critères, pour un individu, s'effectue en même temps que son élaboration. En effet, à chaque filtrage effectué pour maintenir la cohérence de l'individu, les deux variables Coût-global et TFin-global sont également filtrées et réduites. Le modèle correspondant à cet exemple est représenté en figure 3.6.

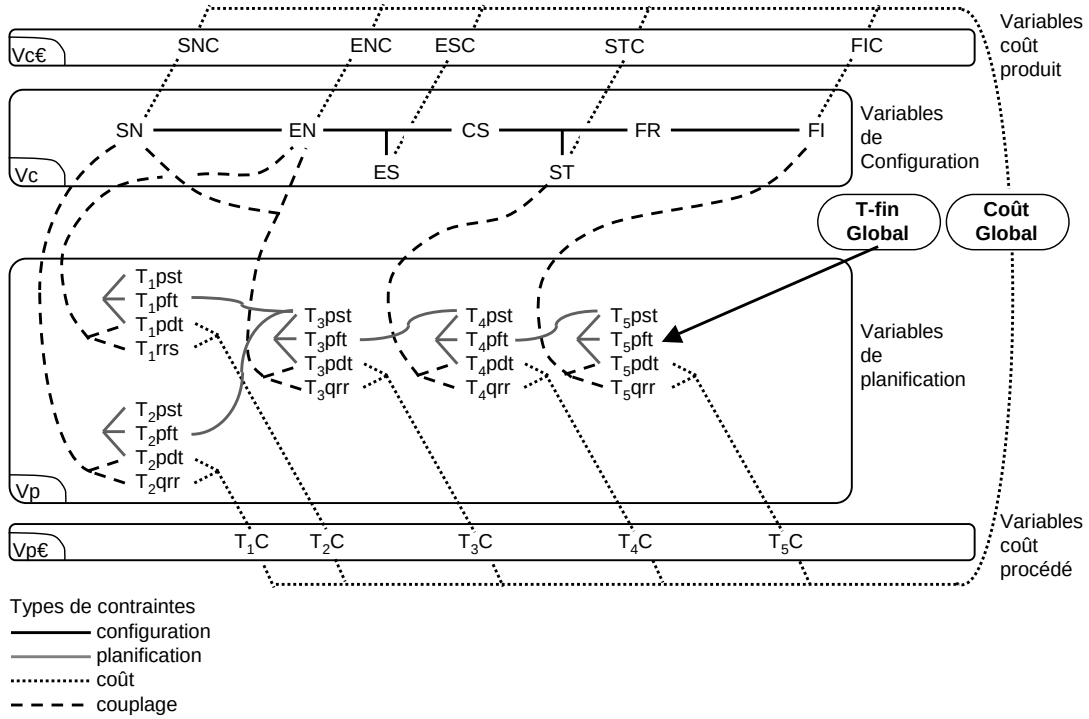


FIGURE 3.6: Modèle de l'avion de tourisme à optimiser

En terme de combinatoire, les tailles des domaines des variables du modèle de configuration sont :

– nombre de sièges (SN) avec six valeurs possibles

– type de motorisation (EN) avec six valeurs possibles

– vitesse de croisière (CS) avec six valeurs possibles

– distance de vol (FR) avec six valeurs possibles

– finition (FI) avec quatre valeurs possibles

– réglage moteur (ES) avec six valeurs possibles

– référence réservoir supplémentaire (ST) avec six valeurs possibles

En ce qui concerne la planification, les tailles des domaines des variables sont :

– T_1 approvisionnement : $T_1 \bullet rrs$, cinq fournisseurs possibles, pas de quantité de ressource,

– T_2 : fabrication : $T_2 \bullet qrr$, cinq quantités possibles, pas de choix de ressource,

– T_3 : assemblage : $T_3 \bullet qrr$, quatre quantités possibles, pas de choix de ressource,

– T_4, montage réservoir : $T_4 \bullet qrr$, quatre quantités possibles, pas de choix de ressource,

– T_5, finition : $T_5 \bullet qrr$, quatre quantités possibles, pas de choix de ressource,

Dans ce modèle, les contraintes font que les valeurs de la référence réservoir supplémentaire (ST) et du réglage moteur (ES) sont respectivement déduites des variables : vitesse et rayon d'action pour ST et vitesse et type de motorisation pour ES. En conséquence, ces deux variables n'entrent pas dans les gènes du chromosome. Le chromosome de chaque individu comporte donc les dix gènes ou variables suivants :

SN	**EN**	**CS**	**FR**	**FI**	$T_1 \bullet rrs$	$T_2 \bullet qrr$	$T_3 \bullet qrr$	$T_4 \bullet qrr$	$T_5 \bullet qrr$
3	1LP	350	600	Standard	t1	1	1	1	1
4	2LP	400	800	Confort	t2	2	2	2	2
6	1HP	450	1000	Luxe	t3	3	3	3	3
8	2HP	500	1200	Custom	t4	4	4	4	4
10	1ECO	550	1400		t5	5			
12	2ECO	600	1600						

Ce qui fait, sans prendre en compte les contraintes, un espace de 8 294 400 solutions différentes. Deux niveaux de contraintes, faible et fort, seront considérés. Le niveau faible supprime 20% de la combinatoire de chaque contrainte tandis que le niveau fort supprime 50% des combinaisons possibles. Pour ces deux niveaux, l'ordre de grandeur du nombre de solutions est pour :

– le niveau faible : 800 000 solutions,

– le niveau fort : 200 000 solutions.

Après une première campagne d'essais préliminaires, les données de fonctionnement de l'algorithme ont été ajustées à :

– taille population = 150,

– taille archive = 100,

– probabilité de mutation = 0.4,

– probabilité de croisement = 0.8.

3.3.3 Résultats expérimentaux

Cette section présente les résultats expérimentaux obtenus sur l'avion de tourisme. Deux types d'expérimentations ont été réalisées : la première prend en compte l'exemple sans aucune réduction faite par l'utilisateur (domaines des variables initiaux), la seconde prend en compte une réduction utilisateur (domaines des variables initialement réduits). Les tests sont lancés sur deux niveaux de problème : faiblement et fortement contraint. Les résultats obtenus pour chaque expérimentation, sont présentés sous forme de nuage de points à différentes étapes de résolution de l'algorithme.

3.3.3.1 Expérimentations sans réduction préalable

L'exemple est considéré sans réduction préalable, c'est-à-dire sans aucune réduction faite par l'utilisateur (domaines des variables initiaux). Pour les deux niveaux de contraintes, l'algorithme évolutionnaire est lancé dix fois pendant 1 heure.

Les résultats, présentés en figure 3.7, montrent, en partie supérieure pour le problème peu contraint et inférieure pour le problème fortement contraint :

- en partie gauche :
  - une représentation de l'espace des solutions avec des individus pris au hasard sous forme d'un nuage de points,
  - pour un lancement pris au hasard, les fronts de Pareto obtenus après l'initialisation et après 60 minutes,
- en partie droite, un zoom sur la zone des fronts de Pareto avec :
  - la même représentation de l'espace des solutions,
  - pour le même lancement, quatre fronts de Pareto après l'initialisation, 15, 30 et 60 minutes.

La partie gauche de la figure 3.7 permet de situer les fronts de Pareto par rapport à l'espace de solutions. La partie droite montre :

- que le front à 15 minutes obtenu après deux générations donne une bonne approximation du front de Pareto,
- que les fronts à 30 et 60 minutes sont pratiquement identiques.

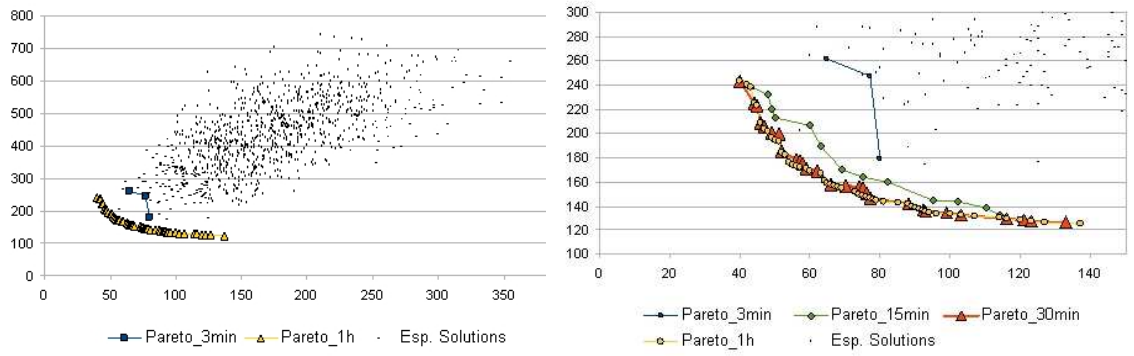
Ces conclusions sont identiques pour les deux problèmes faiblement et fortement contraints.

Associé à cela, afin d'évaluer plus précisément les performances de l'algorithme (vitesse de convergence et diversité des solutions), nous utilisons la métrique d'hypervolume définie dans (Zitzler et Thiele, 1989) illustrée dans la figure 3.8. Cette métrique permet d'évaluer la propriété de convergence et de diversité. Elle est mesurée par l'équation

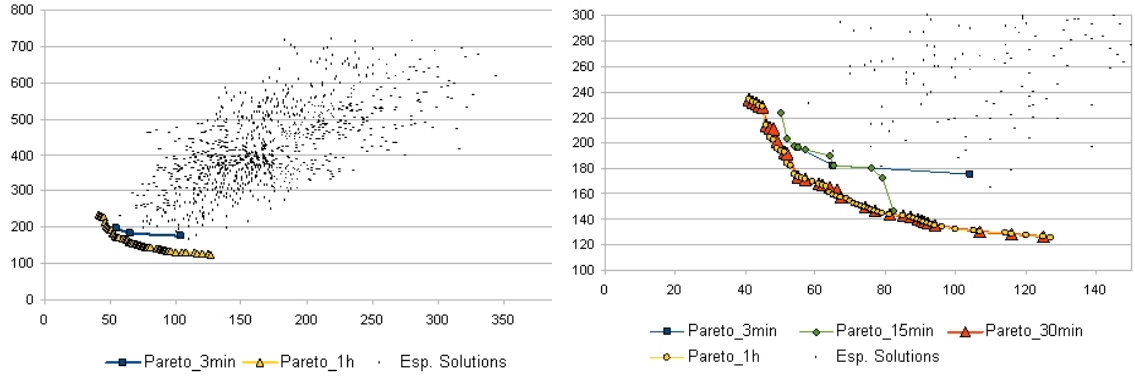
$$HV = \sum_{i=1}^{\overline{NParch}} C_i$$

avec :

- $NParch$ le nombre de solutions Pareto dans l'archive,
- C_i est l'hyper cube qui lie une solution i au point de référence W (qui correspond à la pire valeur pour les deux objectifs).



Problème faiblement contraint



Problème fortement contraint

FIGURE 3.7: Résultats sans réduction préalable

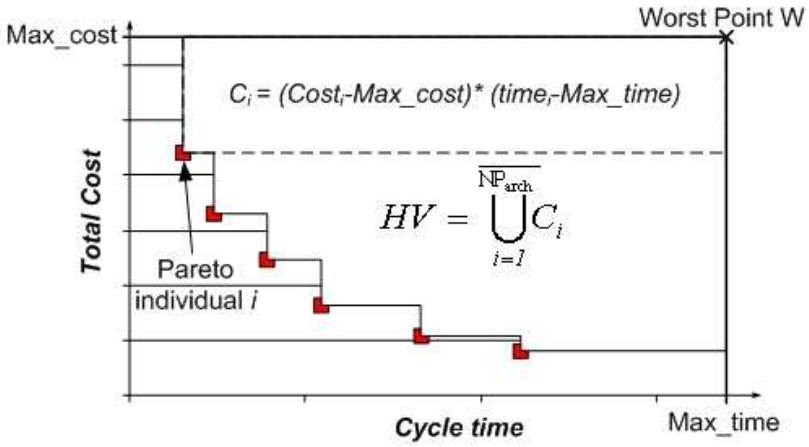


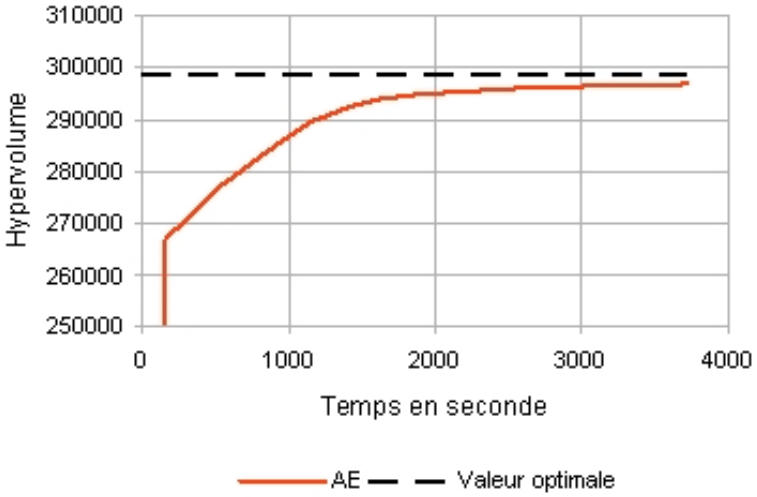
FIGURE 3.8: Métrique de l'hyper volume

Elle correspond à l'union du volume des zones dominées par chaque solution Pareto de l'archive.

Les courbes représentées en figure 3.9 et 3.10 montrent l'évolution de la métrique pour les deux niveaux de contraintes. Cette évolution correspond à la moyenne des 10 lancements. Le tableau de valeurs associés à chaque courbe indique :

- le nombre de générations :NG,
- le temps de calcul TC,
- la valeur de la moyenne de l'hypervolume des 10 lancements HV,
- l'écart type sur les 10 lancements en pourcentage $ET\%$,
- l'écart en % de la moyenne de l'hyper volume par rapport à la solution optimale obtenue par le calcul de toutes les solutions du problème $EM\%$.

Que le problème soit faiblement ou fortement contraint, ces résultats montrent clairement que l'approche proposée permet de déterminer en un quart d'heure une bonne approximation du front de Pareto. L'évolution de l'hyper volume montre qu'en 30 minutes, nous obtenons plus de 98% du front optimal et en 60 minutes plus de 99%.



NG	TC	HV	ET (%)	EM (%)
Gen 0	≈ 3 min (160s)	266996	2.63	10.55
Gen 2	≈ 15 min (867s)	284234	1.44	4.78
Gen 5	≈ 30 min (1891s)	294637	0.67	1.29
Gen 13	≈ 1 h (3718s)	298498	0.53	0.61

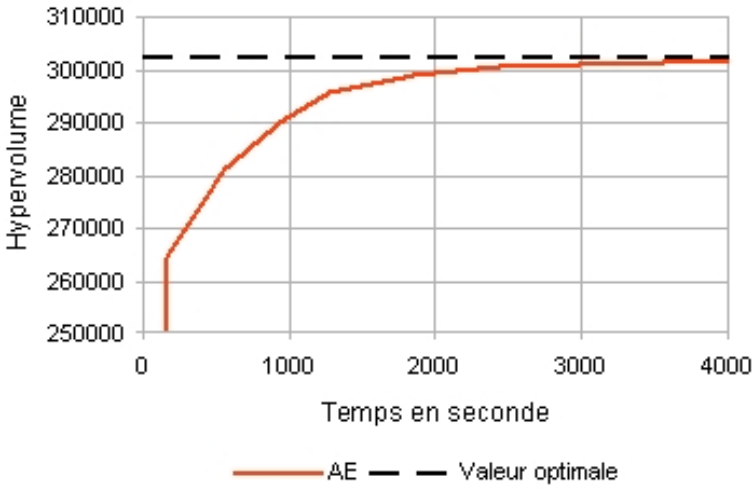
hv optimal : 298498

FIGURE 3.9: Convergence avec l'hyper volume problème faiblement contraint

3.3.3.2 Expérimentations avec réduction préalable

Le même exemple est considéré maintenant avec une réduction préalable des domaines (domaines des variables réduits), correspondant aux besoins de l'utilisateur de l'étape 1. Cette réduction correspond un avion de moyenne gamme caractérisé par :

- nombre de sièges réduit à 6 ou 8 sièges : $SN = \{6, 8\}$ soit deux valeurs conservées sur six,



NG	TC	HV	ET (%)	EM (%)
Gen 0	≈ 3 min (164s)	264219	3.18	12.58
Gen 2	≈ 15 min (934s)	290076	1.38	4.03
Gen 5	≈ 30 min (1874s)	299006	0.50	1.07
Gen 11	≈ 1 h (3587s)	301383	0.26	0.29

hv optimal : 302252

FIGURE 3.10: Convergence avec l'hyper volume problème fortement contraint

– finition réduit à luxe ou confort : $FI = \{luxe, confort\}$ soit deux valeurs conservées sur quatre

Pour les deux niveaux de contraintes (faible et fort) l'algorithme évolutionnaire est lancé pendant 30 mn. Les résultats représentés en figure 3.11 font apparaître, en partie haute pour le problème peut contraint et en partie basse pour le problème fortement contraint :

– pour comparaison une duplication de la figure 3.7 montrant :
  – l'espace des solutions avec des individus pris au hasard sous forme de nuage de points,
  – le front de Pareto obtenus après 30 minutes sans réduction
– pour un lancement pris au hasard, le front de Pareto obtenus après 30 minutes

Pour les deux problèmes faiblement et fortement contraints, la réduction de domaine ne modifie en aucune manière le bon comportement de l'approche proposée. Le front de Pareto obtenu permet à l'utilisateur de visualiser, rapidement, les compromis cycle/coût envisageables et de sélectionner le compromis qui lui convient.

3.3.3.3 Proposition d'exploitation pour des problèmes plus grands

Il est fort probable que sur des problèmes de plus grande taille, comme par exemple, Amilhastre et al. recense $1.4 * 10^{12}$ solutions, l'approche proposée en deux étapes peut s'avérer inopérante. La rapidité d'obtention d'une première approximation du front de Pareto nous permet cependant d'envisager une possibilité de recherche itérative du compromis. La démarche serait alors la suivante :

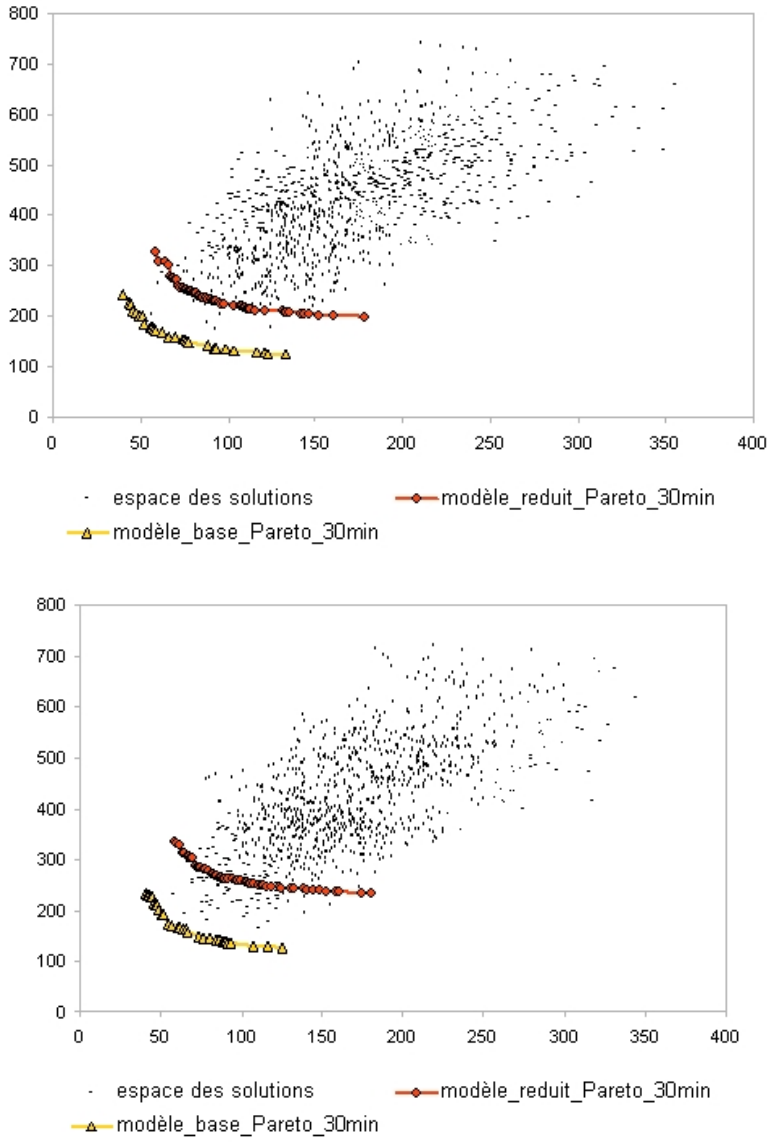


FIGURE 3.11: Résultats avec prise en compte du besoin

- Etape 1 : inchangée, réduction de domaines en filtrant les besoins utilisateurs,
- Étape 2_{bis} : recherche d'une première approximation du front de Pareto avec un temps de calcul borné par exemple à 30 mn ou à quelques générations,
- Étape 3 : sélection d'une zone de l'espace des critères cycle/coût en contraignant cycle et coût et en filtrant ces contraintes,
- Étape 4 : recherche d'une seconde approximation affinant localement la première approximation.

Cette démarche est illustrée dans le bas de la figure 3.12 et comparée à la démarche initiale, en haut de la figure 3.1. Il apparaît clairement que le dégrossissement du front de Pareto de l'étape 2 associé à la sélection de l'espace de solution à affiner de l'étape 3 réduit énormément le travail de recherche de l'étape 4 pour aboutir au même résultat final. Cette proposition constitue une piste de recherche future pour les problèmes de grande taille.

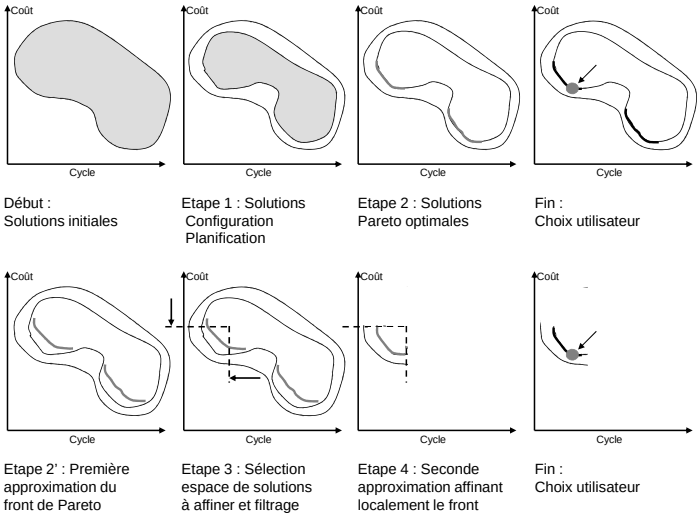


FIGURE 3.12: Démarche proposé pour les problèmes de grande taille

3.3.3.4 Conclusions

L'objectif de chapitre était de proposer une optimisation simultanée du produit configuré et son processus de réalisation en recherchant un compromis cycle/coût. Une approche évolutionnaire conventionnelle et un algorithme éprouvé SPEA2 ont été retenus.

Devant prendre en compte des contraintes agissant sur l'espace de solutions, nous avons adapté l'algorithme SPEA2 en intégrant un processus de filtrage lors de la création initiale des individus, lors de leur croisement et lors de leur mutation. Nous avons restreint nos travaux à une structure figée de problème, c'est-à-dire qu'il n'y a pas d'ajout de variables lors de la recherche de solution. Nous avons retenu en conséquence le modèle de configuration/planification exploitant les *CSP** qui conserve la structure du problème. Nous pensons qu'il doit être tout à fait possible de poursuivre ces travaux et d'adapter nos propositions pour prendre en compte les problèmes avec ajout de variables et les modèles formalisés avec les *DCSP*.

L'exemple de l'avion de tourisme a été complété avec un second critère coût dans le but d'évaluer expérimentalement nos propositions. Les expérimentations ont montré qu'en un quart d'heure, il était tout à fait envisageable d'obtenir un front de Pareto s'approchant à quelques pourcents du front optimal. L'algorithme adapté et la démarche proposés sont donc tout à fait adéquats et constituent une base solide pour optimiser les problèmes de configuration/planification. Il est néanmoins incontournable de noter que ce résultat est lié au problème étudié. Des problèmes de taille plus conséquente doivent être étudiés. Une démarche itérative a été proposée mais non testée pour aborder ceux-ci.

Conclusion et perspectives

Nous avons abordé, dans ce manuscrit, un problème assez peu étudié qui est l'association ou le couplage de la configuration de produit et de la planification du projet de réalisation de celui-ci. Nous tenons à souligner que les travaux ayant traités cette problématique l'ont fait de manière théorique alors qu'un besoin d'outils d'aide à la décision est de plus en plus exprimé par les industriels (Abeille et al., 2009).

Nos travaux permettent d'apporter une aide à la décision pour le couplage de la configuration de produit et de la planification du projet associé, en exploitant deux outils issus de l'Intelligence Artificielle :

- les approches par contraintes pour permettre la prévention des incohérences des solutions et ainsi permettre au décideur de configurer son produit et son projet de réalisation de manière simultanée et interactive. Pour ce faire, les techniques de propagation et de filtrage des contraintes sont exploitées spécifiquement,
- les algorithmes évolutionnaires pour optimiser l'espace de solutions selon les critères coût et délai afin de présenter au décideur, un ensemble réduit de solutions optimisées. Afin de présenter des solutions optimisées et cohérentes avec les contraintes du problème, l'algorithme *SPEA2* intègre des méthodes de filtrage à ses opérateurs de parcours de l'espace de recherche.

Conclusions

Le premier objectif de la thèse était d'étudier comment les approches par contraintes pouvaient être exploitées pour supporter le couplage de la configuration de produit et de la planification de sa production sur un seul niveau de modélisation, tel qu'indiqué dans le chapitre 2. Nous avons démontré, dans ce chapitre, comment nous pouvions propager de manière efficace des décisions de configuration vers la planification et réciproquement des décisions de planification vers la configuration en exploitant les approches par contraintes.

Trois natures de problèmes ont été étudiés :

- un problème élémentaire de configuration et de planification,
- un problème avec entités optionnelles où des éléments peuvent être ajoutés au problème en cours de résolution,
- un problème avec entités optionnelles en exclusion à sélectionner en cours de résolution.

Le couplage de la configuration de produit et de planification de projet de réalisation porte sur les six problèmes identifiés. Nous avons caractérisé de couplage homogène, les trois couplages mettant en relation les problèmes de même nature : problème de configuration élémentaire et

problème de planification élémentaire, problème de configuration avec éléments optionnels et problème de planification avec tâches optionnelles, et problème de configuration avec éléments en exclusion et problème de planification avec tâches en exclusion. Le couplage non homogène met, quant à lui, en relation des problèmes de nature différente, par exemple, problème de configuration élémentaire et problème de planification avec tâches optionnelles.

Les différents couplages homogènes et non homogènes ont montré que la manière de modéliser le problème sous forme de contraintes est liée au niveau de granularité et au niveau de détail de modélisation souhaité. Le problème avec les sous-ensembles, mentionné mais non abordé, reste à documenter afin de permettre de considérer différents niveaux d'abstraction.

Les différents problèmes ont été formalisés en utilisant différentes approches par contraintes. La configuration fait appel le plus souvent à des *CSP* discrets alors que la planification est modélisée par des *CSP* temporels qualitatifs (Bartak et al., 2010). Le couplage est donc un modèle basé sur des *CSP* mixtes car il met en relation des variables discrètes de configuration et numériques de planification.

Nous avons utilisé deux extensions du formalisme *CSP* afin de modéliser le besoin de moduler l'existence d'entités aussi bien en configuration qu'en planification : les *CSP** et les *DCSP*. La principale différence entre ces deux extensions provient de l'ensemble des variables manipulées lors de la recherche de solution :

- pour les *CSP**, la totalité de l'ensemble des variables du problème est filtré et présent dans toutes les solutions du *CSP*. C'est l'ajout d'une valeur particulière $\star$ ou 0 qui permet d'indiquer quelles sont les variables qui n'appartiennent pas à la solution de configuration. Comme souligné précédemment dans le chapitre 2, le formalisme *CSP** réduit l'occurrence d'incohérence mais, il peut rejeter des solutions intéressantes par la considération de contraintes induites par des variables n'appartenant pas à la solution de configuration,
- pour les *DCSP*, seul un sous-ensemble des variables du problème est filtré et présent dans toutes les solutions du *CSP*. Certaine variable du problème possède un état qui permet d'indiquer si la variable appartient ou non à la solution du *CSP* et, par conséquent, à la solution de configuration. Comme indiqué précédemment dans le chapitre 2, le formalisme *DCSP* conserve plus de solutions mais peut conduire à des incohérences lors de l'ajout de variables au problème.

Enfin, les différentes expérimentations logicielles menées sur CoFiADe, ILOG CP 5.5 et ECLiPSe ont montré les possibilités de rendre opérationnelles nos propositions. La comparaison des trois outils de propagation de contraintes nous conduit au fait que CoFiADe, développé pour des besoins de filtrage, répond tout à fait à nos besoins en couplage.

Le second objectif de nos travaux est de réaliser une optimisation simultanée du produit et de son procédé de réalisation en recherchant un compromis cycle/coût. Pour répondre à cet objectif, nous avons utilisé un algorithme *SPEA2* modifié, de manière à utiliser des techniques de propagation de contraintes lors des différentes étapes de l'algorithme, afin de garantir la cohérence des individus aux contraintes du problème. Les contraintes permettent donc de borner l'espace de solutions à optimiser lors de l'initialisation de la population, lors du croisement et de la mutation des individus.

L'algorithme modifié a été testé sur l'exemple de l'avion de tourisme formalisé en *CSP**, dans lequel des variables de coût de composants et de coût de tâches ont été ajoutées. Les expérimentations ont montré qu'il était tout à fait envisageable, en un quart d'heure, d'obtenir un front de Pareto s'approchant à quelques pourcent du front optimal. L'algorithme paraît donc adapté pour l'optimisation des problèmes de configuration/planification.

Les deux approches retenues se sont montrées efficace séparément pour aider à la décision en couplage de la configuration et de la planification. Une utilisation conjointe de ces deux approches a été réalisée selon la démarche suivante :

- une phase de configuration interactive, via la valuation de variables non négociables, permettant de réduire l'espace de solutions à l'aide des approches par contraintes,
- une phase d'optimisation permettant de proposer un ensemble de solutions optimisées selon les critères cycle/coût à l'aide de l'algorithme *SPEA2* sous contraintes.

Perspectives

Nous allons dans cette section, aborder les différents points qui pourront être améliorés ou abordés comme suite de nos travaux :

1. Le problème de couplage que nous avons traité s'est focalisé sur la configuration de produit et la configuration/planification de son projet de réalisation. Nous nous sommes donc limités à des processus très routiniers où les espaces de solutions sont totalement caractérisés. Il serait intéressant d'étendre nos propositions à des processus de conception/planification moins routinier. Des éléments de réponse seront apportés dans la thèse de Joël Abeille [2].
2. Le problème de couplage de la configuration et de la planification a été étudié, dans nos travaux, sur un seul niveau d'abstraction. Il paraît important d'étendre l'étude à plusieurs niveaux d'abstraction afin de prendre en compte le caractère hiérarchique et structuré des produits et des projets.
3. En ce qui concerne la partie optimisation du couplage, elle nécessite un travail complémentaire de comparaison de l'algorithme *SPEA2* sous contraintes avec des méthodes d'optimisation exacte type *Branch and Bound*.
4. Nous avons utilisé, lors de nos expérimentations, le modèle de configuration/planification exploitant les *CSP** qui conserve la structure du problème. Nous pensons qu'il doit être tout à fait possible de poursuivre ces travaux et d'adapter nos propositions pour prendre en compte les problèmes avec ajout de variables formalisés avec les *DCSP*.
5. L'exemple, sur lequel nos propositions ont été testées, est relativement de petite taille 800 000 solutions. Nos propositions devraient être confrontées à un modèle plus conséquent, au moins de taille similaire à l'exemple proposé par Amilhastre et al. et présentant, au moins un troisième critère, telle que le performance.
6. L'utilisation conjointe et itérative des approches par contraintes et des algorithmes évolutionnaires devrait permettre de configurer, planifier et optimiser notre problème de couplage de manière interactive sur des problèmes plus conséquent. En effet, la considération de cycles de filtrage et d'optimisation limitée en nombre de génération et/ou en temps, devrait aboutir à la présentation successive, au décideur, de zones de solutions optimisées de plus en plus réduites et cohérentes avec les contraintes du problèmes. Un tel fonctionnement reste à expérimenter sur un problème de plus grand taille que celui considéré.
7. Les solutions retenues se montrant performantes pour le couplage de la configuration de produits et de la planification de projet, il serait intéressant de les appliquer à d'autres types de problèmes d'ingénierie.

2. thèse s'inscrivant aussi dans le cadre du projet *ATLAS*

Bibliographie

Aamodt, A. et Plaza, E. (1994). Case based reasoning : Fundational issues, methodoligical variations, and system approaches. *Artificial Intelligence Communication AICom*, 7(1) : pages 39–59.

Abeille, J., Goris, G., Vareilles, E., Roux, T., Aldanondo, M., et Coudert, T. (2009). Couplage de la conception de produit et de la planification de projet : Première analyse des pratiques industrielles. In *8émé Congrès international de Génie Indutriel, CIGI*.

AFNOR (1988). *Recommandations pour obtenir et assurer la qualité en conception. Norme X50-127*. AFNOR.

Akao, J. (1997). Qfd : Past, present, and future. In *International Symposium on QFD'97*, Linköping, Sweden.

Albano, L. et Suh, N. (1992). Axiomatique approach to structural design. *Researche in Engineering Design*, 4 : pages 171–183.

Aldanondo, A., Hadj-Hamou, K., Moynard, G., et Lamothe, J. (2003). Mass customization and configuration : Requirement analysis and constraint bases modeling propositions. *Inegrated Computer-Aided Engineering*, 10(2) : pages 177–189.

Aldanondo, M. et Vareilles, E. (2008). Configuration for mass customisation : how to extend product configuration towards requirements and process configuration. *Journal of intelligent Manufacturing*, 19(5) : pages 521–535.

Aldanondo, M., Vareilles, E., Abeille, J., Coudert, T., et Geneste, L. (2010). System design and design planning : an interaction identification. In *MOSIM*.

Allen, J. (1983). Maintaining knowledge about temporal intervals. *ACM*, 11 : pages 123–154.

Altshuller, G. (1984). *Creativity as an Exact Science*. Gordon and Breach Science Publishers.

Amilhastre, J. (1999). *Représentation par automate d'ensemble de solutions de problèmes de satisfaction de contraintes*. Thèse, Université de Montpellier.

Amilhastre, J., Fargier, H., et Marquis, P. (2002). Consistency restoration and explanations in dynamics csps- application to configuration. *Artificial Intelligence*, 135(1-2) : pages 199–234.

ANSI (1998). *ANSI*.

Barker, V., O'Connor, D., Bechant, J., et Soloway, E. (1989). Expert system for configuration at digital : Xco and beyond. *Communication of the ACM*, 32(3) : pages 298–318.

Baron, C. et Rochet, S. (2005). Assistance à la conduite de projet en ingénierie des systèmes-proposition d'une méthode et d'un outil évolutionnaire. *Journal Européen des Systèmes Automatisés (JESA)*, 43(9-10) : pages 1017–1040.

Bartak, R., Salido, M., et Rossi, F. (2010). Constraint satisfaction techniques in planning and scheduling. *Journal of Intelligent Manufacturing*, 21 : pages 5–15. DOI : 10.1007/s10845-008-0203-4.

Belmokhtar, S., Dolgui, A., Delorme, X., et Ignatenko, I. (2007). Optimizing modular machining line design problem with mixed activation mode of machining units. *Decision Making in Manufacturing and Services*, 1(1-2) : pages 35–48.

Belu, N. et Anghel, D. (2008). Using the fbs model for the analysus of the correlation "problem-solution" in the design process. *Annals of Oradea University. Fascicle of Management and Technological Engineering*, VII(XVII) : pages 1219–1222.

Benhamou, F. (1996). Heterogeneous constraint programming. In Verlag, S., editor, *5th international conference on algebraic and logic programming*, pages 62–76, Aachen, Allemagne.

Benhamou, F., Mc Allester, D., et Van Hentenryck, P. (1994). Clp(intervals) revisited. In *ILPS'94*, pages 1–21.

Bessière, C. et Cordier, M. (1993). Arc-consistency and arc-consistency again. In *AAAI*, pages 108–113, Cambridge MA.

Bessière, C. et Régin, J. (1994). An arc-consistency algorithm optimal in the number of constraint checks. In *ECAI Workshop on Constraint Processing*.

Beyer, H. (2001). *The theory of Evolution Strategies*. Natural Computing Series. Springer.

Beyer, H. et Schwefel, H. (2002). Evolution strategies - a comprehensive introduction. *Natural Computing*, 1(1) : pages 3–52.

Bidot, J. (2005). *A general framework integrating techniques for scheduling under uncertainty*. Thèse, Institut National Polytechnique de Toulouse.

Blessing, L. T. M. (1996). Comparison of design models proposed in prescriptive literature. In *Proceedings of the COST A3/COST A4 International research workshop on « The role of design in the shaping of technology »*, Lyon.

Bonjour, E. (2008). *Contributions à l'instrumentation du métier d'architecte système : de l'architecture modulaire du produit à l'organisation du système de conception*. Habilitation à diriger des recherches, Université de Franche-Comté.

Brown, D. et Chandrasekaran, B. (1985). Expert system for a class of mecanical design activity. In Gero, J. S., editor, *knowledge engineering in Computer-Aided Design*, pages 259–282, Amsterdam : North-Holland.

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice : Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press.

Carraway, R. L., Morin, T., et Moskowitz, H. (1990). Genaralized dynamic programming for multicriteria optimization. *European journal of operational research*, 44(1) : pages 95–104.

Cavaillé, J. et Aldanondo, M. (2001). *Gestion de production - fonctions, techniques et outils. Chapitre 3 :Ordonnancement et gestion de production dans l'industrie*, chapitre 3. Hermès Science Publications, Paris.

Chandrasekaran, B. (1990). Design problem solving : a task analysis. In *Artificial Intelligence Magazine*, volume 11, pages 59–71.

Coello Coello, C. (2002). Theoretical and numeric constraint-handling techniques used with eas : a survey of the state of art. *Computer methods in Applied Mechanics and Engineering*, 191(11-12) : pages 1245–1287.

Cohen, L. (1995). *Quality Function Deployment : How to make QFD work for you*. Addison-Wesley.

Danilovic, M. et Browning, T. (2007). Managing complex product development projects with design structure matrices and domain mapping matrices. *International Journal of Project Management*, 25 : pages 300–314.

Deb, K. (2002). *Multi-objective optimization using evolutionary algorithms*. Wiley-inteerscience series in systems and optimization. Wiley, J. & Sons.

Dechter, R.and Meiri, I. et Pearl, J. (1991). Temporal constraint network. *Artificial Intelligence*, 49 : pages 61–95.

Despontin Monsarrat, E. (2004). *Aide à la décision pour une coopération inter-entreprises dans le cadre de la production à la commande*. Thèse, University Toulouse 3 Paul sabatier.

Duharcourt, P. (1969). Introduction à la programmation dynamique. *revue economique*, 20(2) : pages 182–234.

Elmaghraby, S. (1977). *Activity networks : Project planning and control by netword models*. John Wiley Interscience.

Eppinger, S. (2001). innovation at the speed of information. *Harvard Business Review*, 79(1) : pages 149–158.

Esquirol, P. et Lopez, P. (1999). *L'ordonnancement*. Gestion Economica.

Faltings, B. (1994). Arc consistency for continuous variables. In *Artificial Intelligence*, volume 65, pages 363–376.

Fleischanderl, G., Friedrich, G., Jannach, D., et Zenker, M. (1998). configuring large sytstems using generative constraint satisfaction. *IEEE Intelligence Systems*, 13(4) : pages 59–68.

Fogel, L. J., Owens, A., et Walsh, M. J. (1966). *Artificial Intelligence through Simulated Evolution*. John Wiley, New York.

Gelle, E. (1998). *On the generation of locally consistent solution spaces in mixed dynamic constraint problems*. Thèse de doctorat, École Polytechnique Fédérale de Lausanne, Suisse.

Gelle, E. et Faltings, B. (2003). Solving mixed and conditional constraint satisfaction problems. In *Constraints*, pages 107–141.

Gelle, E. et Weigel, R. (1995). Interactive configuration based on incremental constraint satisfaction. In *IFIP*, pages 117–126.

Gentner, D. (1983). Structure-mapping : A theoretical framework for analogy. *Cognitive Science*, 7(2) : pages 155–170.

Gero, J. (1989). A locus for knowledge-based systems in caad education. In *CAAD futures Digital Proceedings*, pages 49–60.

Gero, J. (1990). Design prototypes : A knowledge representation schema for design. *AAAI*, 11(4) : pages 26–36.

Ghedira, K. (2007). *Optimisation combinatoire par méta heuristiques : Origines, concepts et éléments de base, algorithmes canoniques et étendus*. Sciences et technologies. Paris, editions technip édition.

Giaccobi, S. (2009). Méthode de conception de multimatériaux à architecture multicouche : application à la conception d'une canalisation sous-marine. Thèse d'Etat, Université de Bordeaux.

Glover, F. et Laguna, M. (1993). Tabou search. In *Modern Heuristic Techniques for Combinatorial Problems*, Oxford, England.

Goldberg, D. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison Wesley, New York.

Golomb, S. et Baumert, I. (1965). Backtrack programming. *Journal of ACM*, 12(4) : pages 516–524.

Golver, F. et Laguna, M. (1997). *Tabu search*. Kluwer Academic Publisher.

Goncalves-Coelho, A. (2004). Axiomatic design and concurrent engineering paradigm. In *Conference COmputinfg and Solutions in Manufacturing Engineering COSME 04*, Brasov, Roumanie.

Goncalves-Coelho, A., A.J.F., M., et Pereira, Z. (2005). Improving the use of qfd with axiomatic desugn. *CERA*, 13(3).

Gumus, B. (2005). *Axiomatic product development lifecycle*. Thèse, Graduate Faculty of Texas Tech University, U.S.

Haralick, R. et Elliot, G. (1980). Increasing tree search efficiency for constraint satisfaction problem. In *Articial Intelligence*, volume 14, pages 263–313.

Hart, P. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on systems science and cybenetics SSC4*, 4(2) : pages 100–107.

Holland, J. H. (1992). *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor.

Hvam, L., Riis, J., et M., M. (2002). A multi-prescreptive approach for the design of configuration systems. In *European Conference on Artificial Intelligence (ECAI02). Workshop on Configuration*, pages 56–62, Lyon, France.

Institute, P. M. (2004). *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*. Library of Congress Cataloging-in-Publication Data, 3rd edition édition.

Jiao, J., Zhang, L., Pokharel, S., et He, Z. (2007). Identifying generic routings for product families based on text mining nd tree matching. *Decision Support Systems*, 43 : pages 866–883.

Jiao, J., Zhang, L., Zhang, Y., et Pokharel, S. (2008). Association rule mining for product and process variety mapping. *International Journal of Computer Integrated Manufacturing*, 21(1) : pages 111–124.

Junker, U. et Mailharro, D. (2003). The logic of ilig(j) configurator : combining constraint programming with a description logic. In *Proceedings of IJCAI 2003 Workshop on Configuration*, pages 13–20.

Kennedy, J. et Eberhart, R. (1995). Particle swarm optimisation. In *IEEE int. conf. on Neural Networks*, pages 1942–1948, Australie.

Kirkpatrick, S., Gelatt, C., et Vecchi, M. (1983). Optimization by simulated annealing. In *Science*, volume 200, pages 671–680.

Kota, S. et Ward, A. (1991). *Functions, structures and constraints in conceptual design*. Ward, A.C., University of Michigan.

Kowalczyk, R. (1997). Constraint consistent genetic algorithms. In *Proc. of IEEE conf. on evolutionary computation*, pages 343–348.

Koziel, S. et Michalewicz, Z. (1999). Evolutionary algorithms, homomorphous mapping and constrained parameter optimisation. *Evolutionary Computation*, 7(1) : pages 19–44.

Kusiak, A. et Wang, J. (1993). *Decomposition in concurrent design, engineering*. Concurrent Engineering. J.Wiley.

Labrousse, M. et Bernard, A. (2004). Modéliser et gérer les objets d'entreprise : des concepts au système d'information. Présentation GDR-MACS-Nantes, 25 mars.

Lamothe, J., Hadj-Hamou, K., et Aldanondo, M. (2006). An optimization model for selecting a product family and designing its supply chain. *European Journal of Operational Research*, 169(3) : pages 1030–1047.

Lawson, M. et Karandikar, H. (1994). A survey of ce. *CERA Journal*.

Lebbah, Y., Rueher, M., et Michel, C. (2002). A global filtring algorithm for handling systems of quadratic equations and equations. In Verlag, S., editor, *8th International Conference on Principles and Practice of Constraint ProgrammingCP'08*, pages 109–123, London, UK.

Lhomme, O. (1993). Consistency techniques for numeric CSP. In *International Joint Conference on Artificial Intelligence*, pages 232–238, Chambéry, France.

Lindemann, U. (2007). A vision to overcome "chaotic " design for x processes in early phases. In *International conference on Engineering Design (ICED)*, Paris, France.

Lizarralde, I. (2007). *Aide au pilotage d'activité d'ingénierie pour le développement distribué d'un système complexe*. Thèse, Institut National des sciences Appliquées de Toulouse, Toulouse, France.

Lobjois, L. et Lemaitre, M. (1997). Coopération entre méthodes complètes et incomplètes pour la résolution de (V)CSP : une tentative d'inventaire. In *Journées Nationales de la Résolution Pratique de Problèmes NP-complets*, pages 67–73, Rennes, France.

Lévine, P. et Pomerol, J. (1989). *Systèmes interactifs d'aide à la décision et systèmes experts*. Hermès, Paris.

M., S. et Michalewicz, Z. (1998). Sphere operators and their applicability for constrained parameter optimization problems. In Porto, B. e., editor, *Proceedings of the seventh Annual Conference on Evolutionary Programming*. Springer Verlag.

Mackworth, A. (1977). Consistency in networks of relations. In *Artificial Intelligence*, volume 8(1), pages 99–118.

Marciniak, R.and Pagerie, M. (1998). *Gestion de projet : Guide pratique de la reussite de tous vos projets et produits industriels : coûts, delais, qualite*. Weka.

Martin, J. (1998). Evolution of eia 632 from an interim standard to a full standard. *in INCOSE 1989 Symposium*.

McDermott, J. (1982). R1 : A rule-based configurer of computer system. *Artificial intelligence*, 1(19) : pages 39–88.

McDonald, K. et Prosser, P. (2002). A case study of constraint programming for configuration problems. Technical report, APES Research Group. APES-45-2002.

Meiri, I. (1996). Combining qualitative and quantitative constraints in temporal reasoning. *Artificial Intelligence*, 87(1-2) : pages 343–385.

Meredith, J. R. et Mantel, S. J. (1989). *Project mamagement, a managerial approach*. John Wiley et Sons, New York, 2nd edition édition.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Berlin.

Michalewicz, Z. et Janikow, C. (1991). Handling constraints in genetic algorithms. In Belew, R. K. et Booker, L. e., editors, *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 151–157. Morgan Kauffmann.

Michalewicz, Z. et Nazhiyath, G. (1995). Genocop iii : A co-evolutionary algorithm for numerical optimization with nonlinear constraints. In editor, F., editor, *Proceedings of the second IEEE International Conference on Evolutionary Computation*, pages 647–651. IEEE Press.

Mittal, S. et Falkenhainer, B. (1990). Dynamic constraint satisfaction problems. In *AAAI*, pages 25–32, Boston, US.

Mittal, S. et Frayman, F. (1989). Towards a generic model of configuration tasks. In *proceedings of the Eleventh International joint Conference on Artificial Intelligence*, pages 1395–1401.

Montanari, U. (1974). Networks of constraints : fundamental properties and application to picture processing. In *Information sciences*, volume 7, pages 95–132.

Mouhoub, M. et Sukpan, A. (2005a). A new temporal csp framework handling composite variables and activity constraints. In *the 17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'05)*, pages 143–149.

Mouhoub, M. et Sukpan, A. (2005b). A new temporal csp framework handling composite variables and activity constraints. In *The 17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'05)*, pages 143–149, Hong Kong.

Moynard, G. (2003). *Contribution au déploiement de progiciels de configuration dans l'industrie : éléments de modélisation et d'estimation*. Thèse de doctorat, Institut National Polytechnique de Toulouse.

Mulyanto, T. (2002). *Utilisation des techniques de programmation par contraintes pour la conception d'avions*. Thèse de doctorat, École Nationale Supérieure de l'Aéronautique et de l'Espace, Toulouse, France.

Myung, M. Kim, J. et Fogel, D. (1995). Preliminary investigation into a two-stage method of evolutionary optimization on constrained problems. In McDonnell, J., Reynolds, R. G., et Fogel, D. B. e., editors, *Proceedings of the fourth Annual Conference on Evolutionary Programming*, pages 449–463. MIT Press.

Oosterman, B. (2001). *Improving product development projects by matching product architecture and organization*. Thèse, University of Groningen, Netherlands.

Pahl, G. et Beitz, W. (1996). *Engineering Design : a Systematic Approach*. Springer-Verlag.

Pargamin, B. (2002). Vehicule sales configuration : the cluster tree approach. In *European Conference on Artificial Intelligence (ECAI 02), Workshop on Configuragion*, pages 35–40, Lyon, France.

Parmee, I. et Purchase, G. (1994). The development of directed genetic search technique for heavily constrained design spaces. In *Proceedings of the Conference on Adaptive Computing in Engineering Design and Control*, pages 97–102. University of Plymouth.

Pitiot, P. (2009). *Amélioration des techniques d'optimisation combinatoire par utilisation d'un mécanisme de retour d'expérience : Application à la sélection de scénarios en conception préliminaire de produit-projet*. Thèse, Écoles Nationale d'Ingénieur de Tarbes.

Prasad, B. (1997). *Concurrent engineering fundamentals*, volume 1 et 2. Prentice Halls.

Ramat, E. (1997). *Modélisation et planification de projets complexes à contraintes de ressources : le modèle RAIH*. Thèse, Université de Tours.

Rechenberg, I. (1965). *Cyberbetic solution path of an experimental problem*. Royal Aircraft Establishment Library Translation.

Renaud, J., Bonjour, E., Chebel Morello, B., Fuchs, B., et Matta, N. (2008). *Retour et Capitalisation d'expérience, Outils et démarches*. AFNOR.

Richardson, J., Palmer, M., Liepins, G., et Hilliard, M. (2009). Some guidlines for ga with penality functions. In *Third international conference on GA*, pages 175–192. Schaffer, J.D. Editor.

Rosenau, M. et Githens, G. (2005). *Successful Project Management. A step-by-step Approach with Practical Examples*. John Wiley et Sons, New Jersey, fourth edition édition.

Rossi, F., van Beek, P., et Walsh, T., editors (2006). *Handbook of Constraint Programming*. Elsevier. ISBN : 0-444-52726-5.

Sabin, D. et Freuder, E. (1996). Configuration as composite constraint satisfaction. In *Artificial Intelligence and Manufacturing Research Planning Workshop*, pages 153–161.

Sabin, D. et Weigel, R. (1998). Product configuration frameworks - a survey. *IEEE Intelligent System and their Applications*.

Salcedo-Sanz, S. (2009). A survey of repair methods used as constraint handling techniques in evolutionary algorithms. *Computer science review*, pages 175–192.

Sam-Haroud, J. (1995). *Constraint consistency techniques for continuous domains*. Thèse, Ecole Polytechnique Fédérale de Lausanne.

Scaravetti, D. (2004). *Formalisation préalable d'un problème de conception, pour l'aide à la décision en conception préliminaire.* Thèse, Ecole Nationale Supérieure D'Arts et Métiers (ENSAM.

Schoenauer, M. et Xanthakis, S. (1993). Constrained ga optimization. In Forrest, S. e., editor, *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 573–580. Morgan Kaufmann.

Schwefel, H. (1981). *Numerical Optimization of Computer Models*. Wiley, J. & Sons.

Soininen, T.and Niemelä, I. (1999). developing a declarative rule language for application in product configuration. In springer verlag, editor, *Workshop on Practical Aspect of Declarative Language (PADL 99), Lecture Notes in Computer science*, San-Fransisco.

Soininen, T. et Gelle, E. (1999). Dynamic constraint satisfaction in configuration. In *American Association for Artificial Intelligence, Workshop on Configuration*, Orlando, US.

Srinivas, N. et Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3) : pages 221–248.

Steward, D. (1981). The design structure system : a method for managing the design of complex systems. *IEEE Transactions on Engineering Management*, 28(3) : pages 71–74.

Steward, D. et Tate, D. (2000). Integration of axiomatic design and project planning. In *First International Conference on Axiomatic Design*, pages 285–289, Combridge, MA.

Stewart, B ans White, C. (1991). Multiobjectif a*. *Journal of the ACM*, 38(4) : pages 775–814.

Suh, N. (1990). *The principles of Design*. Oxford series on Advance Manufacturing. Oxford University Press, New York.

Suh, N. (2001). *Axiomatic Design : Advances and Application*. The MIT-Pappalardo Series in Mechanical Engineering. Oxford University Press, New York.

Surry, P.D.and Radcliffe, N. et Boyd, I. D. (1995). A multi-objective approach to constrained optimization of gaz supply networks. In Fogarty, T., editor, *Proceedings of the AISB-95 Workshop on Evolutionary Computing*, volume 993, pages 166–180. Springer Verlag.

Talbi, E. (2002). A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8(5) : pages 541–564.

Tiihonen, J. et Soininen, T. (1997). Product configurators - information system support for configurable product. Technical report, TAi Research center and laboratory of information processing science, Product Data Management Group, Helsinki University of Thechnology, Finland.

Tiihonen, J., Soininen, T., Männistö, T., et Sulonen, R. (1996). State-of-the-practice in product configuration - a survey of 10 cases in the finnish industry. Technical report, IIA-Research Center, Helsinki University of Technology Otakaari, Finland.

Tsamardinos, I., Vidal, T., et Pollack, M. (2003). Ctp : A new constraint-based formalism for conditionnal temporal planning. *Constraints*, 8(4) : pages 365–388.

Tsang, E. (1993). Foundations of constraints satisfaction. In *Academic Press*, London.

Van Oudenhove de Saint Géry, T. (2006). *Contribution à l'élaboration d'un formalisme gérant la pertinence pour les problèmes d'aide à la conception à base de contraintes*. Thèse, Institut National Polytechnique de Toulouse.

Vareilles, E. (2005). *Conception et approches par propagation de contraintes : contribution à la mise en oeuvre d'un outils d'aide intéractif*. Thèse, Institut National Polytechnique de Toulouse.

Vareilles, E., Aldanondo, M., et Gaborit, P. (2007). Evaluation and design : a knowledge bases approach. *International Journal of Computer Integrated Manufacturing*, 20(7) : pages 639–653.

Vargas, C. (1995). *Modélisation du processus de conception en ingénierie des systèmes mécaniques. Mise en oeuvre basée sur la propagation de contraintes. Application à la conception d'une culasse automobile*. Thèse de doctorat, École Normale Supérieure de Cachan, France.

Vempaty, N. (1992). Solving constraint satisfaction problems using finite state automata. In *Swartout*, pages 453–458.

Vernat, Y. (2004). *Formalisation et qualification de modèles par contraintes en conception préliminaire*. Thèse de doctorat, Ecole Nationale des Arts et Métier, Bordeaux, France.

Veron, M. (2001). *Modélisation et résolution du problème de configuration industrielle : utilisation des techniques de satisfaction de contraintes*. Thèse de doctorat, Institut National Polytechnique de Toulouse, France.

Vilim, P., Bartak, R., et Cepek, O. (2004). Unary ressource constraint with optional activities. In *In Proceedings CP 2004*, pages 62–76.

Waagen, D., Diercks, P., et McDonnell, J. (1992). The stochastic direction set algorithm : A technique for finding function extrema. In Fogel, D. et Atmar, W. e., editors, *Proceedings of the 1st Annual Conference on Evolutionary Programming*, pages 35–42. Evolutionary Programming Society.

Watson, I. et Marir, F. (1994). Case-based reasoning : A review. *The Knowledge Engineering Review*, 9(4).

Weisbuch, G. (1989). *Dynamique des systèmes complexes : une introduction aux réseaux d'automates*. InerEditions/Editions du CNRS.

Yoshikawa, H. (1989). Design philosophy : the state of the art. *CIRP annals manufacturing technology*, 38(2) : pages 579–586.

Zitzler, E. et Thiele, L. (1989). An evolutionary algorithm for multiobjective optimization : the strength pareto approach. Technical report, Computer Engineering and Networks laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Suisse.

Zitzler, E., Thiele, L., et Laumanns, M. (2001). Spea2 : improving the strength pareto evolutionary algorithm. Technical report, Computer Engineering and Networks laboratory, , Swiss Federal Institute of Technology (ETH), Zurich, Suisse.

Annexes : Implémentation des modèles

.1 Implémentation avec CoFiADe

.1.1 Déclaration des variables CS et EN

```
my $domains =
 qq{
       domain integer CS_dom = {350, 400, 450, 500, 550, 600};
       domain symbolic EN_dom = {"1LP", "2LP", "1HP", "2HP", "1ECO", "2ECO"};
 };
my $variables =
qq{
       variable symbolic EN in EN_dom;
       variable integer CS in CS_dom;
 };
```

.1.2 Implémentation de la contrainte $C_{C2}(CS, EN)$

```
 constraint Cc2 using (CS, EN)
     table {

        {350, !="all"},
        {400, !="all"},
        {450, !="all"},
        {500, !="all"},
        {550, !="1LP"},
        {600, "2HP"},
        {600, "2ECO"},
     };
```

.1.3 Implémenation du réservoir supplémentaire en *CSP**

```
% Déclaration du domaine et de la variable optionnelle ST
 domain symbolic ST_dom = {"*", "ST100", "ST200", "ST300"};
 variable symbolic ST in ST_dom;
% Contrainte Cc5(CS, FR, ST)
 constraint Cc5 using (CS, FR, ST)
     table {
        {350, != 0, "*"},
        {400, 600, "*"},
        {400, 800, "*"},
        {400, 1000, "*"},
        {400, 1200, "ST100"},
```

```
        {400, 1400, "ST200"},
        {400, 1600, "ST300"},
        {450, 600, "*"},
        {450, 800, "*"},
        {450, 1000, "*"},
        {450, 1200, "ST100"},
        {450, 1400, "ST200"},
        {450, 1600, "ST300"},
        {500, 600, "*"},
        {500, 800, "ST100"},
        {500, 1000, "ST100"},
        {550, 600, "*"},
        {550, 800, "ST200"},
        {550, 1000, "ST200"},
        {600, 600, "*"},
        {600, 800, "ST300"},
        {600, 1000, "ST300"},
    };
```

.1.4 Implémenation du réservoir supplémentaire en *DCSP*

```
% Déclaration du domaine de ST et de la variable ST
  domain symbolic ST_dom = {"ST100", "ST200", "ST300"};
  inactive variable symbolic ST in ST_dom;
% Déclaration de la contrainte Cc6 initialement inactif
 freeze constraint Cc6 using (CS, FR, ST)
     table {

        {400, 1200, "ST100"},
        {400, 1400, "ST200"},
        {400, 1600, "ST300"},
        {450, 1200, "ST100"},
        {450, 1400, "ST200"},
        {450, 1600, "ST300"},
        {500, 800, "ST100"},
        {500, 1000, "ST100"},
        {550, 800, "ST200"},
        {550, 1000, "ST200"},
        {600, 800, "ST300"},
        {600, 1000, "ST300"},
    };
% Activation de ST et de Cc6
 constraint Ca1 using (CS, FR) activate (ST, Cc6)
     table {
         {> 380, >1100},
         {>480, >700},
     };
```

.1.5 Implémenation des élements en exclusion FCU et FCA en *CSP**

```
% Déclaration des domaines et des varibles FCA, FCU
        domain symbolic FCA_dom = {"*", "Standard", "Luxe", "Confort"};
        domain symbolic FCU_dom = {"*", "Work","Leisure"};

        variable float FCU in FCU_dom;
        variable float FCA in FCA_dom;
% Déclaration de la contrainte éxclusion
   constraint Cex using (FCA, FCU)
     table {
         {"*","Work"},
```

```
          {"*","Leisure"},
          {"Standard","*"},
          {"Luxe","*"},
          {"Confort","*"},
     };
```

.1.6 Implémenation des élements en exclusion FCU et FCA en *DCSP*

```
   % Déclaration des domaines et des varibles FCA, FCU
          domain symbolic FCA_dom = {"Standard", "Luxe", "Confort"};
          domain symbolic FCU_dom = {"Work","Leisure"};
          domain symbolic Sel_F_dom = {"fca","fcu"};

          inactive variable float FCU in FCU_dom;
          inactive variable float FCA in FCA_dom;
          variable symbolic Sel_F in Sel_F_dom;
  % Déclaration des contraintes d'activation selon la valeur du sélecteur
    constraint Ca2 using (Sel_F) activate (FCA)
        table {
            {"fca"},
        };
    constraint Ca3 using (Sel_F) activate (FCU)
        table {
            {"fcu"},
        };
```

.2 Implémentation avec ILOG CP 5.5

.2.1 Déclaration des variables CS et EN

```
IloIntVar cs;
IloIntVar en;
cs = cp.intVar(new int []{350, 400, 450, 500, 550, 600 });
en = cp.intVar(new int []{lP1, lP2, hP1, hP2, eCO1, eCO2});
```

.2.2 Implémentation de la contrainte $C_{C2}(CS,EN)$

```
 /* en utilisant la méthode allowedAssignments*/
/* Déclaration de chaque tuple*/

ilog.concert.IloIntTupleSet table2= cp.intTable(2);
ilog.concert.IloIntArray tuple21 = cp.intArray(2);
tuple21.set(0,350);
tuple21.set(1,lP1);
table2.add(tuple21);

ilog.concert.IloIntArray tuple22 = cp.intArray(2);
tuple22.set(0, 350);
tuple22.set(1,lP2);
table2.add(tuple22);

ilog.concert.IloIntArray tuple23 = cp.intArray(2);
```

```
tuple23.set(0,350);
tuple23.set(1,hP1);
table2.add(tuple23);

ilog.concert.IloIntArray tuple24 = cp.intArray(2);
tuple24.set(0,350);
tuple24.set(1,eCO1);
table2.add(tuple24);

ilog.concert.IloIntArray tuple25 = cp.intArray(2);
tuple25.set(0,400);
tuple25.set(1,lP1);
table2.add(tuple25);

ilog.concert.IloIntArray tuple26 = cp.intArray(2);
tuple26.set(0, 400);
tuple26.set(1,lP2);
table2.add(tuple26);

        /* La même chose pour les autres tuple de la table
        .
        .
        .

        /* Déclaration de la table comme étant une contrainte*/

IloConstraint cc2 = cp.allowedAssignments(new IloIntVar[]{cs,en}, table2);

        /* Ajout de la contrainte à l'ensemble des contraintes*/
contraintes.add(cc2);
```

.2.3 Implémenation du réservoir supplémentaire en *CSP**

```
/* Contrainte de compatibilité entre cs , fr et st */
      % Déclaration de la table
ilog.concert.IloIntTupleSet table5 = cp.intTable(3);

      % Déclaration d'un tuple de la table
ilog.concert.IloIntArray tuple15 = cp.intArray(3);
tuple15.set(0,350);
tuple15.set(1,600);
tuple15.set(1,0);
table5.add(tuple15);

ilog.concert.IloIntArray tuple251 = cp.intArray(3);
tuple251.set(0,350);
tuple251.set(1,800);
tuple251.set(1,0);
table5.add(tuple251);

% La même chose pour le reste des tuple au nombre de 27
        .
        .
        .

        /* Déclaration de la table comme étant une contrainte*/

IloConstraint cc5 = cp.allowedAssignments(new IloIntVar[]{cs,fr, st}, table5);
        /* Ajout de la contrainte à l'ensemble des contraintes*/
contraintes.add(cc5);
```

.2.4 Implémenation du réservoir supplémentaire en *DCSP*

.2.5 Implémenation des élements en exclusion FCU et FCA en *CSP*$\star$

.2.6 Implémenation des élements en exclusion FCU et FCA en *DCSP*

.3 Implémentation avec ECLiPSe

.3.1 Déclaration des variables CS et EN

```
:-export domain(moteur(un_Mot_ECO, un_Mot_BP,un_Mot_HP,
                deux_Mot_ECO, deux_Mot_BP,deux_Mot_HP)).
EN &:: moteur,
CS #::[350, 400, 450, 500, 550,600],
```

.3.2 Implémentation de la contrainte $C_{C2}(CS,EN)$

```
       % Déclaration de la Contrainte Cc2: CS et EN  %

(
(CS #= 350 , EN &= un_Mot_ECO);
(CS #= 350 , EN &= un_Mot_HP);
(CS #= 350 , EN &= deux_Mot_BP);
(CS #= 350 , EN &= un_Mot_BP);
(CS #= 400 , EN &= un_Mot_BP);
(CS #= 400 , EN &= deux_Mot_BP);
(CS #= 400 , EN &= un_Mot_HP);
(CS #= 400 , EN &= deux_Mot_HP);
(CS #= 400 , EN &= un_Mot_ECO);
(CS #= 400 , EN &= deux_Mot_ECO);
(CS #= 450 , EN &= un_Mot_ECO);
(CS #= 450 , EN &= deux_Mot_ECO);
(CS #= 450 , EN &= deux_Mot_HP);
(CS #= 450 , EN &= un_Mot_HP);
(CS #= 450 , EN &= deux_Mot_BP);
(CS #= 450 , EN &= un_Mot_BP);
(CS #= 500 , EN &= un_Mot_ECO);
(CS #= 500 , EN &= deux_Mot_ECO);
(CS #= 500 , EN &= deux_Mot_HP);
(CS #= 500 , EN &= un_Mot_HP);
(CS #= 500 , EN &= deux_Mot_BP);
(CS #= 500 , EN &= un_Mot_BP);
       (CS #= 550 , EN &= un_Mot_ECO);
(CS #= 550 , EN &= deux_Mot_ECO);
(CS #= 550 , EN &= deux_Mot_HP);
```

```
(CS #= 550 , EN &= un_Mot_HP);
(CS #= 550 , EN &= deux_Mot_BP);
(CS #= 600 , EN &= deux_Mot_ECO);
(CS #= 600 , EN &= deux_Mot_HP)

)
```

.3.3 Implémenation du réservoir supplémentaire en *CSP**

```
  % Déclaration du domaine et de la variable ST
       :-export domain(reservoir( *, sT1, sT2, sT3)).
ST &:: reservoir,
  % Déclaration de la contrainte Cc5: CS, FR et ST
  (
(CS #= 600 , FR #= 800, ST &= sT3);
        (CS #= 600 , FR #= 1000, ST &= sT3);
        (CS #= 400 , FR #= 1600, ST &= sT3);
(CS #= 450 , FR #= 1600, ST &= sT3);
(CS #= 400 , FR #= 1400, ST &= sT2);
(CS #= 450 , FR #= 1400, ST &= sT2);
(CS #= 550 , FR #= 800 , ST &= sT2);
(CS #= 550 , FR #= 1000, ST &= sT2);
(CS #= 400 , FR #= 1200, ST &= sT1);
(CS #= 450 , FR #= 1200, ST &= sT1);
(CS #= 500 , FR #= 800 , ST &= sT1);
(CS #= 500 , FR #= 1000, ST &= sT1);
(CS #= 350 , FR #= 600, ST &= *);
(CS #= 350 , FR #= 800, ST &= *);
(CS #= 350 , FR #= 1000, ST &= *);
(CS #= 350 , FR #= 1200, ST &= *);
(CS #= 350 , FR #= 1400, ST &= *);
(CS #= 350 , FR #= 1600, ST &= *);
(CS #= 400 , FR #= 600, ST &= *);
(CS #= 400 , FR #= 800, ST &= *);
(CS #= 400 , FR #= 1000, ST &= *);
(CS #= 450 , FR #= 600, ST &= *);
(CS #= 450 , FR #= 800, ST &= *);
(CS #= 450 , FR #= 1000, ST &= *);
(CS #= 500 , FR #= 600, ST &= *);
(CS #= 550 , FR #= 600, ST &= *);
(CS #= 600 , FR #= 600, ST &= *)

      )
```

.3.4 Implémenation du réservoir supplémentaire en *DCSP*

```
  % Déclaration du domaine et de la variable ST
       :-export domain(reservoir( sT1, sT2, sT3)).
ST &:: reservoir,

  % Activation de la contrainte Cc6: CS, FR et ST  %

       (
       (((FR #> 1100),(CS #> 380));((FR #> 700),(CS #>480))  )  =>
(

%  La contraintes table activée

     (CS #= 600 , FR #= 800, ST &= sT3);
     (CS #= 600 , FR #= 1000, ST &= sT3);
```

```
            (CS #= 400 , FR #= 1600, ST &= sT3);
     (CS #= 450 , FR #= 1600, ST &= sT3);
     (CS #= 400 , FR #= 1400, ST &= sT2);
     (CS #= 450 , FR #= 1400, ST &= sT2);
     (CS #= 550 , FR #= 800 , ST &= sT2);
     (CS #= 550 , FR #= 1000, ST &= sT2);
     (CS #= 400 , FR #= 1200, ST &= sT1);
     (CS #= 450 , FR #= 1200, ST &= sT1);
     (CS #= 500 , FR #= 800 , ST &= sT1);
     (CS #= 500 , FR #= 1000, ST &= sT1)


       )
```

.3.5 Implémenation des élements en exclusion FCU et FCA en $CSP^{x}\star$

```
  % Déclaration des domaines et des variables FCU et FCA
        :-export domain(finitionCa(*, standard, confort, luxe)).
        :-export domain(finitionCu(*, work, leisure)).
FCU &:: finitionCu,
FCA &:: finitionCa,
  % Déclaration de la contrainte éxclusion entre FCA, FCU

(
 (FCA &= *  ,FCU &\= * );
 (FCA &\= *  ,NEN &= *)

        )
```

.3.6 Implémenation des élements en exclusion FCU et FCA en *DCSP*

```
  % Pas possible car on active pas de variable dans \eclipse
```

Annexes : Code source du couplage en *CSP** implémentation avec CoFiADe

Cette annexe présente le code source du modèle complet de couplage en *CSP** implémenté avec CoFiADe.

```
#Domaines
my $domains =
  qq{
          domain integer SN_dom = {3, 4, 6, 8, 10 ,12};
          domain float SNC_dom = {30, 40, 60, 80, 100, 120};

          domain symbolic EN_dom = {"1LP", "2LP", "1HP", "2HP", "1ECO", "2ECO"};
          domain float ENC_dom = {20, 40, 50, 100, 80};

          domain integer CS_dom = {350, 400, 450, 500, 550, 600};

          domain integer FR_dom = {400, 600, 800, 1000, 1200, 1400, 1600};
          domain symbolic F_dom = {"Standard", "Luxe", "Confort", "Custom"};
          domain float FC_dom = {20, 40, 80, 200};

          domain symbolic GS_dom = {"Small", "Medium", "Big"};
          domain integer NEN_dom = {[1, 2]};

          domain integer ST_dom = {100, 200, 300, 0};
          domain float STC_dom = {0, 10, 20, 30};

          domain float GC_dom = {[0, 10000]};
          domain float DC_dom = {[0, 10000]};
          domain float PC_dom = {[0, 10000]};
};
my $variables =
  qq{
          variable integer SN in SN_dom;
          variable symbolic EN in EN_dom;
          variable integer CS in CS_dom;
          variable integer FR in FR_dom;
          variable symbolic F in F_dom;
          variable symbolic GS in GS_dom;
          variable integer NEN in NEN_dom;
          variable float SNC in SNC_dom;
          variable float ENC in ENC_dom;
          variable float STC in STC_dom;
          variable float FC in FC_dom;
          variable integer ST in ST_dom;
          variable float GC in GC_dom;
          variable float DC in DC_dom;
          variable float PC in PC_dom;
};

my $planning_dom = qq{
          actif group Start;
```

```
          in group Start {
          domain float pst_dom = {0};
          domain float pft_dom = {0};
          domain float plt_dom = {0};
          };

          actif group End;
          in group End {
          domain float pst_dom = {[0, 500]};
          domain float pft_dom = {[0, 500]};
          domain float plt_dom = {0};
          };

          actif group Fab;
          in group Fab {
          domain float pst_dom = {[0, 10000]};
          domain float pft_dom = {[0, 10000]};
          domain float plt_dom = {[10, 90]};
          domain integer qrs_dom = {[1, 3]};
          domain float pc_dom = {[20, 84]};
          };

          actif group Appro;
          in group Appro {
          domain float pst_dom = {[0, 10000]};
          domain float pft_dom = {[0, 10000]};
          domain float plt_dom = {[10, 170]};
          domain float pc_dom = {[20, 140]};
          domain integer rrs_dom = {[1, 5]};
          };

          actif group Assem;
          in group Assem {
          domain float pst_dom = {[0, 10000]};
          domain float pft_dom = {[0, 10000]};
          domain float plt_dom = {[10, 180]};
          domain integer qrs_dom = {[1, 3]};
          domain float pc_dom = {[20, 168]};
          };

          group Mont_Res;
             in group Mont_Res {
             domain float pst_dom = {[0, 10000]};
             domain float pft_dom = {[0, 10000]};
             domain float plt_dom = {0,[5, 40]};
             domain integer qrs_dom = {0, 1, 2, 4};
             domain float pc_dom = {0,[10, 28]};
          };

          group Finish;
          in group Finish {
          domain float pst_dom = {[0, 10000]};
          domain float pft_dom = {[0, 10000]};
          domain float plt_dom = {[5, 40], [60, 100] };
          domain integer qrs_dom = { 1, 3, 2, 4};
          domain float pc_dom = {[10, 28], [35, 45]};
          };
};

my $planning = qq{
          in group Start {
          variable float pst in pst_dom;
          variable float pft in pft_dom;
          variable float plt in plt_dom;
          };

          in group End {
          variable float pst in pst_dom;
          variable float pft in pft_dom;
          variable float plt in plt_dom;
```

```
          };

          in group Fab {
          variable float pst in pst_dom;
          variable float pft in pft_dom;
          variable float plt in plt_dom;
          variable integer qrs in qrs_dom;
          variable float pc in pc_dom;
          };

          in group Appro {
          variable float pst in pst_dom;
          variable float pft in pft_dom;
          variable float plt in plt_dom;
          variable float pc in pc_dom;
          variable integer rrs in rrs_dom;
          };

          in group Assem {
          variable float pst in pst_dom;
          variable float pft in pft_dom;
          variable float plt in plt_dom;
          variable integer qrs in qrs_dom;
          variable float pc in pc_dom;
          };

          in group Mont_Res {
          variable float pst in pst_dom;
          variable float pft in pft_dom;
          variable float plt in plt_dom;
          variable integer qrs in qrs_dom;
          variable float pc in pc_dom;
          };

          in group Finish {
          variable float pst in pst_dom;
          variable float pft in pft_dom;
          variable float plt in plt_dom;
          variable integer qrs in qrs_dom;
          variable float pc in pc_dom;
          };


};

my $constraints_design =
  qq{
    constraint FC1 using (SN, EN)
        table {
           {3, "1LP"},
           {3, "2LP"},
           {3, "1ECO"},
           {3, "2ECO"},
           {4, !="2HP"},
           {6, !="all"},
           {8, !="all"},
           {10, !="1LP"},
           {12, "1HP"},
           {12, "2HP"},
           {12, "1ECO"},
           {12, "2ECO"},
        };

    constraint FC2 using (CS, EN)
        table {
           {350, !="all"},
           {400, !="all"},
           {450, !="all"},
           {500, !="all"},
           {550, !="1LP"},
```

```
            {600, "2HP"},
            {600, "2ECO"},
        };

    constraint FC3 using (CS, FR, ST)
        table {
            {350, != 0, 0},
            {400, 400, 0},
            {400, 600, 0},
            {400, 800, 0},
            {400, 1000, 0},
            {400, 1200, 100},
            {400, 1400, 200},
            {400, 1600, 300},
            {450, 400, 0},
            {450, 600, 0},
            {450, 800, 0},
            {450, 1000, 0},
            {450, 1200, 100},
            {450, 1400, 200},
            {450, 1600, 300},
            {500, 400, 0},
            {500, 600, 0},
            {500, 800, 100},
            {500, 1000, 100},
            {550, 400, 0},
            {550, 600, 0},
            {550, 800, 200},
            {550, 1000, 200},
            {600, 400, 0},
            {600, 600, 0},
            {600, 800, 300},
            {600, 1000, 300},
        };

    constraint FC5 using (SN, GS)
        table {
            {3, "Small"},
            {4, "Small"},
            {6, "Medium"},
            {8, "Medium"},
            {10, "Big"},
            {12, "Big"},
        };

    constraint FC6 using (EN, NEN)
        table {
            {"1LP", 1},
            {"2LP", 2},
            {"1HP", 1},
            {"2HP", 2},
            {"1ECO", 1},
            {"2ECO", 2},
        };
};

my $constraints_design_cost =
qq {
    constraint DC1 using (SN, SNC)
        table {
            {3, 30},
            {4, 40},
            {6, 60},
            {8, 80},
            {10, 100},
            {12, 120},
        };

    constraint DC2 using (EN, ENC)
        table {
```

```
            {"1LP", 20},
            {"2LP", 40},
            {"1HP", 50},
            {"2HP", 100},
            {"1ECO", 40},
            {"2ECO", 80},
        };

    constraint DC3 using (ST, STC)
        table {
            {100, 10},
            {200, 20},
            {300, 30},
            {0, 0}
        };

    constraint DC4 using (F, FC)
        table {
            {"Standard", 20},
            {"Confort", 40},
            {"Luxe", 80},
            {"Custom", 200},
        };
};

my $constraints_num_times =
  qq {

     in group Start {
        constraint Calcul_pft using (pst, pft, plt)
        condition {pft = pst + plt};

        constraint Calcul_pst using (pst, pft, plt)
        condition {pft - plt = pst};

        constraint Calcul_plt using (pst, pft, plt)
        condition {pft - pst = plt};
     };

     in group Fab {
        constraint Calcul_pft using (pst, pft, plt)
        condition {pft = pst + plt};

        constraint Calcul_pst using (pst, pft, plt)
        condition {pft - plt = pst};

        constraint Calcul_plt using (pst, pft, plt)
        condition {pft - pst = plt};
     };

     in group Appro {
        constraint Calcul_pft using (pst, pft, plt)
        condition {pft = pst + plt};

        constraint Calcul_pst using (pst, pft, plt)
        condition {pft - plt = pst};

        constraint Calcul_plt using (pst, pft, plt)
        condition {pft - pst = plt};
     };

    in group Assem {
        constraint Calcul_pft using (pst, pft, plt)
        condition {pft = pst + plt};

        constraint Calcul_pst using (pst, pft, plt)
        condition {pft - plt = pst};

        constraint Calcul_plt using (pst, pft, plt)
        condition {pft - pst = plt};
```

```
    };

    in group End {
       constraint Calcul_pft using (pst, pft, plt)
       condition {pft = pst + plt};

       constraint Calcul_pst using (pst, pft, plt)
       condition {pft - plt = pst};

       constraint Calcul_plt using (pst, pft, plt)
       condition {pft - pst = plt};
    };

    in group Mont_Res {
       constraint Calcul_pft using (pst, pft, plt)
       condition {pft = pst + plt};

       constraint Calcul_pst using (pst, pft, plt)
       condition {pft - plt = pst};

       constraint Calcul_plt using (pst, pft, plt)
       condition {pft - pst = plt};
    };

    in group Finish {
       constraint Calcul_pft using (pst, pft, plt)
       condition {pft = pst + plt};

       constraint Calcul_pst using (pst, pft, plt)
       condition {pft - plt = pst};

       constraint Calcul_plt using (pst, pft, plt)
       condition {pft - pst = plt};
    };
};

my $constraints_anteriority_Start_Fab =
  qq{
constraint S_before_FA using (Start::pft, Fab::pst)
  condition { Start::pft < Fab::pst + 0 };

constraint FA_after_S using (Start::pft, Fab::pst)
  condition { Start::pft +0 < Fab::pst };

};

my $constraints_anteriority_Start_Appro =
  qq{

constraint S_before_Ap using (Start::pft, Appro::pst)
  condition { Start::pft < Appro::pst + 0 };

constraint Ap_after_S using (Start::pft, Appro::pst)
  condition { Start::pft +0 < Appro::pst };

};


my $constraints_anteriority_Fab_Assem =
  qq{

constraint FA_before_As using (Fab::pft, Assem::pst)
  condition { Fab::pft < Assem::pst + 0 };

constraint As_after_FA using (Fab::pft, Assem::pst)
  condition { Fab::pft +0 < Assem::pst };

};

my $constraints_anteriority_Appro_Assem =
```

```
  qq{

constraint Ap_before_As using (Appro::pft, Assem::pst)
  condition { Appro::pft < Assem::pst + 0 };

constraint As_after_Ap using (Appro::pft, Assem::pst)
  condition { Appro::pft +0 < Assem::pst };

};

my $constraints_anteriority_Assem_Mont_Res =
  qq{

constraint As_before_MontRes using (Assem::pft, Mont_Res::pst)
  condition { Assem::pft < Mont_Res::pst + 0 };

constraint MontRes_after_As using (Assem::pft, Mont_Res::pst)
  condition { Assem::pft +0 < Mont_Res::pst };

};


my $constraints_anteriority_Mont_Res_Finish =
  qq{

constraint MontRes_before_Fi using (Mont_Res::pft, Finish::pst)
  condition { Mont_Res::pft < Finish::pst + 0 };

constraint Fi_after_MontRes using (Mont_Res::pft, Finish::pst)
  condition { Mont_Res::pft +0 < Finish::pst };

};

my $constraints_anteriority_Finish_End =
  qq{

constraint Fi_before_E using (Finish::pft, End::pst)
  condition { Finish::pft < End::pst + 0 };

constraint E_after_Fi using (Finish::pft, End::pst)
  condition {Finish::pft +0 < End::pst };

};

my $constraints_project = qq{
    in group Appro {
        constraint plt_pc_rrs using (rrs, plt, pc)
        table {
           {1, [150, 170], [20, 35]},
           {2, [110, 150], [35, 65]},
           {3, [70, 110], [65, 95]},
           {4, [30, 70], [95, 125]},
           {5, [10, 30], [125, 140]},
        };

    };
};

my $constraints_COUPLING =
  qq{
    constraint GS_Fabqrs_Fabplt_Fabpc using (GS, Fab::qrs,
               Fab::plt, Fab::pc)
        table {
        {"Small", 1, [22, 30], [20, 21]},
        {"Small", 2, [12, 22], [21, 24]},
        {"Small", 3, [10, 12], [24, 28]},
        {"Medium", 1, [45, 60], [40, 42]},
        {"Medium", 2, [25, 45], [42, 48]},
        {"Medium", 3, [20, 25], [48, 56]},
        {"Big", 1, [67, 90], [60, 63]},
```

```
        {"Big", 2, [37, 67], [63, 75]},
        {"Big", 3, [30, 37], [75, 84]},
        };

    constraint GS_NEN_Assemqrs_Assemplt_Assempc using (GS, NEN,
               Assem::qrs, Assem::plt, Assem::pc)
        table {
        {"Small", 1, 1, [22, 30], [20, 21]},
        {"Small", 1, 2, [12, 22], [21, 24]},
        {"Small", 1, 3, [10, 12], [24, 28]},
        {"Small", 2, 1, [45, 60], [40, 42]},
        {"Small", 2, 2, [25, 45], [42, 48]},
        {"Small", 2, 3, [20, 25], [48, 56]},
        {"Medium", 1, 1, [45, 60], [40, 42]},
        {"Medium", 1, 2, [25, 45], [42, 48]},
        {"Medium", 1, 3, [20, 25], [48, 56]},
        {"Medium", 2, 1, [90, 120], [80, 84]},
        {"Medium", 2, 2, [50, 90], [84, 96]},
        {"Medium", 2, 3, [40, 50], [96, 112]},
        {"Big", 1, 1, [67, 90], [60, 63]},
        {"Big", 1, 2, [37, 67], [63, 75]},
        {"Big", 1, 3, [30, 37], [75, 84]},
        {"Big", 2, 1, [135, 180], [120, 126]},
        {"Big", 2, 2, [75, 135], [126, 150]},
        {"Big", 2, 3, [60, 75], [150, 168]},
        };

  constraint ST_MontResqrs_MontResplt_MontRespc using (ST,
             Mont_Res::qrs, Mont_Res::plt, Mont_Res::pc)
        table {
        {100, 1, [15, 20], 10},
        {100, 2, [8, 15], [10, 12]},
        {100, 4, [5, 8], [12, 14]},
        {200, 1, [22, 33], 15},
        {200, 2, [11, 22], [15, 18]},
        {200, 4, [8, 11], [18, 21]},
        {300, 1, [30, 40], [20, 21]},
        {300, 2, [15, 30], [21, 25]},
        {300, 4, [10, 15], [25, 28]},
        {0, != 0, != 0,  0}
        };

      constraint F_Finishqrs_Finishplt_Finishpc using (F,
                 Finish::qrs, Finish::plt, Finish::pc)
        table {
        {"Standard", 1, [15, 20], 10},
        {"Standard", 2, [8, 15], [10, 12]},
        {"Standard", 4, [5, 8], [12, 14]},
        {"Confort", 1, [22, 30], 15},
        {"Confort", 2, [11, 22], [15, 18]},
        {"Confort", 4, [8, 11], [18, 21]},
        {"Luxe", 1, [30, 40], [20, 21]},
        {"Luxe", 2, [15, 30], [21, 25]},
        {"Luxe", 4, [10, 15], [25, 28]},
        {"Custom", 1, 60, 45},
        {"Custom", 2, 80, 40},
        {"Custom", 3, 100, 35},
        };
};

my $cost_constraints = qq{

   constraint compute_GC1 using (GC, DC, PC)
     condition {GC = DC + PC};

   constraint compute_GC2 using (GC, DC, PC)
     condition {GC - PC = DC};

   constraint compute_GC3 using (GC, DC, PC)
     condition {GC - DC = PC};
```

```
   constraint compute_DC1 using (DC, SNC, FC, STC, ENC)
     condition {DC = SNC + FC + STC + ENC};

   constraint compute_DC2 using (DC, SNC, FC, STC, ENC)
     condition {DC - FC - STC - ENC = SNC};

   constraint compute_DC3 using (DC, SNC, FC, STC, ENC)
     condition {DC - SNC - STC - ENC =  FC};

   constraint compute_DC4 using (DC, SNC, FC, STC, ENC)
     condition {DC - SNC - FC - ENC =  STC};

   constraint compute_DC5 using (DC, SNC, FC, STC, ENC)
     condition {DC - SNC - FC - STC =  ENC};

   constraint compute_PC1 using (PC, Fab::pc,
               Appro::pc, Assem::pc, Mont_Res::pc, Finish::pc)
     condition {PC = Fab::pc + Appro::pc + Assem::pc + Mont_Res::pc
                + Finish::pc};

   constraint compute_PC2 using (PC, Fab::pc, Appro::pc, Assem::pc,
              Mont_Res::pc, Finish::pc)
     condition {PC - Appro::pc - Assem::pc - Mont_Res::pc - Finish::pc
                = Fab::pc};

   constraint compute_PC3 using (PC, Fab::pc, Appro::pc, Assem::pc,
               Mont_Res::pc, Finish::pc)
     condition {PC - Fab::pc - Assem::pc - Mont_Res::pc - Finish::pc
                = Appro::pc};

   constraint compute_PC4 using (PC, Fab::pc, Appro::pc, Assem::pc,
              Mont_Res::pc, Finish::pc)
     condition {PC - Fab::pc - Appro::pc - Mont_Res::pc - Finish::pc
                = Assem::pc};

   constraint compute_PC5 using (PC, Fab::pc, Appro::pc, Assem::pc,
              Mont_Res::pc, Finish::pc)
     condition {PC - Fab::pc - Appro::pc - Assem::pc - Finish::pc
                = Mont_Res::pc};

   constraint compute_PC6 using (PC, Fab::pc, Appro::pc, Assem::pc,
              Mont_Res::pc, Finish::pc)
     condition {PC - Fab::pc - Appro::pc - Assem::pc - Mont_Res::pc
                = Finish::pc};

};
```

Liste des tableaux

Table des figures

Table des figures

Couplage de la configuration de produit et de projet de réalisation : exploitation des approches par contraintes et des algorithmes évolutionnaires

Dans le contexte actuel de compétitivité des marchés, la maîtrise et l'optimisation des processus de conception et de planification sont nécessaires pour garantir, d'une part la fiabilité et la qualité des produits systèmes ou services conçus et, d'autre part, le cycle de développement et les coûts. Ce constat impose de développer et d'améliorer les méthodes, modèles, techniques et outils relatifs aux processus de conception et de gestion ou de planification.

Les travaux présentés dans cette thèse s'inscrivent dans ce contexte et proposent de mettre en relation ou encore de faire intéragir la configuration de produit avec la planification du projet de réalisation. Le but de ces travaux est d'apporter une aide à la décision pour le couplage de la configuration de produit et de la planification du projet associé, en exploitant deux outils issus de l'Intelligence Artificielle : les approches par contraintes et les algorithmes évolutionnaires. Cette aide à la décision est présentée en deux parties. La première partie décrit l'utilisation des approches par contraintes afin de permettre au décideur de configurer son produit et son projet de réalisation de manière simultanée et interactive. Pour ce faire, les techniques de propagation et de filtrage des contraintes sont exploitées spécifiquement. La deuxième partie s'intéresse à l'exploitation des algorithmes évolutionnaires pour optimiser l'espace de solutions selon les critères coût et délai afin de présenter au décideur, un ensemble réduit de solutions optimisées. Un algorithme SPEA2 modifié en intégrant des méthodes de filtrage dans ses opérateurs de parcours de l'espace de recherche y est présenté. Toutes nos propositions sont illustrées sur un exemple d'avion de tourisme et d'affaire.

Mots-clés : configuration produit, planification projet, approches par contraintes,algorithmes évolutionnaires

Coupling product and project configuration : exploitation of constraints approches and evolutionnary algorithms

In the actual context of market the control and optimization of design processes are essential to ensure on the one hand, the reliability and quality of products, on the other hand the development time and costs. This phenomenon involves the constant development of methodologies, in order to improve the diversity and quality of the product and at the same time to shorten their development time and decrease their cost The work presented in this thesis fits into this context and propose to associate products configuration and production process planning. The aim of this work is to provide decision support for the coupling of configuration products and the associate production process leveraging two tools of Artificial Intelligence : constraints approaches and evolutionary algorithms.

This decision support is presented in two parts. The first part decribes the use of constraints approaches to allow decison-maker to configure product and its production process simultaneously and interactively. For this aim, propagation and filtring techniques are exploited specifically. The second part deals with the use of evolutionary algorithms to optimize the space solutions according to time and cost criteria in order to provide a small set of optimized solutions to the decision-maker. SPEA2 algorithm modified by incorporating filtering methods in its evolutionary operators. All our proposals ara illustrated on an exemple of light aitcraft.

Keywords : product configuration, projet planning, constraints approaches, evolutionary algorithms