Configuration et CSP avec variables à existence conditionnée

Thomas Van Oudenhove

Configuration et CSP avec variables à existence conditionnée

Thomas van Oudenhove*

Directeurs de thèse : Michel Aldanondo et Paul Gaborit

Laboratoire d'accueil :
Centre Génie Industriel
École des Mines d'Albi-Carmaux
Campus Jarlard - Route de Teillet
81013 Albi CT CEDEX 09

Établissement d'inscription :
Institut National Polytechnique de Toulouse
6 allée Emile Monso - ZAC du Palays
BP 34038
31029 Toulouse cedex 4

Résumé

Dans le cadre de la configuration de produit, il a été montré [8] que les contraintes, par le biais des CSP, était une approche adaptée à cette problématique. Cependant, le concept de modularité de la pertinence des variables (variables à existence conditionnée) est encore peu implémenté. Quelques solutions ont été proposées : DCSP [7, 12], CCSP [10] et CSPe [14] ; mais nous n'avons pas connaissance d'évaluation comparée de ces différentes approches. Notre communication présentera nos travaux d'analyse, en termes d'expressivité et de performances, de certaines de ces approches.

Mots-clés

configuration de produit — contraintes — CSP — résolution — filtrage

1 INTRODUCTION

Nous nous plaçons dans le cadre de la configuration de produit (cf. définition 2), pour laquelle nous retenons la définition suivante :

Définition 1 : *Configuration (inspirée de [13])*

Étant donné un modèle représentant le produit configurable, la configuration consiste à capturer de manière cohérente vis à vis du modèle, les souhaits de l'utilisateur pour aboutir à la définition d'un produit réalisable en terme d'une nomenclature de fonctions et/ou de produits.

Définition 2 : *Produit configurable (inspirée de [1])*

Un produit configurable peut être défini par un ensemble de variables correspondant aux composants du produit ou à ses propriétés. Chaque variable possède un domaine de définition (continu ou discret). Il existe des contraintes pouvant conditionner l'existence de variables et/ou restreindre leur domaine de définition. Configurer vise alors à définir les variables existantes et à leur assigner une valeur, tout en respectant les contraintes.

Dans ce cadre, [8] ont montré que les CSP (introduits par [9]) constituaient une approche adaptée aux problématiques de configuration, que nous pouvons différencier en deux types : autonome (l'utilisateur donne ses souhaits, puis on demande au programme de fournir

*vanouden@enstimac.fr

une ou toutes les solutions valides) et interactive (l'utilisateur ajoute des contraintes, visant généralement à réduire des domaines de variables, et le programme fournit les choix restant cohérents). Ces deux types de configuration font appel aux techniques de résolution ou de filtrage des CSP. Pour les techniques de résolution, nous pouvons citer le *Generate & Test* et le *BackTrack*. D'autres techniques de résolution utilisent une partie de filtrage (généralement basée sur l'Arc-Cohérence) ; [2] en propose une bonne synthèse. Quant aux techniques de filtrage, [3] proposent une classification des algorithmes selon leur puissance de filtrage et leur rapidité. En configuration, les principaux algorithmes utilisés sont AC-3 [6] pour le filtrage et *BackTrack*, éventuellement accompagné de AC-3 pour la résolution.

Cependant, beaucoup de produits configurables peuvent être équipés d'options : un composant qui peut être ou ne pas être présent dans certaines configurations. Ainsi, un besoin, non couvert *a priori* par les CSP, est la prise en compte de l'existence de variables et/ou de contraintes selon les besoins de l'utilisateur. Quelques communications ont proposé des solutions. Nous pouvons citer les DCSP (Dynamic CSP) de [7], qui permettent l'activation et/ou la désactivation de variables (concept étendu par les définitions de [12]). Les travaux de [5] proposent d'utiliser une décomposition en CSP standard à chaque nouvelle apparition ou disparition de variables ou de contraintes. Les *Composite* CSP proposés par [10] permettent la gestion de groupes de variables. Les CSP à états [14] associent à chacune des variables un attribut gérant leur existence.

Nous pouvons illustrer le problème des variables conditionnelles par l'exemple 1.

Exemple 1: Voiture simple

Considérons une voiture (luxe ou normale) équipée de vitres (manuelles ou électriques), d'un intérieur (cuir ou tissu) et d'un toit ouvrant (manuel ou électrique) disponible seulement sur la version « luxe ». Des vitres manuelles impliquent un toit ouvrant manuel et un intérieur cuir implique un toit ouvrant électrique. Nous pouvons donc poser ce problème comme un CSP avec la variable ToitOuvrant dont l'existence est conditionnée.

- les variables : Voiture, Vitres, Intérieur, ToitOuvrant (optionnel) ;
- les domaines de ces variables : {luxe, normale}, {manuelles, électriques}, {manuel, électrique}, {cuir, tissu} ;
- les contraintes entre les variables (cf. tableau 1 et figure 1) doivent correspondre à la liste des solutions suivantes :

TAB. 1 – Liste des solutions de l'exemple « voiture simple »

Voiture	luxe	luxe	luxe	luxe	normale	normale	normale	normale
Vitres	manuelles	électriques	électriques	électriques	manuelles	manuelles	électriques	électriques
Intérieur	tissu	tissu	tissu	cuir	tissu	cuir	tissu	cuir
ToitOuvrant	manuel	manuel	électrique	électrique	/	/	/	/

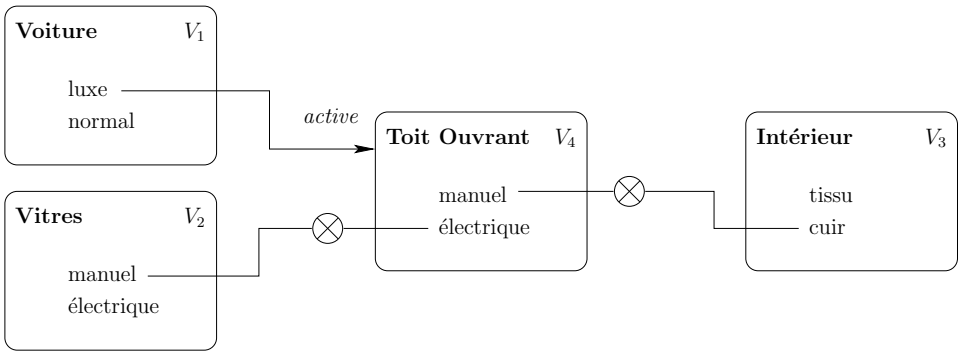

FIG. 1 – Illustration de l'exemple « voiture simple » avec une variable optionnelle

Le schéma de la figure 1 illustre les contraintes. La flèche représente une contrainte d'existence conditionnée : la variable ToitOuvrant est active si et seulement si la variable Voiture est valuée à luxe. Les autres contraintes sont représentées sous forme d'incompatibilités.

Dans ce cadre, nos travaux visent à évaluer l'interêt des CSPe [14, 13] en les confrontant aux approches concurrentes. Dans ce but, notre communication propose une démarche visant à construire une implémentation de référence. Nous proposons tout d'abord les définitions liées aux CSPe. L'implémentation de référence est ensuite abordéee. L'exploitation d'informations spécifiques aux CSPe permet de proposer des heuristiques adaptées, ainsi qu'une première modification de l'algorithme *BackTrack*. Ces propositions sont enfin évaluées par rapport à notre première implémentation.

2 DÉFINITION D'UN FORMALISME ET CRITÈRES D'ÉVALUATION

2.1 DÉFINITIONS

Nous retenons la définition classique d'un CSP :

Définition 3 : *CSP*

*Un CSP (notion introduite par [9]) est considéré comme un triplet (**X**,**D**,**C**) où :*
- ***X** est un ensemble de variables ;*
- ***D** est l'ensemble de leurs domaines de définition ;*
- ***C** un ensemble de contraintes sur les variables.*

Les contraintes définissent les relations entre les variables en limitant les combinaisons de leurs valeurs.

Une solution d'un CSP est ainsi définie de la manière suivante :

Définition 4 : *Solution d'un CSP*

Une solution d'un CSP est une instanciation de toutes les variables telle que toutes les contraintes sont vérifiées.

De plus, nous définissons ainsi la résolution et le filtrage d'un CSP :

Définition 5 : *Résolution d'un CSP*

La résolution d'un CSP consiste à trouver une ou toutes les solutions possibles.

Définition 6 : *Filtrage d'un CSP*

Le filtrage consiste à ajouter des contraintes (valuation de variables par exemple), puis à réduire les domaines de définition des variables en tentant d'y supprimer les valeurs qui n'interviennent dans aucune solution.

Plusieurs types de filtrage existent (cohérence d'arc, chemin-cohérence,...). Chacun est caractérisé par un compromis entre force de filtrage et rapidité d'exécution (cf. [4]). Ce compromis implique que le filtrage n'est généralement pas parfait. Des valeurs incohérentes peuvent donc ne pas être détectées.

Nous adopterons la définition suivante pour les CSPe (issue de [13]) :

Définition 7 : *CSPe*

*Un CSPe est donc un triplet (**V**,**A**,**F**) tel que :*
- ***V** est un ensemble de variables, chacune associée à un attribut d'état (celui-ci est « actif » ou « inactif ») ;*
- ***A** est l'ensemble des domaines de définition des variables ;*
- ***F** est un ensemble de contraintes sur les variables (cf. définition 8).*

Pour bien mesurer les différences entre le concept de CSP et celui de CSPe, nous adoptons cette définition des contraintes dans un CSPe :

Définition 8 : *Contrainte dans un CSPe*

Une contrainte est une condition logique sur les valeurs ou états d'un ensemble de variables, notée $(V_i, V_j, ...)$.

Une contrainte est donc une formule logique $F(V_i, \ldots, V_j)$*, avec, pour chaque variable impliquée dans la formule, une condition sur la valeur de cette variable ou sur son attribut d'état. En notant* S_k *une condition sur l'état et* W_i *une condition sur la valeur de la variable elle-même,* F *s'écrit* $F(W_i, \ldots, W_j, S_k, \ldots, S_l)$.

Postulat 1 (issu de la définition 8) : *Nous faisons le choix (inspiré de [7]) de prendre en compte une contrainte si et seulement si toutes les variables sur lesquelles elle porte sont actives. Dans tous les autres cas, la contrainte est considérée comme satisfaite.*

Il est d'ores et déjà possible d'établir une classification des contraintes : nous pouvons différencier les contraintes portant uniquement sur des attributs d'état de variables F^s des contraintes portant uniquement sur les valeurs des variables F^b et des contraintes mixtes F^{sb}.

Toutes ces définitions étant posées, nous pouvons donc réécrire formellement l'exemple 1 :

- les variables : $\boldsymbol{V}$: {Voiture (V_1), Vitres (V_2), Intérieur (V_3), ToitOuvrant (V_4)} ;
- les domaines : $\boldsymbol{A}$: {{luxe, normale}, {manuelles, électriques}, {tissu, cuir}, {manuel, électrique}} ;
- les contraintes : $\boldsymbol{C}$:

$$\begin{array}{ll} P_1 : & \forall i \in \{1,2,3\}, V_i \textit{ active} \\ P_2 : & (V_2 = \text{ manuel}) \Rightarrow (V_4 = \text{ manuel}) \\ P_3 : & (V_3 = \text{ cuir}) \Rightarrow (V_4 = \text{ électrique}) \\ P_4 : & (V_1 = \text{ luxe}) \Rightarrow (V_4 \textit{ active}) \end{array} \quad (1)$$

Définition 9 : *Solution d'un CSPe*

Une solution d'un CSPe *est une instanciation de tous les attributs d'état et de toutes les variables dont l'attribut d'état a pour valeur « actif » telle que toutes les contraintes à prendre en compte sont satisfaites.*

Les définitions de résolution et filtrage restent les mêmes que pour les CSP.

2.2 CRITÈRES D'ÉVALUATION

Pour évaluer nos propositions en résolution (algorithme *BackTrack*), nous nous baserons sur les critères suivants :

- nombre de solutions atteintes ;
- nombre de *BackTracks* ;
- nombre de nœuds parcourus ;
- nombre de vérifications de contraintes.

Le critère le plus intéressant est le nombre d'évaluation de contraintes. En effet, cette évaluation est l'opération la plus coûteuse. Nos propositions seront d'autant plus efficaces qu'elles permettront une réduction significative du nombre d'évaluations de contraintes. Nos propositions pour améliorer la résolution de CSPe (par traduction en CSP) font l'objet de la section 4.

3 PROPOSITION D'IMPLÉMENTATION

3.1 DÉMARCHE PROPOSÉE

Notre objectif est d'évaluer les performances comparées des CSPe et des CSP. Cette communication se restreint aux techniques de résolution. Nos travaux futurs porteront sur le filtrage

des CSPe. Au niveau de l'expressivité, un CSPe apporte plus d'informations qu'un CSP, dans la mesure où chaque variable est munie d'un attribut d'état représentant sa pertinence dans le problème en cours.

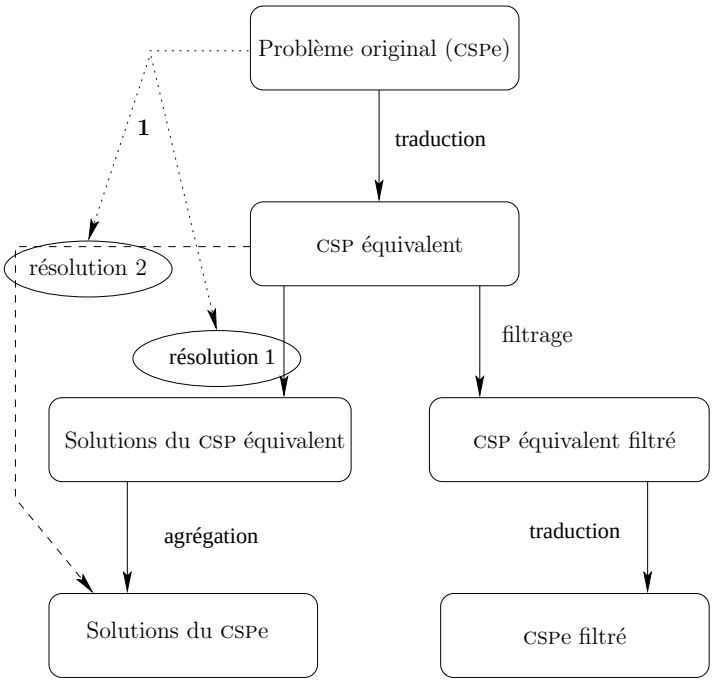

FIG. 2 – Schématisation de l'approche par traductions

Cette démarche consiste à générer un CSP équivalent au CSPe. Ce CSP est ensuite résolu par des algorithmes CSP classiques (*Back-Track*,...) : « résolution 1 ». Les solutions de ce CSP peuvent alors être agrégées pour donner les solutions du CSPe. Pour une résolution du problème, nous proposons deux types d'améliorations :

- la première (« résolution 2 ») consiste à modifier l'algorithme classique de résolution pour tenir compte du fait qu'une variable inactive n'a pas de valeur ;
- d'autres améliorations à base d'heuristiques (flèche **1**) complétent ce dispositif en jouant sur l'ordre des variables à instancier, profitant ainsi des avantages expressifs des CSPe.

3.2 LA TRADUCTION

La traduction d'un CSPe en CSP implique donc le passage d'un triplet ($\boldsymbol{V}$,$\boldsymbol{A}$,$\boldsymbol{F}$) à un autre triplet ($\boldsymbol{X}$,$\boldsymbol{D}$,$\boldsymbol{C}$). L'étape principale de cette traduction consiste à expliciter le CSPe de départ en termes de couples de variables d'état et variables de base. Il ne nous reste plus ensuite qu'à établir les équivalences de la définition 10 entre les deux triplets (($\boldsymbol{V}$,$\boldsymbol{A}$,$\boldsymbol{F}$) et ($\boldsymbol{X}$,$\boldsymbol{D}$,$\boldsymbol{C}$)).

Définition 10 : *CSP équivalent CSPe*

Le passage d'un CSPe *à un* CSP *équivalent se fait grâce aux opérations suivantes :*

$$\left\{\begin{array}{lcl} \boldsymbol{X} & = & \boldsymbol{X^s} \cup \boldsymbol{X^b} \\ & = & \{V_i\} \cup \{\acute{e}tat(V_i)\}, \forall i \\ & = & \textit{union de l'ensemble des variables de base et de l'ensemble des attributs d'états « transformés » en variables d'état} \\ \boldsymbol{D} & = & \boldsymbol{D}(\boldsymbol{X^b}) \cup \boldsymbol{D}(\boldsymbol{X^s}) \\ & = & \boldsymbol{A}(\boldsymbol{V}) \cup \{actif, inactif\} \\ & = & \textit{union des domaines des variables et des deux valeurs possibles pour l'état} \\ \boldsymbol{C} & = & \boldsymbol{C}^s \cup \boldsymbol{C}^b \cup \boldsymbol{C}^{sb} \\ & \sim & \boldsymbol{F}^s \cup \boldsymbol{F}^b \cup \boldsymbol{F}^{sb} \\ & = & \textit{union des contraintes sur les attributs d'états, sur les valeurs des variables et des contraintes mixtes} \end{array}\right. \quad (2)$$

Dans le cas d'un CSPe *traduit en* CSP *et en tenant compte du postulat 1, une contrainte se traduit ainsi :*

$$F(W_i,\ldots,W_j,S_k,\ldots,S_l) \rightarrow \begin{array}{ll} X_i^s = inactif & \vee \\ \ldots & \vee \\ X_j^s = inactif & \vee \\ f(X_i^b,\ldots,X_j^b,X_k^s,\ldots,X_l^s) & \end{array} \quad (3)$$

3.3 L'AGRÉGATION

Cette opération consiste, après une résolution, à transformer les solutions du CSP obtenues en solutions du CSPe initial. Chaque solution du CSPe est une classe d'équivalence d'au moins une solution du CSP. En effet, toutes les solutions du CSP qui ne diffèrent que par les valeurs des variables de base inactives représentent la même solution du CSPe.

4 AMÉLIORATIONS ET MESURE DE PERFORMANCES

Les résultats sont donnés avant l'étape d'agrégation ; il faudra donc prendre cette étape en compte si le nombre de solutions du CSP est supérieur à celui du CSPe. Chacun des tests est la moyenne des résultats obtenus sur 10 tests. L'ordre aléatoire d'instanciation des variables dans l'algorithme *BackTrack* sert de référence pour tester les différentes heuristiques.

4.1 AMÉLIORATIONS À BASE D'HEURISTIQUES

Les premières améliorations consistent à utiliser des heuristiques. Ces heuristiques peuvent être divisées en deux catégories : les heuristiques classiques applicables aux CSP (cf. [11]) et les heuristiques spécifiques aux CSPe. Toutes ces heuristiques jouent sur l'ordre d'instanciation des variables.

Sachant que le CSP est issu d'un CSPe, nous nous proposons d'étudier l'heuristique suivante (inspirée de [7, 13]) : instancier en premier toutes les variables d'état. Cette heurisitique sera ensuite désignée comme « heuristique 1 ».

Une heuristique classique des CSP (heuristique 2) classe les variables selon le nombre de contraintes l'utilisant et selon le cardinal de son domaine. On instancie en premier les variables référencées par le plus grand nombre de contraintes et, en cas d'égalité, on privilégie le plus petit domaine.

Le tableau 2 donne les résultats pour l'exemple 1 et pour l'exemple de la voiture publié dans [7, 13] (8 variables, 19 contraintes).

Critères	Solutions : CSP (CSPe)	*BackTracks*	Nœuds parcourus	Évaluations de contraintes
Exemple 1 : voiture simple				
Ordre aléatoire	12 (8)	110,2	244,4	198
Heuristique 1	12 (8)	21,4	64,8	50,4
Heuristique 2	12 (8)	32,8	87,6	62,4
Exemple de la voiture publié dans [7, 13]				
Ordre aléatoire	1 072 (450)	61 418,8	112 898,8	207 335,2
Heuristique 1	1 072 (450)	680,8	2 988,2	6 381
Heuristique 2	1 072 (450)	794,2	3 228,4	3 824,6

TAB. 2 – Comparaison des différentes heuristiques avec l'algorithme *BackTrack*

L'heuristique 1, par rapport à l'ordre aléatoire, permet des gains appréciables en terme d'évaluation de contraintes. De plus, l'heuristique 2 en est très proche, voire meilleure. Ce résultat qui peut sembler surprenant s'explique par le postulat 1. En effet, avec la traduction que nous adoptons de CSPe en CSP, toutes les variables d'état apparaissent nécessairement

dans au moins autant de contraintes que leurs variables de base associées. De plus, les domaines d'état ont deux valeurs (variables booléennes) alors que les cardinaux des domaines de base sont supérieurs ou égaux à 2. Le domaine de base est donc généralement plus grand. Par conséquent, la variable d'état sera dans la plupart des cas instanciée avant sa variable de base.

4.2 AMÉLIORATIONS PAR MODIFICATION DE L'ALGORITHME

Nous avons modifié l'algorithme de *BackTrack* pour tenir compte du fait que l'on cherche les solutions d'un CSPe. On ne teste alors aucune valeur pour les variables de base inactives. Lorsque l'algorithme doit valuer une variable de base, il vérifie si la variable d'état associée est valuée. Si celle-ci est inactive, on affecte une valeur arbitraire à la variable de base, et toutes les contraintes mettant en jeu cette variable sont trivialement satisfaites (cf. postulat 1). On ne parcourt donc pas tout le domaine de la variable de base.

Le tableau 3 montre les performances que nous obtenons pour l'exemple 1, puis l'exemple de la voiture [7, 13] avec l'algorithme *BackTrack*-CSPe.

Critères	Solutions : CSP (CSPe)	*BackTracks*	Nœuds parcourus	Évaluations de contraintes
Exemple 1 : voiture simple				
Ordre aléatoire	10 (8)	82	182	142
Heuristique 1	8 (8)	18	50	34,8
Heuristique 2	8 (8)	25,6	65,2	45,6
Exemple de la voiture publié dans [7, 13]				
Ordre aléatoire	733,6 (450)	9 061,2	18 264	24 474,8
Heuristique 1	450 (450)	404,4	1 432,4	3 904
Heuristique 2	450 (450)	331	1 338	1 541,8

TAB. 3 – Comparaison des différentes heuristiques avec l'algorithme *BackTrack*-CSPe

Nous pouvons constater que le nombre de solutions n'est plus obligatoirement constant. En effet, une variable de base est valuée si et seulement si sa variable d'état correspondante est inactive. Ainsi, lorsqu'une variable est inactive, il est possible d'avoir une seule solution si la variable d'état est valuée en premier. Si la variable de base est instanciée en premier, le nombre de solutions du CSP équivalentes pour le CSPe est proportionnel au cardinal du domaine de cette variable.

Le nombre de contraintes évaluées pour chacune des heuristiques est évidemment beaucoup plus faible avec l'algorithme *BackTrack*-CSPe.

D'après le tableau 3, nous pouvons constater que l'heuristique 2 peut être meilleure que l'heuristique 1 lors de l'utilisation de l'algorithme *BackTrack*-CSPe. Cet algorithme ne valuant que les variables non-inactives, il est logique que l'heuristique 1 proposée dans la section 4.1 puisse — selon les types de problèmes — apporter moins d'améliorations que l'heuristique 2.

5 CONCLUSION ET PERSPECTIVES

Dans le cadre de la configuration, nous essayons de répondre aux problématiques de variables « optionnelles », dont l'existence est conditionnée par les valeurs ou l'existence d'autres variables. Dans ce but, nous avons concentré nos travaux sur les propositions de [14]. Nous avons donc proposé un formalisme pour la résolution (puis le filtrage) adapté aux CSPe. Nous construisons ensuite une implémentation de référence pour la résolution, basée sur l'algorithme *BackTrack* légèrement modifié pour prendre en compte les spécificités des CSPe. Cette implémentation a été testée.

Le principal objectif à court terme est de mener des expérimentations similaires sur le filtrage. Nous comptons apporter des améliorations pour ces différents niveaux :

- la traduction de CSP filtré en CSPe filtré ;
- l'utilisation d'informations du CSPe pour limiter les opérations de filtrage du CSP, tout en garantissant une puissance de filtrage équivalente.

Les algorithmes pour mettre en œuvre ce filtrage sont en cours de développement.

Références

[1] Michel ALDANONDO, Hélène FARGIER et Mathieu VERON. « *Gestion de production : fonctions, techniques et outils* », Chapitre 7 — Configuration, configurateur et gestion de production. Hermes Science, Paris, 2001.

[2] Roman BARTÁK. « Constraint propagation », 1998.

[3] Romuald DEBRUYNE et Christian BESSIÈRE. « Some Practicable Filtering Techniques for the Constraint Satisfaction Problem ». *In IJCAI'97, International Joint Conference on Artificial Intelligence*, pages 412–417, Nagoya, Japon, 1997.

[4] Romuald DEBRUYNE et Christian BESSIÈRE. « Domain Filtering Consistencies ». *Journal of Artificial Intelligence Research*, 14 :205–230, mai 2001.

[5] Esther GELLE et Boi FALTINGS. « Solving Mixed and Conditional Constraint Satisfaction Problems ». *Constraints*, 8 :107–141, 2003.

[6] A.K. MACKWORTH et E.C. FREUDER. « The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems ». *Artificial intelligence*, 25 :65–74, 1985.

[7] Sanjay MITTAL et Brian FALKENHAINER. « Dynamic Constraint Satisfaction Problem ». *In AAAI'90, American Association for Artificial Intelligence*, Boston, Massachusetts, 1990.

[8] Sanjay MITTAL et Felix FRAYMAN. « Toward a Generic Model of Configuration Tasks ». *In IJCAI'89, International Joint Conference on Artificial Intelligence*, Detroit, Michigan, 1989.

[9] Ugo MONTANARI. « Networks of constraints : fundamental properties and application to picture processing ». *Information sciences*, 7 :95–132, 1974.

[10] Daniel SABIN et Eugene C. FREUDER. « Configuration as Composite Constraint Satisfaction ». *In Proceedings of the (1st) Artificial Intelligence and Manufacturing Research Planning Workshop*, 1996.

[11] M. SAGAUT, *coord.* « *La programmation par contraintes* », Chapitre 4 — Contraintes et Problèmes à Satisfaction de Contraintes. Agence pour la Diffusion de l'Information Technologique, Avril 1996.

[12] Timo SOININEN, Esther GELLE et Ilkka NIEMELÄ. « A Fixpoint Definition of Dynamic Constraint Satisfaction ». *In CP'99, Conference on Principles and practices of constraint programming*, pages 419–433, 1999.

[13] Mathieu VERON. « *Modélisation et résolution du problème de configuration industrielle : utilisation des techniques de satisfaction de contraintes* ». Thèse de doctorat, Institut National Polytechnique de Toulouse, ENI Tarbes, novembre 2001.

[14] Mathieu VERON et Michel ALDANONDO. « Yet Another Approach to CCSP for configuration problem ». *In ECAI'00, European Conference on Artificial Intelligence, Workshop on Configuration*, pages 59–62, Berlin, Allemagne, 2000.