Stratégies hybrides pour des décompositions optimales et efficaces

Philippe Jégou, Samba Ndojh Ndiaye, Cyril Terrioux

Stratégies hybrides pour des décompositions optimales et efficaces

Philippe Jégou [1] **Samba Ndojh Ndiaye** [2] **Cyril Terrioux** [1]

[1] LSIS - UMR CNRS 6168
Université Paul Cézanne (Aix-Marseille 3)
Avenue Escadrille Normandie-Niemen 13397 Marseille Cedex 20, France
[2] LIRIS - UMR CNRS 5205
Université Claude Bernard Lyon 1
43 boulevard du 11 novembre 1918 (Nautibus) 69622 Villeurbanne cedex, France
{philippe.jegou, cyril.terrioux}@univ-cezanne.fr samba-ndojh.ndiaye@liris.cnrs.fr

Résumé

Dans cet article, nous nous intéressons aux méthodes de résolution de réseaux de contraintes par des approches de décompositions structurelles. La *Hiérarchie des Contraintes Traitables* définie dans [9] propose un classement de ces méthodes par rapport à leur complexité temporelle théorique. Cependant, elle crée une très forte dépendance entre les algorithmes de calcul de décompositions et les techniques d'exploitation de ces dernières.

Nous faisons tomber ces cloisons par la définition de nouvelles méthodes basées sur des combinaisons d'approches Tree-Clustering [6], BTD [16], Tree-Decomposition [18] et Hypertree-Decomposition [9].

Nous démontrons leur intérêt d'un point de vue théorique grâce à des résultats portant sur leur complexité temporelle. En outre, cette analyse nous a permis d'enrichir la Hiérarchie des Contraintes Traitables. Finalement, nous justifions notre approche avec des résultats expérimentaux.

1 Introduction

Nous nous intéressons, dans ce papier, à l'efficacité théorique et pratique de méthodes de résolution de CSP exploitant les propriétés topologiques des réseaux de contraintes. Un CSP peut être vu comme un problème qui, étant donné un ensemble fini de variables X prenant leur valeur dans des domaines finis D, vérifie l'existence d'une affectation de ces variables satisfaisant simultanément un ensemble de contraintes C. Une telle affectation est une solution du CSP. De manière plus générale, le problème consiste à trouver une solution voire l'ensemble des solutions. Malheureusement, déterminer l'existence d'une solution est un problème NP-complet. Malgré tout, différentes classes d'algorithmes de résolution de CSP ont été proposées. D'une part, nous avons les algorithmes combinant la technique standard du backtracking et celle du filtrage des domaines tels que FC (Forward Checking [13]) ou MAC (Maintaining Arc Consistency [19]). Même si ces algorithmes sont relativement efficaces en pratique, leur complexité temporelle théorique est évaluée de manière classique par $O(S.m^n)$, avec S la taille du CSP considéré, n le nombre de variables et m la taille maximum des domaines des variables.

Différentes approches ont été proposées pour améliorer cette borne qui se révèle la pire possible. En particulier, les méthodes structurelles exploitent les propriétés topologiques qui existent fréquemment dans les problèmes réels. Généralement, elles reposent sur les propriétés d'une décomposition arborescente [18] ou d'une décomposition hyperarborescente [9] du réseau de contraintes qui capture sa structure et permet donc d'exprimer ses propriétés topologiques. Etant donnée une décomposition arborescente de largeur w, la complexité temporelle des meilleures approches structurelles, basées sur cette décomposition, est en $O(S.m^{w+1})$, avec la garantie d'avoir $w < n$ et dans beaucoup de cas $w \ll n$. De même, si on considère une décomposition hyperarborescente de largeur h, il est possible d'obtenir une complexité temporelle en $O(S.r^h)$, avec r la taille maximum des relations

(tables) associées aux contraintes. [9] a montré que la décomposition hyperarborescente est meilleure que la décomposition arborescente car $h \leq w$. En outre, les auteurs ont introduit un outil formel, la *Hiérarchie des Contraintes Traitables*, pour la comparaison de plusieurs méthodes structurelles. Cet outil théorique considère les classes d'instances de CSP pouvant être résolues en un temps polynomial. Il apparaît dès lors que la méthode structurelle basée sur la décomposition hyperarborescente (notée MHD) [9] est plus performante que le Tree-Clustering basé sur la décomposition arborescente (noté TC) [6] puisque la classe des instances traitables par MHD contient strictement celles traitables par TC.

D'un point de vue théorique, cette hiérarchie est un outil d'un très grand intérêt dans la mesure où elle permet de se reposer sur des critères fiables pour choisir la meilleure approche pour la résolution d'une instance donnée. Cependant, nous savons qu'il peut parfois exister une grande différence entre les performances théoriques et pratiques d'une même méthode. De ce fait, nous allons étendre notre étude de cette hiérarchie en y intégrant des aspects liés au comportement et à l'utilisation pratiques de ces méthodes. Nous pouvons déjà dire que les méthodes de résolution basées sur une décomposition ne sont généralement pas utilisées comme cela est supposé dans la hiérarchie. En effet, cette dernière ne considère que des décompositions optimales. Or, le calcul d'une décomposition optimale est souvent inaccessible car il nécessite un temps qui peut parfois dépasser largement le temps nécessaire à la résolution du problème sans décomposition. Ainsi, en pratique, on préfère généralement utiliser un algorithme heuristique (approximation des valeurs optimales de w ou h) pour trouver une décomposition. Une fois celle-ci obtenue, l'étape suivante consiste à résoudre le CSP décomposé. Dans la hiérarchie, il existe un lien rigide entre le type de la décomposition et la méthode qui exploite celle-ci. Dans cet article, nous faisons tomber ces barrières en créant une séparation entre la méthode de décomposition graphique et la méthode exploitant celle-ci pour résoudre le problème. Pour cela, nous définissons et étudions de nouvelles combinaisons d'approches dans lesquelles un type de décomposition (par exemple une décomposition hyperarborescente) peut être utilisé pour la résolution par une méthode initialement définie pour un autre type de décomposition (par exemple TC).

Ensuite, nous montrons que ce type de méthodes hybrides a un réel intérêt d'un point de vue théorique. En effet, nous obtenons des résultats de complexité originaux permettant d'enrichir la hiérarchie de [9]. Enfin, nous montrons de manière empirique, sur des benchmarks de la dernière compétition CSP (http ://www.cril.univ-artois.fr/CPAI08/), que notre approche semble bien adaptée pour comparer des méthodes structurelles par rapport à leur capacité à résoudre des CSP.

La section 2 rappelle des notations ainsi que des résultats sur la complexité de méthodes énumératives et structurelles. La section 3 introduit et étudie de nouvelles approches qui combinent différents types de décompositions et de méthodes d'exploitation, puis propose de nouvelles bornes de complexité qui nous permettent d'étendre la *Hiérarchie des Contraintes Traitables*. La section 4 présente une comparaison pratique des méthodes ainsi définies et la section 5 donne une conclusion.

2 Rappels

2.1 Notations

Un *problème de satisfaction de contraintes fini* ou *réseau de contraintes fini* (X, D, C, R) est défini par un ensemble fini de variables $X = \{x_1, \dots x_n\}$, un ensemble de domaines finis $D = \{d_1, \dots d_n\}$ (le domaine d_i contient toutes les valeurs possibles pour la variable x_i), et un ensemble C de contraintes sur les variables. Une contrainte $c_i \in C$ sur un sous-ensemble ordonné de variables, $c_i = (x_{i_1}, \dots x_{i_{a_i}})$ (a_i est appelé l'*arité* de la contrainte c_i), est définie par une relation associée $r_i \in R$ de combinaisons autorisées de valeurs des variables de c_i. Nous allons utiliser la même notation pour la contrainte c_i et l'ensemble des variables sur lesquelles elle porte. Nous notons a l'arité maximale des contraintes de C. Sans perte de généralité, nous supposons que chaque variable est contenue dans au moins une contrainte. Une solution de (X, D, C, R) est une affectation de toutes les variables qui satisfait toutes les contraintes. La structure d'un CSP peut être représentée par l'hypergraphe (X, C), appelé l'*hypergraphe de contraintes*.

Dans cet article (de même que dans [9]), nous supposons que les relations ne sont pas vides et peuvent être représentées par des tables comme dans la théorie des bases de données relationnelles. Ensuite, nous notons S la taille d'un CSP (qui vérifie $S \leq n.m + a.r.|C|$ avec $r = max\{|r_i| : r_i \in R\}$). Soient $Y = \{x_1, \dots x_k\}$ un sous-ensemble de X et $\mathcal{A}$ une affectation sur Y. $\mathcal{A}$ peut être vue comme un tuple $\mathcal{A} = (v_1, \dots v_k)$. La *projection* de $\mathcal{A}$ sur un sous-ensemble Y' de Y, notée $\mathcal{A}[Y']$, est la restriction de $\mathcal{A}$ aux variables de Y'. La projection de la relation r_i sur le sous-ensemble Y' de c_i est l'ensemble des tuples $r_i[Y'] = \{t[Y'] | t \in r_i\}$. La jointure des relations sera notée $\bowtie$. La jointure de $\mathcal{A}$ avec la relation r_i est $\mathcal{A} \bowtie r_i = \{t | t \text{ est un tuple sur } Y \cup c_i \text{ et } t[Y] = \mathcal{A} \text{ et } t[c_i] \in r_i\}$.

2.2 Complexité de l'énumération

L'approche basique pour la résolution de CSP repose sur la procédure classique appelée *Backtracking (BT)*. La complexité temporelle de cet algorithme de base est en $O(a.r.|C|.m^n)$ car le nombre de nœuds potentiels développés durant la recherche est m^n et en supposant qu'un test de contrainte $\mathcal{A}[c_i] \in r_i$ est réalisé en $O(a.r)$. Pour simplifier les notations, elle sera exprimée en $O(S.m^n)$. Généralement, cet algorithme n'est jamais utilisé car il est clairement inefficace en pratique. L'approche la plus classique pour améliorer BT est basée sur le filtrage. Le premier algorithme proposé pour ce filtrage est le Forward Checking (FC [13]). Il était initialement défini sur les CSPs binaires. De nombreuses extensions et généralisations de FC ont été proposées pour résoudre des CSPs non binaires ou pour exploiter des techniques de filtrage plus puissantes [13, 19, 2]. Une de ces extensions appelée nFC2 [2] comporte un niveau de filtrage qui semble être un bon compromis entre le coût de cette opération et le bénéfice qu'elle apporte dans la simplification de la résolution. La complexité de nFC2 est également bornée par $O(S.m^n)$ comme cela est indiqué dans [2]. Récemment, [15] a proposé une nouvelle borne qui considère r la taille maximum des relations (tables) associées aux contraintes. Il a été démontré que la complexité temporelle de nFC2 est en $O(S.r^k)$, où k est la taille de $|C'|$, un recouvrement minimum de X. Pour rappel, on dit que C' est un recouvrement minimum de X si C' est recouvrement de X ($C' \subset C$ et $\cup_{c_i \in C'} = X$) et il n'existe pas de recouvrement C'' tel que $|C''| < |C'|$. Ce résultat peut être étendu à tout autre algorithme qui maintient un niveau de filtrage au moins aussi puissant que celui de nFC2. Par exemple, il est valable pour nFCi ($i \geq 2$) et MAC ([19]).

2.3 Méthodes de résolution basées décomposition

La décomposition de réseau de contraintes a été introduite dans [6] avec le Tree-Clustering (TC). TC et d'autres méthodes basées sur cette approche (voir [3]) reposent sur la notion de décomposition arborescente de graphes. Néanmoins, étant donné un CSP n-aire, et de ce fait un hypergraphe de contraintes, nous pouvons continuer à exploiter cette notion grâce au *graphe primal* de ce dernier. Soit $H = (X, C)$ un hypergraphe, le graphe primal de H est le graphe $G = (X, A_C)$ avec $A_C = \{\{x, y\} \subset X : \exists c_i \in C \ t.q. \ \{x, y\} \subset c_i\}$. Ainsi, étant donné un CSP, nous considérons son graphe primal pour définir une décomposition arborescente associée à ce CSP.

Définition 1 *Une décomposition arborescente d'un graphe $G = (X, A_C)$ est une paire (E, T) où $T = (I, F)$ est un arbre avec un ensemble de nœuds I et d'arêtes F, et $E = \{E_i : i \in I\}$ une famille de sous-ensembles de X, telle que chaque sous-ensemble (appelé cluster) E_i est un nœud de T et vérifie :*
(i) $\cup_{i \in I} E_i = X$,
(ii) $\forall \{x, y\} \in A_C$, $\exists i \in I$ avec $\{x, y\} \subset E_i$, et
(iii) $\forall i, j, k \in I$, si k est sur le chemin entre i et j dans T, alors $E_i \cap E_j \subset E_k$.

La largeur w d'une décomposition arborescente (E, T) est égale à $max_{i \in I}|E_i| - 1$. La largeur arborescente w^* de G est la largeur minimale sur toutes ses décompositions arborescentes.

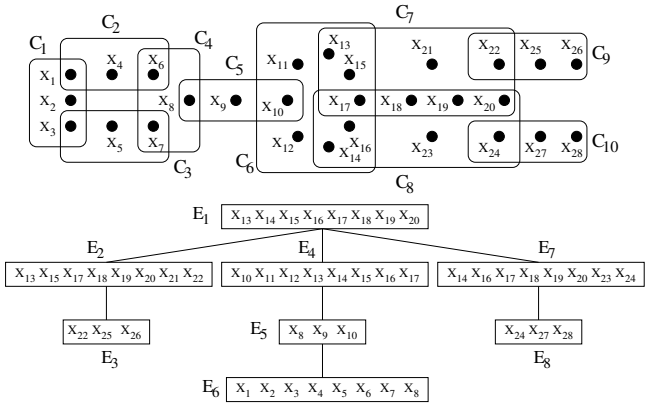

FIG. 1 – Un hypergraphe et une décomposition arborescente optimale.

La figure 1 présente un hypergraphe de contraintes et une décomposition arborescente possible de cet hypergraphe dont la largeur est minimum ($w^* = 7$).

Dans [6], le Tree-Clustering (noté ici TC-1989) a été défini initialement en utilisant un algorithme de complexité temporelle polynomiale (MCS [20]) pour calculer la décomposition arborescente. Plus précisément, la méthode se résume ainsi :

1. Calculer une décomposition arborescente du réseau de contraintes
   (a) Trianguler le graphe primal en utilisant MCS
   (b) Identifier les clusters de variables (les cliques maximales)
   (c) Construire un arbre de clusters (arbre de jointures)
2. Résoudre les sous-problèmes définis par les clusters de variables
3. Résoudre le problème acyclique résultant

La complexité de la première étape est limitée à $O(n^2)$. Cependant, nous n'avons aucune garantie sur l'optimalité du paramètre w. La complexité de la deuxième étape est en $O(S.m^{w+1})$. Ainsi, étant donné

un réseau de contraintes, plus la largeur de la décomposition est petite, plus cette complexité temporelle est petite. Or, la largeur de la décomposition arborescente calculée grâce à MCS peut être très éloignée de l'optimum. En outre, trouver une décomposition arborescente optimale est un problème NP-difficile. Nous faisons remarquer que dans [4] où TC-1989 est appelé Join Tree-Clustering, chaque sous-problème est défini par l'ensemble des variables du cluster et les contraintes dont les variables sont incluses dans ce dernier. Ce qui assure une complexité en $O(|C|.w.log(m).m^{w+1})$ pour la troisième étape. Finalement, le coût total de TC-1989 est en $O(S.m^{w+1})$.

Quant à la complexité en espace, celle-ci est liée à la sauvegarde des solutions des différents sous-problèmes qui vont conduire au nouveau problème acyclique. Elle est en $O(n.a.m^{w+1})$. Cependant, dans [4], Dechter suggère de mémoriser uniquement une partie des solutions, en fait leur projection sur les intersections entre les clusters. De ce fait, la complexité en espace est limitée à $O(n.a.m^s)$, où s est la taille maximum des intersections entre les clusters.

Défini initialement sur les CSPs binaires, TC-1989 a été étendu depuis lors aux CSPs n-aires de différentes manières [4]. Par ailleurs des améliorations de cette méthode ont été proposées pour remédier à certains de ses inconvénients (un espace mémoire requis trop important, résolution complète des sous-problèmes avant le test de la consistance globale du CSP, valeur de w qui peut être assez éloignée de l'optimum). Cependant, malgré ces améliorations, cette méthode demeure peu efficace en pratique. Par exemple, BTD [16] peut être considérée comme une approche efficace d'exploitation de décompositions arborescentes. Cette efficacité pratique vient du fait que BTD utilise un algorithme énumératif guidé par un ordre sur les variables induit par la décomposition arborescente. Cette approche évite généralement la résolution complète de tous les clusters. Comme pour TC-1989, la complexité temporelle de BTD est en $O(S.m^{w+1})$ alors que celle en espace est en $O(n.a.m^s)$. En outre, BTD considère une décomposition arborescente donnée qui peut être obtenue grâce à des algorithmes heuristiques ou exacts.

Alors que, TC-1989 peut être vue comme une méthode orientée variables puisque les clusters sont définis par des ensembles de variables, [9] propose une nouvelle méthode qui considère des clusters de contraintes. Cette approche est basée sur la notion de décomposition hyperarborescente qui est une généralisation de la décomposition arborescente.

Définition 2 *Etant donné un hypergraphe $H = (X, C)$, un hyperarbre de H est un triplet (T, χ, λ) où $T = (N, F)$ est arbre enraciné, et χ et λ sont des fonctions qui associent à chaque sommet $p \in N$ deux ensembles $\chi(p) \subset X$ et $\lambda(p) \subset C$. Si $T' = (N', F')$ est un sous-arbre de T, nous définissons $\chi(T') = \cup_{v \in N'} \chi(v)$. On note $sommets(T)$, l'ensemble N des sommets de T, et $racine(T)$ la racine de T. En outre, quelque soit $p \in N$, T_p est le sous-arbre de T enraciné en p.*

Définition 3 *Une décomposition hyperarborescente de H est un hyperarbre $HD = (T, \chi, \lambda)$ de H qui satisfait les conditions suivantes :*
(i) $\forall c \in C, \exists p \in sommets(T)$ t.q. $c \subset \chi(p)$,
(ii) $\forall x \in X$, l'ensemble $\{p \in sommets(T) : x \in \chi(p)\}$ induit un sous-arbre (connexe) de T,
(iii) $\forall p \in sommets(T), \chi(p) \subset \cup_{c \in \lambda(p)} c$,
(iv) $\forall p \in sommets(T), \cup_{c \in \lambda(p)} c \cap \chi(T_p) \subset \chi(p)$.

Une arête $c \in C$ est fortement couverte dans HD s'il existe $p \in sommets(T)$ tel que $c \subset \chi(p)$ et $c \in \lambda(p)$. Une décomposition hyperarborescente HD est une décomposition complète de H si chaque arête de H est fortement couverte dans HD.

La largeur h d'une décomposition hyperarborescente $HD = (T, \chi, \lambda)$ est $max_{p \in sommets(T)} |\lambda(p)|$. La largeur hyperarborescente h^* de H est la largeur minimum sur toutes ses décompositions hyperarborescentes.

Nous remarquons que les hypergraphes acycliques sont précisément ceux ayant une largeur hyperarborescente égale à 1. Pour la résolution de CSP, nous considérons uniquement des décompositions hyperarborescentes complètes. La figure 2 présente une décomposition hyperarborescente complète de l'hypergraphe de la figure 1.

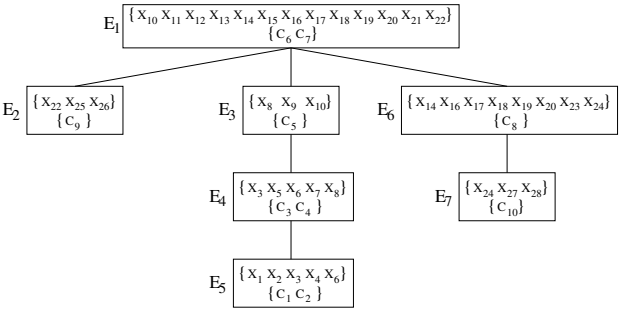

FIG. 2 – Une décomposition hyperarborescente optimale ($h^* = 2$).

La méthode basée sur cette notion de décomposition hyperarborescente (notée ici MHD-1999) a été également proposée dans [9]. Elle procède de la façon suivante :

1. Calculer une décomposition hyperarborescente du CSP
2. Résoudre chaque cluster par une jointure des relations associées aux contraintes du cluster
3. Résoudre le problème acyclique résultant

[9] présente une évaluation de la complexité, en supposant que la première étape peut être réalisée en $O(|C|^{2h^*}.n^2)$ grâce à l'algorithme *opt-k-decomp* défini dans [8]. Pour cela, l'hypergraphe considéré doit avoir une largeur hyperarborescente bornée par une constante. L'algorithme *opt-k-decomp* est donc capable de calculer une décomposition hyperarborescente optimale en un temps polynomial pour tout hypergraphe dont la largeur hyperarborescente est bornée par une constante. Le coût de la résolution des clusters dans la deuxième étape est borné par $O(S.r^h)$. La complexité en espace, en $O(r^h)$, est liée à la taille des relations résultant des jointures au sein des clusters. L'intérêt pratique de MHD-1999 n'a pas été clairement établi jusqu'à présent. Cela est probablement dû au manque d'algorithmes efficaces pour calculer une décomposition hyperarborescente, mais également à l'espace mémoire ($O(n.r^h)$) nécessaire dans la deuxième étape. Néanmoins, d'un point de vue théorique, cette méthode est clairement d'une très grande importance comme cela est démontré dans la *Hiérarchie des Contraintes Traitables* ([9]) (voir la figure 3). Cette hiérarchie donne une comparaison théorique des méthodes les plus connues pour la résolution de CSPs par décomposition d'(hyper)graphes de contraintes. Elle considère uniquement les méthodes D qui procèdent comme suit :

1. Reconnaître en un temps polynomial un CSP P traitable (en vérifiant si sa largeur notée D-*width* par rapport à la décomposition utilisée par D est bornée par une constante)
2. Calculer en un temps polynomial une décomposition de P dont la largeur est inférieure à cette constante
3. Transformer en un temps polynomial P en un CSP acyclique équivalent P'.
4. Résoudre P' en un temps polynomial (en $S^{D-width}$ puisque D-*width* est bornée par une constante).

Ainsi, si on considère MHD-1999, D-*width* $= h^*$. Tout CSP dont l'hypergraphe de contraintes a une largeur hyperarborescente bornée par une constante est traitable (peut être résolu en un temps polynomial) par MHD-1999.

Formellement, soient D_1 et D_2 deux méthodes de résolution basées décomposition. On dit que D_2 *généralise fortement* D_1 (représenté par un arc (D_1,D_2) dans la figure 3) si D_2 *généralise* D_1 (chaque problème traitable par D_1 est également traitable par D_2) et que D_2 *bat* D_1 (il existe une classe de problèmes traitables par D_2 et pas par D_1).

Alors que MHD-1999 est au sommet de la hiérarchie, il faut noter que formellement, TC-1989 ne devrait pas y figurer car cette méthode ne garantit pas l'utilisation d'une décomposition dont la largeur est bornée par la constante de la première étape. En effet, la largeur de la décomposition arborescente calculée par MCS peut être très éloignée de l'optimum et ainsi dépasser cette constante. Par ailleurs, il existe d'autres méthodes de résolution basées décomposition meilleures que MHD (en termes de complexité de résolution du CSP décomposé) : celle basée sur la décomposition hyperarborescente généralisée (MGHD [10]) et celle basée sur la décomposition hyperarborescente fractionnaire (MFHD [12]). Malheureusement ces approches ne peuvent figurer dans la hiérarchie car il n'existe pas d'algorithmes polynomiaux pour la reconnaissance des CSPs traitables dans ce cas, et pour le calcul des décompositions correspondantes. Ainsi, MHD demeure au sommet de la hiérarchie. Enfin, [7] a récemment proposé des outils algorithmiques intéressants pour calculer efficacement de bonnes approximations de décompositions hyperarborescentes optimales.

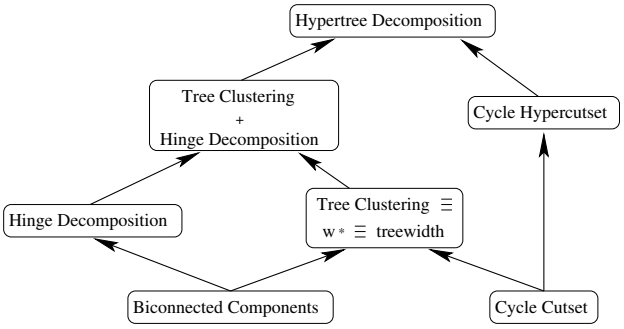

FIG. 3 – La hiérarchie des Contraintes Traitables.

3 De nouvelles méthodes pour résoudre des CSPs décomposés

3.1 Résolution VS Décompositions graphiques

En pratique, les méthodes de résolution basées décomposition ne sont pas véritablement utilisées comme cela est supposé dans la hiérarchie. Généralement, les deux premières étapes sont fusionnées, le calcul d'une décomposition dont la largeur est bornée par une constante permettant de conclure que la largeur du problème P est bornée par cette même constante. En plus, cette décomposition est calculée grâce à un algorithme heuristique à l'image de TC-1989 et non par un algorithme exact trop coûteux. En effet, de ce point de vue pratique, les valeurs optimales de w et h ne sont pas nécessaires voire même totalement inutiles [14]. Ensuite, les étapes 3 et 4 peuvent être considérées séparément comme dans TC-1989 et MHD-1999, ou fusionnées dans une unique étape à

l'image de BTD. Ainsi, dans cet article, nous considérons qu'une méthode de résolution basée décomposition notée DM_{DEC} a pour entrée un CSP P et une décomposition graphique (notée DEC) de P. Dès lors, la méthode DM résout P en utilisant la décomposition graphique considérée DEC. Par exemple, TC_{TD} est l'application de TC au CSP P en considérant une décomposition arborescente optimale tandis que TC_{MCS-TD} travaille avec une décomposition arborescente calculée par MCS pour résoudre P avec TC. De ce fait, TC_{MCS-TD} correspond à TC-1989 alors que TC_{TD} est le TC référencé dans la hiérarchie en considérant une décomposition arborescente optimale. Enfin, MHD_{HD} correspond à MHD-1999. Dans cet article, nous montrons l'intérêt de ce type d'approche par l'introduction de versions hybrides de méthodes $DM \in \{TC, MHD, BTD\}$ et de décompositions graphiques de type décomposition arborescente (TD) ou hyperarborescente (HD), optimales ou heuristiques.

Nous allons maintenant montrer comment des méthodes comme TC ou BTD, basées initialement sur des décompositions arborescentes, peuvent être étendues pour utiliser des décompositions hyperarborescentes, et inversement comment MHD peut être restreintes aux décompositions arborescentes.

3.2 De HD à TD

[15] donne une méthode pour transformer une décomposition hyperarborescente en une arborescente.

En effet, étant donnés un CSP $\mathcal{P} = (X, D, C, R)$ et $HD = (T, \chi, \lambda)$ une décomposition hyperarborescente de $H = (X, C)$ de largeur h, (T, χ) vérifie toutes les conditions requises pour être une décomposition arborescente à l'exception du fait qu'il peut exister un cluster $\chi(p)$ qui soit contenu dans un autre $\chi(p')$, avec $p, p' \in sommets(T)$. La définition suivante permet de construire cette décomposition arborescente grâce à la notion d'arbre de jointures d'un hypergraphe H [4] qui est un arbre connexe dont les nœuds sont les hyperarêtes de H telles que si un sommet x est dans deux nœuds alors x est dans tous les nœuds sur la chaîne reliant les deux nœuds.

Définition 4 *Etant donnée une décomposition hyperarborescente $HD = (T, \chi, \lambda)$ de $H = (X, C)$, $TD(HD) = (E, T')$ est définie par :*
(i) $E = \{\chi(p), p \in sommets(T) | \nexists p' \in sommets(T), \chi(p) \subsetneq \chi(p')\}$,
(ii) $T' = (I', F')$ est un arbre de jointures de l'hypergraphe (X, E).

Propriété 1 *[15] $TD(HD)$ est une décomposition arborescente de $H = (X, C)$.*

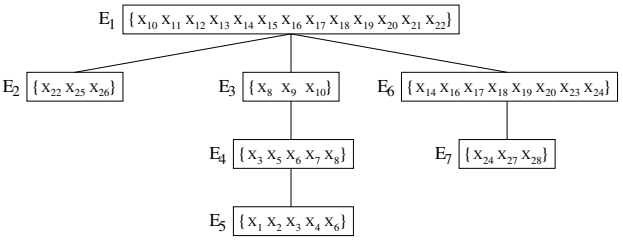

FIG. 4 – Décomposition arborescente induite (à partir de la figure 2).

Dans la figure 4, nous avons une décomposition arborescente qui n'est pas optimale ($w = 12$) et qui est induite par la décomposition hyperarborescente optimale de la figure 2.

3.3 De TD à (G)HD

Inversement, étant donnée une décomposition arborescente $TD = (E, T)$ de $H = (X, C)$, [15] propose une manière de calculer une décomposition hyperarborescente induite. Cette fois-ci, la construction n'est pas immédiate dans la mesure où recouvrir les différents clusters ne suffit pas car la quatrième condition de la définition de décomposition hyperarborescente n'est pas forcément vérifiée. De ce fait, il faut trouver des recouvrements des clusters satisfaisant cette condition supplémentaire. Ce qui peut être une tâche assez ardue. Or, cette condition est présente dans la définition dans le seul but de s'assurer de l'existence d'un algorithme permettant de calculer en un temps polynomial une décomposition hyperarborescente dont la largeur est bornée par une constante si elle existe. Dans notre cas, cet objectif est sans intérêt. Nous avons déjà une décomposition arborescente, et la meilleure décomposition hyperarborescente qu'elle induit est celle dont le recouvrement des clusters est minimum. Aussi, nous ne tiendrons plus compte de cette condition et allons construire une décomposition arborescente généralisée (GHD [10]).

Définition 5 *[10] Une décomposition hyperarborescente généralisée de H est un hyperarbre $HD = (T, \chi, \lambda)$ de H qui satisfait les conditions suivantes :*
(i) $\forall c \in C, \exists p \in sommets(T)$ t.q. $c \subset \chi(p)$,
(ii) $\forall x \in X$, l'ensemble $\{p \in sommets(T) : x \in \chi(p)\}$ induit un sous-arbre (connexe) de T,
(iii) $\forall p \in sommets(T), \chi(p) \subset \cup_{c \in \lambda(p)} c$,

La largeur gh d'une décomposition hyperarborescente généralisée $GHD = (T, \chi, \lambda)$ est $max_{p \in sommets(T)} |\lambda(p)|$. La largeur hyperarborescente généralisée gh^* de H est la largeur minimum sur toutes ses décompositions hyperarborescentes généralisées.

Nous associons un recouvrement minimum $\lambda(p)$ à chaque cluster $E_p \in E$ avec $p \in sommets(T)$.

Définition 6 *Etant donnée une décomposition arborescente $TD = (E, T)$ de $H = (X, C)$, nous définissons $GHD(TD) = (T, \chi, \lambda)$ comme suit :*
(i) $E = \chi$,
(ii) $\lambda = \{$recouvrement minimum de E_p dans $C, \forall p \in sommets(T)\}$.

Propriété 2 *$GHD(TD)$ est une décomposition hyperarborescente généralisée de largeur k, la taille maximum de $\lambda(p)$.*

Néanmoins, cette décomposition hyperarborescente généralisée n'est pas nécessairement complète. Pour la rendre complète, nous devons ajouter, pour toute arête c qui n'est pas fortement couverte, un fils à un nœud p tel que $c \subset E_p$. Soit $GHD_c(TD)$ la décomposition hyperarborescente généralisée complète obtenue de la sorte.

Théorème 1 *$GHD(TD)$ et $GHD_c(TD)$ ont la même largeur.*

La décomposition hyperarborescente généralisée induite par la décomposition arborescente de la figure 1 est facilement obtenue, et sa largeur est 4, parce que la taille maximum des $\lambda(p)$ vaut 4 avec le nœud associé à E_6 ($\{C_1, C_2, C_3, C_4\}$ est le recouvrement minimum).

En outre, si nous considérons gh^*, la largeur hyperarborescente généralisée d'un hypergraphe donné, nous savons que $gh^* \leq h^* \leq 3.gh^* + 1$ ([1]). Nous pouvons en déduire l'existence d'une décomposition hyperarborescente $HD_c(TD)$ de largeur au plus $3.k + 1$.

3.4 De Nouvelles Méthodes

Dorénavant, il est possible d'exploiter une décomposition hyperarborescente avec des méthodes de type TC ou BTD.

Auparavant, nous allons définir une nouvelle extension de TC (notée *TC-2009*) plus appropriée aux CSPs avec des contraintes n-aires. Etant donnés un CSP et une décomposition arborescente $TD = (E, T)$, le sous-problème associé au cluster E_i est défini, à l'image de TC-1989, par le même ensemble de variables E_i. Mais maintenant, l'ensemble des contraintes d'un cluster E_i est $C_{E_i} = \{c_j \in C : c_j \cap E_i \neq \emptyset\}$. Les relations associées à ces contraintes sont $R_{E_i} = \{r_j[c_j \cap E_i] : c_j \in C_{E_i}\}$. TC-2009 comporte également 3 étapes. La première calcule une décomposition arborescente du réseau de contraintes, en utilisant un algorithme de décomposition de (hyper)graphes alors que les deux étapes suivantes restent identiques. Ainsi, cette première étape est maintenant paramétrée par une décomposition graphique quelconque *DEC*. Si nous considérons une décomposition hyperarborescente optimale, nous définissons *TC-2009*$_{HD}$, qui exécute TC sur une décomposition arborescente $TD(HD)$ induite par HD et en considérant comme sous-problèmes, les clusters de variables et les contraintes dont l'ensemble des variables intersecte les clusters. De même, nous pouvons définir une large collection de méthodes à l'instar de *TC-2009*$_{TD}$ (TC basée sur une TD optimale), *TC-2009*$_{MCS(TD)}$ (TC basée sur une TD calculée par MCS), BTD_{HD} (BTD basée sur une HD optimale), BTD_{TD} (BTD basée sur une TD optimale), $BTD_{HMIN(HD)}$ (BTD basée sur une HD calculée grâce à une heuristique), $BTD_{HMIN(TD)}$ (BTD basée sur une TD calculée grâce à une heuristique), $BTD_{HMIN(TD)+HMAX(HD)}$ (BTD basée sur une TD optimale dont le nombre de contraintes dans les clusters a été maximisé grâce à une heuristique), etc.

Supposons que la largeur de décomposition hyperarborescente soit h dans le cadre de l'analyse de la complexité de *TC-2009*$_{HD}$. Chaque sous-problème (cluster) est résolu indépendamment (étape 2) en utilisant un algorithme de type nFC2. Grâce aux résultats présentés dans [15], le coût de la résolution d'un cluster E_i est maintenant en $O(S_i.r^{k_i})$, où S_i est la taille du sous-problème associé à E_i, et $k_i = k_{(E_i, C_{E_i})}$ (i.e. le paramètre associé au recouvrement minimum de E_i). Il faut noter que la taille de l'ensemble de solutions dans E_i est bornée par $O(r^{k_i})$. De ce fait, le coût total pour la résolution du CSP décomposé dans sa totalité est $O(S.r^k)$ où $k = max\{k_i : i \in I\}$. En plus, nous avons $k \leq h$. Nous pouvons donc établir que :

Théorème 2 *La complexité en temps de TC-2009*$_{HD}$ *et de BTD_{HD} est en $O(S.r^h)$.*

Ce résultat donne une présentation plus précise de la *Hiérarchie des Contraintes Traitables* puisque TC-2009$_{HD}$ et BTD$_{HD}$ sont au même niveau (sommet) dans la hiérarchie. Cela démontre que *TC-2009*$_{HD}$ est au moins aussi performante que MHD-1999. Plus précisément, la complexité temporelle de *TC-2009*$_{HD}$ et donc celle de BTD_{HD} sont identiques à celle de MHD-1999. Une autre conséquence de ce résultat est que nous disposons d'une nouvelle implémentation de MHD avec BTD_{HD} qui hérite de la même complexité en temps que MHD tout en limitant drastiquement la complexité en espace. Ainsi, cette approche peut potentiellement bénéficier d'une certaine efficacité pratique.

4 Expérimentations

Dans cette section, nous allons présenter les expérimentations qui ont été menées pour évaluer l'inté-

	FC	$BTD_{HMIN(TD)+HMAX(HD)}$		$BTD_{HMIN(TD)}$		$BTD_{HMIN(HD)}$			
CSP	temps	temps	w	temps	w	temps1	temps2	w	h
geo-1	18,01	14,19	26	18,03	19	MO	8,99	19	10
geo-9	10,40	19,00	26	5,46	18	TO	5,27	18	10
geo-12	14,18	17,50	31	17,68	22	MO	28,66	23	12
geo-19	17,04	30,64	23	170,28	18	TO	159,80	18	10
geo-41	11,46	11,12	24	8,03	19	12,93	8,09	19	10
geo-47	0	23,03	18	56,52	16	46,58	0,73	17	9
geo-52	10,96	10,91	31	7,51	19	8,70	7,44	19	10
geo-55	0,87	7,08	24	3,98	19	76,35	3,87	19	10
geo-57	5,61	6,69	26	6,19	23	4,68	6,16	23	12
geo-62	5,62	5,78	24	8,70	22	TO	11,93	20	11
geo-70	10,30	10,45	36	14,18	21	TO	13,96	21	11
geo-73	2,03	5,11	29	34,61	20	TO	8,17	21	11
geo-75	0,05	52,32	24	2,44	19	69,59	0,66	19	10
geo-86	12,81	13,50	27	15,97	23	TO	15,67	23	12
geo-94	16,82	14,40	27	18,68	18	TO	23,25	19	10
ren-3	TO	11,15	12	10,67	10	42,34	20,56	12	3
ren-6	TO	3,75	13	3,71	10	1279,55	2,70	13	3
ren-12	TO	11,85	12	11,64	10	134,24	10,49	10	3
ren-16	TO	10,36	12	6,04	10	3,65	6,43	13	3
ren-17	TO	5,42	12	9,59	10	145,06	3,41	11	3
ren-18	TO	12,09	11	12,23	11	39,34	10,46	12	3
ren-19	TO	12,07	11	7,74	9	49,25	16,36	11	3
ren-23	TO	2,81	11	12,87	9	28,82	3,79	12	4
ren-24	TO	8,05	14	7,97	11	35,01	7,63	12	4
ren-30	TO	9,81	13	3,80	10	202,05	8,25	11	3
ren-35	TO	12,28	12	7,32	10	57,33	13,19	11	3
ren-36	TO	1,78	13	3,80	11	225,76	4,88	13	4
ren-37	TO	17,01	15	13,68	12	13,64	21,16	14	4
ren-39	TO	35,55	16	13,45	12	746,24	1,79	12	5
ren-40	TO	8,60	14	5,86	11	65,55	9,01	12	4
ren-42	TO	3,44	15	2,48	12	TO	2,50	12	3
ren-47	TO	21,31	14	53,71	12	324,20	80,25	12	5

TAB. 1 – Temps de résolution (en s) des méthodes FC, $BTD_{HMIN(TD)+HMAX(HD)}$, $BTD_{HMIN(TD)}$ et $BTD_{HMIN(HD)}$, et paramètres des décompositions sur les instances des classes modifiedRenault (111 variables) et geom (50 variables) issues des benchmarks de la compétition CPAI08.

rêt pratique des méthodes que nous venons de définir. Les instances CSP que nous avons considérées sont issues de la compétition CPAI08 (http ://www.cril.univ-artois.fr/CPAI08/).

Les implémentations existantes de TC et MHD ne permettent pas de résoudre ces instances à cause de l'espace mémoire trop important qu'elles requièrent ou du temps beaucoup trop long qu'elles mettent pour résoudre séparément les sous-problèmes d'une décomposition. Ainsi, nous avons utilisé BTD qui a déjà montré son efficacité sur des CSPs structurés.

Par ailleurs, calculer une décomposition (hyper)arborescente optimale est un problème NP-difficile. La durée d'exécution des techniques exactes est trop importantes (voir [14]). En plus, il n'existe aucune garantie sur l'efficacité pratique de l'utilisation de ces décompositions. De ce fait, nous préférons des heuristiques avec une meilleure complexité temporelle pour calculer nos décompositions. Donc, nous ne considérerons ni BTD_{HD}, ni BTD_{TD}. Dans un premier temps, nous avons testé $BTD_{HMIN(HD)}$ et $BTD_{HMIN(TD)}$. Nous définissons $BTD_{HMIN(HD)}$ comme une extension de BTD qui gère les contraintes de manière analogue à MHD. Ainsi, lors de la résolution d'un cluster, seules les contraintes données par la HD sont prises en compte. En outre, pour calculer les HD, nous avons utilisé les heuristiques *Bucket Elimination for Hypertree* [7] et *det-k-decomp* [11] dont les implémentations sont disponibles sur internet (www.dbai.tuwien.ac.at/proj/hypertree/). Dans [11], elles sont évaluées comme les meilleures techniques pour calculer les HD. Dans $BTD_{HMIN(TD)}$, les TD sont calculées grâce à l'algorithme de triangulation Minfill qui est bien connu pour ses excellents résultats.

Les expérimentations ont été conduites sur un PC doté d'un Pentium IV de 3,2GHz et de 1Go de RAM, avec un système d'exploitation Linux. Pour chaque problème, la durée limite de résolution est bornée à 1800s. Au-delà, le problème est considéré comme non résolu et cela est symbolisé par TO. Les résultats que nous présentons portent sur les instances des classes *modifiedRenault* et *geom*. Dans le tableau 1, les résultats de la version classique de $BTD_{HMIN(TD)}$ sont reportés dans la colonne temps1.

Comme nous nous y attendions, $BTD_{HMIN(HD)}$ a des performances très pauvres. Elle échoue dans la résolution de beaucoup d'instances (TO ou l'espace mémoire requis dépasse 1GB, ceci étant symbolisé par MO). Sa durée d'exécution moyenne dépasse 248s. En effet, les sous-problèmes dans une HD sont très difficiles à résoudre à cause du nombre restreint de contraintes considérées. Ce petit nombre de contraintes affaiblit la puissance des techniques de filtrage qui contribuent grandement à l'efficacité de l'énumération dans ces sous-problèmes. $BTD_{HMIN(TD)}$ se comporte largement mieux grâce à un nombre beaucoup plus grand de contraintes dans les clusters qui deviennent plus faciles à résoudre. Elle réussit à résoudre toutes les instances avec une durée moyenne de 6,67s.

Pour confirmer cette observation, nous avons essayé de mettre en lumière l'impact du nombre de contraintes dans les clusters des TD. Ainsi, nous avons considéré les méthodes $BTD_{HMIN(TD)+HMIN(HD)}$ et $BTD_{HMIN(TD)+HMAX(HD)}$. Dans $BTD_{HMIN(TD)+HMIN(HD)}$, nous avons utilisé Minfill et MCS pour calculer deux TD et nous choisissons ensuite celle qui minimise le nombre maximum de contraintes dans les clusters, tandis que dans $BTD_{HMIN(TD)+HMAX(HD)}$, nous choisissons celle qui maximise ce nombre.

Nous observons que $BTD_{HMIN(TD)+HMIN(HD)}$ et $BTD_{HMIN(TD)}$ obtiennent les mêmes résultats rassemblés dans la même colonne. $BTD_{HMIN(TD)+HMAX(HD)}$ donne les meilleurs résultats puisque les clusters très contraints sont plus facile à résoudre. Sa durée de résolution moyenne est 5,42s. En outre, les performances de $BTD_{HMIN(HD)}$ sont drastiquement améliorées si on prend en compte toutes les contraintes possibles (dans la colonne temps2) à l'image de $BTD_{HMIN(TD)}$, tandis que les bornes de complexité théorique sont préservées. Néanmoins, ces résultats restent en dessous de ceux de la méthode $BTD_{HMIN(TD)+HMAX(HD)}$. En effet, la durée moyenne de résolution de cette approche est de 6,05s.

Il faut préciser que FC, tout seul, échoue dans la résolution de près de la totalité des instances de la classe *modifiedRenault*, qui ont de bonnes propriétés topologiques (w est en moyenne inférieur à $n/10$). Alors que, ces résultats dans la classe *geom* sont meilleurs par rapport à ceux des méthodes de résolution basées décomposition car la taille de ces problèmes est plus petite et la qualité de leur propriétés topologiques est assez faible (w est en moyenne très proche de $n/2$).

En résumé, nous avons noté premièrement que calculer une TD avec une petite largeur est plus intéressant en pratique (pour la résolution) que le calcul d'une bonne HD. En effet, cette TD donnent de meilleurs résultats en pratique, de même que des bornes de complexité de qualité par rapport à la taille des clusters et de leur recouvrement minimum. En outre, ses clusters peuvent être résolus plus facilement quand ils contiennent plus de contraintes. Cela est justifié par le fait que plus un problème est contraint, plus il est facile à résoudre (ce qui peut être démontré aisément grâce à des arguments probabilistes).

5 Discussion et Conclusion

Nous avons proposé de nouvelles approches basées sur la combinaison de méthodes de décomposition de CSPs. Plus précisément, nous avons introduit deux méthodes optimales, $TC\text{-}2009_{HD}$ et $BTD\text{-}2009_{HD}$, qui exploitent la décomposition hyperarborescente et TC ou BTD pour résoudre des réseaux de contraintes. Cette approche permet d'avoir de meilleures bornes de complexité tout en héritant de l'efficacité pratique des méthodes énumératives telles que nFC2, une des techniques les plus performantes dans la résolution de CSP. Ensuite, nous avons enrichi la hiérarchie des Contraintes Traitables en mettant à jour le sommet de la hiérarchie où nous retrouvons désormais les méthodes $TC\text{-}2009_{HD}$ et $BTD\text{-}2009_{HD}$. Finalement, nous avons obtenu des résultats expérimentaux qui montrent l'intérêt pratique de notre approche.

Ce travail constitue une extension des résultats présentés dans [15]. D'une part, nous montrons, pour les méthodes de résolution basées sur la décomposition de réseaux, l'utilité d'une différenciation de la phase de décomposition et de la phase de résolution. Cette distinction nous permet de définir une plus large palette de méthodes via des hybridations entre les techniques existantes, et elle fournit également une meilleure lecture de la *Hiérarchie des Contraintes Traitables*, tout en la complétant. De plus, nous montrons expérimentalement comment il est possible d'exploiter ces méthodes hybrides. En particulier, ce travail propose des pistes nouvelles pour le choix des décompositions de réseaux et pour celui des méthodes de résolution qui permettront de les exploiter au mieux.

Nos résultats diffèrent de celui de [4] (théorème 7.28, page 231). Dans cet article, Dechter propose la résolution d'une décomposition arborescente de CSP en r^{hw} où hw est obtenu en faisant la somme du nombre de contraintes inclues dans chaque cluster et du nombre de variables du cluster qui ne sont pas couvertes par ces contraintes, puis en prenant le maximum de ces valeurs sur tous les clusters. Pour $TC\text{-}2009_{HD}$ et $BTD\text{-}2009_{HD}$, la complexité en temps est limitée à r^h (où h est la largeur hyperarborescente généralisée induite). De ce point de vue, notre résultat est meilleur.

Nos résultats diffèrent également de ceux de [5] qui démontrent de manière empirique que la borne fournie par la largeur arborescente est souvent meilleure que celle de la largeur hyperarborescente dans les réseaux de contraintes probabilistes ou déterministes. Dans le rapport technique constituant une extension de cet article, il est dit que l'approche And/Or Search Graph garantit une borne de complexité temporelle qui dépend de la largeur d'une décomposition hyperarborescente donnée. Mais, cette méthode ne considère qu'une sous classe des décompositions hyperarborescentes possibles. Cela veut dire que sa complexité est moins forte que celle de MHD, alors que celle de $TC\text{-}2009_{HD}$ et $BTD\text{-}2009_{HD}$ dépendent de la largeur d'une décomposition hyperarborescente généralisée. Donc, la complexité de ces dernières est au moins équivalente à celle de MHD.

Une poursuite naturelle de ce travail pourrait être l'étude de décompositions graphiques qui combinent l'optimisation des paramètres w (qui serait minimisé) et h (qui serait maximisé). Par ailleurs, il semble naturel d'étendre cette analyse au COP (optimisation sous contraintes) ou aux modèles graphiques probabilistes comme dans [5] (et [17]).

Références

[1] I. Adler, G. Gottlob, and M. Grohe. Hypertree width and related hypergraph invariants. *Eur. J. Comb.*, 28(8) :2167–2181, 2007.

[2] C. Bessière, P. Meseguer, E. C. Freuder, and J. Larrosa. On forward checking for non-binary constraint satisfaction. *Artificial Intelligence*, 141 :205–224, 2002.

[3] R. Dechter. *Constraint processing*. Morgan Kaufmann Publishers, 2003.

[4] R. Dechter. *Tractable Structures for Constraint Satisfaction Problems*, pages 209–244. Chapter in the *Handbook of Constraint Programming* F. Rossi, T. Walsh and P. van Beek, 2006.

[5] R. Dechter, L. Otten, and R. Marinescu. On the Practical Significance of Hypertree vs. Tree Width. In *ECAI*, pages 913–914, 2008.

[6] R. Dechter and J. Pearl. Tree-Clustering for Constraint Networks. *Artificial Intelligence*, 38 :353–366, 1989.

[7] A. Dermaku, T. Ganzow, G. Gottlob, B. MacMahan, N. Musliu, and M. Samer. Heuristic Methods for Hypertree Decompositions. In *MICAI 2008*, 2008.

[8] G. Gottlob, N. Leone, and F. Scarcello. On Tractable Queries and Constraints. In *Proceedings of the 18th Symposium on Principles of Database Systems (PODS-99)*, pages 21–32, 1999.

[9] G. Gottlob, N. Leone, and F. Scarcello. A Comparison of Structural CSP Decomposition Methods. *Artificial Intelligence*, 124 :343–282, 2000.

[10] G. Gottlob, N. Leone, and F. Scarcello. Robbers, marshals, and guards : game theoretic and logical characterizations of hypertree width. *Journal of Computer and System Sciences (JCSS)*, 66(4) :775–808, 2003.

[11] G. Gottlob and M. Samer. A backtracking-based algorithm for hypertree decomposition. *ACM Journal of Experimental Algorithmics*, 13, 2008.

[12] M. Grohe and D. Marx. Constraint solving via fractional edge covers. In *SODA*, pages 289–298, 2006.

[13] R. Haralick and G. Elliot. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14 :263–313, 1980.

[14] P. Jégou, S. N. Ndiaye, and C. Terrioux. Computing and exploiting tree-decompositions for solving constraint networks. In *CP*, pages 777–781, 2005.

[15] P. Jégou, S.N. Ndiaye, and C. Terrioux. A New Evaluation of Forward Checking and its Consequences on Efficiency of Tools for Decomposition of CSPs. In *ICTAI*, pages 486–490, 2008.

[16] P. Jégou and C. Terrioux. Hybrid backtracking bounded by tree-decomposition of constraint networks. *Artificial Intelligence*, 146 :43–75, 2003.

[17] K. Kask, R. Dechter, J. Larrosa, and A. Dechter. Unifying tree decompositions for reasoning in graphical models. *Artificial Intelligence*, 166 :165–193, 2005.

[18] N. Robertson and P.D. Seymour. Graph minors II : Algorithmic aspects of treewidth. *Algorithms*, 7 :309–322, 1986.

[19] D. Sabin and E. Freuder. Contradicting Conventional Wisdom in Constraint Satisfaction. In *Proc. of ECAI*, pages 125–129, 1994.

[20] R. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13 (3) :566–579, 1984.