Calcul et exploitation de décompositions arborescentes pour la résolution de CSP

Samba Ndojh Ndiaye

LSIS - UMR CNRS 6168
Université Paul Cézanne (Aix-Marseille 3)
Av. Escadrille Normandie-Niemen
13397 Marseille Cedex 20 (France)
samba-ndojh.ndiaye@univ-cezanne.fr

Résumé

Les méthodes exploitant les décompositions arborescentes pour résoudre des réseaux de contraintes semblent constituer les meilleures approches en terme de complexité théorique en temps. Néanmoins, les études menées dans le cadre des décompositions arborescentes avaient essentiellement pour but d'optimiser des critères graphiques. Toutefois, nous avons observé que ces critères graphique n'étaient pas les plus pertinents pour une résolution efficace en pratique. De ce fait, nous menons ici une étude sur la qualité de ces décompositions au niveau de l'efficacité pratique des méthodes de résolution structurelles. De même, notre étude s'étend à une bonne exploitation des décompositions calculées.

1 Introduction

Le formalisme CSP (Constraint Satisfaction Problem) offre un cadre puissant pour la représentation et la résolution efficace de nombreux problèmes. Formuler un problème en termes de CSP consiste en la définition d'un ensemble X de variables, qui doivent chacune être affectées dans leur domaine (fini) respectif, tout en satisfaisant un ensemble C de contraintes qui expriment des restrictions sur les différentes affectations possibles. Une solution est une affectation de toutes les variables qui satisfait toutes les contraintes. Le problème d'existence de solution est NP-Complet. La méthode usuelle pour la résolution de CSP est basée sur la recherche de type "backtracking" (FC, MAC [1]) avec une complexité théorique en temps de $O(e.d^n)$ où n est le nombre de variables, e le nombre de contraintes, et d la taille maximum des domaines. La meilleure borne connue à ce jour, est fournie par la "tree-width" du CSP (généralement noté w) : $O(n.d^{w+1})$. On peut consulter [2] pour une étude récapitulative des différentes méthodes qui permettent d'atteindre cette borne telles que le Tree-Clustering [3] et ses extensions. Ces méthodes sont fondées sur la notion de décomposition arborescente du réseau de contraintes. Selon l'instance, le gain effectif en temps de calcul par rapport à une approche de type backtracking, peut être considérable. Toutefois, la complexité en espace, linéaire pour les méthodes de type backtracking, peut rendre l'approche par décomposition totalement inopérante. Cette complexité peut être réduite à $O(n.s.d^s)$ où s est la taille maximum des intersections entre clusters [4]. Ces résultats théoriques sont valables également pour les CSPs Valués [5]. De nombreux travaux, fondés sur cette démarche, ont été réalisés, mais nous allons ici nous focaliser sur la méthode BTD [6] qui semble constituer l'approche la plus efficace pour la résolution de CSP par décomposition arborescente de graphes. L'objectif du présent travail est d'améliorer ces méthodes d'un point de vue pratique en proposant des heuristiques pour le calcul de décompositions de meilleure qualité mais aussi pour une meilleure exploitation de celles-ci.

Cela doit permettre de trouver une réponse à l'inadéquation entre la complexité théorique de ces méthodes et leur résultats pratiques catastrophiques.

Cet article est organisé comme suit. Dans la section 2, nous rappelons les notions de base propres aux méthodes exploitant la décomposition arborescente de graphe. La section 3 présente une étude comparative des différentes méthodes de triangulation, tandis que la section 4 donne des stratégies pour une meilleure exploitation des décompositions. Enfin, la dernière section conclut cet article tout en donnant des pistes pour des travaux futurs.

2 Décomposition arborescente de CSP

Dans cet article, et sans manque de généralité, nous considérerons uniquement les CSPs binaires, à savoir ceux dont les contraintes ne portent que sur des couples de variables. Dans un tel cas, la structure d'un CSP peut être représentée par le graphe (X, C), qui est appelé *graphe de contraintes*. Les sommets de ce graphe correspondent alors aux variables de X et les arêtes entre couples de sommets correspondent à l'existence de contraintes entre les couples de variables associées aux sommets.

La méthode BTD (pour Backtracking with Tree-Decomposition) est basée sur la notion de décomposition arborescente de graphes, notion formellement définie par Robertson et Seymour dans [7]. Etant donné un graphe $G = (X, C)$, une *décomposition arborescente* de G est une paire $(E, \mathcal{T})$ avec $\mathcal{T} = (I, F)$ un arbre et $E = \{E_i : i \in I\}$ une famille de sous-ensembles de X, telle que chaque sous-ensemble (ou cluster) E_i est un noeud de $\mathcal{T}$ et vérifie : (i) $\cup_{i \in I} E_i = X$, (ii) pour toute arête $\{x, y\} \in C$, il existe $i \in I$ avec $\{x, y\} \subseteq E_i$, et (iii) pour tout $i, j, k \in I$, si k figure sur un chemin de i à j dans $\mathcal{T}$, alors $E_i \cap E_j \subseteq E_k$.
La largeur d'une décomposition arborescente $(E, \mathcal{T})$ est égale à $max_{i \in I}|E_i| - 1$. La *largeur d'arborescence* ou *tree-width* de G (notée w) est la largeur minimale par rapport à toutes les décompositions arborescentes de G.
BTD s'appuie sur la notion d'ordre compatible pour l'instanciation des variables [6]. La complexité en temps de BTD est $O(n.d^{w+1})$. Malheureusement, l'obtention d'une décomposition arborescente optimale, à savoir une décomposition arborescente dont la largeur est w, constitue un problème NP-difficile [8]. Afin de résoudre ce problème, de nombreux travaux exploitent une approche fondée sur la notion de graphe *triangulé* (voir [9] pour une introduction aux graphes triangulés). Un graphe est dit triangulé s'il tolère un *ordre d'élimination parfait*, c'est-à-dire un ordre sur les sommets $\sigma = (x_1, \ldots, x_n)$ tel que le voisinage ultérieur de tout sommet x_i (sommets x_j voisins figurant après x_i dans l'ordre σ) forme une clique.
Le lien entre graphes triangulés et décomposition arborescente est évident. En effet, étant donné un graphe triangulé G, l'ensemble des cliques maximales $E = \{E_1, E_2, \ldots, E_k\}$ de G correspond à la famille de sous-ensembles associée à la décomposition arborescente. Comme tout graphe G n'est pas nécessairement triangulé, une décomposition arborescente de G peut être obtenue par une triangulation de G, c'est-à-dire une opération qui calculera un graphe triangulé G' à partir de G. Plus précisément, nous appelons *triangulation* l'ajout à G d'un ensemble d'arêtes C' tel que $G' = (X, C \cup C')$ soit triangulé.
La largeur d'arborescence de G' est alors égale à la taille de la plus grande clique du graphe résultant G' moins un. La largeur d'arborescence (tree-width) de G est alors égale à la plus petite largeur d'arborescence pour toutes les triangulations de G.
Généralement, lors de l'exploitation des décompositions arborescentes pour la résolution de CSP, on considère des approximations de triangulations optimales. Ainsi, la complexité en temps est alors $O(m.d^{w^+ + 1})$ où $w^+ + 1$ est la taille du plus grand cluster (ou clique) et nous avons $w + 1 \leq w^+ + 1 \leq n$. La complexité en espace est alors $O(n.s.d^s)$ où s est la taille maximale des *séparateurs*, c'est-à-dire la taille maximum des intersections entre paires de clusters (on a nécessairement $s \leq w^+$). L'une des conséquences immédiates de l'existence de ces bornes de complexité est que, pour garantir *a priori* de bonnes implémentations, il faut impérativement minimiser les valeurs de w^+ ainsi que de s. Nous nous proposons de mener une étude sur la qualité des décompositions arborescentes non pas d'un point de vue purement graphique mais aussi pratique.

3 Triangulations

Dans [10], nous avons considéré le problème de la recherche de "bonnes" décompositions arborescentes en termes de "bonnes" triangulations. Plusieurs algorithmes et approches ont été proposés pour les triangulations. Dans tous les cas, l'objectif consiste à minimiser soit le nombre d'arêtes ajoutées, soit la taille des cliques dans le graphe triangulé. On peut classer les approches possibles en 4 classes :

1. **Triangulations optimales.** Le problème traité est NP-difficile. Donc la complexité des algorithmes proposés est exponentielle. Ce type d'algorithmes a évidemment un intérêt pratique nécessairement limité cause de leur coût dramatique.
2. **Algorithmes d'approximation.** Ces algorithmes garantissent l'approximation d'un optimum par un facteur constant. Leur complexité est généralement polynomiale en temps (dans la tree-width) [11]. Malheureusement, pour le moment, cette approche semble également irréaliste. En effet, la complexité du dernier algorithme de Amir est $O(n^3.log_4(nk^5).log(k))$ avec de plus une constante cachée (au sens de la notation "grand O") supérieure à 850 (d'après [12]). De plus, Amir indique dans ses expérimentations (voir [13]) que sur les benchmarks testés, son algorithme prend de 6 minutes à 6 jours, selon l'instance, alors même qu'une heuristique naïve comme *min-degré* (voir plus loin), donne sur les mêmes benchmarks respectivement de 1 seconde à 2 minutes, avec de plus des résultats dont la qualité est significativement meilleure (l'approximation de la largeur d'arborescence est de l'ordre de 50 % inférieure) !
3. **Triangulation minimale.** Une triangulation minimale calcule un ensemble C' tel que pour tout sous-ensemble $C" \subset C'$, le graphe $G' = (V, C \cup C")$ n'est pas triangulé. Il faut préciser qu'une triangulation peut être minimale sans être optimale (minimum). L'intérêt de ce type d'approche est relatif notamment à l'existence d'algorithmes de complexité polynomiale. Par exemple LEX-M [14] et plus récemment LB [15], ont présentés des algorithmes en $O(ne')$ où e' est le nombre d'arêtes figurant dans le graphe après sa triangulation.
4. **Triangulation heuristique.** Ces approches construisent en général un ordre (dynamiquement) et ajoutent des arêtes dans le graphe, de sorte qu'en fin de traitement, l'ordre obtenu soit un ordre d'élimination parfait pour le graphe résultant G'. La complexité de ces méthodes est en général polynomiale (souvent même linéaire) mais elles n'offrent, en contrepartie, aucune garantie de minimalité. Cette approche serait justifiée en pratique [16]. En effet, Kjærulff a observé entre autres, qu'au contraire des résultats attendus, ces heuristiques produisent des triangulations raisonnablement proches de l'optimum. De plus, elles sont en général très faciles à implémenter. Plusieurs heuristiques sont généralement considérées, et la complexité en temps de leur implémentation varie de $O(n + e')$ à $O(n(n + e'))$ pour la plus coûteuse (*min-fill*).

Jusqu'à présent, les expérimentations présentées dans la littérature semblent indiquer que les approches 1 (recherche d'optimalité) et 2 (approximation) n'ont que très peu d'intérêt pratique car leur temps d'exécution, comme première étape d'une résolution de CSP, est bien trop coûteux par rapport au profit qu'elle pourrait offrir en termes d'approximation de la largeur d'arborescence. Aussi, nous pensons qu'il est préférable de concentrer nos effort sur les approches 3 et 4.

Dans [10], nous avons fait une étude comparative entre plusieurs algorithmes de triangulation minimaux (LB et LEX-M) et heuristiques(MCS, Min-fill et Min-esp). *Min-esp* est une méthode similaire à min-fill et basée sur la notion d'espérance mathématique du nombre de solutions d'un problème, [17]. Elle ordonne les sommets de 1 à n, en sélectionnant comme nouveau sommet, le sommet qui conduira si l'on complète le sous-graphe induit par ses voisins non encore numérotés à une clique d'espérance du nombre de solutions minimum.
Les méthodes structurelles tirent profit de la structure du problème pour améliorer la résolution, mais cela suppose que le problème possède une structure intéressante (ce qui

p	t	LEX-M		LB		min-fill		MCS		min-exp	
		w^+	time	w^+	time	w^+	time	w^+	time	w^+	time
10%	215	18.50	5.53	14.00	3.95	15.97	10.66	14.03	4.06	23.50	4.34
20%	237	22.00	4.07	14.00	4.37	16.33	6.74	14.00	3.53	23.03	3.81
30%	257	23.30	82.79	14.00	2.85	17.20	5.49	15.03	3.81	20.37	1.20
40%	285	24.90	78.22	14.00	1.11	15.33	1.21	15.33	5.88	17.10	1.57

TAB. 1 – [CSP] durée d'exécution (en s) et valeur de w^+ pour des instances de la classe $(150, 25, 15, t, 5, 15)$ après le retrait de $p\%$ arêtes. (avec s limité à 5).

est généralement le cas des problèmes réels) contrairement aux problèmes aléatoires classiques. De ce fait nous avons menés nos expérimentations sur des CSP présentant cette propriét'e. Le tableau 1 contient les durées d'exécution de BTD pour des décompositions arborescentes calculées par ces différentes triangulations sur des CSP aléatoires structurés. Ces derniers sont générés en utilisant le modèle introduit dans [6], qui utilise un arbre de cliques aléatoire, dont la taille des cliques et celle des séparateurs sont bornées. Une fois le graphe de contraintes généré, nous avons supprimé un certain pourcentage p d'arêtes ($p = 10, 20, \ldots$).

On observe que LB et MCS parviennent à retrouver la structure du problème et calculent ainsi des décompositions avec une treewidth de très bonne qualité. Elles obtiennent aussi avec min-exp les meilleurs résultats au niveau de la durée d'exécution de BTD. Ces méthodes ont des performances très proches mais LB semble la plus robuste avec une meilleure treewidth. La surprise vient des très bons résultats obtenus par min-exp. En effet malgré des treewidth de qualité souvent médiocre, elle conduit à d'excellentes durées d'exécution. Cela confirme que la treewidth n'est pas le seul critère pertinent pour évaluer la qualité d'une décomposition. D'autres critères liés par exemple à l'aspect sémantique des contraintes, sont à prendre à compte.

Pour conclure, on peut estimer que le problème de l'obtention d'une "bonne" décomposition, au sens de son utilité pour la résolution de CSP demeure ouvert. Néanmoins, des éléments nouveaux sont apparus ici, qui peuvent maintenant mieux nous guider dans le traitement de cette question. En effet, dans cette section, plusieurs pistes sont ouvertes. Tout d'abord, nous avons pu observer que la recherche d'une triangulation optimale n'est pas nécessairement l'objectif le plus pertinent si l'on triangule en vue de résoudre un CSP. Ainsi, des heuristiques exploitables en temps polynomial pourraient s'avérer suffisantes pour calculer des décompositions arborescentes de très bonne qualité en vue d'une résolution efficace. D'autant plus que l'optimalité ne garantit aucunement une résolution plus efficace dans la pratique.

4 Exploitation des décompositions arborescentes

Dans [18], nous avons étudié des stratégies pour une meilleure exploitation des décompositions arborescentes. En effet, sans heuristique pertinente dans les choix d'affectation des variables, les algorithmes basés sur le backtracking tels que FC ou MAC sont généralement dans l'impossibilité de résoudre efficacement nombre d'instances. Aussi, pour améliorer les performances d'une méthode telle que BTD, et au-delà, des méthodes à base de décomposition, il est nécessaire de s'appuyer sur des heuristiques performantes. Pour être efficaces, ces heuristiques doivent être dynamiques, c'est-à-dire que les choix opérés doivent être élaborés au gré de la recherche et non au préalable de façon statique et définitive. Nous avons donc défini plusieurs classes d'ordres sur les variables de plus en plus dynamiques mais qui garantissent de bonnes bornes de complexité. Les ordres de la classe 3 permettent un choix de variables totalement dynamique à l'intérieur des clusters et un choix dynamique du prochain cluster à visiter parmi les clusters fils des clusters totalement instanciés. Ils permettent également de maintenir la borne de complexité de BTD.

De même, nous avons proposé deux nouvelles extensions pour la borne de complexité tem-

CSP (n, d, w, t, s, n_c, p)	w^+	s	Class 3		Class 4			
			A F	B G	B D	A C	A F	B G
$(150, 25, 15, 215, 5, 15, 10)$	13.0	12.2	2.45	5.34	2.75	2.17	2.08	2.65
$(150, 25, 15, 257, 5, 15, 30)$	12.1	11.4	4.97	3.55	1.41	1.05	1.13	1.30
$(250, 20, 20, 129, 5, 20, 30)$	16.8	15.8	6.19	7.84	33.93	4.61	4.41	34.20
$(250, 20, 20, 146, 5, 20, 40)$	15.9	15.2	4.51	17.99	11.38	3.17	3.17	10.63
$(250, 25, 15, 211, 5, 25, 10)$	13.0	12.3	13.37	18.12	5.86	7.71	6.65	6.44
$(250, 25, 15, 253, 5, 25, 30)$	12.3	11.8	5.14	5.26	2.80	3.71	3.52	3.06
$(250, 20, 20, 99, 10, 25, 10)$	17.9	17.0	M	M	66.94	63.15	62.99	66.33
$(500, 20, 15, 123, 5, 50, 10)$	13.0	12.5	7.31	7.54	5.48	4.50	4.41	5.86
$(500, 20, 15, 136, 5, 50, 20)$	12.9	12.1	27.01	15.11	4.86	4.92	3.94	5.24

TAB. 2 – Paramètres w^+ et s de la décomposition arborescente et durée d'exécution (en s) sur des CSP aléatoires structurés avec mdd_{dyn} pour les classes 2, 3 et 4.

porelle qui confère une importance accrue à l'apport et aux bénéfices des heuristiques([19]). En effet, là où la marge qui nous était laissée de choisir la prochaine variable devant être affectée était restreinte à w^+ variables au plus, nous disposons maintenant d'un choix parmi $w^+ + k$ variables, tout en conservant une borne de complexité théorique. Cette démarche permet d'assurer une meilleure maîtrise de l'espace requis en obtenant une complexité en espace de $O(n.s'.d^{s'})$ où $s' \leq s$. Ces extensions sont basées sur la notion de *décompositions arborescentes $k-$recouvrantes orientées* [19]. Suivant qu'on utilise une (Classe 4) ou plusieurs (Classe 5) *décompositions arborescentes $k-$recouvrantes orientées* lors de la résolution, nous disposons de nouvelles bornes de complexité en temps ($O(exp(w^+ + k))$ pour la classe 4 et $O(exp(2(w^+ + k)))$ pour la classe 5) qui offrent une liberté accrue aux heuristiques de choix de variables.

Pour calculer des ordres de qualité dans ces différentes classes, nous avons élaboré plusieurs heuristiques. Les critères sur lesquels sont fondées ces heuristiques sont liés aux propriétés topologiques du réseau de contrainte et de sa décomposition, ainsi qu'aux traits sémantiques de l'instance traitée. De même, nous avons défini plusieurs heuristiques de fusion de clusters qui permettent de calculer des *décompositions arborescentes $k-$recouvrantes orientées*. Le tableau 2 permet de comparer la classe 3 et la classe 4. Nous avons utilisé comme critère de fusion de clusters celui basé sur la taille des séparateur (on fusionne deux clusters si la taille de leur séparateur dépasse 5). Malgré l'augmentation de la taille des clusters, la classe 4 obtient les meilleurs résultats grâce à l'apport considérable de l'heuristique dynamique de choix de variables. En outre la maîtrise de s permet de limiter l'espace mémoire requis et ainsi de résoudre les instances de la classe $(250, 20, 20, 99, 10, 25, 10)$ où la classe 3 échoue à cause de la trop grande quantité de mémoire requise. Il est tout de même nécessaire de faire un compromis entre la treewidth de la décomposition et la liberté donnée à l'heuristique dynamique de choix de variables et ainsi trouver la bonne valeur de k au delà de laquelle BTD ne tire plus assez profit de la structure du problème.

5 Conclusion

Dans cet article, nous avons présenté une étude qui a pour objectif de rendre opérationnelles les méthodes de résolutions strcuturelles de CSP. En effet, il existe une inadéquation entre les bonnes bornes de complexité de ces méthodes et leurs performances très pauvres en pratiques. L'objectif est donc de leur pourvoir des heuristiques pertinentes pour le calcul de décompositions arborescentes de qualité et pour une meilleure exploitation de ces dernières, et tout cela d'un point de vue pratique. Nous avons dans un premier temps comparé différentes méthodes de triangulation. Il s'est avéré que les critères graphiques (la treewidth) n'étaient pas les plus pertinents ou du moins les seuls pertinents pour évaluer la qualité d'une décomposition arborescente. Nous avons aussi montré que les triangulations minimales et heuristiques permettent de calculer de très bonnes décompositions de manière très efficace. Dans un second temps, nous avons proposé des stratégies pour une meilleure exploitation des décompositions avec des heuristiques de choix de variables plus dynamiques

tout en maintenant de bonnes bornes de complexité. Les expérimentations menées ont mis en lumière une amélioration très significative des performances de BTD. Des travaux futurs devraient permettre d'accroître cela avec de nouveaux critères pour fusionner les clusters et ainsi avoir de bonnes valeurs de k. De même cette étude doit être étendue aux VCSP.

Références

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

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

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

[4] R. Dechter and Y. El Fattah. Topological Parameters for Time-Space Tradeoff. *Artificial Intelligence*, 125 :93–118, 2001.

[5] C. Terrioux and P. Jégou. Bounded backtracking for the valued constraint satisfaction problems. In *Proceedings of CP*, pages 709–723, 2003.

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

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

[8] S. Arnborg, D. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal of Discrete Mathematics*, 8 :277–284, 1987.

[9] M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press. New-York, 1980.

[10] P. Jégou, S. N. Ndiaye, and C. Terrioux. Sur la gnration et l'exploitation de dcompositions pour la rsolution de rseaux de contraintes. In *Actes des JFPC'2005*, pages 149–158, 2005.

[11] E. Amir. Efficient approximation for triangulation of minimum treewidth. In *Proceedings of UAI*, pages 7–15, 2001.

[12] V. Bouchitté, D. Kratsch, H. Muller, and I. Todinca. On treewidth approximations. *Discrete Appl. Math.*, 136(2-3) :183–196, 2004.

[13] E. Amir. Approximation algorithms for treewidth, 2002. http ://reason.cs.uiuc.edu/eyal/paper.html.

[14] D. Rose, R. Tarjan, and G. Lueker. Algorithmic Aspects of Vertex Elimination on Graphs. *SIAM Journal on computing*, 5 :266–283, 1976.

[15] A. Berry. A Wide-Range Efficient Algorithm for Minimal Triangulation. In *Proceedings of SODA*, january 1999.

[16] U. Kjaerulff. Triangulation of Graphs - Algorithms Giving Small Total State Space. Technical report, Judex R.R. Aalborg., Denmark, 1990.

[17] B. Smith. The Phase Transition and the Mushy Region in Constraint Satisfaction Problems. In *Proceedings of ECAI*, pages 100–104, 1994.

[18] P. Jégou, S. N. Ndiaye, and C. Terrioux. Heuristiques pour la recherche énumérative bornée : Vers une libération de l'ordre. In *Actes des JFPC'2006*, pages 219–228, 2006.

[19] P. Jégou, S. N. Ndiaye, and C. Terrioux. An extension of complexity bounds and dynamic heuristics for tree-decompositions of CSP. In *Proceedings of CP*, 2006.