Sur la pertinence des décompositions arborescentes optimales pour la résolution de CSP*

Philippe Jégou[1] **Hélène Kanso**[2] **Cyril Terrioux**[1]

[1] Aix Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France
[2] Effat University, Jeddah, Arabie Saoudite
{philippe.jegou, cyril.terrioux}@lis-lab.fr hkanso@effatuniversity.edu.sa

Résumé

La notion de décomposition arborescente est un sujet important pour l'étude et la résolution de problèmes NP-difficiles en intelligence artificielle, et notamment en programmation par contraintes. D'un point de vue théorique, son exploitation, dans le cadre général des modèles graphiques (réseaux bayésiens, CSP (pondérés), ...), a conduit, sous certaines hypothèses, à la définition d'algorithmes de résolution polynomiaux. Par ailleurs, ces dernières années, des solveurs l'exploitant ont fait montre de leur intérêt pratique pour la résolution de problèmes de décision, d'optimisation ou de comptage.

Dans cet article, nous nous intéressons aux décompositions arborescentes optimales et à leur intérêt pratique pour la résolution d'instances CSP. La motivation de ce travail est double. D'une part, des progrès considérables ont été accomplis au niveau des méthodes calculant de telles décompositions, rendant envisageable leur exploitation pratique. D'autre part, la complexité temporelle des méthodes de résolution exploitant une décomposition arborescente est directement liée à la largeur de la décomposition employée et donc employer une décomposition optimale permet théoriquement de minimiser cette complexité. Nous évaluons d'abord la capacité des méthodes calculant des décompositions optimales à décomposer des instances CSP avant de mesurer l'apport de ces décompositions optimales au niveau de l'efficacité de la résolution.

Ce papier est un résumé de [2].

1 Contexte

Une instance CSP (pour Problème de Satisfaction de Contraintes) est définie par la donnée d'un triplet (X, D, C), où $X = \{x_1, \ldots, x_n\}$ est un ensemble de n variables, $D = \{d_{x_1}, \ldots, d_{x_n}\}$ est un ensemble de domaines finis de taille au plus d, et $C = \{c_1, \ldots, c_e\}$ est un ensemble de e contraintes. Chaque contrainte c_i est un couple $(S(c_i), R(c_i))$, où $S(c_i) = \{x_{i_1}, \ldots, x_{i_k}\} \subseteq X$ définit la *portée* de c_i, et $R(c_i) \subseteq d_{x_{i_1}} \times \cdots \times d_{x_{i_k}}$ est une *relation de compatibilité*. La structure d'une instance CSP est donnée par un hypergraphe, appelé *hypergraphe de contraintes*, dont les sommets correspondent aux variables et les arêtes aux portées des contraintes. Une affectation d'un sous-ensemble de X est dite *cohérente* si toutes les contraintes portant sur ce sous-ensemble sont satisfaites. Une *solution* est une affectation cohérente de toutes les variables.

Déterminer si une instance CSP possède une solution est un problème NP-complet. Il en découle que les algorithmes de résolution habituellement employés ont une complexité en $O(exp(n))$. Toutefois, certains algorithmes, comme BTD [3], sont capables de fournir de meilleures bornes de complexité temporelle en exploitant une *décomposition arborescente* [4] de l'hypergraphe de contraintes pour identifier des sous-problèmes indépendants.

Définition 1 *Une* décomposition arborescente *d'un graphe $G = (X, C)$ est un couple (E, T) où $T = (I, F)$ est un arbre (I est un ensemble de nœuds et F un ensemble d'arêtes) 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) pour chaque 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 est sur un chemin de i à j dans T, alors $E_i \cap E_j \subseteq E_k$. La largeur d'une décomposition est égale à $max_{i \in I}|E_i| - 1$. La largeur arborescente dite* treewidth *w^* de G est la largeur minimale pour toutes les décompositions arborescentes de G.*

*Ce travail est soutenu par l'Agence Nationale de la Recherche dans le cadre du projet DEMOGRAPH (ANR-16-C40-0028)

Leur complexité temporelle peut alors s'exprimer en $O(exp(w))$, avec w la largeur de la décomposition arborescente employée ($w < n$), pour une complexité spatiale en $O(exp(s))$, avec s la taille de la plus grande intersection entre deux clusters ($s \leq w$). D'un point de vue théorique, elle semble d'autant plus intéressante que w est proche de w^* et, à ce titre, le calcul d'une décomposition arborescente de petite largeur paraît constituer une étape importante en vue de la résolution efficace d'instances CSP.

2 Calculs de décompositions

De nombreuses méthodes de calculs de décompositions arborescentes ont été proposées dans la littérature. Nous pouvons les diviser en trois catégories : les méthodes exactes, les méthodes approchées et les méthodes heuristiques. Les méthodes exactes ont pour objectif de calculer des décompositions optimales (c'est-à-dire dont la largeur est w^*). Ce problème étant NP-difficile, elles ont une complexité en temps exponentielle, rendant généralement trop coûteuse leur utilisation. Toutefois, des progrès considérables ont été réalisés dans le cadre du défi PACE [1], rendant désormais possible la décomposition de graphes ayant plusieurs centaines de sommets contre quelques dizaines précédemment. Les méthodes approchées, elles, fournissent des garanties sur la qualité de la largeur obtenue. Malheureusement, à l'heure actuelle, elles s'avèrent trop coûteuses en temps, surtout vis-à-vis de la qualité des décompositions calculées. Enfin, les méthodes heuristiques permettent de calculer rapidement des décompositions arborescentes, mais n'offrent aucune garantie sur la qualité de la largeur obtenue.

3 Résultats expérimentaux

Nous comparons les méthodes exactes *larisch*, *tamaki* et *bannach* [1] à des méthodes heuristiques avec, d'une part, Min-Fill et MCS (qui visent à minimiser la largeur) et, d'autre part, Connected et Small-sep (dont l'objectif est de calculer des décompositions adaptées à la résolution d'instances CSP). Pour cela, nous avons considéré 7 597 instances CSP au format XCSP3.

Du point de vue de la capacité à décomposer, les méthodes heuristiques parviennent à décomposer quasiment toutes les instances. Si les méthodes exactes ne font pas aussi bien dans le temps imparti (1 800 s par instance), elles réussissent tout de même à décomposer au moins 75 % des instances. De plus, grâce aux largeurs optimales ainsi calculées, nous avons pu établir que Min-Fill produit des décompositions dont la largeur est souvent proche de l'optimum.

Maintenant, si nous nous intéressons à l'efficacité de la résolution avec un algorithme comme BTD (figure 1), nous pouvons à nouveau constater que les méthodes heuristiques conduisent à obtenir de meilleurs résultats. Au niveau de ces heuristiques, Connected et Small-sep permettent à BTD de résoudre plus d'instances qu'avec Min-Fill ou MCS. Une des explications réside dans la valeur du paramètre s. En effet, nous avons pu observer que, pour Min-Fill et MCS comme pour les méthodes exactes, la valeur de s est souvent proche, voire égale, de la largeur de la décomposition calculée. Si nous considérons uniquement les temps de résolution (c'est-à-dire sans compter le temps de décomposition), l'emploi de décompositions optimales peut se révéler pertinent pour certaines instances. Par contre, si nous prenons en considération le temps total, le calcul de décompositions optimales reste souvent encore trop coûteux pour permettre une résolution d'instances CSP efficace.

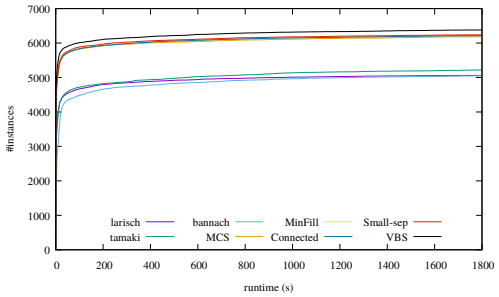

FIGURE 1 – Nombre cumulé d'instances résolues.

4 Conclusion

Contrairement à ce que peut laisser penser la théorie, l'emploi d'une décomposition optimale n'est pas le gage d'une résolution d'instances CSP plus efficace. Différents paramètres influent sur l'efficacité de la résolution, la largeur de la décomposition employée n'étant que l'un d'eux. Il est à noter que nous avons réalisé des observations similaires dans le cadre de la résolution d'instances WCSP ou du problème #CSP.

Références

[1] H. DELL, C. KOMUSIEWICZ, N. TALMON et M. WELLER : The PACE 2017 Parameterized Algorithms and Computational Experiments Challenge : The Second Iteration. *In IPEC*, pages 30 :1–30 :12, 2018.

[2] P. JÉGOU, H. KANSO et C. TERRIOUX : On the Relevance of Optimal Tree Decompositions for Constraint Networks. *In Proceedings of ICTAI*, pages 738–743, 2018.

[3] P. JÉGOU et C. TERRIOUX : Hybrid backtracking bounded by tree-decomposition of constraint networks. *AIJ*, 146:43–75, 2003.

[4] N. ROBERTSON et P.D. SEYMOUR : Graph minors II : Algorithmic aspects of treewidth. *Algorithms*, 7:309–322, 1986.