Un compromis temps-espace pour la résolution de réseaux de contraintes par décomposition

Philippe Jégou, Cyril Terrioux

Un compromis temps-espace pour la résolution de réseaux de contraintes par décomposition

Philippe Jégou **Cyril Terrioux**

LSIS - UMR CNRS 6168
Université d'Aix-Marseille 3
Av. Escadrille Normandie-Niemen
13397 Marseille Cedex 20 (France)
{philippe.jegou, cyril.terrioux}@univ.u-3mrs.fr

Résumé

Nous revenons ici sur une méthode de résolution de CSP par décomposition introduite dans [16] et qui est appelée Regroupement Cyclique. Alors que [16] se limitait à présenter uniquement les principes de la méthode, dans cette contribution, nous montrons comment celle-ci peut être rendue opérationnelle, notamment par une exploitation idoine des propriétés des sous-graphes triangulés. Dans un second temps, nous présentons des résultats formels qui démontrent que le Regroupement Cyclique réalise effectivement un compromis temps-espace en termes de complexités théoriques. Nous concluons cet article en présentant quelques résultats expérimentaux qui montrent que le Regroupement Cyclique. peut être efficace en pratique.

Abstract

We study here a CSP decomposition method introduced in [16] and called Cyclic-Clustering. While [16] only presents the principles of the method, this paper explains how this method can be made operational by exploiting good properties of triangulated induced subgraphs. After, we give formal results which show that Cyclic-Clustering proposes a time-space trade-off w.r.t. theoretical complexities. Finally, we present some preliminary experiments which show that Cyclic-Clustering based on an hybrid adaptation of Tree-Clustering called BTD [18] may be efficient in practice.

1 Introduction

Le formalisme CSP (Constraint Satisfaction Problem) constitue un cadre puissant pour la représentation et la résolution efficace de nombreux problèmes. En particulier, de nombreux problèmes académiques ou réels peuvent être formulés dans ce cadre qui permet notamment l'expression de problèmes NP-complets. Généralement, les CSP sont résolus par différentes versions de la recherche de type "backtrack" dont la complexité est exponentielle dans la taille de l'instance. Néanmoins, des résultats théoriques tels que ceux présentés dans [10] ont montré que, pour le cas de certaines instances, de meilleures bornes de complexité pouvaient être proposées. Ces nouvelles bornes sont obtenues en appliquant des méthodes de décomposition qui fonctionnent en exploitant des propriétés topologiques du réseau de contraintes, c'est-à-dire du graphe (ou de l'hypergraphe) représentant la structure du problème. Par exemple, dans [6, 9], deux stratégies de décomposition ont été proposées, la méthode de l'*ensemble coupe-cycle* (CCM) et le *Tree-Clustering* (TC). Néanmoins, alors que leur intérêt théorique semble tout à fait avéré (voir [13] pour une telle analyse), leur avantage sur un plan pratique est loin d'être démontré.

D'abord dans [16] et puis dans [8], des compromis entre complexité en temps et en espace ont été proposés afin de rendre cette classe d'approches utilisable en pratique. Plus récemment encore, dans [18], une approche hybride de la décomposition de CSP a été proposée. Cette approche est hybride au sens où elle combine le backtracking avec la décomposition structurelle. Cette méthode a montré son intérêt pratique pour la résolution de CSP difficiles. Aussi, il semble que de tels compromis ainsi que la notion de méthodes hybrides, permettent de proposer des implémentations réalistes pour les méthodes structurelles. Cet article étudie cette orientation en analysant et en essayant de rendre exploitable la méthode dite du *Regroupement*

cyclique (Cyclic-Clustering, noté CC [16]). Il faut rappeler ici que l'article [16] se limitait essentiellement à la description de ses motivations et de ses principes. Par exemple, il ne fournissait aucune indication précise pour une exploitation pratique de CC. De plus, aucune analyse théorique, ni pratique, n'était présentée dans [16], et n'a depuis, été réalisée. Aussi, au regard des derniers développements réalisés au niveau de la décomposition de CSP, et en particulier des approches qui exploitent des compromis entre temps et espace, il nous a semblé pertinent d'étudier l'approche de la méthode CC.

La première étape de cette méthode, telle qu'elle était introduite dans [16], consistait en la détermination des cliques maximales du réseau de contraintes initial (du graphe donc). Cette approche peut se révéler d'un intérêt limité car le nombre de cliques maximales dans un réseau de contraintes peut être exponentiel dans le nombre de ses variables [24]. Aussi, l'exploitation pratique de CC requiert-elle une étude approfondie afin de contourner déjà cette première difficulté. Ceci est rendu possible par la démarche suivante, qui s'appuie sur la notion de sous-graphe induit triangulé. Alors que TC fonctionne en approximant la décomposition arborescente optimale d'un réseau de contraintes, notre approche de CC détermine d'abord un sous-graphe du réseau de contraintes du problème considéré, sous-graphe qui se révèle déjà structuré en arbre. Cette partie, interne au CSP de départ, possède donc des propriétés très utiles pour une résolution efficace. De plus, une propriété, simple mais très utile sur les hypergraphes, nous permet de construire, sur la base de ce sous-graphe, un CSP équivalent au problème de départ. Une partie de celui-ci sera arborescente, alors que l'autre correspondra exactement à un ensemble coupe-cycle. L'analyse du réseau est fondée sur des propriétés des graphes triangulés.

Notre nouvelle description de CC nous permet de présenter de nouveaux résultats sur la complexité des méthodes de décompositions, qui tendent à prouver que cette technique constitue un compromis entre TC et CCM. Par exemple, nous démontrons ici que les complexités théoriques en temps et en espace de CC se situent entre celles de TC et de CCM.

Finalement, nous présentons une implémentation de CC qui est basée sur BTD [18]. Cette implémentation permet à CC d'obtenir des résultats expérimentaux assez intéressants en pratique.

Cet article est organisé comme suit. La section 2 introduit les définitions essentielles au sujet du formalisme CSP et des méthodes de décomposition comme TC et CCM. La section 3 introduit les fondations théoriques de CC, puis la section suivante présente une comparaison formelle entre CC, TC et CCM. La section 5 traite des questions relatives à la mise en oeuvre effective de CC et présente quelques résultats expérimentaux préliminaires.

2 Réseaux de contraintes et méthodes de décomposition

2.1 Problèmes de satisfaction de contraintes

Un *problème de satisfaction de contraintes* (CSP) également appelé *réseau de contraintes*, consiste en la donnée d'un ensemble de variables qui doivent être affectées chacune à une valeur issue d'un domaine fini, de telle sorte que toutes les contraintes soient satisfaites. Formellement, un CSP est ainsi défini par un quadruplet (X, D, C, R). X est un ensemble $\{x_1, \ldots, x_n\}$ de n variables. Chaque variable x_i prend sa valeur dans un domaine fini associé D_i, domaine figurant dans D. Les variables sont soumises à des contraintes définies par C. Chaque contrainte C_i est définie par la donnée d'un ensemble $\{x_{i_1}, \ldots, x_{i_{j_i}}\}$ des variables qu'elle contraint ainsi que par celle d'une relation R_i (de R) telle que R_i représente l'ensemble des tuples sur $D_{i_1} \times \cdots \times D_{i_{j_i}}$ qui permettent de satisfaire la contrainte. Une solution du CSP est une affectation de toutes les variables à des valeurs qui permettent de satisfaire toutes les contraintes.

Pour un CSP $\mathcal{P}$, l'hypergraphe (X, C) est appelé hypergraphe de contraintes. Un CSP est dit binaire si toutes ses contraintes sont binaires, i.e. elles concernent chacune exactement deux variables. Dans ce cas, (X, C) est en fait un graphe (appelé graphe de contraintes) associé à (X, D, C, R). Pour un CSP donné, le problème consiste soit à trouver toutes les solutions, soit une solution, ou encore savoir s'il en existe une. Le problème de décision (existence d'une solution) est connu pour être NP-complet, celui du calcul d'une solution étant NP-difficile.

En général, les CSP sont résolus par différentes versions de la recherche backtrack, dont la complexité est alors exponentielle dans la taille de l'instance considérée. Par conséquent, de nombreux travaux du domaine ont cherché à améliorer l'efficacité de la recherche. L'un des résultats essentiels du domaine, présenté dans le contexte limité des CSP binaires, est donné dans [10]. Dans cet article, E. Freuder introduit une procédure de prétraitrement dont l'objet est de construire un bon ordre sur les variables, avant l'exécution d'un algorithme de recherche. E. Freuder exhibe ensuite une condition qui permet d'affirmer qu'aucun backtrack ne sera opéré pendant la recherche. Cette condition est liée à la fois à une propriété structurelle vérifiée par le réseau de contraintes, ainsi qu'à la vérification par celui-ci, d'une propriété de cohérence locale. La pro-

priété la plus intéressante s'énonce alors par :

Théorème 1 ([10]) *Un CSP binaire arc-consistant dont le graphe de contraintes est acyclique admet une solution et il existe un ordre de recherche glouton (i.e. sans backtrack) pour la trouver.*

Notons dès à présent que cette propriété est également vérifiée pour le cas de contraintes d'arité quelconque, et dont le réseau est donc constitué par un hypergraphe de contraintes [15]). Cette propriété montre que la résolution effective (*tractability*) d'un CSP est intimement liée à des propriétés topologiques, c'est-à-dire structurelles, de son graphe de contraintes sous-jacent.

Cette propriété a conduit plusieurs chercheurs à proposer des méthodes qui permettraient de résoudre des CSP quelconques, à savoir des CSP cycliques. On retiendra principalement la méthode dite de l'ensemble coupe-cycle [6] et le tree-clustering [9] que nous présentons succinctement dans les sections suivantes.

2.2 Tree-clustering (TC)

Parmi les méthodes basées sur la notion de *décomposition arborescente de graphes* [21], nous avons choisi, sans manque de généralité, et pour des raisons de simplicité, de décrire TC. Cependant, notre travail peut être adapté à d'autres méthodes de décomposition également basées sur la décomposition arborescente de graphes [13, 7].

TC procède en construisant un CSP défini à partir de regroupements de variables (clusters) de sorte que l'interaction existant entre clusters soit alors structurée en arbre. Le nouveau CSP est équivalent à celui d'origine (i.e. il possède le même ensemble de solutions), mais l'hypergraphe de contraintes associé est maintenant acyclique. Aussi, la propriété concernant les CSP acycliques est alors applicable à cette nouvelle représentation du problème de départ. Pour présenter TC dans le détail, puis ensuite le Regroupement Cyclique, nous devons rappeler quelques définitions issues de la théorie des graphes et des hypergraphes :

Définition 1 *La 2-section (ou graphe primal) d'un hypergraphe (X, C) est le graphe (X, E) avec $E = \{\{X_i, X_j\} \mid \exists C_k \in C$ telle que $\{X_i, X_j\} \subseteq C_k\}$.*

Définition 2 *Un graphe est dit triangulé si tout cycle de longueur au moins égale à 4 possède une corde, i.e. une arête joignant deux sommets non-consécutifs le long du cycle.*

Définition 3 *Etant donné un graphe $G = (X, C)$ et un ordre $\sigma = (x_1, x_2, \ldots, x_n)$ sur X, les successeurs d'un sommet x_i sont les éléments de l'ensemble $\{x_j \in X \setminus \{x_i\} : \{x_i, x_j\} \in C$ and $j > i\}$. Nous dirons qu'un sommet x_i est simplicial pour σ si les successeurs de x_i sont deux à deux adjacents. Nous dirons qu'un ordre $\sigma = (x_1, x_2, \ldots, x_n)$ de X est un ordre d'élimination parfait si, pour $i = 1, \ldots, n$, x_i est simplicial pour l'ordre σ.*

Théorème 2 ([11]) *Un graphe $G = (X, C)$ est dit triangulé si et seulement si X admet un ordre d'élimination parfait dans G.*

Différentes définitions équivalentes de la notion d'acyclicité d'hypergraphe ont été proposées dans la littérature. Ici, nous considérerons celle relative aux graphes triangulés :

Définition 4 *Un hypergraphe est dit conforme si toutes les cliques maximales de sa 2-section correspondent à une arête dans l'hypergraphe original.*

Définition 5 ([2, 3]) *Un hypergraphe est acyclique s'il est conforme et si sa 2-section est triangulée.*

La procédure TC comporte plusieurs étapes. La première est basée sur la triangulation de la 2-section. Cette opération, qui est réalisée bien sûr uniquement si ce graphe n'est pas déjà triangulé, ajoute de nouvelles arêtes afin de produire une 2-section triangulée [23], dont l'hypergraphe doit ensuite être déterminé. C'est l'objet de la seconde étape, puisqu'elle va identifier les cliques maximales $\{\mathcal{C}'_1, \ldots, \mathcal{C}'_m\}$ dans la 2-section triangulée [12]. L'étape suivante fournit l'hypergraphe acyclique dont les arêtes correspondent aux cliques maximales de la 2-section. Cet hypergraphe possède nécessairement par construction, une 2-section triangulée et il est conforme. La complexité en temps de ces différentes étapes est $O(n + e')$ où n est le nombre de sommets et e' le nombre d'arêtes du graphe triangulé. Une fois ces étapes réalisées, il faut alors définir le CSP résultant qui est obtenu en résolvant chaque sous-problème défini par les clusters (cliques maximales de l'hypergraphe) $\mathcal{C}'_1, \ldots, \mathcal{C}'_m$, chaque sous-problème étant constitué des variables appartenant au cluster correspondant. Les contraintes à considérer sont alors celles figurant à l'origine entre les variables. La résolution de chaque sous-problème va produire une liste de tuples consistants qui correspondent aux solutions. La complexité en temps de ces différentes étapes est $O(m.d^{W+1})$ où d est la taille du plus grand domaine et $W + 1$ l'arité de la plus grande contrainte dans le nouveau CSP (elle est égale à la taille de la plus grande clique dans la 2-section triangulée). Finalement, nous obtenons un CSP globalement consistant en utilisant un filtrage par *inter-consistance* ou *pairwise-consistency* [15], qui équivaut à l'exécution de la consistance d'arc sur les tuples. La complexité

en temps est $O(n.d^{W+1}.(W+1).log(d))$. Notons que la complexité en espace est $O(m.d^{W+1})$ mais qu'il est possible de la limiter à $O(m.d^S)$, comme c'est par exemple le cas avec l'adaptation de TC fournie par la méthode "Cluster Tree Elimination" [7]). Dans ce cas, S est la taille de la plus grande intersection entre paires de clusters, c'est-à-dire la taille maximum des séparateurs minimaux du graphe.

Enfin, il faut rappeler ici que le problème qui consiste à calculer la meilleure décomposition arborescente, c'est-à-dire la décomposition pour laquelle la valeur de W est minimum, constitue un problème qui est NP-difficile. Pour plus de détails, voir [7].

Par la suite, nous identifierons sous le sigle TC, les méthodes basées comme TC [9], sur la décomposition arborescente de réseaux de contraintes. Il s'agit par exemple de *Join Tree Elimination* [7] ou encore de *Cluster Tree Elimination* [7].

Alors que ces méthodes semblent fournir les approches les plus efficaces pour la résolution de CSP, du moins pour ce qui concerne la complexité théorique en temps (voir l'analyse de [13]), leur intérêt pratique est hélas très limité. Les raisons de cet état de fait sont que le temps de calcul effectif est finalement très important, et de plus qu'elles nécessitent l'allocation et l'exploitation d'un espace mémoire trop important pour résoudre des cas pratiques réels. Aussi, dans [18], une approche alternative, appelée BTD, a été proposée. Son objet est de limiter la taille de l'espace requis, et d'éviter également de nombreux traitements redondant lors de la recherche. Alors que TC calcule toutes les solutions de chaque cluster, puis mémorise les affectations (tuples) correspondantes sur les intersections entre clusters (pour les meilleures approches), BTD ne calcule qu'une partie de ces solutions et donc, n'en enregistre que certaines. Dans BTD, ces affectations sont appelées *goods* si elle peuvent être étendues à des affectation consistantes sur la partie de CSP qui est connectée à cette intersection, et elles sont appelées *nogoods* si ce n'est pas possible. Aussi, comme pour les autres méthodes procédant par décomposition arborescente et basées sur TC, la complexité en espace de BTD est limitée car elle est bornée par $O(m.d^S)$. Toutefois, ce pire des cas n'est généralement pas atteint en pratique grâce à l'enregistrement des goods et des nogoods. Du fait de ce compromis, BTD peut résoudre des instances que ni TC, ni ses diverses optimisations ne peuvent appréhender.

2.3 La méthode de l'ensemble coupe-cycle (CCM)

Un ensemble coupe-cycle d'un graphe (respectivement d'un hypergraphe) est un ensemble de sommets dont la suppression induit un graphe acyclique (resp. un hypergraphe acyclique). CCM [6] est basée sur le fait que l'affectation de variables change, d'une certaine façon, la connectivité effective du graphe de contraintes traité. Aussi, lors d'une recherche, dès que toutes les variables d'un ensemble coupe-cycle ont été affectées, le graphe de contraintes résultant ne possède alors plus aucun cycle. Le sous-problème associé est alors acyclique et le théorème 1 peut donc être exploité. Une propriété permet de résumer la méthode :

Propriété 1 *Si toutes les variables qui appartiennent à un ensemble coupe-cycle d'un graphe de contraintes (respectivement un hypergraphe de contraintes) sont affectées en satisfaisant les contraintes qui portent sur elles, et si le CSP résultant d'une application d'un filtrage par consistance d'arc (respectivement par inter-consistance) ne voit aucun domaine vide, alors le problème admet des solutions et tolère un ordre de recherche glouton.*

Le test de consistance d'un CSP peut donc s'opérer, d'abord en affectant de façon consistante, les variables d'un ensemble coupe-cycle, puis en propageant par consistance d'arc l'effet de cette affectation. La taille de l'ensemble coupe-cycle constitue alors la "hauteur" du backtracking nécessaire à la résolution du problème. Plus précisément, pour un CSP binaire, la complexité en temps de CCM est $O(e.d^{K+2})$ où e est le nombre de contraintes et K la taille de l'ensemble coupe-cycle. Le coût nécessaire à la recherche d'une affectation consistante sur l'ensemble coupe-cycle est $O(d^K)$. Pour chaque solution de l'ensemble coupe-cycle (i.e. une affectation consistante), il est nécessaire de résoudre le CSP acyclique induit. La complexité en temps est alors de $O((n-K).d^2)$, et par conséquent, nous obtenons globalement $O(e.d^{K+2})$. On peut constater que le paramètre le plus important dans cette complexité est K, et que le problème d'optimisation visant à minimiser K est NP-difficile [20]. Cette méthode peut être étendue aux CSP non-binaires, la complexité en temps étant alors $O(e.r.log(r).d^K)$ où r est la taille (nombre de tuples) de la relation la plus grande associée aux contraintes dans le CSP [17]. Notons qu'après l'affectation de l'ensemble coupe-cycle, le CSP acyclique induit peut dans ce cas être résolu par le même type de procédure que TC, car on se situe alors dans un CSP qui n'est pas nécessairement binaire.

2.4 Comparaison des méthodes

L'avantage le plus important de TC concerne la nature du résultat effectivement obtenu par la méthode. TC calcule une représentation du problème qui permet l'obtention des solutions sans backtracking à l'issue de la phase de prétraitement. A l'opposé, CCM n'offre pas

ce type d'avantage car toute solution doit être calculée à l'aide d'une recherche par backtracking, qui sera bornée par la taille de l'ensemble coupe-cycle.

Tout d'abord dans [4], puis récemment dans [13], il a été démontré que si l'on considère les valeurs optimales de W et K, nous aurons $W+1 \leq K+2$. Ainsi, TC est théoriquement meilleur que CCM pour ce qui concerne la complexité en temps. Toutefois, ni TC ni CCM n'ont à ce jour démontré leur réel intérêt pratique. Comme cela est affirmé dans [18], ce phénomène est dû à la complexité en espace observée en pratique pour TC, alors que pour CCM, cet état de fait est probablement dû à la complexité en temps.

3 Mise en oeuvre du regroupement cyclique (CC)

Afin d'éviter les inconvénients de TC et de CCM, une approche alternative appelée *regroupement cyclique* (CC) a été introduite dans [16]. Notons que cet article se limitait uniquement à présenter les idées de l'approche sans éclaircissement sur la mise en oeuvre effective de la méthode. Tel que présenté dans [16], CC fonctionne en quatre étapes. La première étape consiste en la recherche de toutes les cliques maximales du graphe de contraintes initial. La deuxième étape considère chaque clique maximale pour résoudre le sous-problème associé. Le résultat consiste en un hypergraphe de contraintes dont les contraintes correspondent aux différents sous-problème résolus. Les deux dernières étapes déterminent d'abord un ensemble coupe-cycle, puis pour terminer, lancent l'exécution de CCM sur ce nouveau CSP. L'exécution de CC n'est malheureusement pas efficace pour des raisons d'ordre à la fois théorique et pratique. Par exemple, il est bien connu que pour un graphe quelconque, le nombre de ses cliques maximales peut être exponentiel dans le nombre de ses sommets [24] et ainsi, conduire à une première étape difficilement réalisable en pratique. Dans cet article, nous présentons une autre approche pour la mise en oeuvre de CC, originellement introduite dans [17]. Cette approche est basée sur l'exploitation des propriétés combinatoires et algorithmiques des graphes triangulés. La figure 1 résume l'approche. La première étape de CC (a) identifie un sous-graphe triangulé (Triangulated Induced Subgraph, noté TIS) du graphe (tous les sommets et arêtes appartiennent au TIS, exceptés les sommets 2 et 4 et les arêtes en pointillés). La deuxième étape génère un CSP n-aire acyclique basé sur le TIS (b). La dernière étape résout le CSP initial en appliquant la méthode de l'ensemble coupe-cycle sur le nouveau problème, puisque les sommets qui ne figurent pas dans le TIS constituent un ensemble coupe-cycle du nouveau

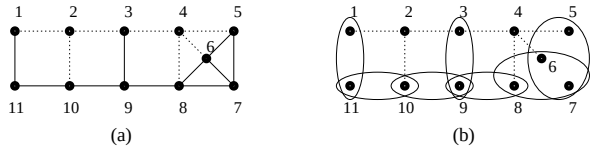

FIG. 1 – Une décomposition par regroupement cyclique.

problème. Dans cet exemple, ce sont les sommets 2 et 4. Les justifications théoriques de cette approche sont basées sur les propriétés des TIS.

Définition 6 *Etant donné un graphe $G = (X, C)$, si T est un sous-ensemble de X, $G(T) = (T, C(T))$ est le sous-graphe de G induit par les sommets de T, c'est-à-dire $C(T) = \{\{x_i, x_j\} \in C : x_i, x_j \in T\}$. Le graphe $G(T)$ est un sous-graphe induit triangulé (TIS) de G si et seulement si $G(T)$ est triangulé. $G(T)$ est TIS Maximal (MTIS) si T est maximal pour l'inclusion (il n'est pas inclus strictement dans un sous-ensemble $T' \subseteq X$ tel que $G(T')$ soit triangulé.*

Dans [1], Balas et Yu définissent un algorithme appelé `TRIANG` pour le calcul d'un MTIS. Sa complexité en temps est $O(n+e)$. Notons que le MTIS n'est pas nécessairement de taille maximum, mais qu'il s'agit toujours d'un sous-graphe maximal au sens de l'inclusion. Ceci n'est pas surprenant parce que le problème de la recherche d'un sous-graphe maximum triangulé est un problème NP-difficile [20, 25].

Pour simplifier, et sans manque de généralité, nous présentons notre approche uniquement sur les graphes de contraintes, donc les CSP binaires. Pour étendre cette méthode aux CSP n-aires, il suffit de considérer la 2-section comme dans TC. Etant donné le graphe de contraintes initial, la première étape consiste en la recherche d'un TIS. En exploitant les propriétés des hypergraphes acycliques, la seconde étape génère un hypergraphe qui possède deux types d'arêtes. Le premier type d'arêtes correspond aux cliques maximales du TIS et le second aux arêtes du graphe initial qui ne figurent pas dans le sous-graphe. La 2-section de cet hypergraphe correspond au graphe de contraintes initial et on connaît alors un ensemble coupe-cycle de cet hypergraphe, ensemble correspondant aux sommets qui ne figurent pas dans le TIS. Cet hypergraphe permet de générer un nouveau CSP dont toutes les arêtes correspondent aux nouvelles contraintes. Les tuples des relations de compatibilité de ces nouvelles contraintes sont produits en énumérant toutes les affectations partielles consistantes des variables appartenant aux contraintes. La dernière étape consiste en l'application de CCM.

Définition 7 *Etant donné un graphe $G = (X, C)$:*
- *$H_{max}(G) = (X, \mathcal{C}')$ désigne l'hypergraphe induit par les cliques maximales de G, c'est-à-dire que $\mathcal{C}'$ est l'ensemble des cliques maximales de G.*
- *étant donné un sous-ensemble Y de X, $E_G(Y) = \{c \in C : c \cap Y \neq \emptyset\}$, i.e. $E_G(Y)$ est l'ensemble des arêtes qui possèdent au moins un sommet dans Y.*

Définition 8 *Etant donnés un hypergraphe $H = (X, C)$ et Y, un sous-ensemble de X :*
- *$H(Y) = (Y, C')$ désigne l'hypergraphe induit par l'ensemble de sommets Y, où $C' = \{c' \subseteq Y : \exists c \in C, c' \subseteq c$ et c' est maximal $\}$.*
- *si Y est tel que $H(X \setminus Y)$ est acyclique, alors Y est un ensemble coupe-cycle de H.*

Propriété 2 *Etant donnés un graphe $G = (X, C)$ et un sous-ensemble T de X tel que $G(T)$ est triangulé, alors l'hypergraphe $H_{max}(G(T))$ est acyclique.*

Notons dès à présent que pour des questions de place, nous ne donnerons ici aucune preuve. Le lecteur intéressé pourra les trouver dans [19].

Propriété 3 *Etant donné un graphe $G = (X, C)$, un sous-ensemble T de X et $\mathcal{C}'$ l'ensemble des cliques maximales de $G(T)$, c'est-à-dire $H_{max}(G(T)) = (T, \mathcal{C}')$, alors si $G(T)$ est triangulé, $X \setminus T$ est un ensemble coupe-cycle de l'hypergraphe $H_T = (X, \mathcal{C}' \cup E_G(X \setminus T))$.*

Propriété 4 *Etant donné un graphe $G = (X, C)$ et T est un sous-ensemble de X ; si $G(T)$ est un MTIS de G, alors $Y = X \setminus T$ est un ensemble coupe-cycle minimal de $H_T = (X, \mathcal{C}' \cup E_G(Y))$ où $\mathcal{C}'$ est l'ensemble des cliques maximales de $G(T)$.*

Définition 9 *Etant donné un CSP binaire $\mathcal{P}_G = (X, D, C, R)$ de graphe $G = (X, C)$, et avec T sous-ensemble de X tel que $G(T)$ est triangulé, le CSP induit par T sur $\mathcal{P}_G$ est $\mathcal{P}_{H_T} = (X, D, C_T, R_T)$ qui est défini par :*
- *$C_T = \mathcal{C}' \cup E_G(Y)$ où $\mathcal{C}'$ est l'ensemble des cliques maximales de $G(T)$, c'est-à-dire $H_{max}(G(T)) = (T, \mathcal{C}')$ et $Y = X \setminus T$*
- *$R_T = \{R_i \in R : C_i \in E_G(Y)\} \cup \{R'_i : \mathcal{C}'_i \in \mathcal{C}'$ et $R'_i = \bowtie_{C_j \subseteq \mathcal{C}'_i} R_j\}$ où le symbole $\bowtie$ désigne l'opérateur de jointure de la théorie des bases de données relationnelles.*
- *$H_T = (X, C_T)$.*

Si $G(T)$ est un MTIS de G, alors $\mathcal{P}_{H_T}$ est appelé CSP maximal induit par T sur $\mathcal{P}_G$.

Nous appliquons cette définition à l'exemple donné dans la figure 1. Ici, $T = X \setminus \{2, 4\}$ et donc $C_T = \{\{1,2\}, \{1,11\}, \{2,3\}, \{2,10\}, \{3,4\}, \{3,9\}, \{4,5\}, \{4,6\}, \{4,8\}, \{5,6,7\}, \{6,7,8\}, \{8,9\}, \{9,10\}, \{10,11\}\}$. Les relations associées aux contraintes binaires correspondent aux relations initiales alors que les nouvelles contraintes, définies par des relations ternaires, sont obtenues en résolvant les sous-problèmes associés. Aussi, nous avons $R_{\{5,6,7\}} = R_{\{5,6\}} \bowtie R_{\{5,7\}} \bowtie R_{\{6,7\}}$ et $R_{\{6,7,8\}} = R_{\{6,7\}} \bowtie R_{\{6,8\}} \bowtie R_{\{7,8\}}$.

Finalement, il faut que, le CSP induit maximal $\mathcal{P}_{H_T}$ possède exactement le même ensemble de solutions que $\mathcal{P}_G$.

Théorème 3 *Soit $\mathcal{P}_G = (X, D, C, R)$ un CSP binaire de graphe $G = (X, C)$, T un sous-ensemble de X tel que $G(T)$ est un TIS, et $\mathcal{P}_{H_T} = (X, D, C_T, R_T)$ le CSP induit par T on $\mathcal{P}_G$. L'ensemble de solutions de $\mathcal{P}_G$, noté $Sol_{\mathcal{P}_G}$, est égal à l'ensemble de solutions $\mathcal{P}_{H_T}$, noté $Sol_{\mathcal{P}_{H_T}}$.*

Le théorème 3 résume la méthode : étant donné un CSP binaire $\mathcal{P}_G = (X, D, C, R)$ de graphe G, il est suffisant de déterminer un ensemble T de sommets tel que $G(T)$ est un TIS de G. Pour trouver l'ensemble T, nous pouvons utiliser la procédure `TRIANG`(G, T). Après, nous considérons $G(T)$ et nous calculons ses cliques maximales $\mathcal{C}'$ grâce à une procédure adéquate `CliquesMaxTriangulated`$(G(T), \mathcal{C}')$. A l'étape suivante, nous générons le CSP induit par T sur $\mathcal{P}_G$, $\mathcal{P}_{H_T} = (X, D, C_T, R_T)$ en résolvant chaque sous-problème correspondant à chaque clique dans $\mathcal{C}'$. Nous notons cette procédure `Generate`$(\mathcal{P}_G, T, \mathcal{C}', \mathcal{P}_{H_T})$. Finalement, puisque le sous-ensemble $X \setminus T$ est un ensemble coupe-cycle minimal de l'hypergraphe H_T, nous pouvons appliquer la méthode de l'ensemble coupe-cycle généralisée pour résoudre $\mathcal{P}_{H_T}$, en appelant la procédure `CycleCutsetMethod`$(\mathcal{P}_{H_T}, X \setminus T, Sol_{\mathcal{P}_G})$.

```
1. CyclicClustering(P_G, Sol_{P_G})
2. Begin
3.       TRIANG(G, T) ;
4.       CliquesMaxTriangulated(G(T), C') ;
5.       Generate(P_G, T, C', P_{H_T}) ;
6.       CycleCutsetMethod(P_{H_T}, X \ T, Sol_{P_G})
7. End
```

FIG. 2 – Regroupement Cyclique.

Théorème 4 *Etant donné un CSP binaire $\mathcal{P}_G$, la procédure `CyclicClustering` calcule $Sol_{\mathcal{P}_G}$.*

Théorème 5 *La complexité en temps de `CyclicClustering` est $O(e + n.a.d^{a+k})$ et la complexité en espace est $O(n.s.d^s)$ où s est la taille de la plus grande intersection entre deux clusters, c'est-à-dire la taille maximale des séparateurs minimaux dans le TIS et k est la taille de l'ensemble coupe-cycle.*

Notons que la première étape de cette approche permet de déterminer les paramètres de la complexité, a et k en $O(n+e)$ fournissant ainsi un nouvelle mesure de la complexité d'un CSP dont l'évaluation peut être obtenue en un temps linéaire, sous l'hypothèse de l'emploi de `TRIANG`. Un autre avantage qui est offert par CC réside dans le fait que la "hauteur" totale de backtracking est en fait décomposée en deux parties différentes : l'une relative à la résolution de l'ensemble coupe-cycle (le paramètre k), et l'autre pour résoudre les sous-problèmes (le paramètre a). Finalement, nous pouvons constater que la complexité est bornée par $exp(a+k)$.

Pour appliquer efficacement cette méthode, il nous faut d'abord résoudre un problème d'optimisation car la première étape consiste indirectement, à déterminer les paramètres les plus importants pour ce qui concerne la complexité : a et k. Aussi, il serait intéressant de minimiser k, ce qui revient à maximiser $|T|$. Il nous semble que l'emploi de l'algorithme `TRIANG` ne soit pas nécessairement, dans ce cas, la meilleure approche envisageable. Toutefois, nous ne connaissons pas, à ce jour, de résultats concernant ce problème d'optimisation.

4 Comparaison théorique

Nous considérons ici W et S pour TC, K pour CCM et a, s et k pour CC. Bien que les valeurs optimales de W, S, K, a, s et k soient difficiles à déterminer en pratique car tous les problèmes d'optimisation associés sont NP-difficiles, l'analyse proposée ici prend en compte les valeurs optimales de ces paramètres.

Tout d'abord, il est clair que $s \leq S$. Notons qu'il est facile d'exhiber des exemples de CSP pour lesquels $k + a \ll K$. Plus généralement et formellement, notre premier résultat nous permet d'affirmer que le Regroupement Cyclique est théoriquement meilleur que CCM pour ce qui concerne la complexité en temps.

Théorème 6 *Etant donné un CSP binaire $\mathcal{P}_G = (X, D, C, R)$, si $\mathcal{P}_G$ possède un ensemble coupe-cycle de taille K, alors il existe nécessairement une décomposition par Regroupement Cyclique dont les paramètres a et k vérifient $k + a \leq K + 2$.*

Notons que pour la complexité en espace, CCM est meilleure que CC puisque l'espace requis est linéaire dans le nombre de variables, soit limité à n.

Le théorème suivant nous permet d'affirmer que la décomposition par Regroupement Cyclique est systématiquement meilleure que le tree-clustering pour ce qui concerne la complexité en espace. La preuve de ce résultat est basée sur le fait que la valeur $W + 1$ est toujours plus grande que la taille de la plus grande clique dans un graphe.

Théorème 7 *Etant donné un CSP binaire $\mathcal{P}_G = (X, D, C, R)$, pour toutes les décompositions par Regroupement Cyclique dont les paramètres sont a et k, et pour toutes les décompositions par tree-clustering de paramètre W, nous avons $a \leq W + 1$.*

Le théorème suivant est clairement plus intéressant puisqu'il établit le lien entre les complexités en temps de TC et de CC.

Théorème 8 *Etant donné un CSP binaire $\mathcal{P}_G = (X, D, C, R)$, s'il existe une décomposition par Regroupement Cyclique dont les paramètres sont a et k, alors il existe une décomposition par tree-clustering de paramètre W vérifiant $W + 1 = k + a$.*

En résumé, pour les complexités théoriques, on a la garantie d'obtenir :
- Pour le temps : $CCM \leq CC \leq TC$
- Pour l'espace : $TC \leq CC \leq CCM$

où la notation $X \leq Y$ signifie que la méthode X possède une complexité moins bonne que celle de Y.

5 Mise en oeuvre du Regroupement Cyclique avec BTD

5.1 Deux approches naturelles : CC-BTD$_1$ et CC-BTD$_2$

Du fait de l'intérêt pratique limité de TC, il est naturel d'implémenter CC en remplaçant TC, après l'affectation de l'ensemble coupe-cycle, par BTD dont l'efficacité est largement meilleure que celle de TC [18]. Aussi, nous montrerons ici comment il est possible d'exploiter BTD et nous présenterons une optimisation de son emploi dans CC.

Pour implémenter CC, une approche naturelle peut consister en l'exécution de BTD dès qu'une nouvelle affectation consistante de l'ensemble coupe-cycle est trouvée. Cette approche, notée **CC-BTD$_1$**, est immédiatement envisageable et de plus, elle garantit l'obtention des bornes de complexité fournies par CC. Néanmoins, cette approche n'est pas nécessairement la plus efficace. En effet, dès que CC réalise une nouvelle affectation consistante de l'ensemble coupe-cycle, il faut exécuter BTD, qui calculera et enregistrera notamment à nouveau des "goods" et des "nogoods". Toutefois, il est clair qu'une partie de ce calcul peut être évitée. En effet, considérons une affectation consistante de l'ensemble coupe-cycle. BTD calculera et enregistrera les goods et les nogoods associés à l'affectation considérée. En effet, certains nogoods sont

obtenus parce que l'affectation d'une certaine variable de l'ensemble coupe-cycle conduit à l'inconsistance d'une partie de l'affectation des futures variables. Pour une autre affectation de l'ensemble coupe-cycle, la même affectation de la même variable pourrait cette fois-ci fournir un good. Aussi, étant donnée une affectation consistante de l'ensemble coupe-cycle, il semble impossible d'exploiter les goods et les nogoods produits par un précédent appel de BTD. Néanmoins, il est possible d'exploiter les nogoods produits lors d'un appel préliminaire de BTD. Nous noterons cette seconde approche **CC-BTD$_2$**. Ici, avant de lancer CC, nous réalisons un appel préliminaire de BTD. Il est clair que tous les nogoods enregistrés correspondent à des affectations qui ne pourront jamais être étendues à des affectations complètes consistantes, en particulier en incluant les variables qui appartiennent à l'ensemble coupe-cycle. Aussi, ces nogoods peuvent être exploités pour traiter toutes les affectations consistantes de l'ensemble coupe-cycle pour stopper la recherche. D'autre part, les goods calculés ne peuvent pas être exploités directement car ils ont été calculés sur des sous-problèmes qui ne sont pas nécessairement compatibles avec toute affectation de l'ensemble coupe-cycle.

Avant de présenter des expérimentations sur ces deux approches, notons que l'intégration de BTD dans CC offre déjà un intérêt théorique :

Théorème 9 *Etant donné un CSP $\mathcal{P} = (X, D, C, R)$ et une décomposition par Regroupement Cyclique avec un TIS $G(T)$ et des paramètres a et k. La complexité de **CC-BTD**$_i$ ($i = 1, 2$) est $O(e + n.a.d^{a+k})$ alors que la complexité en espace est $O(n.s.d^s)$ où s est la taille de la plus grande intersection entre deux cliques maximales de $G(T)$.*

5.2 Resultats expérimentaux

Tout d'abord, avant de présenter les résultats que nous avons obtenus, il nous faut noter que l'efficacité de CC dépend fortement du nombre de solutions (ou affectations partielles consistantes) de l'ensemble coupe-cycle. En effet, dans le pire des cas, CC calculera toutes les solutions de l'ensemble coupe-cycle. Aussi, pour des raisons d'efficacité, CC doit s'appuyer sur un ensemble coupe-cycle qui en recèle peu, ce qui conduit directement à la question d'un calcul d'ensemble coupe-cycle qui satisferait ce type de propriété. Une façon de procéder peut consister à calculer un MTIS en utilisant l'algorithme de Balas et Yu [1]. En faisant ainsi, nous obtenons certes un ensemble coupe-cycle de taille raisonnable, mais qui possède parfois trop de solutions. Aussi, avons-nous préféré construire un ensemble coupe-cycle éventuellement de plus grande taille, mais qui recèlerait beaucoup moins de solutions en exploitant non pas l'algorithme TRIANG de Balas et Yu, mais une méthode heuristique sans garantie de minimalité. Cette méthode procède en choisissant un sommet (une variable), en l'enlevant du graphe de contraintes, et en répétant ce processus jusqu'à ce que le graphe résultant soit triangulé. Les sommets supprimés forment alors un ensemble coupe-cycle. Afin de limiter le nombre de solutions de l'ensemble coupe-cycle, lors de chaque étape, le sommet choisi est celui qui correspond à la variable qui est *a priori* la plus contrainte, soit, celle qui partage le plus de contraintes avec les variables déjà sélectionnées.

Nous avons d'abord expérimenté CC-BTD$_i$ sur des instances de CSP binaires aléatoires classiques et nous avons comparé les résultats obtenus avec Forward-Checking [14], MAC [22] et FC-BTD [18]. Pour ordonner les variables dans FC, MAC ou même, dans le traitement des clusters pour FC-BTD et CC-BTD$_i$, nous avons utilisé l'heuristique classique *dom/deg* [5]. Cette heuristique choisit en premier les variable x_i qui minimisent le ratio $\frac{|d_{x_i}|}{|\Gamma_{x_i}|}$ avec d_{x_i} le domaine courant de x_i, et Γ_{x_i} l'ensemble des variables partageant une contrainte avec x_i. Les instances aléatoires dites classiques, ont été produites en utilisant le générateur écrit par D. Frost, C. Bessière, R. Dechter et J.-C. Régin. Ce générateur considère 4 paramètres : n, d, e et t. Il produit ainsi des CSP d'une classe (n, d, e, t) possédant n variables, dont le domaine est de taille d, et il existe au total e contraintes binaires ($0 \leq e \leq \frac{n(n-1)}{2}$) pour lesquelles t tuples sont interdits ($0 \leq t \leq d^2$). Les classes que nous avons considérées se situent dans le voisinage de la transition de phase. Pour chaque classe, nous avons résolu 50 instances, qui dans chaque cas, possédaient un graphe de contraintes connexe.

Les résultats sont donnés dans [19]. Nous avons constaté que CC-BTD$_1$ et CC-BTD$_2$ obtiennent des résultat similaires. L'existence de tels résultats n'est en soi pas surprenant, parce que, comme cela a déjà été observé dans [18], il existe très peu de nogoods structurels dans les instances aléatoires classiques, pour des raisons évidentes intrinsèquement liées à la nature de ces instances. Ainsi, il n'y a que très peu de nogoods qui peuvent être exploités, et qui de fait le seront. Ceci explique que CC-BTD$_i$ possède des performances inférieures à FC ou FC-BTD. Toutefois, CC-BTD$_i$ supplante MAC quand FC s'avère meilleur que MAC. De plus, en dépit de l'absence de propriétés adéquates pour l'exploitation d'une telle méthode structurelle, nous n'avons pas observé de dégradation significative des performances de cette méthode, par rapport aux résultats offerts par FC ou FC-BTD.

Comme il est bien connu que les instances de CSP binaires aléatoires classiques ne recèlent aucune structure pertinente en vue de l'exploitation qui pourrait

Classe	W	K	$a+k$	FC	MAC	FC-BTD	CC-BTD$_1$	CC-BTD$_2$
(50,15,15,75,5,15,80,50)	27,96	46,70	30	42,00	89,98	10,14	4,76	3,35
(50,15,15,76,5,15,80,50)	28,10	46,38	30	24,68	85,14	23,62	3,35	2,44
(50,15,15,71,5,20,130,50)	28,28	50,34	35	46,42	159,96	40,70	0,35	0,35
(50,15,15,72,5,20,130,50)	29,14	50,40	35	211,36	699,47	128,53	0,29	0,30
(50,15,15,77,5,15,80,30)	27,96	46,40	30	1,89	8,64	1,58	4,89	1,49
(50,15,15,78,5,15,80,30)	28,10	46,28	30	39,98	89,56	1,51	3,45	2,15

TAB. 1 – Classes d'instances structurées aléatoires. Paramètres W, K et $a+k$ des décompostions et temps d'exécution (en secondes) de FC, FC-BTD, CC-BTD$_1$ et CC-BTD$_2$.

en être faite pour ce type de décomposition, il nous a semblé nécessaire de réaliser des tests en s'appuyant cette fois-ci, sur des instances de CSP binaires aléatoires structurées. Par instances structurées, nous entendons des instances dont les graphes de contraintes vérifient des propriétés qui ont un sens pour des méthodes telles que CC. Formellement, notre générateur aléatoire prend 8 paramètres. Il construit des CSP de la classe $(n, d, a, t, s, k, e_1, e_2)$ avec $n+k$ variables, chacune possédant un domaine de taille d. Pour chaque contrainte, t tuples sont interdits. Concernant la structure, le générateur construit d'abord un arbre de cliques sur n sommets (comme dans [18]) où a est la taille de la plus grande clique et s celle de la plus grande intersection entre paires de cliques. Notons que tout arbre de cliques est un TIS. Les k variables restantes correspondent alors à un ensemble coupe-cycle pour lesquelles e_1 contraintes sont présentes. Finalement, nous rajoutons e_2 contraintes afin de connecter les variables de l'ensemble coupe-cycle et celles de l'arbre de clique. De cette manière, nous générons des problèmes qui recèlent une structure exploitable. Malheureusement, nous ne possédons par de méthode qui soit en mesure de reconnaître systématiquement une telle structure. Puisque notre objectif est de vérifier l'intérêt de CC-BTD$_i$ dans le cadre de problèmes structurés, et non de juger de la qualité de la méthode du TIS, et donc de celle de l'ensemble coupe-cycle, la structure sera fournie en entrée de CC-BTD$_i$. Les classes de CSP qui ont fait l'objet d'expérimentations sont toutes situées au voisinage de la transition de phases (voir table 1). Pour chaque classe considérée, nous avons résolu 50 instances. Enfin, pour chaque instance, le graphe de contraintes est connexe.

La table 1 montre que CC-BTD$_i$ peut surpasser MAC, FC et FC-BTD quand la structure du problème possède de telles propriétés. Notons par ailleurs qu'en ce qui concerne les paramètres de la complexité (W, K, a et k), TC fournit, comme l'indiquent les résultats théoriques de la section 4, les meilleurs résultats. La résolution qui suit ne reprend cependant pas cette tendance. Cela nous incite à penser que le découpage en deux étapes distinctes de la partie "exponentielle" dans CC, constitue un avantage sur le plan pratique. Nous pouvons aussi observer que CC-BTD$_2$ est meilleur que CC-BTD$_1$. Aussi, les nogoods enregistrés pendant la phase de traitement préliminaire réalisée par l'appel à BTD permettent-ils à CC-BTD$_2$ d'éviter certaines zones de l'espace de recherche potentiel.

Pour résumer, CC-BTD$_i$ peut constituer une approche intéressante pour la résolution de problèmes structurés. La difficulté essentielle concerne alors le problème de reconnaissance de la structure du problème que la méthode pourrait exploiter, en particulier, l'aptitude qu'elle peut avoir à calculer un TIS de bonne qualité, c'est-à-dire conduisant à un ensemble coupe-cycle recelant peu de solutions. Une conséquence immédiate de ce constat est qu'il est maintenant nécessaire de développer des méthodes calculant des TIS avec cet objectif, avant de pouvoir totalement affirmer l'intérêt pratique de CC-BTD$_i$.

6 Conclusion

Dans ce papier, nous avons étudié le Regroupement Cyclique [16]. Alors que [16] se limitait à présenter les principes de la méthode, nous avons expliqué comment cette méthode peut être mise en oeuvre effectivement par l'exploitation de propriétés de sous-graphes triangulés. Nous avons ensuite montré que le Regroupement Cyclique offre un compromis entre temps et espace sur le plan de la complexité théorique. Pour cela, nous avons démontré que sa complexité en temps est inférieure à celle de la méthode de l'ensemble coupe-cycle alors que pour ce qui concerne sa complexité en espace, elle est inférieure à celle du Tree-Clustering. Finalement, nous avons présenté les résultats d'expérimentations qui montrent que le Regroupement Cyclique, quand il s'appuie sur une adaptation hybride du Tree-Clustering appelée BTD [18], peut effectivement être très efficace en pratique, ceci à partir du moment où l'instance à résoudre recèle des propriétés structurelles adaptées et reconnaissables.

Il demeure un travail important qui reste à dé-

velopper, principalement dans deux directions. Tout d'abord, nous devons étudier les outils algorithmiques pour calculer des TIS de meilleure qualité. La seconde direction concerne l'étude visant à exploiter au mieux les informations produites par BTD pendant la phase de traitement préliminaire dans CC-BTD$_2$.

Références

[1] E. Balas and C. Yu. Finding a maximum clique in an arbitrary graph. *Siam J. on Computing*, 15(4) :1054–1068, 1986.

[2] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database schemes. *J. ACM*, 30 :479–513, 1983.

[3] C. Berge. *Hypergraphes – Combinatoire des ensembles finis.* Gauthier-Villars, 1987.

[4] U. Bertele and F. Brioschi. *Nonserial Dynamic Programming.* Academic Press, 1972.

[5] C. Bessière and J.-C. Régin. MAC and Combined Heuristics : Two Reasons to Forsake FC (and CBJ ?) on Hard Problems. In *Proceedings of CP'96*, pages 61–75, 1996.

[6] R. Dechter. Enhancement Schemes for Constraint Processing : Backjumping, Learning, and Cutset Decomposition. *Artificial Intelligence*, 41 :273–312, 1990.

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

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

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

[10] E. Freuder. A Sufficient Condition for Backtrack-Free Search. *JACM*, 29 :24–32, 1982.

[11] D.R. Fulkerson and O. Gross. Incidence matrices and interval graphs. *Pacific J. Math.*, 15 :835–855, 1965.

[12] F. Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM Journal on Computing*, 1 (2) :180–187, 1972.

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

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

[15] P. Janssen, P. Jégou, B. Nouguier, and M.C. Vilarem. A filtering process for general constraint satisfaction problems : achieving pairwise-consistency using an associated binary representation. In *Proceedings of IEEE Workshop on Tools for Artificial Intelligence*, pages 420–427, 1989.

[16] P. Jégou. Cyclic-Clustering : a compromise between Tree-Clustering and the Cycle-Cutset method for improving search efficiency. In *Proceedings of the European Conference on Artificial Intelligence (ECAI-90)*, pages 369–371, 1990.

[17] P. Jégou. *Contribution à l'étude des problèmes de satisfaction de contraintes : Algorithmes de propagation et de résolution – Propagation de contraintes dans les réseau dynamiques.* PhD thesis, Université des Sciences et Techniques du Languedoc, January 1991. In french.

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

[19] P. Jégou and C. Terrioux. A time-space trade-off for constraint networks decomposition. Technical report, Laboratoire des Sciences de l'Information et des Systèmes (LSIS), 2004. Available at www.lsis.org.

[20] M. Krishnamoorthy and N. Deo. Node deletion NP-complete problems. *SIAM J. on Computing*, 8(4) :619–625, 1979.

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

[22] D. Sabin and E. Freuder. Contradicting Conventional Wisdom in Constraint Satisfaction. In *Proceedings of the European Conference on Artificial Intelligence (ECAI-94)*, pages 125–129, 1994.

[23] 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.

[24] I. Tomescu. Sur le nombre de cliques maximales d'un graphe et quelques problèmes sur les graphes parfaits. *Revue roumaine de Mathématiques Pures et Appliquées*, 16 :1115–1126, 1970. In french.

[25] M. Yannakakis. Node and Edge-deletion NP-complete problems. In *Proceedings ACM Symp. on Theory of Comput.*, pages 253–264, 1978.