HAL
archives-ouvertes.fr

Sur la complexité des algorithmes de backtracking et quelques nouvelles classes polynomiales pour CSP

Achref El Mouelhi, Philippe Jégou, Cyril Terrioux, Bruno Zanuttini

▶ **To cite this version:**

Achref El Mouelhi, Philippe Jégou, Cyril Terrioux, Bruno Zanuttini. Sur la complexité des algorithmes de backtracking et quelques nouvelles classes polynomiales pour CSP. Simon de Givry. Huitièmes Journées Francophones de Programmation par Contraintes - JFPC 2012, May 2012, Toulouse, France. 2012, Actes des Huitièmes Journées Francophones de Programmation par Contraintes. <hal-00830458>

HAL Id: hal-00830458
https://hal.inria.fr/hal-00830458

Submitted on 5 Jun 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Sur la complexité des algorithmes de backtracking et quelques nouvelles classes polynomiales pour CSP *

Achref El Mouelhi[1] **Philippe Jégou**[1] **Cyril Terrioux**[1] **Bruno Zanuttini**[2]

[1] LSIS - UMR CNRS 7296, Aix-Marseille Université
Avenue Escadrille Normandie-Niemen, 13397 Marseille Cedex 20
[2] GREYC, Université de Caen Basse-Normandie, CNRS UMR 6072, ENSICAEN
Campus II, Boulevard du Maréchal Juin, 14032 Caen Cedex
{achref.elmouelhi, philippe.jegou, cyril.terrioux}@lsis.org
bruno.zanuttini@unicaen.fr

Résumé

L'étude des classes polynomiales, pour les problèmes de satisfaction de contraintes (CSP), constitue depuis longtemps un domaine de recherche important qui s'avère aujourd'hui très actif. Cependant, les travaux réalisés jusqu'à présent se sont révélés pour l'essentiel théoriques. En effet, ils se cantonnent en général à la définition de classes d'instances pour lesquelles des algorithmes polynomiaux ad hoc, à la fois pour la reconnaissance et pour la résolution, sont proposés. Ces algorithmes ne peuvent être, en fait, utilisés que pour le traitement d'une classe d'instances donnée. Ils s'avèrent ainsi difficilement exploitables en pratique, et ne sont donc pas exploités au sein de solveurs généraux. L'intérêt pratique des classes polynomiales est ainsi très limitée.

Dans cet article, nous abordons la question des classes polynomiales CSP d'un point de vue différent de l'approche classique, en nous intéressant aux algorithmes que l'on peut retrouver dans les systèmes de résolution opérationnels. Pour cela, nous étudions d'abord la complexité d'algorithmes génériques de résolution de CSP tels que le Forward-Checking par exemple. Cette étude s'appuie sur l'exploitation d'un paramètre issu de la théorie des graphes, et qui permet de proposer de nouvelles bornes de complexité. La mise en relation de ces nouvelles bornes avec certains résultats issus de la théorie des graphes nous permet d'exhiber de nouvelles classes polynomiales. De cette façon, nous montrons comment des algorithmes classiques de résolution de CSP peuvent traiter efficacement en pratique ainsi qu'en théorie, des instances de CSP, sans devoir reconnaître au préalable leur appartenance à d'éventuelles classes polynomiales.

*Ce travail est soutenu par l'Agence Nationale de la Recherche dans le cadre du projet TUPLES (ANR-2010-BLAN-0210).

Abstract

The question of tractable classes of constraint satisfaction problems (CSPs) has been studied for a long time, and is now a very active research domain. However, studies of tractable classes are typically very theoretical. They usually introduce classes of instances together with polynomial time algorithms for recognizing and solving them, and the algorithms can be used only for the new class.

In this paper, we address the issue of tractable classes of CSPs from a different perspective. We investigate the complexity of classical, generic algorithms for solving CSPs (such as Forward Checking). We introduce a new parameter for measuring their complexity and derive new complexity bounds. By relating the complexity of CSP algorithms to graph-theoretic parameters, our analysis allows us to point at new tractable classes, which can be solved directly by the usual CSP algorithms in polynomial time, and without the need to recognize the classes in advance.

1 Introduction

Les Problèmes de Satisfaction de Contraintes (CSP, [18]) constituent un formalisme important de l'Intelligence Artificielle pour exprimer et résoudre efficacement un large éventail de problèmes réels. Un réseau de contraintes (ou CSP) est constitué d'un ensemble

de variables X, dont chacune doit être affectée à une valeur issue de son domaine (fini) associé, de sorte que ces affectations satisfassent tout un ensemble fini C de contraintes.

Décider si un CSP donné possède une solution est un problème NP-complet. Aussi, les approches classiques proposées pour la résolution de ce problème sont basées sur des algorithmes de recherche, dont la complexité temporelle, dans le pire des cas est de l'ordre de $O(min(n, e).d^n)$ où n est le nombre de variables, e le nombre de contraintes et d la taille du plus grand domaine. L'amélioration de l'efficacité de ce type d'algorithmes s'appuie généralement sur des techniques de filtrages opérés lors de la recherche (en plus d'autres techniques telles que les heuristiques de choix de variables). Avec l'aide de ces techniques, en dépit de leur complexité théorique exponentielle, des algorithmes tels que Forward-Checking [11] (noté FC), RFL (pour Real Full Look-ahead, [14]) ou MAC (pour "Maintaining Arc-Consistency", [19]) pour les CSP binaires, ou nFC$_i$ pour le cas de versions non-binaires [1] s'avèrent très efficaces dans la pratique sur un large éventail de problèmes réels.

Dans une direction orthogonale, d'autres travaux portent sur l'efficacité de la résolutions des CSP en définissant des *classes polynomiales*. Une classe polynomiale est un sous-ensemble d'instances de CSP qui peuvent être reconnues, puis résolues en utilisant des algorithmes de complexité polynomiale. Différents types de classes polynomiales ont été introduites. Certaines d'entre elles sont basées sur la *structure* du réseau de contraintes, par exemple l'acyclicité du réseau [8] ou plus généralement, les réseaux dont la largeur arborescente est bornée par une constante [10]. D'autres classes sont basées sur des restrictions de contraintes (on parle alors de *langage de contraintes*). Par exemple, les contraintes Zero-One-All (ZOA) restreignent les relations de compatibilité à certaines formes [4]. Plus récemment, des classes dites *hybrides* ont été proposées. On peut citer notamment la classe des instances vérifiant la Broken-Triangle Property [5].

Malheureusement, la plupart des classes polynomiales se présentent rarement en pratique, ce qui diminue d'autant leur intérêt. En revanche, comme évoqué ci-dessus, des algorithmes tels que FC, RFL, MAC, ou nFC$_i$, dont la complexité théorique est exponentielle, sont à la base de systèmes pratiques pour la résolution de contraintes, et leurs résultats pratiques sont souvent impressionnants en termes de temps de calcul, alors même qu'ils n'exploitent *a priori* aucune classe polynomiale.

Dans cet article, nous essayons d'amoindrir le fossé existant entre les travaux théoriques sur les classes polynomiales et l'efficacité pratique des méthodes usuelles, cela en tentant de fournir des éléments de réponse à la question portant sur les raisons de l'efficacité observée pour des algorithmes tels que (n)FC ou (n)RFL. Nous le faisons en réévaluant leur complexité en temps en utilisant un nouveau paramètre, à savoir le nombre de cliques maximales dans la *micro-structure* [12] d'une instance ou dans la *micro-structure généralisée* pour le cas non-binaire. Pour le cas binaire, si $\omega_{\#}(\mu(P))$ exprime le nombre de cliques maximales figurant dans la micro-structure d'un CSP P, nous montrons que la complexité d'un algorithme tel que FC est en $O(n^2 d \cdot \omega_{\#}(\mu(P)))$. Cela fournit une nouvelle perspective pour l'étude de l'efficacité des algorithmes de type backtracking en l'associant à un paramètre bien connu en théorie des graphes. En particulier, en réutilisant des résultats connus de la théorie des graphes, nous proposons de nouvelles classes polynomiales de CSP. Le trait saillant de ces classes est qu'elles sont résolues en temps polynomial par des algorithmes à la fois *très généraux* et *largement utilisés*, *sans requérir la nécessité de disposer d'algorithmes de reconnaissance de classes*. À cet égard, notre étude est très proche dans l'esprit de l'étude menée par Rauzy dans le cadre la satisfiabilité de formules propositionnelles et le comportement de l'algorithme DPLL sur les cas connus de classes polynomiales SAT [15].

Cet article est organisé comme suit. Nous présentons d'abord les notations classiques de CSP et les définitions et propriétés de base de la micro-structure. Ensuite, nous présentons notre analyse de la complexité de BT, FC, et RFL sur les CSPs binaires, puis nous introduisons la notion de *micro-structure généralisée* de sorte à étendre notre étude aux CSP non-binaires et donc à des algorithmes de la classe nFC$_i$. Nous mettons ensuite en évidence de nouvelles classes polynomiales issues de la théorie des graphes, qui peuvent ainsi être exploitées dans le domaine des CSP. Enfin, nous évoquons les perspectives offertes pour l'extension de ce travail qui nous apparaît à ce jour comme préliminaire

2 Preliminaries

2.1 Notions de base

Avant d'examiner l'analyse classique de la complexité des algorithmes usuels, nous rappelons quelques notions de base sur les CSP et leur micro-structure.

Définition 1 (CSP) *Un* problème de satisfaction de contraintes *fini (CSP) est un triplet* (X, D, C)*, où* $X = \{x_1, \ldots, x_n\}$ *est un ensemble de variables,* $D = \{D(x_1), \ldots, D(x_n)\}$ *est un ensemble de domaines finis de valeurs, un pour chaque variable, et*

$C = \{c_1, \ldots, c_e\}$ *est un ensemble fini de contraintes. Chaque contrainte* c_i *est un couple* $(S(c_i), R(c_i))$*, où* $S(c_i) = \{x_{i_1}, \ldots, x_{i_k}\} \subseteq X$ *est la* portée *ou* scope *de* c_i*, et* $R(c_i) \subseteq d(x_{i_1}) \times \cdots \times d(x_{i_k})$ *est sa* relation de compatibilité*. L'*arité *de* c_i *est* $|S(c_i)|$*.*

Une contrainte binaire de portée $\{x_i, x_j\}$ sera notée c_{ij}. Un *CSP binaire* est un CSP pour lequel toutes les contraintes sont binaires. Sinon (cas général), le CSP est dit *n-aire*.

Nous supposons que toutes les variables figurent dans la portée d'au moins une contrainte et que pour une portée donnée, il existe au plus une contrainte. Cela ne pose pas de problème dans le contexte de ce travail puisque deux contraintes portant sur le même ensemble de variables peuvent être fusionnées en une seule en prenant l'intersection de leurs relations.

Définition 2 (affectation, solution) *Étant donné un CSP* (X, D, C)*, une* affectation *de valeurs à* $Y \subseteq X$ *est un ensemble de paires* $t = \{(x_i, v_i) \mid x_i \in Y\}$ *(que l'on écrira* $t = (v_1, \ldots, v_k)$ *quand aucune confusion ne se présente), avec* $v_i \in D(x_i)$ *pour tout* i*. Une affectation de* $Y \subseteq X$ *est dite* cohérente *(ou* consistante *ou bien encore* solution partielle*) si toutes les contraintes* $c \in C$ *dont la portée* $S(c) \subseteq Y$ *sont satisfaites, i.e.,* $t[S(c)] \in R(c)$ *où* $t[S(c)]$ *est la restriction de* t *à* $S(c)$*. Une* solution *est une affectation cohérente de* X*.*

Notation 1 (paramètres) *Afin d'exprimer la complexité des algorithmes, nous utiliserons les notations suivantes :*

- n *exprime le nombre de variables d'un CSP*
- d *est la cardinalité du plus grand domaine*
- e *est le nombre de contraintes*
- a *est l'arité maximale pour toutes les contraintes*
- r *est le nombre de tuples de la plus grande relation*

Étant donné un CSP, la question fondamentale est de décider s'il possède une solution, problème bien connu comme étant NP-complet. Afin d'étudier les CSP et essayer de contourner cette difficulté, différents points de vue peuvent être adoptés. Concernant les *CSP binaires*, l'un d'eux s'appuie sur la notion de *micro-structure d'une instance*, qui correspond à son graphe de compatibilité et dont nous rappelons la définition. Intuitivement, les sommets de ce graphe codent les valeurs et ses arêtes codent leur compatibilité.

Définition 3 (micro-structure) *Étant donné un CSP* $P = (X, D, C)$*, la* micro-structure *de* P *est un graphe non orienté* $\mu(P) = (V, E)$ *avec :*

- $V = \{(x_i, v_i) : x_i \in X, v_i \in D(x_i)\}$,
- $E = \{\ \{(x_i, v_i), (x_j, v_j)\}\ \mid\ i \neq j, c_{ij} \notin C$ *ou* $c_{ij} \in C, (v_i, v_j) \in R(c_{ij})\}$

En d'autres termes, la micro-structure d'un CSP binaire P contient une arête pour toutes les paires de sommets, sauf pour les sommets issus d'un même domaine et pour les sommets correspondant à des paires qui sont interdites par certaines contraintes. On peut constater que la micro-structure d'un CSP est un graphe n-parti, car il n'existe pas d'arête reliant les sommets issus d'un même domaine. Dans cet article, nous allons étudier la complexité des algorithmes de résolution de CSP classiques en s'appuyant sur les cliques qui figurent dans la micro-structure.

Définition 4 (clique) *Un* graphe complet *est un graphe simple dans lequel chaque paire de sommets distincts est reliée par une arête. Une* k-clique *dans un graphe non orienté est un sous-ensemble de* k *sommets induisant un graphe complet (tous les sommets sont deux à deux adjacents). Une* clique maximale *est une clique qui n'est pas un sous-ensemble propre d'une autre clique. Nous noterons par* $\omega_{\#}(G)$ *le nombre de cliques maximales dans un graphe* G*.*

Le résultat suivant se déduit directement du fait que dans une micro-structure, les sommets d'une clique correspondent à des valeurs compatibles issues de domaines différents.

Proposition 1 *Étant donné un CSP binaire* P *et sa micro-structure* $\mu(P)$*, une affectation* $(v_1, ..., v_n)$ *de* X *est une solution de* P *ssi* $\{(x_1, v_1), ..., (x_n, v_n)\}$ *est une* n*-clique de* $\mu(P)$*.*

On peut facilement constater que la transformation d'un CSP P en sa micro-structure $\mu(P)$ peut être réalisée en temps polynomial. On en déduit directement l'existence d'une réduction polynomiale du problème de décision lié à l'existence de solution d'un CSP binaire donné vers le problème d'existence d'une clique de taille donnée dans un graphe non-orienté (appelé généralement *problème de la clique*). Cette transformation qui offre un point de vue nouveau pour l'étude des CSP en fournissant un lien avec des notions issues de la Théorie des Graphes a d'abord été exploitée par [12] qui a proposé des classes polynomiales de CSPs basées sur l'existence de classes polynomiales pour le problème de la clique (*graphes chordaux* [9]). Une approche identique a été considérée pour le cas des classes polynomiales *hybrides* [3].

2.2 Complexité des Algorithmes de Backtracking

Nous passons maintenant brièvement en revue la complexité des algorithmes qui nous intéressent ici : BT, FC, RFL pour les CSP binaires, et nFC$_i$ pour le cas non-binaire. Ces algorithmes couvrent toutes les approches qui utilisent le backtracking et les approches

de type prospectif au sens lookahead (la question des choix d'ordres de variables et de valeurs ne sera pas considérée ici).

L'algorithme de *Backtracking* (noté BT et également appelé *Backtracking chronologique*) est une procédure d'énumération récursive. Il commence par une affectation vide et dans le cas général, étant donnée une solution partielle courante $(v_1, v_2, \ldots, v_i)$, il choisit une nouvelle variable x_{i+1} et cherche à affecter des valeurs de $D(x_{i+1})$ à x_{i+1}. Le seul contrôle effectué consiste alors à vérifier que l'affectation résultante $(v_1, v_2, \ldots, v_i, v_{i+1})$ est cohérente. Dans l'affirmative, BT continue avec cette nouvelle solution partielle en l'étendant à une nouvelle variable non encore affectée (appelée *variable future*). Sinon (si $(v_1, v_2, \ldots, v_i, v_{i+1})$ n'est pas compatible), BT essaie une autre valeur de $D(x_{i+1})$. S'il n'y a plus de valeur inexplorée, BT se trouve dans une impasse, et il *désinstancie* x_i (il effectue un *backtrack*).

Il est facile de voir que la recherche effectuée par BT correspond à un parcours en profondeur d'abord d'un arbre sémantique appelé *arbre de recherche*, et dont la racine est un tuple vide, tandis que les nœuds situés au $i^{ème}$ niveau sont des i-uplets qui représentent les affectations des variables le long du chemin correspondant dans l'arbre. Les nœuds de cet arbre qui correspondent à des solutions partielles sont appelés *nœuds consistants*, tandis que les autres nœuds sont appelés *nœuds inconsistants*. Le nombre de nœuds dans l'arbre de recherche est au plus $\Sigma_{0 \leq i \leq n} d^i = \frac{d^{n+1}-1}{d-1}$, par conséquent, il est en $O(d^n)$. Ainsi, la complexité de BT est bornée par le nombre de nœuds multiplié par le coût à chaque nœud. En supposant qu'un test de contrainte peut être réalisé en $O(a)$, la complexité de BT est de $O(min(n, e)ad^n)$.

BT peut être considéré comme un algorithme générique. Des algorithmes basés sur BT et utilisés en pratique réalisent un certain travail supplémentaire à chaque nœud de l'arbre de recherche, à savoir, ils suppriment des valeurs incompatibles dans le domaine des variables futures (procédé appelé *filtrage*). Dans le cas des CSP *binaires*, FC supprime les valeurs incompatibles avec la dernière affectation réalisée, tandis que RFL applique la consistance d'arc (AC) sur les variables futures. La complexité de FC peut être bornée par $O(nd^n)$. En utilisant un algorithme en $O(ed^2)$ pour réaliser AC, la complexité de la RFL est en $O(ed^2d^{n-1}) = O(ed^{n+1})$. Dans le cas des CSP *n-aires*, les algorithmes de la classe nFC$_i$ ($i = 0, 1 \ldots 5$) recouvrent l'application partielle ou totale de *la consistance d'arc généralisée* (GAC) sur un sous-ensemble de contraintes impliquant à la fois les variables affectées et les variables futures. Dans chaque cas, le filtrage est réalisé après chaque affectation de variable. Ainsi, la complexité des techniques de type nFC$_i$ dépend du coût du filtrage. Pour nFC$_5$ qui réalise le filtrage le plus puissant, la complexité est alors en $O(n^e ard)$. C'est la mème chose si l'on considère la version n-aire de RFL qui maintient GAC à chaque nœud. Dans ce qui suit, nous noterons par nBT et nRFL les versions n-aires de BT et de RFL [1].

Nous tenons à souligner ici que les algorithmes (n)BT, FC, (n)RLF ou nFC$_i$ peuvent utiliser un ordre *dynamique* sur les variables, c'est-à-dire que le choix de la future variable (x_{i+1}) à explorer peut être décidé après chaque nouvelle affectation. Notons cependant qu'ici, lorsque l'affectation de la valeur v_{i+1} conduit à une impasse, nous ne considérons pas la possibilité de remplacer x_{i+1} par une autre variable future (quand bien même il resterait encore des valeurs à explorer pour x_{i+1}) comme le fait par exemple MAC. Au contraire, nous nous en tenons ici aux algorithmes qui changent la variable courante seulement après avoir épuisé son domaine.

3 Une Nouvelle Analyse de la Complexité pour les CSP binaires

Nous arrivons maintenant au cœur de notre contribution, à savoir une analyse de la complexité des algorithmes classiques en termes de paramètres liés à la micro-structure. Dans ce qui suit, nous disons qu'un nœud de l'arbre de recherche est un *nœud consistant maximalement profond* s'il est cohérent et s'il ne possède pas de nœud fils cohérent (sur la variable suivante dans l'ordre). Ainsi, un tel nœud correspond soit à une solution, soit à une solution partielle qui ne peut être étendue de façon cohérente sur la variable suivante.

Le résultat suivant peut-être considéré comme central pour notre étude.

Proposition 2 *Étant donné un CSP binaire $P = (X, D, C)$, il existe une application injective de l'ensemble des nœuds consistants maximalement profonds explorés par BT vers l'ensemble des cliques maximales de $\mu(P)$.*

Preuve : Soit $(v_1, v_2, \ldots, v_i)$ un nœud consistant maximalement profond exploré par BT. Par définition, $(v_1, v_2, \ldots, v_i)$ est une solution partielle, et donc pour tout $1 \leq j, k \leq i$, soit il n'existe pas de contrainte c_{jk} dans C dont la portée est $\{x_j, x_k\}$, soit (v_j, v_k) figure dans la relation $R(c_{jk})$. Dans les deux cas, $\{(x_j, v_j), (x_k, v_k)\}$ constitue une arête de $\mu(P)$. Donc $\{(x_1, v_1), \ldots, (x_i, v_i)\}$ est une clique de $\mu(P)$ et donc,

1. Notons que nBT correspond exactement au même algorithme que BT, ce qui n'est pas le cas de nRFL puisqu'il faut alors considèrer la consistance d'arc généralisée GAC

elle est incluse dans au moins une clique maximale de $\mu(P)$. Nous la noterons $Cl(v_1, v_2, \ldots, v_i)$.

Nous montrons maintenant que Cl constitue une application injective de l'ensemble des nœuds consistants maximalement profonds vers l'ensemble des cliques maximales. Par construction de BT, si $(v_1, v_2, \ldots, v_i)$ et $(v'_1, v'_2, \ldots, v'_{i'})$ sont deux nœuds maximalement profonds explorés, ils doivent différer d'au moins une valeur sur une variable. Précisément, ils doivent différer au moins sur le nœud où les chemins correspondants se diffèrencient dans l'arbre de recherche, et donc sur les nœuds correspondant à une certaine variable x_j affectée à une certaine valeur sur un chemin, et à une autre valeur sur l'autre [2]. Comme il n'y a pas d"arête de $\mu(P)$ connectant deux valeurs d'une même variable, il ne peut pas y avoir de clique maximale contenant à la fois $(v_1, v_2, \ldots, v_i)$ et $(v'_1, v'_2, \ldots, v'_{i'})$, donc Cl est nécessairement une application injective. □

En utilisant cette propriété, nous pouvons facilement borner le nombre de nœuds figurant dans un arbre de recherche induit par une recherche de type backtracking, et donc aussi sa complexité en temps, en termes de propriété liée à sa micro-structure. Comme il est d'usage, nous supposons qu'un test de contrainte (vérifier si $(v_i, v_j) \in R(c_{ij})$) est réalisable en temps constant.

Proposition 3 *Le nombre de nœuds $N_{BT}(P)$ figurant dans l'arbre de recherche développé par BT pour résoudre un CSP binaire $P = (X, D, C)$, vérifie $N_{BT}(P) \leq nd \cdot \omega_{\#}(\mu(P))$. Sa complexité en temps est en $O(n^2 d \cdot \omega_{\#}(\mu(P)))$.*

Preuve : Considérons d'abord le nombre de nœuds consistants. Parce que n'importe quel nœud de l'arbre de recherche figure à une profondeur au plus égale à n et que le chemin de la racine à un nœud consistant contient uniquement des nœuds consistants, comme corollaire direct de la Proposition 2, nous obtenons que l'arbre de recherche contient au plus $n \cdot \omega_{\#}(\mu(P))$ nœuds consistants. Maintenant, par définition de BT, un nœud consistant possède au plus d fils (un par valeur candidate pour la variable suivante), et les nœuds incompatibles n'en ont pas. Il s'ensuit que l'arbre de recherche a au plus $nd \cdot \omega_{\#}(\mu(P))$ nœuds de toute nature.

La complexité en temps se déduit directement puisque chaque nœud correspond à l'extension de l'affectation partielle à une variable supplémentaire x_{i+1}, ce qui implique au plus un test de contrainte par autre variable déjà affectée (vérifier la satisfaction de $c_{j(i+1)}$ pour chaque x_j déjà affecté), et comme il n'en existe nécessairement moins de n, on obtient le résultat.□

On peut constater dans l'énoncé de la Proposition 3 ainsi que dans les suivantes, que le nombre de cliques maximales $\omega_{\#}(\mu(P))$ pourrait être remplacé par le nombre de cliques maximales *de taille au plus* $n-1$. Ceci parce que dès qu'un chemin exploré est contenu dans une n-clique, c'est-à-dire, dans une solution, aucun retour arrière ne se produira ultérieurement sur ce chemin.

Nous analysons maintenant FC et RFL. Il apparaît clairement que la Proposition 2 vaut également pour chacun de ces algorithmes. Le nombre de nœuds explorés découle du fait que seuls les nœuds consistants sont explorés. La complexité en temps se déduit du fait qu'au plus n domaines futurs sont filtrés par FC et que AC est réalisable en un temps $O(ed^2)$ par RFL.

Proposition 4 *Le nombre de nœuds $N_{FC}(P)$ figurant dans l'arbre de recherche développé par FC ou par RFL pour résoudre un CSP binaire $P = (X, D, C)$, vérifie $N_{FC}(P) \leq n \cdot \omega_{\#}(\mu(P))$. La complexité en temps de FC est en $O(n^2 d \cdot \omega_{\#}(\mu(P)))$, et celle de RFL est en $O(ned^2 \cdot \omega_{\#}(\mu(P)))$.*

Il est important de constater que la taille relative des arbres de recherche de BT et FC fournie par l'analyse classique et celle obtenue par l'approche que nous proposons sont les mêmes. Notamment, dans le pire des cas, l'arbre de recherche de BT est d fois plus grand en nombre de nœuds que celui de FC sur la même instance.

4 Une Nouvelle Analyse pour les CSP Non-Binaires

4.1 Micro-structure Généralisée

Nous passons maintenant à l'étude des CSP n-aires. Dans un premier temps, nous étendons la notion de micro-structure à ce cas général, puis nous analysons la complexité des techniques de type nFC$_i$ en termes de nombre de cliques maximales.

Notons que la généralisation de la notion de la micro-structure a d'abord été proposée dans [3]. Toutefois, cette notion est basée sur les hypergraphes et n'a pas vraiment été exploitée jusqu'à présent. En revanche, notre notion se situe toujours dans le cadre des graphes. Notre *micro-structure généralisée* est obtenue en considérant comme sommets les tuples figurant dans les relations du CSP plutôt que les valeurs (x_i, v_i) utilisées dans le cas binaire.

2. Nous utilisons ici l'hypothèse selon laquelle l'algorithme explore toutes les valeurs d'une variable avant le réordonnancement des variables futures

Définition 5 (micro-structure généralisée) *Étant donné un CSP $P = (X, D, C)$ (pas nécessairement binaire), la* micro-structure généralisée *de P est un graphe non-orienté $\mu_G(P) = (V, E)$ avec :*

- $V = \{(c_i, t_i) : c_i \in C, t_i \in R(c_i)\}$,
- $E = \{ \{(c_i, t_i), (c_j, t_j)\} \mid i \neq j, t_i[S(c_i) \cap S(c_j)] = t_j[S(c_i) \cap S(c_j)]\}$

Comme pour la micro-structure des CSP binaires, il existe une relation directe entre cliques et solutions de CSP :

Proposition 5 *Un CSP P possède une solution ssi $\mu_G(P)$ possède une clique de taille e.*

Preuve : Par construction, $\mu_G(P)$ est e-parti et chaque clique contient au plus un sommet (c_i, t_i) par contrainte $c_i \in C$. Donc, les e-cliques de $\mu_G(P)$ correspondent exactement à des cliques ne possédant qu'un sommet (c_i, t_i) par contrainte $c_i \in C$. De plus, et encore par construction de $\mu_G(P)$, tout couple de sommets (c_i, t_i), (c_j, t_j) joints par une arête vérifie $t_i[S(c_i) \cap S(c_j)] = t_j[S(c_i) \cap S(c_j)]$. Donc tous les t_i d'une clique sont connectés ensemble, et il s'ensuit que les e-cliques de $\mu_G(P)$ correspondent exactement à des ensembles de tuples t qui sont connectés avec d'autres tuples permis par les contraintes, c'est-à-dire qu'ils correspondent à des solutions de P. □

On peut observer que la généralisation de la micro-structure correspond à la micro-structure de la *représentation duale* d'un CSP [6]. À ce titre, on peut aussi constater que notre généralisation correspond en fait au *line-graph* ou *graphe des intersections* (c'est le graphe dual) de l'hypergraphe proposé dans [3], auquel nous aurions rajouté des arêtes pour le cas où deux portées de contraintes ne s'intersecteraient pas. Enfin, et dans le même esprit que celui de notre approche, nous pourrions proposer d'autres généralisations de micro-structure en considérant toute représentation graphique de CSP n-aire, dès que la représentation a le même ensemble de solutions - à une bijection près - que l'instance d'origine (par exemple le *codage par variables cachées* [17]). Toutefois, par manque de place, nous ne traitons pas ces questions ici.

4.2 Complexité de nBT et nFC

Nous étudions maintenant la complexité des algorithmes de résolution des CSP n-aires. Dans un premier temps, nous posons une hypothèse sur l'ordre dans lequel ces algorithmes explorent les variables, puis nous discutons de cette restriction.

Définition 6 (ordre compatible) *Soit P un CSP. Un ordre total $(x_1, x_2, \ldots x_n)$ sur X est dit* compatible avec les contraintes de C *s'il y a k contraintes $c_{i_1}, c_{i_2}, \ldots c_{i_k}$ dans C ($1 \leq k \leq e$) qui vérifient :*

- $\bigcup_{1 \leq l \leq k} S(c_{i_l}) = X$
- *il y a k variables $x_{i_1}, x_{i_2}, \ldots x_{i_k}$ telles que pour tout ℓ dans $\{1, \ldots, k\}$, $x_\ell \in S(c_{i_\ell})$ et $\bigcup_{1 \leq j \leq \ell} S(c_{i_j}) = \{x_i \mid i = 1, \ldots, i_\ell\}$.*

En d'autres termes, l'ordre est tel que les variables figurant dans la portée d'une première contrainte c_{i_1} apparaissent toutes d'abord, puis toutes les variables figurant dans la portée d'une certaine contrainte C_{i_2} apparaissent ensuite (sauf pour celles qui figurent déjà dans c_{i_1}), etc. Les variables $x_{i_1}, \ldots, x_{i_k}$ dans la définition sont telles que x_{i_j} est la dernière variable affectée dans la portée de c_{i_j}. Nous nous considèrerons ces variables comme étant des *jalons* dans l'ordre.

Par exemple, avec la notation de la définition, nous devons avoir $S(c_{i_1}) = \{x_1, x_2, \ldots x_{i_1}\}$ et $S(c_{i_1}) \cup S(c_{i_2}) = \{x_1, x_2, \ldots x_{i_1}, x_{i_1+1}, \ldots x_{i_2}\}$. La variable x_{i_1} est un jalon (dernière variable affectée dans la portée de c_{i_1}) de même que x_{i_2} pour c_{i_2}.

Dans l'hypothèse où un tel ordre sur les variables est utilisé, nous pouvons donner une généralisation de la Proposition 2.

Proposition 6 *Soit P un CSP n-aire. Supposons que nBT explore les variables dans un ordre compatible avec les contraintes de C. Alors, il existe une application injective de l'ensemble des nœuds consistants maximalement profonds (x_i, v_i) de l'arbre de recherche tels que x_i est un jalon, vers l'ensemble des cliques maximales de $\mu_G(P)$.*

Preuve : Soit t une affectation correspondant à un nœud tel que défini, et notons x_{i_j} pour la dernière variable affectée dans t (x_{i_j} est un jalon par hypothèse). De même, soit t' une autre affectation maximale compatible avec le jalon $x_{i_{j'}}$ comme dernière variable. Notons T l'ensemble $\{t[S(c)] \mid c \in C, S(c) \subseteq \{x_1, \ldots, x_{i_j}\}\}$, c'est-à-dire, l'ensemble de toutes les projections de t sur les portées de toutes les contraintes totalement affectées t, et idem pour T'. Alors T (resp. T') est inclus dans une clique maximale $Cl(t)$ (resp. $Cl(t')$) de $\mu_G(P)$). Maintenant, supposons $j' \geq j$ (sans manque de généralité). Avec $t \neq t'$, le fait que t soit maximalement consistant, et le fait que t' soit consistant, permet d'affirmer que t' diffère de t d'au moins une variable x_ℓ avec $\ell \leq i_j$. Ainsi, cette variable est affectée différemment dans chaque cas, et il existe une certaine contrainte c_{i_ℓ} telle que $t[S(c_{i_\ell})]$ est différent de $t'[S(c_{i_\ell})]$, et donc, t et t' ne peuvent être inclus dans une même clique. Donc, Cl définit une application injective de l'ensemble des affectations considérées vers l'ensembles des cliques maximales de $\mu_G(P)$. □

En utilisant cette propriété, nous pouvons borner le nombre de nœuds dans un arbre de recherche induit

par une recherche de type backtracking, ainsi que sa complexité en temps, relativement à la micro-structure généralisée.

Proposition 7 *Soit $P = (X, D, C)$ un CSP n-aire. Supposons que nBT utilise un ordre sur les variables qui est compatible avec les contraintes de C. Le nombre de nœuds $N_{nBT}(P)$ dans l'arbre de recherche de nBT sur P vérifie $N_{nBT}(P) \leq nd^a \cdot \omega_{\#}(\mu_G(P))$. Sa complexité en temps est en $O(nea \cdot d^a \cdot \omega_{\#}(\mu_G(P)))$.*

Preuve : Du fait de la Proposition 6, le sous-arbre induit par l'arbre de recherche sur les jalons contient au plus $\omega_{\#}(\mu_G(P))$ nœuds. Pour atteindre un jalon à partir du précédent, c'est-à-dire, pour étendre une affectation de $x_1, \ldots, x_{i_j}$ à l'affectation de $x_1, \ldots, x_{i_{j+1}}$ (avec les notations de la définition 6), nBT explore au plus a variables (ceci par définition d'un ordre compatible avec les contraintes). Par conséquent, il explore au plus d^a combinaisons de valeurs (nBT n'a aucune raison pour exclure une affectation avant d'affecter toutes les variables figurant dans la portée d'une contrainte). Puisqu'une branche contient au plus n nœuds, et donc finalement n jalons, on obtient le résultat.

La complexité en temps se déduit directement puisque chaque nœud nécessite au plus e tests de contraintes, chacun en $O(a)$ avec une structure de données appropriée. □

Un résultat similaire est valable pour nFC_i ($i \geq 2$). Néanmoins, nous devons tenir compte du coût supplémentaire dû à l'application de GAC, qui est $O(e \cdot a \cdot r)$ à chaque nœud.

Toutefois, on peut noter que, contrairement à nBT, et en raison de l'utilisation de GAC, nFC_i explore seulement les r tuples autorisés par c lors de l'exploration des variables dans $S(c)$, plutôt que toutes les d^a combinaisons de valeurs.

Proposition 8 *Soit $P = (X, D, C)$ un CSP n-aire. Supposons que nFC_i ($i \geq 2$) utilise un ordre sur les variables qui est compatible avec les contraintes de C. Le nombre de nœuds $N_{nFC_i}(P)$ dans l'arbre de recherche de nFC_i sur P vérifie $N_{nFC_i}(P) \leq nr \cdot \omega_{\#}(\mu_G(P))$. Sa complexité en temps est en $O(nea \cdot r^2 \cdot \omega_{\#}(\mu_G(P)))$.*

Ce résultat est également valable pour nRFL.

4.3 Complexité sans hypothèse sur l'ordre

On pourrait estimer que notre restriction posée sur les ordres d'affectation des variables en termes de compatibilité avec les contraintes n'est pas respectée par tous les ordres raisonnables. Par exemple, il n'y a aucune raison en général, pour qu'une heuristique très usitée telle que *dom / deg* respecte ce type d'ordres. Nous montrons donc ici que ce type de restriction s'avère nécessaire.

Pour montrer cela, nous construisons une famille d'instances qui possèdent un nombre linéaire de cliques dans leur micro-structure généralisée (linéaire en son nombre de sommets), mais pour lesquelles nFC_5 explore un arbre de recherche de taille exponentielle (dans le nombre de sommets) pour un certain ordre sur variable.

Les instances (X, D, C) de cette famille sont construites comme suit. Avec e le nombre de contraintes, nous considérons deux variables distinctes x_0 et x'_0 de X qui seront communes à toutes les contraintes, et nous aurons $X \setminus \{x_0, x'_0\}$ qui est partitionné en e ensembles $X_1, \ldots, X_e$ (X_i sera associé à c_i). Ainsi, chaque contrainte $c_i \in C$ possède une portée $S(c_i) = \{x_0, x'_0\} \cup X_i$. Maintenant, le domaine est $\{v_0, \ldots, v_{e-1}\}$ pour toutes les variables ($d = e$). Enfin, les tuples permis pour une contrainte c_i sont précisément de la forme $\{(x_0, v_j), (x'_0, v_{i+j}), \ldots\}$, pour $j = 1, \ldots, e$ (les indices sont pris modulo e) et sans restriction pour les affectations de X_i. On note que pour $i \neq i'$, les restrictions pour les tuples permis par c_i et $c_{i'}$ sur $\{x_0, x'_0\}$ ne coïncident jamais, de sorte qu'il n'existe aucune arête dans $\mu_G(P)$. Ainsi, le nombre de cliques dans $\mu_G(P)$ exactement égal au nombre de sommets $|V| = e^{2+(n-2)/e}$.

D'autre part, supposons que nFC_5 explore toutes les variables dans les ensembles X_i et explore seulement x_0 et x'_0 après elles. Ainsi, puisque que toutes les valeurs pour x_0 possèdent un support dans toutes les contraintes, et de même pour x'_0, aucune valeur ne sera retirée avant d'atteindre x_0 ou x'_0, et donc toutes les $e^{n-2} \sim |V|^e$ combinaisons de valeurs seront explorées, c'est-à-dire exponentiellement plus que le nombre de cliques figurant dans $\mu_G(P)$.

5 Quelques Classes Classes Polynomiales pour le Backtracking

Le nombre de cliques dans un graphe peut croître de façon exponentielle avec la taille du graphe [20] et il en est de même du nombre $\omega_{\#}(G)$ de *cliques maximales* dans un graphe G [13]. Toutefois, pour certaines classes de graphes, le nombre de cliques maximales peut être borné par un polynôme en la taille du graphe. Si la micro-structure (généralisée) d'un (d'une famille de) CSP P appartient à l'une de ces classes, l'analyse que nous avons présentée dans les sections précédentes permet de conclure que P est résoluble en temps polynomial par des algorithmes classiques d'énumération, et ce, *sans avoir à reconnaître l'appartenance de l'instance à cette classe.*

Dans cette section, nous étudions plusieurs classes

de graphes vérifiant cette propriété et nous commentons leur pertinence en termes de problèmes de satisfaction de contraintes.

5.1 Graphes "Triangle-Free" ou Bipartis

Nous rappelons qu'un *k-cycle* dans un graphe $G = (V, E)$ est une séquence $(v_1, v_2, \dots v_{k+1})$ de sommets distincts, sauf pour $v_1 = v_{k+1}$, vérifiant $\forall i, 1 \leq i \leq k, \{v_i, v_{i+1}\} \in E$.

Un graphe *triangle-free* est un graphe non-orienté sans 3-cycle. Il est facile de voir que le nombre de cliques maximales dans un graphe *triangle-free* est exactement égal au nombre de ses arêtes. En effet, chaque arête est une clique, et par définition, il ne peut pas exister de plus grande clique. Avec notre analyse, nous pouvons affirmer que pour la classe des CSP dont la micro-structure (généralisée) est *triangle-free*, les algorithmes (n)BT, (n)FC et (n)RFL sont des procédures de résolution fonctionnant en temps polynomial. Notons cependant que cette classe de CSP constitue d'une certaine façon un cas *dégénéré*, puisque excepté pour les instances ayant au plus deux variables (cas binaire) ou deux contraintes (cas n-aire), les instances avec micro-structure (généralisée) *triangle-free* sont incohérentes.

Un autre cas dégénéré est constitué par la classe des graphes bipartis. Un graphe est dit *biparti* s'il ne contient pas de cycle impair. Encore une fois, un graphe biparti ne peut contenir de clique de plus de deux sommets, et donc aucune affectation partielle de plus de trois variables ne sera considérée par BT (dont la complexité sera ainsi en $O(d^3)$).

Nous passons maintenant à des classes plus intéressantes, qui aussi, contiennent pour l'essentiel des CSP incohérents, mais pour lesquelles notre analyse fournit une meilleure complexité en temps que celle offerte par l'analyse classique.

5.2 Graphes Planaires, Toroïdaux, et "Embedded"

Définition 7 (planaire) *Un graphe* planaire *est un graphe qui peut être dessiné dans le plan sans que deux arêtes ne se chevauchent.*

[20] a prouvé que le nombre cliques d'un graphe planaire $G = (V, E)$ est au plus $8(|V| - 2)$.

Définition 8 (toroïdal) *Un graphe* toroïdal *est un graphe qui peut être dessiné sur un tore sans que deux arêtes ne se chevauchent.*

[7] ont montré que tout graphe toroïdal possède au plus $8(|V| + 9)$ cliques et que tout graphe intégrable ("Embedded") dans une surface possède un nombre linéaire de cliques puisque majoré par $8(|V| + 27)$ au pire. Puisque la micro-structure $\mu(P)$ (resp. $\mu_G(P)$) d'un CSP P contient nd sommets (resp. er sommets), alors si $\mu(P)$ (resp. $\mu_G(P)$) appartient à l'une de ces classes de graphes, alors $\omega_\#(\mu(P))$ (resp. $\omega_\#(\mu_G(P))$) possède au plus $O(nd)$ cliques (resp. $O(er)$).

Grâce aux Propositions 3-4 et 7-8, nous obtenons immédiatement le résultat suivant.

Théorème 1 *Soit Em qui désigne la classe de tous les CSP dont les micro-structure sont planaires, toroïdales, ou intégrables dans une surface. Alors les instances de Em sont résolues en un temps*

- $O(n^2 d \cdot \omega_\#(\mu(P))) = O(n^3 d^2)$ *par BT or FC,*
- $O(ned^2 \cdot \omega_\#(\mu(P))) = O(n^2 ed^3)$ *par RFL,*
- $O(nead^a \cdot \omega_\#(\mu_G(P))) = O(ne^2 ard^a)$ *par nBT,*
- $O(near^2 \cdot \omega_\#(\mu_G(P))) = O(ne^2 ar^3)$ *par nFC$_i$ ou nRFL.*

Rappelons que cette famille de graphes ne peut contenir en tant que mineur ni une 8-clique (pour les graphes toroïdaux), ou ni une 5-clique ou $K_{3,3}$ (pour les graphes planaires). Cela a pour conséquence, en particulier, que tous les CSP binaires (resp. n-aires) dans *Em* sur au moins 8 variables (resp. contraintes) sont incohérents. On peut donc dire raisonnablement que cette classe est pour le moins dégénérée. Néanmoins, si l'on se réfère à l'analyse classiques de la complexité, on constate que, par exemple, BT résout ces instances en un temps $O(d^8)$, pour le cas où d est grand, ce temps est significativement supérieur à $O(n^3 d^2)$.

5.3 Graphes CSG

Nous étudions maintenant la classe des *Graphes CSG* qui a été introduite par [2] et qui généralise la classe des *graphes chordaux* (dits aussi *triangulés*).

Étant donné un graphe (V, E) et un ordre $v_1, \dots, v_{|V|}$ sur ses sommets, nous noterons $N^+(v_i)$ pour le *voisinage ultérieur* de v_i, c'est-à-dire, $N^+(v_i) = \{v_j \in V | \{v_i, v_j\} \in E, i < j\}$. Pour $V' \subseteq V$, nous notons $G(V')$ le sous-graphe *induit* par E sur V', soit, $G(V') = (V', E')$ où $E' = \{\{x, y\} \mid x, y \in V' \text{ et } \{x, y\} \in E\}$.

Définition 9 (Graphes CSG) *La classe de graphes CSG^k est définie inductivement comme suit.*

- *CSG^0 est la classe des graphes complets.*
- *Étant donné $k > 0$, CSG^k est la classe des graphes $G = (V, E)$ tels qu'il existe un ordre $\sigma = (v_1, ..., v_{|V|})$ sur V vérifiant que pour $i = 1, \dots, |V|$, le graphe $G(N^+(v_i))$ est un graphe CSG^{k-1}.*

La classe de graphes CSG généralise la classe des graphes complets (graphes CSG^0) et surtout la classe

des graphes chordaux (graphes CSG^1). Comme les graphes chordaux, les graphes CSG possèdent des propriétés intéressantes. Par exemple, ils peuvent être reconnus en temps polynomial. De plus il a été démontré dans [2] que les graphes CSG^k possèdent au plus $|V|^k$ cliques maximales, et un algorithme de complexité polynomiale en $O(|V|^{2(k-1)}(|V|+|E|))$ a été proposé pour les énumérer.

L'existence de ces deux algorithmes confère à la classe des CSP qui ont une micro-structure (généralisée) CSG^k le statut de classe polynomiale pour toute valeur k fixée. Nous sommes cependant en mesure de montrer que des algorithmes *génériques* tels que (n)BT, FC, nFC_i ou (n)RFL s'exécutent en temps polynomial sur de tels CSP, sans même qu'il soit nécessaire de reconnaître l'appartenance des instances traitées à cette classe (et donc sans avoir besoin de calculer la micro-structure). À nouveau, ce résultat est issu du nombre de cliques maximales par l'application des Propositions 4 et 7-8.

Théorème 2 *Pour tout entier k fixé, la classe des CSP dont la micro-structure (généralisée) est CSG^k peut être résolue en un temps*

- $O(n^2d \cdot \omega_{\#}(\mu(P))) = O(n^{k+2}d^{k+1})$ *par BT et FC,*
- $O(ned^2 \cdot \omega_{\#}(\mu(P))) = O(n^{k+1}ed^{k+2})$ *par RFL,*
- $O(nead^a \cdot \omega_{\#}(\mu_G(P))) = O(ne^{k+1}ar^kd^a)$ *par nBT,*
- $O(near^2 \cdot \omega_{\#}(\mu_G(P))) = O(ne^{k+1}ar^{k+2})$ *par nFC_i ou nRFL.*

Nous pouvons constater de plus que les complexités temps sont même meilleures que celles issues de l'algorithme dédié. Par exemple, celui-ci calcule la micro-structure d'un CSP binaire et énumère toutes les cliques maximales ou bien s'arrête dès qu'une n-clique est trouvée. Ainsi, la complexité en temps est en $O((nd)^{2(k-1)}(nd+n^2d^2)) = O((nd)^{2k})$. Néanmoins, nous pouvons noter que l'algorithme est défini pour des graphes CSG^k quelconques, alors que les micro-structures (généralisées) de CSP sont des graphes très particuliers.

Il semble, en termes de CSP, que les graphes CSG soient généralement moins restrictifs que les classes précédentes de graphes. Par exemple, il est possible d'avoir des graphes CSG possédant des n-cliques (resp. des e-cliques) pour toute valeur de n (resp. e), contrairement notamment au cas des graphes planaires. En particulier, il existe des CSP cohérents possédant une micro-structure (généralisée) qui est un graphe CSG^k. C'est le cas notamment pour la classe CSG^0 qui sont exactement les CSPs binaires cohérents de domaines monovalents (une valeur par domaine) ou les CSP n-aires cohérents avec exactement un tuple autorisé par relation. Néanmoins, les CSP qui ont une micro-structure (généralisée) qui est un graphe CSG^1 peuvent être cohérents ou non et il est facile de construire un CSP avec plusieurs solutions, ce qui correspond à une collection de cliques de taille n (cas binaire) ou e (cas n-aire). En outre, contrairement aux classes des sections précédentes, dans les graphes CSG^k (avec $k \geq 1$), il n'y a pas de restriction sur les valeurs de n, d, e, a ou r. Toutefois, cet ensemble de classes de CSP doit encore être étudié en détail pour évaluer son intérêt pratique.

6 Discussion et Perspectives

Cet article portait sur l'analyse de la complexité temporelle des algorithmes génériques classiques de résolution de CSP dans une perspective nouvelle et surtout différente des études menées jusque'à présent. Notre analyse exprime la complexité en termes de nombre de cliques maximales dans la micro-structure (généralisée) du CSP à résoudre. Cette analyse révèle que pour l'essentiel, le backtracking et le forward checking visitent chaque clique maximale de la micro-structure (généralisée) au plus une fois.

A partir de cette analyse, nous déduisons des classes polynomiales de CSP qui peuvent être résolues par des algorithmes classiques en temps polynomial, *sans avoir à reconnaître que l'instance figure dans la classe.* Aussi, les résultats obtenus apportent un éclairage nouveau sur l'analyse de la complexité des CSP.

La première perspective de ce travail est constituée par l'étude de classes de graphes possédant un nombre polynomial de cliques maximales. L'étude de [16] revêt ici un intérêt particulier car elle a permis de caractériser précisément ces classes de graphes sur la base de graphes d'intersection.

Une autre perspective importante consiste à mettre en évidence des liens qui pourraient exister entre notre analyse et les classes polynomiales obtenues par des approches différentes. Il devrait être assez clair notamment que nos classes sont orthogonales aux classes polynomiales basées sur la structure. Par exemple, il n'existe aucune raison pour qu'un CSP arborescent ne possède pas un nombre exponentiel de cliques. Plus généralement, les classes basées sur la structure nécessitent généralement des algorithmes dédiés, et les algorithmes de backtracking génériques sont de complexité polynomiales sur elles uniquement s'ils procèdent, par exemple, via l'exploitation d'un ordre d'affectation des variables spécifique. Par contre, pour les classes définies par un langage de contraintes, il serait possible que certaines soient capturées par notre analyse, tout comme c'est le cas pour le problème de satisfiabilité dans les travaux de [15]. Enfin, les classes hybrides sont beaucoup plus proches dans l'esprit de notre approche

et l'étude en profondeur des liens entre ces classes et notre analyse constitue donc une perspective naturelle à court terme.

En ce qui concerne les solveurs, la complexité d'algorithmes tels que MAC, dont le comportement est différent de ceux qui sont analysés ici, doit également être étudiée, car cette complexité ne s'inscrit pas naturellement dans l'évaluation que nous avons proposée.

Enfin, une perspective importante porte sur l'extension de notre étude aux autres généralisations possibles de la micro-structure pour le cas des problèmes de satisfaction de contraintes non-binaires.

7 Remerciements

Les auteurs tiennent à remercier les relecteurs anonymes de cet article qui ont permis déjà d'en améliorer la rédaction mais aussi, qui ont suggéré des pistes intéressantes pour la poursuite de ce travail.

Références

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

[2] A. Chmeiss and P. Jégou. A generalization of chordal graphs and the maximum clique problem. *Information Processing Letters*, 62 :111–120, 1997.

[3] David A. Cohen. A New Classs of Binary CSPs for which Arc-Constistency Is a Decision Procedure. In *Proceedings of CP 2003*, pages 807–811, 2003.

[4] M. Cooper, D. Cohen, and P. Jeavons. Characterising Tractable Constraints. *Artificial Intelligence*, 65(2) :347–361, 1994.

[5] M. Cooper, Peter Jeavons, and Andras Salamon. Generalizing constraint satisfaction on trees : hybrid tractability and variable elimination. *Artificial Intelligence*, 174 :570–584, 2010.

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

[7] Vida Dujmovic, Gasper Fijavz, Gwenaël Joret, Thom Sulanke, and David R. Wood. On the maximum number of cliques in a graph embedded in a surface. *European J. Combinatorics*, 32(8) :1244–1252, 2011.

[8] E. Freuder. A Sufficient Condition for Backtrack-Free Search. *JACM*, 29 (1) :24–32, 1982.

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

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

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

[12] P. Jégou. Decomposition of Domains Based on the Micro-Structure of Finite Constraint Satisfaction Problems. In *Proceedings of AAAI 93*, pages 731–736, Washingtown, DC, 1993.

[13] J. W. Moon and L. Moser. On cliques in graphs. *Israel Journal of Mathematics*, 3 :23–28, 1965.

[14] B. Nadel. *Tree Search and Arc Consistency in Constraint-Satisfaction Algorithms*, pages 287–342. In *Search in Artificial Intelligence*. Springer-Verlag, 1988.

[15] Antoine Rauzy. Polynomial restrictions of SAT : What can be done with an efficient implementation of the Davis and Putnam's procedure. In U. Montanari and F. Rossi, editors, *Proc. International Conference on Principles of Constraint Programming (CP 1995)*, pages 515–532. Springer Verlag, 1995.

[16] Bill Rosgen and Lorna Stewart. Complexity results on graphs with few cliques. *Discrete Mathematics and Theoretical Computer Science*, 9 :127–136, 2007.

[17] F. Rossi, C. Petrie, and V. Dhar. On the equivalence of constraint satisfaction problems. In *Proceedings of the 9th European Conference on Artificial Intelligence*, pages 550–556, 1990.

[18] F. Rossi, P. van Beek, and T. Walsh. *Handbook of Constraint Programming*. Elsevier, 2006.

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

[20] David R. Wood. On the maximum number of cliques in a graph. *Graphs and Combinatorics*, 23 :337–352, June 2007.