Une famille de classes polynomiales de CSP basée sur la microstructure

Martin Cooper, Philippe Jégou, Cyril Terrioux

Une famille de classes polynomiales de CSP basée sur la microstructure *

Martin Cooper[1] **Philippe Jégou**[2] **Cyril Terrioux**[2]

[1] IRIT, University of Toulouse III,
31062 Toulouse, France
[2] Aix-Marseille Université, LSIS UMR CNRS 7296
Avenue Escadrille Normandie-Niemen
13397 Marseille Cedex 20, France
cooper@irit.fr {philippe.jegou, cyril.terrioux}@lsis.org

Résumé

L'étude des classes polynomiales constitue une question importante en intelligence artificielle, en particulier au niveau des problèmes de satisfaction de contraintes. Dans ce contexte, la propriété BTP fournit une classe importante de l'état de l'art. Dans cet article, nous proposons d'étendre et de généraliser cette classe en introduisant la propriété k-BTP (et la classe des instances satisfaisant cette propriété) où le paramètre k est une constante donnée. Ainsi, nous avons 2-BTP = BTP, et pour $k > 2$, k-BTP est une relaxation de BTP au sens où k-$BTP \subsetneq (k+1)$-BTP. En outre, nous montrons que si k-TW est la classe d'instances ayant une largeur arborescente bornée par une constante k, alors k-$TW \subsetneq (k+1)$-BTP. Au niveau de la complexité, nous montrons que les instances satisfaisant k-BTP et qui vérifient la k-cohérence-forte sont reconnaissables et résolubles en temp polynomial. Nous étudions aussi la relation entre k-BTP et l'approche de W. Naanaa qui a proposé un outil théorique connu sous le vocable *directional rank* afin d'étendre les classes polynomiales de manière paramétrée. Enfin, nous proposons une étude expérimentale de 3-BTP qui montre l'intérêt pratique de cette classe.

Abstract

The study of tractable classes is an important issue in Artificial Intelligence, especially in Constraint Satisfaction Problems. In this context, the Broken Triangle Property (BTP) is a state-of-the-art microstructure-based tractable class which generalizes well-known and previously-defined tractable classes. In this paper, we propose to extend and to generalize this class using a more general approach based on a parameter k which is a given constant. To this end, we introduce the k-BTP property (and the class of instances satisfying this property) such that we have 2-BTP = BTP, and for $k > 2$, k-BTP is a relaxation of BTP in the sense that k-$BTP \subsetneq (k+1)$-BTP. Moreover, we show that if k-TW is the class of instances having tree-width bounded by a constant k, then k-$TW \subsetneq (k+1)$-BTP. Concerning tractability, we show that instances satisfying k-BTP and which are strong k-consistent are tractable, that is, can be recognized and solved in polynomial time. We also study the relationship between k-BTP and the approach of Naanaa who proposed a set-theoretical tool, known as the directional rank, to extend tractable classes in a parameterized way. Finally we propose an experimental study of 3-BTP which shows the practical interest of this class.

*Ce travail est soutenu par l'Agence Nationale de la Recherche dans le cadre du projet TUPLES (ANR-2010-BLAN-0210) ainsi que par l'EPSRC grant EP/L021226/1.

1 Introduction

Identifier des fragments polynomiaux, généralement appelés *classes polynomiales*, est une question importante en intelligence artificielle, en particulier dans les problèmes de satisfaction de contraintes (CSP). De nombreuses études ont abordé cette question, notamment dès les débuts de l'intelligence artificielle. Ces résultats sont souvent de nature théorique avec, dans certains cas, la mise en évidence de classes polynomiales qui peuvent très souvent être considérées comme artificielles, au sens où il s'avère difficile, voire impossible, de les exploiter pour la résolution d'instances du monde réel. Cela étant, certaines classes polynomiales ont cependant effectivement été utilisées en

pratique, comme notamment les classes définies par les réseaux de contraintes de largeur arborescente bornée [6, 11]. Plus récemment, le concept de classe hybride a été introduit notamment avec la classe BTP [1]. Cette classe contient strictement des classes polynomiales structurelles (comme les CSP arborescents) et des classes polynomiales définies par restriction de langage. Un avantage majeur de cette classe, en plus de sa généralisation de classes polynomiales déjà connues, est lié à son intérêt pratique. En effet, les instances de cette classe peuvent être résolues en temps polynomial à l'aide d'algorithmes tels que MAC (Maintaining Arc-Consistency [17]) et RFL (Real Full Look-ahead [15]), algorithmes qui sont généralement mis en œuvre dans les solveurs. Cela permet à BTP d'être directement utilisée en pratique. En outre, cette particularité peut aussi aider à expliquer théoriquement l'efficacité pratique, souvent remarquable des solveurs, alors même que la complexité en temps des algorithmes qu'ils implémentent est exponentielle dans le pire des cas.

Dans cette contribution, nous revenons sur ce type d'approche en généralisant la classe polynomiale BTP dont la définition s'appuie sur l'exclusion de certains motifs (appelés *triangles cassés*) dans le graphe de microstructure associé à toute instance de CSP binaire. Des travaux très récents allant dans la même direction ont introduit la classe ETP [13] qui généralise BTP en relaxant certaines de ses conditions, puisque certains triangles cassés sont tolérés dans ETP alors qu'ils sont interdits pour BTP. Ici, nous proposons une généralisation plus large appelée k-BTP qui étend ces travaux selon deux axes. Tout d'abord, dans le même esprit que ETP, la nouvelle classe tolère la présence d'un nombre plus grand de triangles cassés, généralisant ainsi strictement ETP (et par conséquent BTP). Deuxièmement, la classe k-BTP est paramétrée par une constante k de sorte à offrir une version générique et donc plus large, qui montre son intérêt théorique pour des valeurs quelconques de k, bien qu'en pratique, le cas pour lequel $k = 3$ doit probablement constituer la classe la plus intéressante. Ainsi, alors que BTP est définie pour des ensembles de 3 variables et ETP pour des ensembles de 4 variables, k-BTP est définie sur la base d'ensembles de $k+1$ variables où k est une constante fixée. Dans cette approche, $BTP = 2\text{-}BTP$ tandis que $ETP \subsetneq 3\text{-}BTP$. Ainsi, cette approche rend possible une généralisation stricte de ces deux classes. De plus, k-BTP conserve certaines de leurs propriétés intéressantes ainsi que certains des avantages pratiques. Notamment, nous montrons que les algorithmes classiques comme MAC ou RFL peuvent résoudre des instances appartenant à k-BTP en temps polynomial, en supposant que ces instances vérifient la *k-cohérence-forte* [9]. En outre, nous mettons en évidence les relations de cette classe avec des classes polynomiales structurelles et hybrides de la littérature. Nous montrons en particulier que la classe des réseaux de contraintes dont la largeur arborescente est bornée par k est strictement incluse dans la classe $k+1$-BTP. Ce résultat donne une première réponse à une question posée très récemment par M. Vardi et qui portait sur les relations pouvant exister entre ETP et la classe polynomiale induite par les instances de largeur arborescente bornée [18]. Nous revenons également sur un résultat récent mais relativement méconnu qui a été proposé par W. Naanaa [14] et dont nous étudions les relations avec k-BTP.

Dans la partie 2, nous rappelons les définitions des classes polynomiales BTP et ETP. Dans la partie 3 nous définissons la nouvelle classe k-BTP et nous montrons que les instances de cette classe peuvent être reconnues en temps polynomial. En outre, nous montrons que sous l'hypothèse supplémentaire de vérification de la *k-cohérence-forte*, les instances vérifiant k-BTP peuvent être résolues en temps polynomial et que les algorithmes standards (comme MAC ou RFL) peuvent les résoudre polynomialement. Dans la partie 4 nous étudions les relations entre k-BTP et plusieurs classes polynomiales de la littérature, tandis que dans la partie 5, nous présentons des résultats expérimentaux sur la présence de cette classe au sein des benchmarks utilisés par la communauté, ainsi que sur la résolution de ses instances.

2 Préliminaires

Formellement, un *problème de satisfaction de contraintes* (*CSP*) aussi appelé *réseau de contraintes* est un triplet (X, D, C), où $X = \{x_1, \ldots, x_n\}$ est un ensemble de n variables, $D = (D_{x_1}, \ldots, D_{x_n})$ est une liste de domaines finis de valeurs, un par variable, et $C = \{c_1, \ldots, c_e\}$ est un ensemble de e contraintes. Chaque contrainte c_i est une paire $(S(c_i), R(c_i))$, 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 $|S(c_i)|$. Dans cet article, nous ne considérerons que le cas des CSP binaires, c'est-à-dire des CSP pour lesquels toutes les contraintes ont pour arité 2. Pour simplifier la notation, nous noterons c_{ij} la contrainte portant sur x_i et x_j. La structure d'un réseau de contraintes est représentée par un graphe appelé *graphe de contraintes*, dont les sommets correspondent aux variables et les arêtes aux portées des contraintes. L'affectation des variables d'un sous-ensemble Y de X est dite *cohérente* si elle ne viole aucune contrainte dont la portée est incluse dans Y. Nous utiliserons la notation $R(c_{ij})[a]$ pour représenter l'ensemble des

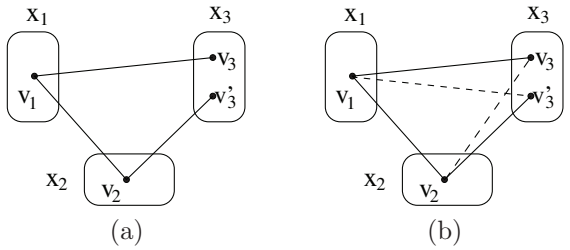

FIGURE 1 – Une instance ne vérifiant pas BTP (a) et une instance BTP (b) par rapport à l'ordre $x_1 < x_2 < x_3$ si l'une des deux arêtes en pointillés est présente.

valeurs de D_{x_j} compatibles avec $a \in D_{x_i}$. Ainsi, s'il y a une contrainte dont la portée est $\{i, j\}$, alors $R(c_{ij})[a] = \{b \in D_{x_j} | (a, b) \in R(c_{ij})\}$; s'il n'y pas de contrainte dont la portée est $\{i, j\}$, alors, par défaut, $R(c_{ij})[a] = D_{x_j}$. Nous rappelons la propriété BTP introduite dans [1].

Définition (BTP) Une instance de CSP binaire (X, D, C) satisfait la *Broken Triangle Property* (BTP) par rapport à un ordre sur les variables $<$ si, pour tout triplet de variables (x_i, x_j, x_k) tel que $i < j < k$, si $(v_i, v_j) \in R(c_{ij})$, $(v_i, v_k) \in R(c_{ik})$ et $(v_j, v'_k) \in R(c_{jk})$, alors soit $(v_i, v'_k) \in R(c_{ik})$, soit $(v_j, v_k) \in R(c_{jk})$. Si aucun de ces deux couples n'existe, (v_i, v_j, v_k, v'_k) est appelé *triangle cassé sur x_k par rapport à x_i et x_j*.

S'il existe au moins un triangle cassé sur x_k par rapport à x_i et x_j, (x_i, x_j, x_k) est appelé *triplet cassé sur x_k par rapport à x_i and x_j*. Soit BTP l'ensemble des instances pour lesquelles il existe un ordre sur les variables tel que BTP est vérifiée par rapport à cet ordre. La propriété BTP est relative à la compatibilité entre valeurs des domaines et peut donc être représentée graphiquement (figure 1) à l'aide du graphe de microstructure. Par exemple, dans la figure 1 (a), il y a un triangle cassé sur x_3 par rapport aux variables x_1 et x_2 puisque nous avons $(v_1, v'_3) \notin R(c_{13})$ et $(v_2, v_3) \notin R(c_{23})$ tandis que $(v_1, v_2) \in R(c_{12})$, $(v_1, v_3) \in R(c_{13})$ et $(v_2, v'_3) \in R(c_{23})$ est vérifiée. Aussi, (x_1, x_2, x_3) est un triplet cassé sur x_3 par rapport à x_1 et x_2. Par contre, dans la figure 1 (b), si l'une des deux arêtes en pointillés (c'est-à-dire des 2-uplets) figure dans la microstructure, la propriété BTP sera vérifiée pour tous les ordres sur les variables.

Très récemment, un travail autour de la propriété BTP a conduit à proposer une propriété voisine, appelée *ETP* pour *Extendable-Triple Property* [13] qui est basée sur une relaxation des conditions de BTP, en considérant quatre variables plutôt que trois, et en tolérant l'existence de certains triangles cassés.

Définition (ETP) Une instance de CSP binaire (X, D, C) satisfait la *Extendable-Triple Property* (ETP) par rapport à un ordre sur les variables $<$ si et seulement si, pour tout quadruplet de variables (x_i, x_j, x_k, x_l) tel que $i < j < k < l$, il existe au plus un triplet cassé sur x_l parmi (x_i, x_j, x_l), (x_i, x_k, x_l) et (x_j, x_k, x_l).

De cette façon, une instance de CSP binaire peut satisfaire la propriété ETP quand bien même elle contiendrait deux triplets cassés parmi (x_i, x_j, x_k, x_l), un sur x_k, et un autre sur x_l, alors qu'aucun n'est autorisé avec BTP. Ainsi, ETP généralise strictement BTP puisqu'une instance peut satisfaire ETP en invalidant BTP alors que l'inverse est faux. Une conséquence immédiate est que la classe des instances satisfaisant BTP est strictement incluse dans la classe des instances satisfaisant ETP (notée ETP) comme indiqué dans le théorème 1 de [13]. i.e. $BTP \subsetneq ETP$. Comme dans le cas de BTP, ETP nous permet de définir une classe polynomiale mais pour cela, il faut imposer une propriété supplémentaire liée au niveau de cohérence locale qui doit être vérifiée par les instances. Alors que l'ensemble des instances satisfaisant BTP définit une classe polynomiale, l'ensemble des instances satisfaisant ETP requiert de plus la satisfaction de la *cohérence de chemin forte* (*Strong-Path-Consistency* [9]), c'est-à-dire la cohérence d'arc et la cohérence de chemin. Néanmoins, ces instances vont conserver certaines des propriétés intéressantes vérifiées par les instances satisfaisant BTP, comme par exemple leur capacité à être résolues en temps polynomial par des algorithmes usuels tels que MAC ou RFL. Dans la partie suivante, nous introduisons une nouvelle propriété qui généralise BTP mais également ETP.

3 k-BTP : définition et propriétées

Définition (k-BTP) Une instance de CSP binaire $P = (X, D, C)$ satisfait la propriété k-BTP pour un k donné $(2 \leq k < n)$ par rapport à un ordre sur les variables $<$ si et seulement si, pour tout sous-ensemble de $k + 1$ variables $x_{i_1}, x_{i_2}, \ldots x_{i_{k+1}}$ tel que $i_1 < i_2 < \ldots < i_{k-1} < i_k < i_{k+1}$, il existe au moins un triplet de variables $(x_{i_j}, x_{i_{j'}}, x_{i_{k+1}})$ avec $1 \leq j \neq j' \leq k$ tel qu'il n'existe pas de triangle cassé sur $x_{i_{k+1}}$ par rapport à x_{i_j} et $x_{i_{j'}}$.

On notera k-BTP l'ensemble des instances pour lesquelles il existe un ordre sur les variables tel que k-BTP est vérifiée par rapport à cet ordre. On peut constater que 2-BTP est exactement BTP alors que 3-BTP inclut ETP. À partir de là, on peut immédiatement étendre le théorème 1 de [13] puisque $BTP \subsetneq$

$ETP \subsetneq 3\text{-}BTP$. Mais au-delà, un résultat plus général, qui est une conséquence immédiate de la définition de k-BTP peut être formulé :

Théorème 1 *Pour tout $k \geq 2$, on a :*

$$k\text{-}BTP \subsetneq (k{+}1)\text{-}BTP$$

Pour analyser la polynomialité du traitement de k-BTP, nous montrons maintenant que les instances de cette classe peuvent être reconnues en temps polynomial :

Théorème 2 *Étant donnée une instance de CSP binaire (X, D, C) et une constante k avec $2 \leq k < n$, il existe un algorithme de complexité polynomiale pour déterminer un ordre sur les variables $<$ tel que cette instance satisfait k-BTP par rapport à $<$, ou pour déterminer qu'un tel ordre n'existe pas.*

Preuve : Comme dans la preuve correspondante pour BTP [1] et ETP [13], nous définissons une instance de CSP notée P_o qui est cohérente si et seulement si un ordre permettant de vérifier k-BTP existe. Plus précisément, cette instance possède une variable o_i de domaine $\{1, \ldots, n\}$ par variable x_i de X. La valeur de o_i représente la position de la variable x_i dans l'ordre. Nous ajoutons une contrainte concernant $\{o_{i_1}, o_{i_2}, \ldots o_{i_k}, o_{i_{k+1}}\}$ et imposant la condition $o_{i_{k+1}} < max(o_{i_1}, o_{i_2}, \ldots o_{i_k})$ pour chaque $k{+}1$-uplet de variables $(x_{i_1}, x_{i_2}, \ldots x_{i_k}, x_{i_{k+1}})$ tel que chaque triplet de variables $(x_{i_j}, x_{i_{j'}}, x_{i_{k+1}})$ avec $1 \leq j \neq j' \leq k$ possède au moins un triangle cassé sur $x_{i_{k+1}}$ par rapport à x_{i_j} et $x_{i_{j'}}$. Si P_o possède une solution, un ordre total $<$ sur les variables peut être produit à partir de l'ordre partiel donné par les valeurs des variables o_i. Alors, pour chaque $k{+}1$-uplet de variables $(x_{i_1}, x_{i_2}, \ldots x_{i_k}, x_{i_{k+1}})$, avec $i_1 < \ldots < i_{k+1}$, nous avons au moins un triplet de variables $(x_{i_j}, x_{i_{j'}}, x_{i_{k+1}})$ avec $1 \leq j \neq j' \leq k$ qui n'a pas de triangle cassé sur $x_{i_{k+1}}$ par rapport à x_{i_j} et $x_{i_{j'}}$. En effet, si ce n'était pas le cas, alors la contrainte $o_{i_{k+1}} < max(o_{i_1}, o_{i_2}, \ldots o_{i_k})$ aurait été imposée, ce qui serait en contradiction avec $i_1 < \ldots < i_{k+1}$. Donc, si P_o possède une solution, nous avons un ordre satisfaisant la propriété k-BTP.

Inversement, considérons un ordre satisfaisant la propriété k-BTP et supposons que P_o n'a pas de solution. Cela signifie qu'au moins une contrainte $o_{i_{k+1}} < max(o_{i_1}, o_{i_2}, \ldots o_{i_k})$ est violée. Donc chaque triplet de variables $(x_{i_j}, x_{i_{j'}}, x_{i_{k+1}})$ avec $1 \leq j \neq j' \leq k$ possède au moins un triangle cassé sur $x_{i_{k+1}}$, ce qui est impossible puisque cet ordre vérifie la propriété k-BTP. Ainsi P_o possède une solution si et seulement si (X, D, C) admet un ordre satisfaisant la propriété k-BTP. Nous montrons maintenant que P_o peut être construit et résolu en temps polynomial. Trouver tous les triplets cassés peut être réalisé en $O(n^3.d^4)$, tandis que la définition des contraintes $o_{i_{k+1}} < max(o_{i_1}, o_{i_2}, \ldots o_{i_k})$ peut être réalisée en $O(n^{k+1})$. Donc P_o peut être calculé en $O(n^3.d^4 + n^{k+1})$. En outre, P_o peut être résolu en temps polynomial en établissant la cohérence d'arc généralisée puisque ses contraintes sont *max-closed* [12]. □

Nous analysons maintenant la complexité de la résolution des instances de la classe $k\text{-}BTP$. Afin d'assurer la polynomialité de $k\text{-}BTP$, nous considérons une condition supplémentaire qui porte sur la vérification par les instances de la *k-cohérence-forte* dont nous rappelons la définition.

Definition (k-cohérence-forte [9]) *Une instance de CSP binaire satisfait la i-cohérence si toute affectation cohérente de $i-1$ variables peut être étendue à une affectation cohérente sur toute $i^{ème}$ variable. Une instance de CSP binaire satisfait la k-cohérence-forte si elle satisfait la i-cohérence pour tout i tel que $1 \leq i \leq k$.*

La k-cohérence-forte et $k\text{-}BTP$ permettent de définir une nouvelle classe polynomiale :

Théorème 3 *Soit P une instance de CSP binaire telle qu'il existe une constante k avec $2 \leq k < n$ pour laquelle P satisfait k-BTP par rapport à un ordre sur les variables $<$ ainsi que la k-cohérence-forte. Alors l'instance P est cohérente et une solution peut être trouvée en temps polynomial.*

Preuve : Considérons un ordre pour l'affectation des variables correspondant à l'ordre $<$. Comme l'instance satisfait la k-cohérence-forte, elle satisfait la cohérence d'arc et par conséquent, aucun de ses domaines n'est vide et chaque valeur possède au moins un support dans chacun des autres domaines. De plus, comme l'instance satisfait la k-cohérence-forte, il existe une affectation cohérente des k premières variables. Maintenant, et plus généralement, supposons que nous disposons d'une affectation cohérente $(u_1, u_2, \ldots u_{l-1}, u_l)$ pour les l premières variables $x_1, x_2, \ldots x_{l-1}, x_l$ dans l'ordre, avec $k \leq l < n$. Nous montrons que cette affectation peut être étendue de façon cohérente sur la variable x_{l+1}. Pour montrer cela, nous devons prouver que $\cap_{1 \leq i \leq l} R(c_{il+1})[u_i] \neq \emptyset$, c'est-à-dire qu'il existe au moins une valeur dans le domaine de x_{l+1} qui est compatible avec l'affectation $(u_1, u_2, \ldots u_{l-1}, u_l)$.

Nous prouvons d'abord cela pour $l = k$. Considérons l'affectation cohérente $(u_1, u_2, \ldots u_{k-1}, u_k)$ sur les k premières variables. Considérons une $k{+}1^{ème}$ variable

x_{k+1} apparaissant plus loin dans l'ordre. Puisque P satisfait k-BTP, il existe au moins un triplet de variables $(x_j, x_{j'}, x_{k+1})$ avec $1 \leq j \neq j' \leq k$ tel qu'il n'existe pas de triangle cassé sur x_{k+1} par rapport à x_j et $x_{j'}$. D'après le lemme 2.4 présenté dans [1], nous avons :

$$(R(c_{jk+1})[u_j] \subseteq R(c_{j'k+1})[u_{j'}])$$

ou

$$(R(c_{j'k+1})[u_{j'}] \subseteq R(c_{jk+1})[u_j])$$

Sans manque de généralité, supposons que $R(c_{jk+1})[u_j] \subseteq R(c_{j'k+1})[u_{j'}]$ et $j < j'$. Puisque P satisfait la k-cohérence-forte, nous savons que l'affectation partielle de $(u_1, u_2, \ldots, u_j, \ldots u_{k-1}, u_k)$ sur $k-1$ variables excluant l'affectation $u_{j'}$ pour $x_{j'}$ peut être étendue de façon cohérente sur la variable x_{k+1}. De plus, nous savons que $R(c_{jk+1})[u_j] \subseteq R(c_{j'k+1})[u_{j'}]$ et par la cohérence d'arc, $R(c_{i_j i_{k+1}})[u_j] \neq \emptyset$. Par conséquent, $(u_1, u_2, \ldots, u_j, \ \ldots, \ u_{j'}, \ \ldots, u_k, u_{k+1})$ est une affectation cohérente des $k+1$ premières variables.

Notons que cette preuve est également valide pour tous les sous-ensembles de $k+1$ variables tels que x_{k+1} apparaît plus loin dans l'ordre $<$, et pas seulement pour les $k+1$ premières variables $x_1, x_2, \ldots x_{k-1}, x_k$ et x_{k+1}.

Maintenant, nous démontrons la propriété pour l avec $k < l < n$. C'est-à-dire que nous montrons qu'une affectation cohérente $(u_1, u_2, \ldots u_{l-1}, u_l)$ peut être étendue à une $(l+1)^{ème}$ variable. Comme hypothèse d'induction, nous supposons que chaque affectation cohérente sur $l-1$ variables peut être étendue à une $l^{ème}$ variable qui apparaît plus loin dans l'ordre considéré.

Considérons une affectation cohérente $(u_1, u_2, \ldots u_{l-1}, u_l)$ sur les l premières variables. Soit $(u_{i_1}, u_{i_2}, \ldots u_{i_k})$ une affectation partielle de $(u_1, u_2, \ldots u_{l-1}, \ u_l)$ sur k variables. Comme P satisfait k-BTP, et comme $k < l < n$, pour tous les sous-ensembles de k variables $x_{i_1}, x_{i_2}, \ldots x_{i_k}$, nous savons qu'il existe un triangle qui n'est pas cassé sur x_{l+1} par rapport à x_{i_j} et $x_{i_{j'}}$, avec x_{i_j} et $x_{i_{j'}}$ apparaissant dans les variables $x_{i_1}, x_{i_2}, \ldots x_{i_k}$. Aussi, sans manque de généralité, nous pouvons considérer que $i_1 \leq i_j < i_{j'} \leq i_k \leq l$ et que nous avons $R(c_{i_j l+1})[u_{i_j}] \subseteq R(c_{i_{j'} l+1})[u_{i_{j'}}]$. Notons que x_{i_j} et $x_{i_{j'}}$ peuvent être permutées dans l'ordre si c'est nécessaire.

Maintenant, considérons l'affectation cohérente $(u_1, u_2, \ldots u_{l-1}, u_l)$ sur les l premières variables. Par hypothèse d'induction, chaque affectation partielle de $(u_1, u_2, \ldots u_{l-1}, u_l)$ sur $l-1$ variables peut être étendue à une affectation cohérente sur x_{l+1} avec une valeur compatible u_{l+1}. Maintenant, considérons l'affectation partielle sur $l-1$ variables où $u_{i_{j'}}$ n'apparaît pas. Cette affectation est par exemple $(u_1, u_2, \ldots u_{i_j}, \ldots u_{l-1}, u_l, u_{l+1})$. Comme nous avons $R(c_{i_j l+1})[u_{i_j}] \subseteq R(c_{i_{j'} l+1})[u_{i_{j'}}]$, la valeur $u_{i_{j'}}$ est également compatible avec u_{l+1}, et donc l'affectation $(u_1, u_2, \ldots u_{i_j}, \ldots u_{i_{j'}}, \ldots u_{l-1}, u_l, u_{l+1})$ sur les $l+1$ premières variables est une affectation cohérente.

Ainsi, toute affectation cohérente $(u_1, u_2, \ldots u_{l-1}, u_l)$ sur $(x_1, x_2, \ldots x_{l-1}, x_l)$ peut être étendue à une $(l+1)^{ème}$ variable, pour tout l avec $k < l < n$. Et plus généralement, nous avons démontré que toute affectation cohérente sur l variables, pas nécessairement consécutives dans l'ordre (comme le sont les l premières variables), peut être étendue à une affectation cohérente sur toute $(l+1)^{ème}$ variable qui apparaît après ces l variables dans l'ordre $<$ associé à k-BTP. Ainsi, l'hypothèse d'induction est vérifiée pour l'étape suivante.

Notons que cette preuve démontre également qu'une instance qui satisfait la k-cohérence-forte et k-BTP par rapport à un ordre $<$ est cohérente.

Finalement, étant donné un ordre $<$, nous montrons que trouver une solution peut être réalisé en temps polynomial. Étant donnée une affectation cohérente $(u_1, u_2, \ldots u_l)$ avec $l < n$, trouver une valeur compatible u_{l+1} pour la variable suivante x_{l+1} est réalisable en recherchant dans son domaine une valeur compatible, sachant que la taille du domaine est majorée par d. Pour chaque valeur, nous devons vérifier les contraintes connectant la variable x_{l+1} ce qui peut être réalisé en $O(e_{l+1})$ si la variable suivante x_{l+1} possède e_{l+1} voisins parmi les variables précédentes. Puisque $\Sigma_{1 \leq l < n} e_{l+1} = e$, le coût total pour trouver une solution est $O((n+e).d)$. □

Dans la suite, nous noterons k-BTP-SkC, la classe des instances satisfaisant k-BTP et vérifiant la k-cohérence-forte. Une des propriétés les plus intéressantes de la classe BTP est le fait que les instances de cette classe peuvent être résolues en temps polynomial en utilisant des algorithmes classiques (tels que MAC ou RFL), cela mêmes qui sont mis en œuvre dans la plupart des solveurs. La propriété suivante établit un résultat similaire pour k-BTP-SkC. En effet, la preuve du théorème 3 nous permet de montrer que des algorithmes tels que BT (Backtracking), MAC et RFL peuvent résoudre (i.e. trouver une solution) toute instance de la classe k-BTP-SkC en temps polynomial :

Théorème 4 *Étant donné une instance P de CSP binaire et un ordre sur les variables $<$ tel que P satisfait k-BTP par rapport à $<$ et vérifie la k-cohérence-forte, les algorithmes BT, MAC et RFL trouvent une solution de l'instance P en temps polynomial.*

Preuve : Comme l'instance satisfait la k-cohérence-forte, BT utilisant l'ordre $<$ pour l'affectation des

variables peut trouver une affectation cohérente sur $x_1, x_2, \ldots x_{k-1}$ et x_k. De plus, étant donné l avec $k < l < n$, il est montré dans la preuve du théorème 3 qu'une affectation cohérente $(u_1, u_2, \ldots u_{l-1}, u_l)$ sur $x_1, x_2, \ldots x_{l-1}$ et x_l peut être étendue à une $(l+1)^{ème}$ variable, c'est-à-dire sur x_{l+1}. Pour trouver une affectation de x_{l+1}, nous devons chercher une valeur compatible dans son domaine. Cela est réalisable en $O(e_{l+1}.d)$ en supposant que x_{l+1} possède e_{l+1} voisins dans les variables précédentes. Donc, comme pour la preuve du théorème 3, trouver une solution de P est globalement réalisable en $O((n+e).d)$. Si nous considérons maintenant des algorithmes tels que MAC ou RFL, par le même raisonnement, nous montrons que leur complexité est limitée à $O(n.(n+e).d^2)$ en raison du coût supplémentaire du filtrage par cohérence d'arc effectué après chaque affectation de variable. □

Dans la partie 5, nous discutons de l'intérêt de la classe k-BTP d'un point de vue pratique. Avant cela, dans la partie suivante, nous étudions les relations existant entre k-BTP et certaines classes polynomiales de la littérature.

4 Relations existant entre k-BTP et d'autres classes polynomiales

Nous considérons tout d'abord une classe polynomiale très importante, à la fois dans le cadre des CSP, mais bien au-delà, la classe basée sur la notion de décomposition arborescente de graphes [16].

Définition (Décomposition arborescente) *Étant donné un graphe $G = (X, C)$, une* décomposition arborescente *de G est une paire (E, T) où $T = (I, F)$ est un arbre et $E = \{E_i : i \in I\}$ une famille de sous-ensembles (appelés* clusters*) de X, telle que chaque cluster E_i est un nœud de T et vérifie :*

(i) $\cup_{i \in I} E_i = X$,

(ii) pour chaque arête $\{x, y\} \in C$, il existe $i \in I$ avec $\{x, y\} \subseteq E_i$, et

(iii) pour tout $i, j, k \in I$, si k est sur un chemin de i vers j dans T, alors $E_i \cap E_j \subseteq E_k$.

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

Soit k-TW la classe des instances de CSP binaires tels que leur largeur arborescente est inférieure ou égale à une constante k. Il est bien connu que k-TW constitue une classe polynomiale [11]. Récemment, M. Vardi a posé une question sur les relations qui pourraient exister entre k-TW et ETP ou d'autres généralisations de BTP [18]. Les deux théorèmes qui suivent donnent une première réponse à cette question.

Théorème 5 *k-$TW \subsetneq (k+1)$-BTP.*

Preuve : Nous montrons tout d'abord que k-TW $\subseteq (k+1)$-BTP. Il est bien connu que si la largeur arborescente d'une instance de CSP binaire est bornée par k, il existe un ordre $<$ sur les variables, tel que pour $x_i \in X$, $|\{x_j \in X : j < i \text{ et } c_{ji} \in C\}| \leq k$ [6]. Maintenant, considérons un sous-ensemble de $k+2$ variables $x_{i_1}, x_{i_2}, \ldots x_{i_k}, x_{i_{k+1}}, x_{i_{k+2}}$ tel que $i_1 < i_2 < \ldots < i_{k-1} < i_k < i_{k+1} < i_{k+2}$. Puisque la largeur arborescente est bornée par k, nous savons qu'il existe au plus k contraintes $c_{i_j i_{k+2}} \in C$. Donc, il y a au moins un triplet de variables $(x_{i_j}, x_{i_{j'}}, x_{i_{k+2}})$ avec $1 \leq j \neq j' \leq k$ tel que $c_{i_j i_{k+2}} \notin C$ ou $c_{i_{j'} i_{k+2}} \notin C$. Sans manque de généralité, supposons qu'il n'y a pas de contrainte $c_{i_j i_{k+2}} \in C$. Ainsi, il n'y a aucun triangle cassé sur $x_{i_{k+2}}$ par rapport à x_{i_j} et $x_{i_{j'}}$ parce que toutes les valeurs de $D_{x_{i_j}}$ sont compatibles avec toutes les valeurs de $D_{x_{i_{k+2}}}$. Ainsi, l'instance de CSP considérée satisfait la propriété $(k+1)$-BTP. Finalement, il est facile de définir des instances dont la largeur arborescente est strictement supérieure à k et qui satisfont la propriété $(k+1)$-BTP. Par exemple, nous pouvons considérer une instance de CSP monovalente (i.e. dont la taille des domaines vaut un) et dont le graphe de contraintes est complet, et possédant une solution. La largeur arborescente de cette instance est $n-1$ alors qu'elle satisfait k-BTP pour toutes les valeurs possibles de k. □

Le coût de la vérification de cohérence d'instances de k-TW est du même ordre que celui de l'obtention de la $(k+1)$-cohérence-forte, qui est $O(n^{k+1}d^{k+1})$. Néanmoins, cela ne nous permet pas d'établir une inclusion formelle de k-TW dans $(k+1)$-BTP-$S(k+1)C$ qui est une classe polynomiale tandis $(k+1)$-BTP n'est pas nécessairement une classe polynomiale. Mais si l'on note k-TW-$S(k+1)C$, la classe des instances de CSP binaires appartenant à k-TW et qui satisfont la $(k+1)$-cohérence-forte, nous obtenons directement le résultat suivant :

Théorème 6 *Pour toute valeur $k < n$, on a :*

$$k\text{-}TW\text{-}S(k+1)C \subsetneq (k+1)\text{-}BTP\text{-}S(k+1)C.$$

Récemment, la classe polynomiale BTP a également été généralisée avec la définition de la propriété $\forall\exists$-BTP [2], d'une manière différente de celle que nous proposons ici, mais également en remarquant que tous les triangles cassés n'ont pas besoin d'être interdits.

Nous allons montrer que ces deux généralisations sont orthogonales.

Définition (∀∃-BTP) *Une instance de CSP binaire P satisfait la propriété* ∀∃-BTP *par rapport à un ordre des variables $<$ si et seulement si, pour chaque paire de variables x_i, x_k telles que $i < k$, pour tout valeur $v_i \in D_{x_i}$, $\exists v_k \in D_{x_k}$ telle que $(v_i, v_k) \in R(c_{ik})$ et pour toute variable x_j avec $j < k$ et $j \neq i$, et pour toute valeur $v_j \in D_{x_j}$ et pour toute valeur $v'_k \in D_{x_k}$, (v_i, v_j, v_k, v'_k) n'est pas un triangle cassé sur x_k par rapport à x_i et x_j. Soit ∀∃-BTP l'ensemble des instances pour lesquelles* ∀∃-BTP *est vérifiée par rapport à un certain ordre sur les variables.*

La classe ∀∃-BTP peut être résolue et reconnue en temps polynomial [2]. Elle constitue une classe polynomiale qui inclut strictement BTP car elle n'interdit pas tous les triangles cassés. Puisque k-BTP n'interdit également pas tous les triangles cassés, il est naturel de comparer ces deux classes. Nous analysons cela pour le cas particulier $k = 3$, mais le même argument vaut pour toute valeur de $k \geq 3$.

Théorème 7 *Même pour les ensembles d'instances de CSP binaires qui vérifient la cohérence de chemin forte, les propriétés* 3-BTP *et* ∀∃-BTP *sont incomparables.*

Preuve : Prenons une instance P^* pour laquelle chaque domaine D_{x_k} contient une valeur a^* telle que pour toutes les autres variables x_i, pour toutes les valeurs $v_i \in D_{x_i}$, $(v_i, a^*) \in R(c_{ik})$. On remarque que P^* satisfait ∀∃-BTP puisqu'il ne peut y avoir aucun triangle cassé de la forme (v_i, v_j, a^*, v'_k), la valeur a^* étant compatible avec toutes les affectations à toutes les autres variables. Il est facile de compléter une tellle instance P^* de sorte qu'elle ne satisfasse pas 3-BTP pour tout ordre sur les variables en rajoutant des triangles cassés sur d'autres valeurs de domaine que a^*.

Considérons une instance à trois variables notée P_3, avec des domaines $\{0, \ldots, 7\}$ et qui possède les trois contraintes suivantes :

$$\begin{aligned} x_1 = x_2 \quad &\wedge \quad x_1 \equiv x_2 + 1 \pmod 8 \\ x_2 = x_3 \quad &\wedge \quad x_2 \equiv x_3 + 2 \pmod 8 \\ x_1 = x_3 \quad &\wedge \quad x_1 \equiv x_3 + 4 \pmod 8 \end{aligned}$$

Ainsi, P_3 vérifie la cohérence de chemin forte et satisfait trivialement 3-BTP (car il n'y a que trois variables), mais P_3 ne satisfait pas ∀∃-BTP, quel que soit l'ordre des variables. □

Nous considérons maintenant une classe polynomiale très générale récemment découverte par Naanaa [14] et qui mérite sans aucun doute d'être mieux connue.

Soit E un ensemble fini et soit $\{E_i\}_{i \in I}$ une famille finie de sous-ensembles de E. La famille $\{E_i\}_{i \in I}$ est dite *indépendante* si et seulement si pour tout $J \subset I$,

$$\bigcap_{i \in I} E_i \ \subset \ \bigcap_{j \in J} E_j.$$

Notons que $\{E_i\}_{i \in I}$ ne peut être indépendante si $\exists j \neq j' \in I$ tels que $E_j \subseteq E_{j'}$ puisque dans ce cas et avec $J = I \setminus \{j'\}$, nous devrions avoir

$$\bigcap_{i \in I} E_i \ = \ \bigcap_{j \in J} E_j.$$

Définition (Rang Directionnel) *Soit P une instance de CSP binaire dont les variables sont totalement ordonnées par $<$. Le* rang directionnel *de la variable x_m est la taille k de la plus grande affectation cohérente $(a_1, \ldots, a_k)$ à un ensemble de variables $x_{i_1}, \ldots, x_{i_k}$ (avec $i_1 < \ldots < i_k < m$) telle que la famille des ensembles $\{R(c_{i_j m})[a_j]\}_{j=1,\ldots,k}$ est indépendante. Le rang directionnel de P (par rapport à l'ordre $<$ de ses variables) est le rang directionnel maximal sur toutes ses variables.*

Naanaa a montré que si P est une instance de CSP binaire qui a un rang directionnel ne dépassant pas k et s'il vérifie la $(k+1)$-cohérence-forte, alors P est globalement cohérente [14]. On note DR-k, l'ensemble de ces instances. Naanaa souligne que certaines classes polynomiales connues telles que les instances de CSP binaires avec des contraintes *connected row convex* [7], ont un rang directionnel borné.

Si une instance de CSP binaire P est $(k+1)$-BTP, alors aucune variable peut avoir un rang directionnel plus grand que k. Ceci est dû au fait que pour toute variable x_m et toute affectation $(a_1, \ldots, a_{k+1})$ d'un ensemble de variables $x_{i_1}, \ldots, x_{i_{k+1}}$ avec $i_1 < \ldots < i_{k+1} < m$, par la définition de $(k+1)$-BTP, nous devons avoir $R(c_{i_j m})[a_j] \subseteq R(c_{i_{j'} m})[a_{j'}]$ pour certains $j \neq j' \in \{1, \ldots, k+1\}$. Donc, comme observé ci-dessus, les ensembles $\{R(c_{i_j m})[a_j]\}_{j=1,\ldots,k+1}$ ne peuvent pas être indépendants. Il en résulte que la polynomialité de $(k+1)$-BTP-$S(k+1)C$ est également un corollaire du résultat de Naanaa [14]. Toutefois, la propriété $(k+1)$-BTP, bien que subsumée par DR-k, peut être détectée en $O(n^k d^k + n^3 d^4)$ ce qui inférieur comparativement à $O(n^{k+1} d^{k+1})$ pour DR-k.

5 Expérimentations

Dans cette section, nous comparons l'intérêt pratique des classes polynomiales BTP, ETP-SPC, k-BTP-SkC et DR-(k-1), où SPC notera la cohérence

de chemin forte, soit la 3-cohérence-forte. Nous ne considérons que les cas pour lesquels $k = 3$, puisque l'établissement de la k-cohérence-forte devient trop coûteuse en temps, mais aussi du fait que pour $k > 3$, cela peut conduire à l'ajout de contraintes d'arité $k-1$. Nos expériences concernent 2 373 instances binaires de la troisième compétition de solveurs CSP [1]. Il s'agit donc d'instances de CSP binaires issues des benchmarks utilisés pour les évaluations et les comparaisons de solveurs. Nous mettons d'abord en évidence l'existence d'instances appartenant à certaines des classes polynomiales considérées ici. Ce sont les mêmes benchmarks qui ont été utilisés dans [13]. Ensuite, nous évaluons l'impact de ces classes polynomiales sur l'efficacité des solveurs. La plupart des solveurs de l'état de l'art reposant sur les algorithmes MAC ou RFL, nous présentons donc ici les résultats obtenus sur MAC mais on peut noter que nous avons observé des résultats similaires avec RFL.

Puisque les classes polynomiales ETP-SPC, 3-BTP-SPC et DR-2 exigent la vérification de la cohérence de chemin forte, nous devons d'abord réaliser un tel filtrage sur chaque instance avant de vérifier si elle appartient aux classes considérées, ceci dans le même esprit que [8]. Ce faisant, 628 instances ont été détectées comme incohérentes et donc elles appartiennent trivialement à toutes ces classes polynomiales. 85 des instances restantes appartiennent à 3-BTP-SPC, alors que 87 ont un rang directionnel inférieur ou égal à deux. Parmi ces instances, nous avons respectivement 71 et 76 instances appartenant à BTP-SPC et à ETP-SPC. Les différences entre ces classes polynomiales sont bien mises en évidence par certaines instances de la famille bqwh-15-106 puisque nous pouvons observer toutes les configurations possibles des relations d'inclusions $BTP\text{-}SPC \subsetneq ETP\text{-}SPC \subsetneq 3\text{-}BTP\text{-}SPC \subsetneq DR\text{-}2$. Par exemple, l'instance bqwh-15-106-13 appartient à toutes les classes polynomiales considérées alors que les instances bqwh-15-106-28, bqwh-15-106-16 et bqwh-15-106-76 appartiennent seulement respectivement à trois, deux ou une de ces classes polynomiales. Le tableau 5 présente certaines instances appartenant aux classes ETP-SPC, 3-BTP-SPC ou DR-2. Ce tableau fournit également la largeur arborescente w de ces instances et leur largeur arborescente w' une fois que le filtrage par cohérence de chemin forte a été appliqué. Quand la largeur arborescente exacte est inconnue (rappelons que le calcul d'une décomposition arborescente optimale est un problème NP-difficile), nous donnons son encadrement par un intervalle. Nous pouvons noter la diversité de ces instances (instances académiques, aléatoires ou issues du monde réel). Certaines de ces instances appartiennent à 3-BTP-SPC ou DR-2 grâce à leur structure. Par exemple, graph12-w0 et hanoi-7 ont un graphe de contraintes acyclique tandis que la largeur arborescente de domino-100-100 et crossword-m1-uk-puzzle01 est de deux. Cependant, la plupart des instances ont une largeur arborescente supérieure à deux. En outre, dans la plupart des cas, l'application de SPC peut augmenter de manière significative la largeur arborescente d'origine de ces instances. Par exemple, la largeur arborescente de l'instance driverlogw-09-sat est initialement majorée par 108 et est égale à 629 après l'application du filtrage SPC. Cette augmentation est expliquée par les paires de valeurs qui sont interdites par SPC et dont la mise en évidence conduit à la représentation de contraintes induites dont l'ajout a pour effet de densifier le graphe de contraintes. Lorsque SPC interdit une paire de valeurs (v_i, v_j) pour une paire donnée de variables (x_i, x_j), le filtrage supprime (v_i, v_j) de la relation $R(c_{ij})$ si la contrainte c_{ij} existe. Cependant, si la contrainte c_{ij} n'existe pas encore, le filtrage SPC doit d'abord l'ajouter au problème. Dans un tel cas, du fait des contraintes supplémentaires et de leur nombre, la largeur arborescente peut augmenter de manière significative. Il est à noter que pour les instances considérées dont la largeur arborescente est d'au plus deux initialement, la largeur arborescente demeure inchangée après l'application du filtrage SPC.

En ce qui concerne la résolution, toutes les instances appartenant à 3-BTP-SPC ou DR-2 sont résolues par MAC sans aucun retour arrière, sauf l'instance driverlogw-04c-sat qui n'y a recours qu'une seule fois. Il faut noter que MAC n'a pas connaissance de l'ordre sur les variables nécessaire à la satisfaction de 3-BTP ni d'ailleurs de l'ordre associé à un rang directionnel au plus égal à deux. Dans la plupart des cas, nous avons d'ailleurs observé que l'instance de CSP utilisée dans la preuve du théorème 2 afin de calculer un ordre sur les variables approprié n'a pas de contrainte. Ainsi, tout ordre des variables est approprié. En revanche, pour une douzaine d'instances, ce CSP possède plusieurs contraintes mais reste nettement sous-contraint et le réseau de contraintes correspondant possède plusieurs composantes connexes. Il en résulte que le CSP associé à l'ordre possède généralement un grand nombre de solutions. Il est donc très probable que MAC exploite implicitement pour l'affectation des variables un ordre approprié. Par exemple, le CSP associé à l'ordre pour vérifier si l'instance bqwh-15-106-76 (qui possède 106 variables) a un rang directionnel majoré par deux possède 65 composantes connexes et admet plus de 33 millions de solutions.

Certaines des instances sont résolues efficacement par MAC sans retour arrière, même si elles ne font

1. Voir http ://www.cril.univ-artois.fr/CPAI08.

Instance	n	w	w'	BTP-SPC	ETP-SPC	3-BTP-SPC	DR-2
bqwh-15-106-13	106	[7, 48]	104	oui	oui	oui	oui
bqwh-15-106-16	106	[6, 45]	99	non	non	oui	oui
bqwh-15-106-28	106	[7, 52]	105	non	oui	oui	oui
bqwh-15-106-76	106	[6, 44]	100	non	non	non	oui
bqwh-15-106-77	106	[7, 50]	100	non	non	oui	oui
bqwh-18-141-33	141	[7, 64]	134	oui	oui	oui	oui
bqwh-18-141-57	141	[7, 66]	137	oui	oui	oui	oui
domino-100-100	100	2	2	oui	oui	oui	oui
domino-5000-500	5000	2	2	oui	oui	oui	oui
driverlogw-04c-sat	272	[19, 56]	[214, 221]	non	non	non	oui
driverlogw-09-sat	650	[39, 108]	629	oui	oui	oui	oui
fapp17-0300-10	300	[6, 153]	[6, 154]	oui	oui	oui	oui
fapp18-0350-10	350	[5, 192]	[12, 199]	oui	oui	oui	oui
fapp23-1800-9	1800	[6, 1325]	[41, 1341]	oui	oui	oui	oui
graph12-w0	680	1	1	oui	oui	oui	oui
graph13-w0	916	1	1	oui	oui	oui	oui
hanoi-7	126	1	1	oui	oui	oui	oui
langford-2-4	8	7	7	oui	oui	oui	oui
lard-83-83	83	82	82	non	non	oui	oui
lard-91-91	91	90	90	non	non	oui	oui
os-taillard-4-100-0	16	[3, 9]	15	oui	oui	oui	oui
os-taillard-4-100-9	16	[3, 9]	15	oui	oui	oui	oui
scen5	400	[11, 32]	[167, 188]	non	non	oui	oui

TABLE 1 – Quelques instances appartenant à BTP-SPC, ETP-SPC, 3-BTP-SPC ou DR-2 après l'application de SPC, avec leur largeur arborescente w et la largeur arborescente w' des instances une fois SPC appliquée.

pas partie de l'une des classes polynomiales étudiées. Aussi, nous considérons maintenant la notion de *backdoor* [19] avec l'objectif de proposer des explications au sujet de cette efficacité, dans le même esprit que dans [13]. Un *backdoor* est un ensemble de variables défini par rapport à une classe polynomiale, tel que, une fois ces variables affectées, le sous-problème induit se situe dans cette classe. Ici, nous nous intéressons à des backdoors qui sont découverts implicitement par MAC lors de l'affectation de certaines variables. En effet, après quelques affectations et l'application du filtrage, la partie restante du problème peut devenir finalement polynomiale. Afin de tenter d'observer ce phénomène, nous allons évaluer le nombre de variables qui doivent être affectées avant que MAC ne trouve implicitement un backdoor défini par rapport à l'une des classes étudiées ici. Sur les 50 instances examinées ici, nous avons observé que MAC trouve un backdoor relatif à BTP après avoir affecté plus de variables que pour les autres classes considérées. Les nombres de variables affectées nécessaires pour trouver un backdoor respectivement pour ETP et 3-BTP sont très proches, voire égaux dans la plupart des cas. En considérant DR-2, nous gagnons quelques variables par rapport à ETP et 3-BTP. Par exemple, MAC a besoin d'affecter au plus cinq variables avant de trouver un backdoor par rapport à 3-BTP ou DR-2 pour 14 instances contre 12 et 4 instances pour ETP et BTP [2]. Bien sûr, les instances qui en résultent ne satisfont pas nécessairement la cohérence de chemin forte et cela ne nous permet pas d'exploiter le théorème 4 pour expliquer l'efficacité de MAC. Néanmoins, comme évoqué ci-dessus, le grand nombre de solutions du CSP correspondant à l'existence de "bons ordres" permet probablement à MAC d'exploiter implicitement un ordre approprié.

6 Conclusion

Cet article présente une nouvelle famille de classes polynomiales pour les CSP binaires, k-BTP, et dont la polynomialité est associée à un niveau donné de k-cohérence-forte. Cette famille est basée sur une hiérarchie de classes d'instances dont la classe BTP constitue le cas de base (BTP = 2-BTP). Alors que BTP est définie sur des sous-ensembles de 3 variables, les classes k-BTP sont définies sur des ensembles de k+1 variables, tout en relaxant les conditions restrictives imposées par BTP. Nous avons montré que k-BTP hérite de certaines des propriétés de BTP, comme par

2. Notons que ces instances ne comprennent pas toutes celles mentionnées dans [13] puisque certaines d'entre elles appartiennent déjà à 3-BTP-SPC et/ou DR-2.

exemple la possiblité d'être résolue en temps polynomial en utilisant des algorithmes standards tels que MAC. Nous avons également démontré que k-BTP généralise strictement la classe des instances dont la largeur arborescente est bornée par une constante et nous avons analysé les relations avec la classe basée sur la notion de *rang directionnel* récemment introduite par Naanaa. Pour évaluer l'intérêt pratique de la classe k-BTP, une analyse expérimentale est présentée. Elle se concentre sur le cas particulier de 3-BTP. Cette analyse montre un avantage significatif de k-BTP, comparativement à BTP et aux CSP de largeur arborescente bornée.

Un développement de ces travaux semble nécessaire pour déterminer si la condition correspondant à la k-cohérence-forte est réellement nécessaire ou si une condition plus faible suffirait pour garantir la polynomialité. En effet, les expériences ont montré que MAC peut résoudre sans retour arrière certaines instances appartenant à 3-BTP même lorsque ces instances ne vérifient pas le niveau correspondant de cohérence. D'un point de vue théorique et pratique, un défi intéressant porte sur la recherche du niveau minimum de cohérence requis parmi les différents types de cohérences locales telles que PIC [10], maxRPC [4] ou SAC [5]. En outre, l'étude d'une relaxation de la condition k-BTP devrait être abordée de manière à étendre la classe des instances qui peuvent être résolues en temps polynomial, mais dans une direction différente de celle proposée dans [14], même si d'autres développements théoriques et expérimentaux semblent clairement nécessaires pour véritablement apprécier toutes les conséquences du résultat de Naanaa. Enfin, il pourrait être intéressant d'étudier, déjà sur le cas particulier de 3-BTP, une approche similaire à celle introduite dans [3] où une nouvelle opération de réduction réalisable en temps polynomial basée sur la fusion des valeurs de domaines est proposée.

Références

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

[2] Martin C. Cooper. Beyond consistency and substitutability. In *Proceedings of CP*, pages 256–271, 2014.

[3] Martin C. Cooper, Achref El Mouelhi, Cyril Terrioux, and Bruno Zanuttini. On broken triangles. In *Proceedings of CP*, pages 9–24, 2014.

[4] R. Debruyne and C. Bessière. From restricted path consistency to max-restricted path consistency. In *Proceedings of CP*, pages 312–326, 1997.

[5] R. Debruyne and C. Bessière. Domain Filtering Consistencies. *Journal of Artificial Intelligence Research*, 14 :205–230, 2001.

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

[7] Yves Deville, Olivier Barette, and Pascal Van Hentenryck. Constraint satisfaction over connected row convex constraints. *Artificial Intelligence*, 109(1-2) :243–271, 1999.

[8] A. El Mouelhi, P. Jégou, and C. Terrioux. Hidden Tractable Classes : from Theory to Practice. In *Proceedings of ICTAI*, pages 437–445, 2014.

[9] E. Freuder. A Sufficient Condition for Backtrack-Free Search. *Journal of the ACM*, 29 (1) :24–32, 1982.

[10] E. Freuder and C.D. Elfe. Neighborhood inverse consistency preprocessing. In *Proceedings of AAAI*, pages 202–208, 1996.

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

[12] P. Jeavons and M. Cooper. Tractable constraints on ordered domains. *Artificial Intelligence*, 79(2) :327–339, 1995.

[13] P. Jégou and C. Terrioux. The Extendable-Triple Property : a new CSP Tractable Class beyond BTP. In *Proceedings of AAAI*, 2015.

[14] Wady Naanaa. Unifying and extending hybrid tractable classes of csps. *Journal of Experimental and Theoretical Artificial Intelligence*, 25(4) :407–424, 2013.

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

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

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

[18] Moshe Vardi. Private communication. January 2015.

[19] Ryan Williams, Carla P. Gomes, and Bart Selman. Backdoors to typical case complexity. In *Proceedings of IJCAI*, pages 1173–1178, 2003.