Sur une classe polynomiale hybride pour les CSP d'arité quelconque *

Achref El Mouelhi **Philippe Jégou** **Cyril Terrioux**

LSIS - UMR CNRS 7296
Aix-Marseille Université
Avenue Escadrille Normandie-Niemen
13397 Marseille Cedex 20 (France)

{achref.elmouelhi, philippe.jegou, cyril.terrioux}@lsis.org

Abstract

Exhiber de nouvelles classes polynomiales pour les CSP, c'est-à-dire, des classes de problèmes de satisfaction de contraintes pour lesquelles il existe des algorithmes de reconnaissance et de résolution de complexité polynomiale, constitue un axe de recherche fondamental dans l'étude des CSP. Dans ce domaine, le concept de classe hybride, qui permet de combiner à la fois des restrictions de langages et par exemple la vérification des propriétés structurelles, est une approche qui a déjà montré son intérêt. Ici, nous étudions une classe hybride pour les CSP non binaires. Pour cela, nous revenons sur la classe BTP proposée dans [5], au niveau binaire et pour laquelle les auteurs ont proposé une version concernant les CSP non binaires. Nous développons ce travail d'abord en fournissant une nouvelle définition, certes équivalente, mais cette fois-ci, énoncée dans les termes d'une propriété sémantique associée aux relations de compatibilités. Cette classe, appelée *DBTP*, est ensuite comparée à certaines classes parmi les plus connues de la littérature. En particulier, nous démontrons que DBTP, bien que s'appuyant sur BTP lui est incomparable, et qu'elle capture certaines classes bien connues dont notamment les CSP β-acycliques, lui conférant ainsi le statut de classe hybride, au sens où elle conjugue à la fois des aspects sémantiques mais aussi structurels.

1 Introduction

Une instance de CSP $P = (X, D, C)$ est définie par la donnée d'un ensemble X de n variables (notées $x_1, ..., x_n$), d'un ensemble de domaines $D = \{d_1, ..., d_n\}$ (d_i est l'ensemble des valeurs possibles pour la variable x_i) et un ensemble C de e contraintes (notées $c_1, ..., c_e$). Chaque contrainte c_i porte sur un ensemble $S(c_i)$ de r_i variables (appelé *portée* de c_i) et autorise un ensemble de tuples défini sur $\prod_{x_j \in S(c_i)} d_j$ exprimé par une relation de compatibilité $R(c_i)$. r_i note l'*arité* de la contrainte c_i. Nous noterons r l'arité maximum et $\rho = max\{|R(c_i)|\}$. En général, on distingue les contraintes binaires dont l'arité vaut 2 des contraintes d'arité quelconque (parfois dite n-aires). Aussi, les CSP binaires (CSP dont toutes les contraintes sont binaires) sont généralement considérés de façon différente des CSP dont les contraintes sont d'arité quelconque. Pour les CSP binaires, nous noterons c_{ij} la contrainte portant sur x_i et x_j. Que ce soit pour les CSP binaires ou les CSP d'arité quelconque, le problème d'existence d'une solution (i.e. une affectation de valeur à chaque variable qui satisfait toutes les contraintes) est NP-Complet.

Bien que ce problème soit NP-complet, il n'en demeure pas moins qu'il existe des classes d'instances qui peuvent être à la fois reconnues et résolues en temps polynomial. Cela leur confère le statut de "classes polynomiales". Elles se définissent sur la base de propriétés vérifiées par les instances. On peut recenser deux grands types de propriétés. Les premières concernent des traits structurels du réseau de contraintes. Par exemple, il est bien connu qu'un CSP binaire arborescent peut être résolu en temps linéaire [8]. Un autre type de propriétés s'énonce en termes de restrictions sur le langage définissant les contraintes. Ces

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

restrictions concernent les domaines et/ou les relations de compatibilité associées aux contraintes. C'est par exemple le cas des contraintes de la classe "0-1-Tous" ("ZUT") [4]. Plus récemment, un autre type de classes polynomiales a été mis en évidence, comme par exemple la classe BTP [5]. Leur intérêt porte notamment sur le fait qu'elle peuvent prendre en compte à la fois des restrictions relevant du langage qui les exprime ainsi que des aspects structurels. Ces classes sont souvent identifiées sous le vocable de "classes hybrides".

Dans cette contribution, nous étudions une classe hybride appelée DBTP pour "*Dual BTP*." Elle s'appuie sur le concept de "Triangle Cassé" qui est à la base de BTP, et constitue cependant une classe polynomiale très différente. Alors que BTP n'est définie qu'au niveau des contraintes binaires, DBTP est définie pour les CSP dont les contraintes sont d'arité quelconque.

DBTP peut s'exprimer via l'expression d'une propriété relative à la compatibilité entre triplets de tuples figurant dans des triplets de relations de compatibilité. Cependant, cette propriété peut également être considérée comme l'expression de BTP appliquée sur la représentation duale d'un CSP. C'est d'ailleurs ainsi qu'elle a pour la première fois été évoquée dans [5]. Néanmoins, nous verrons que DBTP ne constitue pas pour autant une généralisation de BTP aux contraintes d'arité quelconque puisque pour le cas particulier des CSP binaires, BTP et DBTP sont formellement différentes (cf. théorème 12). Nous verrons également que cette classe polynomiale couvre simultanément des classes structurelles comme les CSP β-*acycliques* ainsi que des classes définies par des restrictions de langages. Nous montrerons également que DBTP est incomparable avec plusieurs autres classes bien connues de la littérature (par exemple *ZUT*, *row-convex* ou *max-closed* [4, 18, 11]).

En plus de ces résultats théoriques, nous avons prouvé que DBTP constitue une propriété conservatoire pour les filtrages classiques comme notamment la cohérence d'arc. Il semblerait ainsi que DBTP recèle un réel intérêt pratique puisque les instances DBTP peuvent être résolues en temps polynomial par l'usage d'algorithmes similaires à MAC [17] notamment.

Dans la partie 2, nous introduisons la classe DBTP dont nous fournissons les principales caractéristiques. Puis, dans la partie 3, nous étudions les relations qui peuvent exister entre BTP et DBTP pour le cas notamment des CSP binaires, et nous montrons que DBTP inclut la classe des CSP β-acycliques. La partie 4 examine les liens entre DBTP et d'autres classes polynomiales. Enfin, du fait des relations entre BTP et l'hyper-3-cohérence, nous étudions les liens entre (D)BTP et l'(hyper-)k-cohérence avant de conclure et de proposer différentes perspectives.

2 DBTP : définition et propriétés

Nous présentons d'abord la propriété DBTP d'un point de vue relationnel :

Définition 1 (Dual Broken-Triangle Property) *Un CSP $P = (X, D, C)$ vérifie la **Dual Broken Triangle Property** (DBTP) par rapport à un ordre $\prec$ sur les contraintes si pour tout triplet de contraintes (c_i, c_j, c_k) tel que $c_i \prec c_j \prec c_k$, pour tout $t_i \in R(c_i)$, $t_j \in R(c_j)$ et $t_k, t'_k \in R(c_k)$ tels que*
- $t_i[S(c_i) \cap S(c_j)] = t_j[S(c_i) \cap S(c_j)]$
- $t_i[S(c_i) \cap S(c_k)] = t_k[S(c_i) \cap S(c_k)]$
- $t'_k[S(c_j) \cap S(c_k)] = t_j[S(c_j) \cap S(c_k)]$

alors
- *soit* $t'_k[S(c_i) \cap S(c_k)] = t_i[S(c_i) \cap S(c_k)]$
- *soit* $t_j[S(c_j) \cap S(c_k)] = t_k[S(c_j) \cap S(c_k)]$

où $t[Y]$ note la restriction du tuple t aux variables du sous-ensemble $Y \subseteq X$.

Nous notons DBTP l'ensemble de ces instances.

Une caractérisation alternative s'appuie sur la propriété BTP [5] et la notion d'instance duale [6]. Le *dual* du CSP $P = (X, D, C)$ est le CSP binaire $P^d = (X^d, D^d, C^d)$ où chaque contrainte c_i de C correspond à la variable x_i^d de X^d dont le domaine d_i^d est défini par l'ensemble des tuples t_i de $R(c_i)$, et une contrainte c_{ij}^d de C^d relie deux variables x_i^d et x_j^d de X^d si les contraintes associées c_i et c_j de C partagent au moins une variable. La relation $R(c^d)$ est alors définie par les tuples $(t_i, t_j) \in d_i^d \times d_j^d$ tels que $t_i[S(c_i) \cap S(c_j)] = t_j[S(c_i) \cap S(c_j)]$. Il est bien connu que tout CSP P possède une solution ssi P^d possède une solution.

Nous rappelons maintenant la propriété BTP :

Définition 2 (Broken Triangle Property [5]) *Une instance de CSP (X, D, C) vérifie 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 $x_i < x_j < x_k$, telles que $(v_i, v_j) \in R(c_{ij})$, $(v_i, v_k) \in R(c_{ik})$ 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 tuples n'existe, (v_i, v_j), (v_i, v_k) et (v_j, v'_k) est appelé **Triangle Cassé sur** x_k.*
Nous notons BTP l'ensemble de ces instances.

Le théorème suivant met en évidence les liens entre DBTP et BTP sur l'instance duale :

Théorème 1 *Un CSP $P = (X, D, C)$ vérifie DBTP par rapport à un ordre $\prec$ sur les contraintes ssi le dual de P vérifie BTP par rapport à l'ordre $\prec$.*

Preuve : P vérifie DBTP par rapport à l'ordre $\prec$

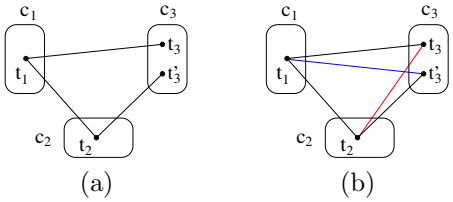


FIGURE 1 – Illustration de la propriété DBTP sur trois contraintes c_1, c_2 et c_3.

$\Leftrightarrow$ pour tout triplet de contraintes (c_i, c_j, c_k) tel que $c_i \prec c_j \prec c_k$, pour tout $t_i \in R(c_i)$, $t_j \in R(c_j)$ et $t_k, t'_k \in R(c_k)$ tels que $t_i[S(c_i) \cap S(c_j)] = t_j[S(c_i) \cap S(c_j)]$, $t_i[S(c_i) \cap S(c_k)] = t_k[S(c_i) \cap S(c_k)]$ et $t'_k[S(c_j) \cap S(c_k)] = t_j[S(c_j) \cap S(c_k)]$ alors soit $t'_k[S(c_i) \cap S(c_k)] = t_i[S(c_i) \cap S(c_k)]$, soit $t_j[S(c_j) \cap S(c_k)] = t_k[S(c_j) \cap S(c_k)]$

$\Leftrightarrow$ pour tout triplet de variables (x_i^d, x_j^d, x_k^d) tel que $x_i^d \prec x_j^d \prec x_k^d$, pour tout $t_i \in d_i^d$, $t_j \in d_j^d$ et $t_k, t'_k \in d_k^d$ tels que $(t_i, t_j) \in R(c_{ij}^d)$, $(t_i, t_k) \in R(c_{ik}^d)$ et $(t_j, t'_k) \in R(c_{jk}^d)$ alors soit $(t_i, t'_k) \in R(c_{ik}^d)$, soit $(t_j, t_k) \in R(c_{jk}^d)$

$\Leftrightarrow$ P^d vérifie BTP par rapport à l'ordre $\prec$. □

C'est d'ailleurs en termes d'expression duale que DBTP a d'abord été exprimée dans [5] puisque dans cet article, les auteurs ont évoqué le fait qu'un CSP dual, et donc binaire, pouvait vérifier BTP.

Sur la base de cette propriété, nous pouvons observer graphiquement la propriété DBTP sur la microstructure de l'instance duale. La *microstructure* [13] d'un CSP binaire $P = (X, D, C)$ est le graphe non-orienté $\mu(P) = (V, E)$ où $V = \{(x_i, v_i) : x_i \in X, v_i \in d_i\}$ et $E = \{ \{(x_i, v_i), (x_j, v_j)\} : i \neq j, c_{ij} \notin C$ ou $(v_i, v_j) \in R(c_{ij})\}$. La figure 1 présente la microstructure de l'instance duale d'un instance P concernant trois contraintes. Dans la figure 1(a), nous pouvons observer la présence d'un triangle cassé sur c_3 si nous considérons l'ordre $c_1 \prec c_2 \prec c_3$ et ainsi, P ne vérifie pas DBTP par rapport à cet ordre. Au contraire, dans la figure 1(b), si soit t_1 et t'_3 (arête bleue), soit t_2 et t_3 (arête rouge) sont compatibles, alors P vérifie DBTP relativement à l'ordre $\prec$.

Nous pouvons noter que DBTP est très différente de BTP. En particulier, une instance binaire peut vérifier DBTP tout en ne vérifiant pas BTP. Par exemple, l'instance binaire décrite dans la figure 2 est DBTP par rapport à l'ordre $c_{ij} \prec c_{jk} \prec c_{ik}$ mais elle n'est pas BTP. Il n'est pas surprenant que DBTP n'implique pas BTP car, même si l'instance originale et son expression duale représentent le même problème, leurs structures et microstructures sont très différentes. Les liens entre DBTP et BTP seront étudiés de façon plus détaillée dans la partie 3 qui leur est dédiée.

Nous montrons maintenant que la classe des CSPs vérifiant DBTP constitue une classe polynomiale. Pour

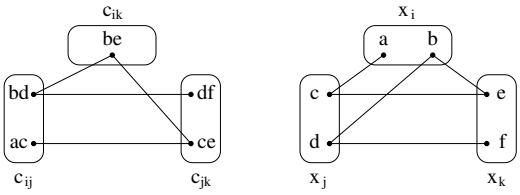


FIGURE 2 – Une instance vérifiant DBTP (a) mais pas BTP (b).

cela, et du fait du théorème 1, les preuves s'appuient sur les mêmes schémas que ceux utilisés dans [5].

Lemme 1 *Tout CSP $P = (X, D, C)$ qui vérifie DBTP par rapport à un ordre $\prec$ sur les contraintes peut être résolu en $O(e^2.r.\rho^2)$.*

Preuve : La première étape consiste à construire le dual de P, ce qui peut être réalisé en $O(e^2.r.\rho^2)$. Ensuite, comme le dual de P est BTP, nous savons qu'il peut être résolu en $O(e^2.\rho^2)$ [5]. Ainsi, la complexité globale est en $O(e^2.r.\rho^2)$. □

Le lemme 2 exprime le fait qu'un ordre $\prec$ sur les contraintes et associé à DBTP peut être calculé (le cas échéant) en temps polynomial.

Lemme 2 *Étant donné un CSP $P = (X, D, C)$, déterminer si un ordre $\prec$ sur les contraintes tel que P est DBTP par rapport à $\prec$ existe (et le trouver le cas échéant) peut être réalisé en temps polynomial.*

Preuve : Un algorithme possible consiste à calculer d'abord le dual de P, puis à déterminer si un ordre $\prec$ tel que le dual de P est BTP existe comme proposé dans [5]. Chaque étape est polynomiale (voir la preuve précédente et [5]). Par conséquent, la complexité globale est polynomiale. □

Du fait de ces deux lemmes, nous pouvons déduire le théorème suivant :

Théorème 2 *DBTP est une classe polynomiale.*

Nous étudions maintenant ce qu'il en est de la propriété DBTP dans le cas de l'application d'un algorithme de filtrage sur une instance DBTP. Une classe $\mathcal{C}$ d'instances de CSP est dite **conservatoire** par rapport à un filtrage de cohérence ϕ si elle est fermée pour ϕ, c'est-à-dire, si le problème obtenu après l'application de ϕ à toute instance de $\mathcal{C}$ appartient à la classe $\mathcal{C}$. Une propriété est dite conservatoire si elle définit une classe d'instances conservatoires.

Propriété 1 *DBTP est conservatoire pour tout filtrage qui se limite à supprimer des valeurs dans les domaines ou des tuples dans les relations.*

Preuve : Considérons un CSP P vérifiant DBTP par rapport à un ordre donné sur les contraintes. La suppression d'une valeur du domaine d'une variable x de P conduit à une suppression de tuples pour les contraintes dont la portée contient x. En d'autres termes, cela revient à supprimer des valeurs dans les domaines de variables du dual de P. Par conséquent, dans les deux cas, les suppressions de valeurs ou de tuples dans l'instance d'origine conduisent à supprimer les valeurs des variables duales. Comme BTP est conservatoire pour les cohérences filtrant les domaines, le dual de P, après ces suppressions, vérifie encore BTP. Par conséquent, P demeure DBTP. □

Cette propriété est donc valable pour tout filtrage de domaine (par exemple pour la cohérence d'arc généralisée ou la cohérence inverse de chemin [2]), qu'il soit appliqué sur l'instance originale ou son instance duale. C'est également le cas pour l'intercohérence ou pairwise-consistency (introduite dans le cadre de la Théorie des Bases de Données Relationelles [1]) dont l'application est équivalente à celle de l'application de la cohérence d'arc sur l'instance duale.

Définition 3 (Intercohérence [10]) *Un CSP $P = (X, D, C)$ est **intercohérent ou pairwise-consistant** ssi $\forall\ 1 \leq i \leq e$, $R(c_i) \neq \emptyset$ et $\forall\ 1 \leq i < j \leq e$, $R(c_i)[S(c_i) \cap S(c_j)] = R(c_j)[S(c_i) \cap S(c_j)]$.*

Comme MAC [17] maintient la cohérence d'arc (notée AC) à chaque étape de la recherche, nous pouvons définir MPWC comme l'algorithme correspondant au maintien de l'intercohérence.

Théorème 3 *Si $P = (X, D, C)$ vérifie DBTP, alors MPWC résout P en temps polynomial pour tout ordre.*

Preuve : Comme l'intercohérence sur P est équivalente à la cohérence d'arc sur le dual de P [10], l'application de MPWC sur P est équivalente à celle de MAC sur le dual de P. De plus, comme P est DBTP, P^d est BTP et ainsi, selon le théorème 7.6 de [5], MPWC résout P en temps polynomial. □

Ce résultat est également vérifié pour MAC mais dans le cas particulier où tout couple de contraintes partage au plus une variable. Avant de le montrer, il nous faut rappeler deux résultats sur la cohérence d'arc et l'intercohérence.

Lemme 3 (Propriété 8.1 page 146 dans [12]) *Soit $P = (X, D, C)$ un CSP tel que $\forall c_i, c_j \in C, |S(c_i) \cap S(c_j)| \leq 1$. Si l'instance P vérifie la cohérence d'arc, alors elle vérifie l'intercohérence.*

Lemme 4 *Soit $P = (X, D, C)$ un CSP vérifiant la cohérence d'arc et tel que $\forall c_i, c_j \in C, |S(c_i) \cap S(c_j)| \leq 1$. Si l'instance P' obtenue à partir de P en supprimant certaines valeurs et en appliquant AC ne possède pas de domaine vide, alors son dual vérifie la cohérence d'arc.*

Preuve : Considérons P et P' obtenu à partir de P en supprimant certaines valeurs et en appliquant AC, tel qu'il ne possède pas de domaine vide. Puisque P' vérifie la cohérence d'arc, d'après le lemme 3, il vérifie aussi l'intercohérence. Ainsi, comme l'intercohérence sur P est équivalente à la cohérence d'arc sur le dual de P [10], le dual de P' vérifie la cohérence d'arc. □

Théorème 4 *Si $P = (X, D, C)$ est tel que $\forall c_i, c_j \in C, |S(c_i) \cap S(c_j)| \leq 1$, et qu'il vérifie la cohérence d'arc ainsi que DBTP, alors MAC peut résoudre P en temps polynomial.*

Preuve : Si après l'obtention de la cohérence d'arc, aucun domaine, ni relation n'est vide, alors P vérifie l'intercohérence et possède une solution. D'après le lemme 4, le problème obtenu après la suppression de valeurs et le filtrage par cohérence d'arc demeure intercohérent. Par conséquent, lors de l'application de MAC sur le problème initial, nous maintenons également l'intercohérence. De plus, comme l'intercohérence est équivalente à la cohérence d'arc sur le problème dual [10], le théorème 7.6 de [5] fait que MAC résout P en temps polynomial puisque le dual est BTP. □

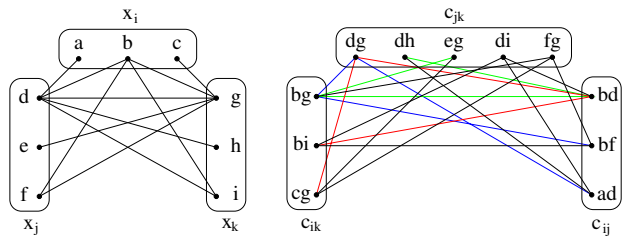
Ce théorème est bien entendu vérifié pour tout CSP binaire.

3 Relations entre DBTP et BTP

La classe $DBTP$ diffère nécessairement de la classe BTP puisque $DBTP$ peut contenir des instances non-binaires alors que BTP n'a été définie qu'au niveau binaire, ce qui conduit à ne se poser la question de leur comparaison que dans ce cadre. Comme le montre la figure 2, une instance peut vérifier DBTP mais pas BTP. Inversément, une instance peut vérifier BTP sans qu'elle ne vérifie DBTP. Ce cas est illustré dans la figure 3 (où les triangles cassés coloriés prouvent que DBTP n'est pas vérifiée). Ces résultats étaient prévisibles, puisque, même si l'instance d'origine et son dual représentent le même problème, leur structure et leur microstructure sont en fait différentes. Ainsi, à partir des exemples des figures 2 et 3, nous obtenons :

Théorème 5 *Soit $P = (X, D, C)$ un CSP binaire.*
- *P vérifie DBTP $\not\Rightarrow$ P vérifie BTP,*
- *P vérifie BTP $\not\Rightarrow$ P vérifie DBTP.*

Les résultats précédents reposent sur la présence de triangles cassés dans la microstructure de l'instance ou



(a) (b)

FIGURE 3 – Une instance vérifiant BTP (a) mais pas DBTP (b).

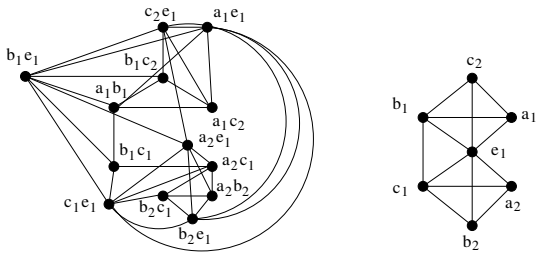
de son instance duale. Dans chaque cas, ces triangles cassés concernent des valeurs qui pourraient être supprimées par un certain filtrage comme la cohérence d'arc notamment. Ainsi, comme DBTP et BTP sont conservatoires par rapport aux filtrages de domaines, nous focalisons notre étude sur les instances binaires qui satisfont la cohérence d'arc et ainsi que l'intercohérence (du fait du lemme 4). Sous ces hypothèses, nous déduisons le lemme suivant :

Lemme 5 *Étant donné un CSP binaire* $P = (X, D, C)$ *vérifiant la cohérence d'arc, si pour un triplet* (x_i, x_j, x_k) *de variables, nous disposons d'un triangle cassé, alors il existe deux triangles cassés pour le triplet* (c_{ij}, c_{ik}, c_{jk}) *dans l'instance duale.*

Preuve : Soient x_i, x_j, $x_k \in X$ telles que $(v_i, v_j) \in R(c_{ij})$, $(v_i, v_k) \in R(c_{ik})$, $(v_j, v'_k) \in R(c_{jk})$, $(v_i, v'_k) \notin R(c_{ik})$ et $(v_j, v_k) \notin R(c_{jk})$. Comme P est intercohérent, il existe des valeurs $v'_i \in d_i$ et $v'_j \in d_j$ telles que $v_i \neq v'_i$, $v_j \neq v'_j$, $(v'_i, v'_k) \in R(c_{ik})$ et $(v'_j, v_k) \in R(c_{jk})$. Aussi, $((v_i, v_j), (v_i, v_k))$, $((v_i, v_j), (v_j, v'_k))$ et $((v_i, v_k), (v'_j, v_k))$ forment un triangle cassé sur c_{jk} pour le triplet (c_{ij}, c_{ik}, c_{jk}). Il en est de même pour $((v_i, v_j), (v_j, v'_k))$, $((v_i, v_j), (v_i, v_k))$ et $((v_j, v'_k), (v'_i, v'_k))$ sur c_{ik}. □

Par conséquent, quand un triangle cassé pour un triplet (x_i, x_j, x_k) impose la condition $x_k < max(x_i, x_j)$ sur l'ordre $<$ des variables, cela revient à imposer les deux conditions $c_{jk} \prec max(c_{ij}, c_{ik})$ et $c_{ik} \prec max(c_{ij}, c_{jk})$ pour le triplet (c_{ij}, c_{ik}, c_{jk}) sur l'ordre $\prec$ des contraintes. Il s'ensuit que toute instance binaire arc-cohérente et intercohérente qui satisfait BTP et dispose de deux triangles cassés pour deux variables différentes d'une même triplet de variables ne peut satisfaire DBTP puisque nous obtiendrions tous les triangles cassés possibles pour le triplet correspondant de contraintes.

Inversément, considérons une instance binaire avec neuf variables $\{x_a, x_b, \ldots, x_i\}$. Nous définissons cet exemple en reproduisant plusieurs fois un même motif qui est tel que chaque valeur apparaissant dans une occurrence de ce motif n'apparaît dans aucune autre occurrence.



(a) (b)

FIGURE 4 – Morceau d'une instance non BTP mais vérifiant DBTP, la cohérence d'arc et l'intercohérence.

Ce motif consiste en un triangle cassé sur une variable z pour un triplet (x, y, z) (c'est-à-dire qui impose la condition $z < max(x, y)$ sur $<$) et chaque valeur des variables x, y et z est liée à une valeur donnée d'une variable qui n'est pas impliquée dans ce triplet. Nous reproduisons ce schéma 9 fois de telle sorte que les conditions suivantes soient imposées : $x_a < max(x_b, x_c)$, $x_b < max(x_e, x_h)$, $x_c < max(x_e, x_g)$, $x_d < max(x_a, x_g)$, $x_e < max(x_a, x_i)$, $x_f < max(x_d, x_e)$, $x_g < max(x_h, x_i)$, $x_h < max(x_b, x_d)$ et $x_i < max(x_c, x_f)$. La figure 4(b) décrit ce motif pour le triplet (x_a, x_b, x_c), un triangle cassé sur x_a (correspondant à la condition $x_a < max(x_b, x_c)$) et une variable indépendante x_e tandis que la figure 4 (a) décrit la partie correspondante dans l'instance duale. En faisant cela, la microstructure de notre CSP binaire et celle de son instance duale ont 9 composantes connexes. On peut noter que cette instance n'est pas BTP parce que les 9 conditions rendent impossible la construction d'un ordre approprié sur les variables. En revanche, il est DBTP (par rapport à l'ordre $c_{ab} \prec c_{ac} \prec c_{ad} \prec c_{bf} \prec c_{bh} \prec c_{ci} \prec c_{df} \prec c_{dh} \prec c_{ef} \prec c_{ei} \prec c_{gh} \prec c_{gi} \prec c_{af} \prec c_{bc} \prec c_{bd} \prec c_{ce} \prec c_{cg} \prec c_{de} \prec c_{dg} \prec c_{fi} \prec c_{hi} \prec c_{ae} \prec c_{ag} \prec c_{be} \prec c_{bg} \prec c_{cd} \prec c_{cf} \prec c_{di} \prec c_{fh} \prec c_{ai} \prec c_{bi} \prec c_{eg} \prec c_{eh} \prec c_{fg} \prec c_{ah} \prec c_{ch}$), et il vérifie la cohérence d'arc et l'intercohérence.

Néanmoins, pour des niveaux plus élevés de cohérence, il n'en est pas nécessairement de même. Nous pouvons notamment démontrer qu'une instance binaire chemin-cohérente et DBTP est BTP :

Théorème 6 *Si un CSP binaire* P *vérifie DBTP par rapport à un ordre* $\prec$ *sur les contraintes et s'il vérifie la chemin-cohérence, alors* P *vérifie BTP par rapport à tout ordre* $<$ *sur les variables.*

Preuve : Supposons que P ne vérifie pas BTP. Alors, pour tout ordre $<$ sur les variables, il existe un triplet $x_i < x_j < x_k$ tel que $\exists v_i \in d_i, v_j \in d_j$ et $v_k, v'_k \in d_k$, $(v_i, v_j) \in R(c_{ij}), (v_i, v_k) \in R(c_{ik})$ et $(v_j, v'_k) \in R(c_{jk})$, $(v_j, v_k) \notin R(c_{jk})$ et $(v_i, v'_k) \notin R(c_{ik})$. Comme P vérifie la cohérence de chemin, $\exists v'_i \in d_i, v'_j \in d_j$ et $v''_k \in d_k, (v_i, v'_j) \in R(c_{ij}), (v'_i, v_j) \in R(c_{ij})$, $(v'_i, v'_k) \in R(c_{ik})$, $(v_i, v''_k) \in R(c_{ik})$, $(v'_j, v_k) \in R(c_{jk})$ et

$(v_j, v''_k) \in R(c_{jk})$. Par conséquent, il est facile de voir qu'il n'y a aucun ordre sur les contraintes tel que P satisfasse DBTP, et donc P ne satisfait pas DBTP. Ainsi, nous obtenons une contradiction et P satisfait BTP. □

Nous étudions maintenant le cas des CSP acycliques pour lesquels [5] a déjà prouvé que ces CSP binaires vérifient BTP. Nous allons montrer que cela est également vrai pour DBTP. Soit $TREE$ l'ensemble des CSP binaires dont le graphe de contraintes est acyclique.

Théorème 7 *$TREE \subsetneq DBTP$.*

Preuve : Soit $DUAL\text{-}TREE$ l'ensemble des CSP binaires dont chaque élément est le dual d'une instances de $TREE$. Comme cela a été montré dans [5], $DUAL\text{-}TREE \subsetneq BTP$. Par conséquent, $TREE \subsetneq DBTP$. □

Ce résultat peut être étendu aux CSP d'arité quelconque. Pour cela, nous devons considérer la notion de cyclicité dans les hypergraphes, pour lesquels différents degrés ont été définis [1]. Ici, nous nous intéressons en particulier à l'α-acyclicité et à la β-acyclicité. Nous allons montrer que les CSP β-acycliques vérifient DBTP, alors que ce n'est pas le cas pour les CSP α-acycliques. Nous rappelons d'abord la définition de la β-acyclicité d'un hypergraphe (de contraintes).

Définition 4 ([9]) *Une séquence $(c_1, ..., c_m, c_{m+1})$ avec $m \geq 3$ telle que $(c_1, ..., c_m)$ sont distincts et $c_1 = c_{m+1}$ est un cycle de Graham si chaque $\Delta_i = S(c_i) \cap S(c_{i+1})$ $(1 \leq i \leq m)$ n'est pas vide, et chaque fois que $i \neq j$, Δ_i et Δ_j sont incomparables (i.e. $\Delta_i \not\subseteq \Delta_j$ et $\Delta_j \not\subseteq \Delta_i$). $H = (X, C)$ est un hypergraphe β-**acyclique** ssi il ne possède pas de cycle de Graham.*

Il a récemment été montré dans [7] que les hypergraphes β-acycliques peuvent être définis en appliquant les deux règles suivantes, qui doivent mener à l'hypergraphe vide :

(1) Si une hyperarête est vide, elle est retirée de C.

(2) Si un sommet est un point de type "nest" (i.e. l'ensemble des hyperarêtes le contenant est une chaîne pour la relation d'inclusion), alors il est retiré de H (i.e. il est retiré de X ainsi que des hyperarêtes qui le contiennent).

Théorème 8 ([7]) *Un hypergraphe H est β-acyclique ssi, après l'application successive des deux règles jusqu'à ce qu'aucune ne puisse l'être, nous obtenons l'hypergraphe vide.*

En utilisant ces définitions, nous pouvons maintenant établir le théorème suivant :

Théorème 9 *Étant donné un CSP (X, D, C), il existe un ordre sur les contraintes $\prec$, tel que $\forall c_i, c_j, c_k \in C$ tel que $c_i \prec c_j \prec c_k$, nous avons $S(c_i) \cap S(c_k) \subseteq S(c_j) \cap S(c_k)$ ou $S(c_j) \cap S(c_k) \subseteq S(c_i) \cap S(c_k)$ ssi (X, D, C) possède un hypergraphe de contraintes β-acyclique.*

Preuve : ($\Rightarrow$) Par contraposition. Nous montrons que si l'hypergraphe de contraintes (X, C) est β-cyclique, alors, il ne peut exister d'ordre sur les contraintes.

Considérons un hypergraphe de contraintes (X, C) qui est β-cyclique. Il possède donc un cycle de Graham, que l'on notera par la séquence d'hyperarêtes $(c_1, ..., c_m, c_{m+1})$. Considérons un ordre quelconque sur les contraintes $\prec$. Nécessairement, parmi les contraintes de ce cycle, il existe une contrainte maximum c_k par rapport à l'ordre $\prec$. Considérons ses deux voisines dans le cycle, notées c_i et c_j (avec $c_i, c_j \prec c_k$). Par définition des cycles de Graham, nous savons que $S(c_i) \cap S(c_k)$ et $S(c_j) \cap S(c_k)$ sont incomparables. Aussi, nous n'avons ni $S(c_i) \cap S(c_k) \subseteq S(c_j) \cap S(c_k)$ ni $S(c_j) \cap S(c_k) \subseteq S(c_i) \cap S(c_k)$, et donc aucun ordre correct sur les contraintes $\prec$ ne peut exister.

($\Leftarrow$) Nous utilisons ici le théorème 8. Étant donné un CSP β-acyclique (X, D, C) d'hypergraphe H qui admet un ordre $\prec$ correct sur les contraintes, nous allons montrer que :

(1) Une hyperarête $S(c_i)$ est vide ssi H sans $S(c_i)$ admet un ordre et est β-acyclique.

(2) Un sommet x de H est un point de type "nest" tel que H admet un ordre ssi H sans x admet un ordre et est β-acyclique.

On voit immédiatement que la propriété est vérifiée par l'application de la règle (1). Considérons donc la règle (2). Supposons que pour un hypergraphe H nous disposons d'un ordre $\prec$. Aussi, $\forall c_i, c_j, c_k \in C$ tel que $c_i \prec c_j \prec c_k$, nous avons $S(c_i) \cap S(c_k) \subseteq S(c_j) \cap S(c_k)$ ou $S(c_j) \cap S(c_k) \subseteq S(c_i) \cap S(c_k)$. Nous avons 5 cas à considérer :

1. $x \notin S(c_i) \cup S(c_j) \cup S(c_k)$: après la suppression de x, ni $S(c_i) \cap S(c_k)$, ni $S(c_j) \cap S(c_k)$ n'ont changées. La propriété est donc vérifiée.
2. $x \in S(c_i) \cap S(c_j) \cap S(c_k)$: après la suppression de x, ce sommet disparaît de chaque intersection $S(c_i) \cap S(c_k)$ et $S(c_j) \cap S(c_k)$, et ainsi, la propriété est vérifiée.
3. x n'appartient qu'à un seul des ensembles $S(c_i)$ ou $S(c_j)$ ou $S(c_k)$: donc x n'appartient à aucune intersection, et donc la propriété est vérifié après la suppression.
4. $x \in S(c_i) \cap S(c_j)$ et $x \notin S(c_k)$: donc x n'appartient ni à $S(c_i) \cap S(c_k)$, ni à $S(c_j) \cap S(c_k)$, et donc la propriété est vérifiée après la suppression.

5. $x \in S(c_i) \cap S(c_k)$ et $x \notin S(c_j)$ (ou symétriquement $x \in S(c_j) \cap S(c_k)$ et $x \notin S(c_i)$) : donc, avant la suppression, nous avions nécessairement $S(c_i) \cap S(c_k) \not\subseteq S(c_j) \cap S(c_k)$ et $S(c_j) \cap S(c_k) \subsetneq S(c_i) \cap S(c_k)$. Ainsi, après la suppression de x, nous avons au moins $S(c_j) \cap S(c_k) \subseteq S(c_i) \cap S(c_k)$.

Donc, nous avons montré que si nous pouvons réduire tout hypergraphe, ce qui ne contredit pas la propriété sur l'ordre, et donc, nécessairement, l'hypergraphe d'origine est β-acyclique et admet un ordre approprié sur les contraintes. □

Nous pouvons noter que ce théorème explique pourquoi la condition énoncée dans le lemme 4.6 de [5] est vérifiée indépendamment de la portée des contraintes. Ce lemme et le théorème précédent nous permettent d'obtenir le théorème 10 où β-$ACYCLIC$ est l'ensemble des CSP pour lesquels l'(hyper)graphe de contraintes est β-acyclique.

Théorème 10 $TREE \subsetneq \beta\text{-}ACYCLIC \subsetneq DBTP$.

Toutefois, l'équivalence proposée dans le lemme 4.6 [5] est seulement vérifiée dans le sens inverse (ce qui ne compromet pas la preuve du théorème 10). Pour s'en rendre compte, il suffit de considérer une instance binaire avec 3 variables monovalentes (une seule valeur par domaine) mutuellement connectées (chaque contrainte admet l'unique tuple possible). Son dual satisfait BTP bien que le graphe de contraintes ne soit pas β-acyclique. Cet exemple de CSP peut bien entendu être généralisé à des tailles de domaines et à un nombre de variables quelconques en reproduisant un motif identique.

Nous montrons maintenant que si α-$ACYCLIC$ est l'ensemble des CSP dont l'(hyper)graphe de contraintes est α-acyclique, alors les ensembles α-$ACYCLIC$ et $DBTP$ sont incomparables. Pour cela, nous rappelons que l'α-acyclicité d'un (hyper)graphe de contraintes peut être définie en utilisant la "running intersection property" [1], à savoir :

Définition 5 (X, C) *est un hypergraphe* ***α-acyclique*** *ssi il existe un ordre* $(c_1, ..., c_e)$ *tel que*

$$\forall k, 1 < k \leq e, \exists j < k, (S(c_k) \cap \bigcup_{i=1}^{k-1} S(c_i)) \subseteq S(c_j).$$

Considérons un CSP possédant six variables $x_a, \ldots x_f$ et quatre contraintes dont les portées sont respectivement $\{x_a, x_b, x_c\}$, $\{x_a, x_b, x_d\}$, $\{x_a, x_c, x_e\}$ et $\{x_b, x_c, x_f\}$. La figure 5 présente son hypergraphe de contraintes (a) et la microstructure de son dual (b). Nous pouvons constater que cette instance est α-acyclique mais qu'elle ne vérifie pas DBTP puisqu'aucun ordre approprié sur les contraintes ne peut exister. De plus, il est bien connu que β-$ACYCLIC \subsetneq \alpha$-$ACYCLIC$ [1]. Par conséquent, si l'on note par $A \perp B$

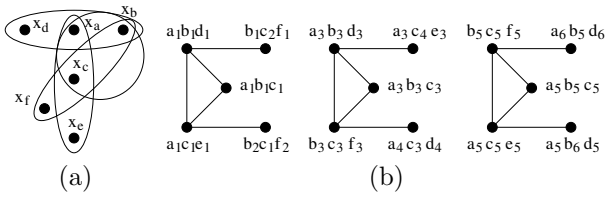


FIGURE 5 – Un CSP α-acyclique (a) qui ne vérifie pas DBTP (b).

le fait que deux classes polynomiales sont incomparables (i.e. nous n'avons ni $A \subseteq B$ ni $B \subseteq A$), nous obtenons le théorème suivant :

Théorème 11 $\alpha\text{-}ACYCLIC \cap DBTP \neq \emptyset$ *et* α-$ACYCLIC \perp DBTP$.

Dans cette partie, nous avons donc globalement établi le résultat suivant :

Théorème 12 $BTP \cap DBTP \neq \emptyset$ *et* $BTP \perp DBTP$.

Dans la partie suivante, nous étudions le lien entre DBTP et quelques autres classes polynomiales.

4 DBTP vs Quelques classes polynomiales

4.1 Cas des CSP binaires

Comme DBTP et BTP sont deux classes différentes, nous nous concentrons d'abord sur certaines classes traitables incluses dans BTP. Ces classes dont les définitions sont rappelées ci-dessous s'appuient sur des restrictions de langages de contraintes.

Définition 6 (Row-convex [18]) *Un CSP binaire* $P = (X, D, C)$ *est dit* ***row-convex*** *par rapport à un ordre* $<$ *sur les variables et à un ordre sur les valeurs, si, pour chaque contrainte* c_{ij} *de* C *avec* $x_i < x_j$, $\forall v_i \in d_i$, $\{v_j \in d_j | (v_i, v_j) \in R(c_{ij})\} = [a_j..b_j]$ *pour* a_j, $b_j \in d_j$ *où* $[a_j..b_j]$ *représente le valeurs de* d_j *entre* a_j *et* b_j *par rapport à l'ordre sur les valeurs.*
Nous noterons RC l'ensemble des instances row-convex.

Définition 7 (0-1-tous [4]) *Un CSP binaire CSP* $P = (X, D, C)$ *est dit* ***0-1-tous*** *si pour chaque contrainte* c_{ij} *de* C, *pour chaque valeur* $v_i \in d_i$, c_{ij} *vérifie l'une des conditions suivantes :*

- *(ZERO) pour chaque valeur* $v_j \in d_j, (v_i, v_j) \notin R(c_{ij})$,
- *(UN) il y a une valeur unique* $v_j \in d_j$ *telle que* $(v_i, v_j) \in R(c_{ij})$,
- *(TOUS) pour chaque valeur* $v_j \in d_j, (v_i, v_j) \in R(c_{ij})$.

Nous noterons ZUT l'ensemble des instances vérifiant la propriété 0-1-tous.

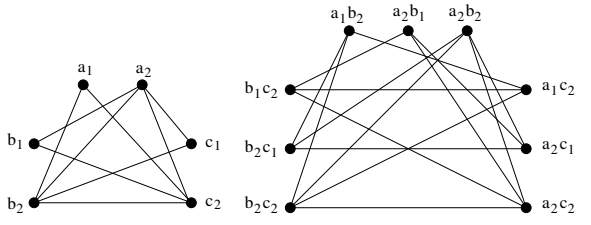


FIGURE 6 – Un CSP appartenant à RC, ZUT et RRM (a) mais qui ne vérifie pas DBTP (b).

Définition 8 (Renamable right monotone [5]) *Un CSP binaire $P = (X, D, C)$ est dit **renamable right monotone** par rapport à un ordre $<$ sur les variables, si, pour $2 \leq j \leq n$, chaque domaine d_j peut-être ordonné par $\preccurlyeq_j$ de telle sorte que pour chaque contrainte c_{ij} de C avec $x_i < x_j$, $\forall v_i \in d_i, v_j, v'_j \in d_j$, si $(v_i, v_j) \in R(c_{ij})$ et $v_j \preccurlyeq_j v'_j$ alors $(v_i, v'_j) \in R(c_{ij})$.*
Nous noterons RRM l'ensemble de ces instances.

Le théorème suivant montre que ces classes polynomiales partagent des instances avec DBTP mais sont cependant différentes.

Théorème 13 *$RC \cap DBTP \neq \emptyset$ et $RC \perp DBTP$.*
$ZUT \cap DBTP \neq \emptyset$ et $ZUT \perp DBTP$.
$RRM \cap DBTP \neq \emptyset$ et $RRM \perp DBTP$.

Preuve : Si l'on considère le CSP binaire de la figure 6(a), il est *0-1-tous*, *row convex*, et *renamable right monotone* par rapport aux ordres lexicographique sur les valeurs et les variables. Cependant, comme le montre la figure 6(b), cette instance n'est pas DBTP. Inversement, toute instance non binaire DBTP ne peut appartenir à RC, ZUT ou RRM.

Afin de prouver que $DBTP$ intersecte RC, ZUT et RRM, il suffit de considérer un CSP binaire monovalent à trois variables et trois contraintes qui est cohérent. En effet, ce type d'instance satisfait à la fois $DBTP$, RC, ZUT et RRM. □

La classe suivante repose sur le nombre de cliques maximales figurant dans la microstructure :

Définition 9 (Maximal clique bounded [15]) *Un CSP $P = (X, D, C)$ est dit **maximal clique bounded** si sa microstructure possède un nombre polynomial de cliques maximales.*
Nous noterons CL l'ensemble de ces instances.

Théorème 14 *$CL \cap DBTP \neq \emptyset$ et $CL \perp DBTP$.*

Preuve : Tout CSP binaire monovalent et cohérent possède une seule clique maximale et il est DBTP. Donc, l'intersection n'est pas vide.

Considérons maintenant un CSP binaire tel que sa microstructure recèle un nombre polynomial de cliques maximales. Nous rajoutons à ce CSP binaire des variables supplémentaires avec d'autres valeurs et contraintes supplémentaires correspondant à l'instance représentée dans la figure 1, de telle sorte que ces valeurs ne soient pas compatibles avec celles de la première partie de ce CSP. De cette façon, il possède un nombre polynomial de cliques maximales dans sa microstructure mais il n'est pas DBTP. Inversement, toute instance de $DBTP$ non binaire ne peut appartenir à CL. □

En ce qui concerne les classes basées sur des structures restreintes, nous avons prouvé dans le théorème 10 que $TREE \subsetneq DBTP$.

4.2 CSP d'arités quelconques

Nous considérons d'abord certaines classes polynomiales connues basées sur des restrictions de langages de contraintes comme la classe *max-closed*.

Définition 10 (Max-closed [11]) *Un CSP $P = (X, D, C)$ est dit **max-closed** si pour chaque contrainte c d'arité r_c, $\forall (v_1, v_2, \ldots, v_{r_c}), (v'_1, v'_2, \ldots, v'_{r_c}) \in R(c)$, $(max(v_1, v'_1), max(v_2, v'_2), \ldots, max(v_{r_c}, v'_{r_c})) \in R(c)$.*
Nous notons MC l'ensemble de ces instances.

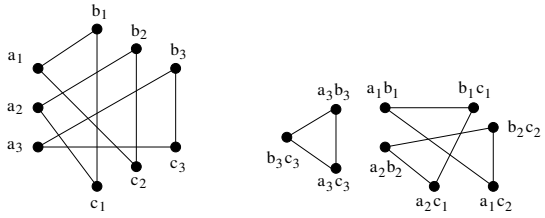
Théorème 15 *$MC \cap DBTP \neq \emptyset$ et $MC \perp DBTP$.*

Preuve : La preuve de $MC \cap DBTP \neq \emptyset$ et de $MC \not\subseteq DBTP$ est similaire à celle du théorème 13. En ce qui concerne $DBTP \not\subseteq MC$, tout CSP ayant deux variables et une contrainte binaire est DBTP mais pas nécessairement *max-closed*. □

Définition 11 (incrementally functional [3]) *Un $P = (X, D, C)$ est dit **incrementally functional** s'il existe un ordre $<$ sur les variables, tel que pour $1 \leq i < n$, chaque solution de $P[\{x_1, \ldots, x_i\}]$ peut s'étendre au plus à une solution de $P[\{x_1, \ldots, x_{i+1}\}]$ où, pour $X' \subseteq X, P[X']$ notant le CSP (X', D', C') où $D' = \{d_i | x_i \in X'\}$ et $C' = \{c' | c \in C$ tel que $S(c) \cap X' \neq \emptyset, S(c') = S(c) \cap X'$ et $R(c') = \{t[S(c')] | t \in R(c)\}\}$.*
Nous notons $IFUN$ l'ensemble de ces instances.

Théorème 16 *$IFUN \cap DBTP \neq \emptyset$ et $IFUN \perp DBTP$.*

Preuve : Afin de prouver que l'intersection n'est pas vide, nous considérons un CSP avec quatre variables monovalentes $x_1, \ldots, x_4$ et trois contraintes ternaires c_1, c_2 et c_3 telles que $S(c_1) = \{x_1, x_2, x_3\}$, $R(c_1) = \{(v_1, v_2, v_3)\}$, $S(c_2) = \{x_1, x_2, x_4\}$, $R(c_2) = \{(v_1, v_2, v_4)\}$, $S(c_3) = \{x_2, x_3, x_4\}$ et $R(c_3) = \{(v_2, v_3, v_4)\}$. Cette instance est *incrementally functional* (en utilisant la numérotation des



(a) (b)

FIGURE 7 – Une instance *incrementally functional* (a) mais qui ne vérifie pas DBTP (b).

variables comme ordre) et DBTP. L'instance présentée dans la figure 7 est *incrementally functional* mais pas DBTP. Inversement, toute instance DBTP ayant plusieurs solutions ne peut pas être *incrementally functional*. □

Définition 12 (Dual CB [15]) *$P = (X, D, C)$ est dit **dual maximal clique bounded (DMCB)** si la microstructure de son instance duale possède un nombre polynomial de cliques maximales. Nous notons DCL l'ensemble de ces instances.*

Théorème 17 *$DCL \cap DBTP \neq \emptyset$ et $DCL \perp DBTP$.*

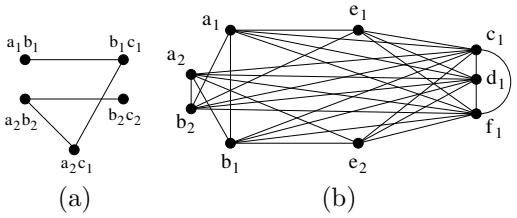
Preuve : Considérons la première instance définie dans la preuve du théorème 16. La microstructure de son instance duale possède une seule clique maximale et l'instance est DBTP. Ainsi, l'intersection n'est pas vide. En ce qui concerne l'exemple représenté dans la figure 7, elle possède un nombre polynomial de cliques maximales dans la microstructure de son instance duale mais n'est pas DBTP. Inversement, une instance binaire dont le graphe de contraintes est une étoile et pour lequel chaque domaine dispose de plusieurs valeurs est DBTP mais possède un nombre de cliques maximales dans sa microstructure duale non borné polynomialement. □

Maintenant, nous introduisons une nouvelle classe polynomiale basée sur une restriction du langage de contraintes.

Définition 13 (Triangulaire) *Un $P = (X, D, C)$ est dit **triangulaire** par rapport à un ordre sur les contraintes $\prec$ ssi $\forall c_i, c_j, c_k$, $c_i \prec c_j \prec c_k$, $\forall t_i \in R(c_i), t_j \in R(c_j), t_k \in R(c_k)$, if*
- *$t_i[S(c_i) \cap S(c_j)] = t_j[S(c_i) \cap S(c_j)]$ et*
- *$t_i[S(c_i) \cap S(c_k)] = t_k[S(c_i) \cap S(c_k)]$.*

alors, $t_j[S(c_j) \cap S(c_k)] = t_k[S(c_j) \cap S(c_k)]$.
Nous notons TR l'ensemble de ces instances.

Théorème 18 *Si un CSP P est triangulaire par rapport à un ordre $\prec$, alors P vérifie DBTP par rapport à $\prec$.*



(a) (b)

FIGURE 8 – (a) Une instance vérifiant DBTP mais qui n'est pas triangulaire. (b) Partie d'une instance vérifiant l'hyper-3-cohérence orientée mais pas BTP.

Preuve : Supposons que P soit triangulaire mais pas DBTP. Alors, il existe trois contraintes c_i, c_j et c_k, $c_i \prec c_j \prec c_k$, $t_i \in R(c_i)$, $t_j \in R(c_j)$ et $t_k, t'_k \in R(c_k)$ telles que $t_i[S(c_i) \cap S(c_j)] = t_j[S(c_i) \cap S(c_j)]$, $t_i[S(c_i) \cap S(c_k)] = t_k[S(c_i) \cap S(c_k)]$, $t'_k[S(c_j) \cap S(c_k)] = t_j[S(c_j) \cap S(c_k)]$, $t'_k[S(c_i) \cap S(c_k)] \neq t_i[S(c_i) \cap S(c_k)]$ et $t_j[S(c_j) \cap S(c_k)] \neq t_k[S(c_j) \cap S(c_k)]$. Comme P est triangulaire par rapport à l'ordre $\prec$, nous devons avoir $t'_k[S(c_i) \cap S(c_k)] = t_i[S(c_i) \cap S(c_k)]$ et $t_j[S(c_j) \cap S(c_k)] = t_k[S(c_j) \cap S(c_k)]$, ce qui n'est pas possible puisque P ne vérifie pas DBTP. □

Théorème 19 *$TR \subsetneq DBTP$.*

Preuve : Le théorème 18 montre que $TR \subseteq DBTP$. L'instance présentée dans la figure 8(a) vérifie DBTP mais n'est pas triangulaire. □

En ce qui concerne les classes basées sur des restrictions de structures, nous avons prouvé dans les théorèmes 10 et 11 que β-$ACYCLIC \subsetneq DBTP$, α-$ACYCLIC \cap DBTP \neq \emptyset$ et α-$ACYCLIC \perp DBTP$.

Une autre classe polynomiale importante basée sur une restriction de structure porte sur la largeur d'arbre. Avant de l'évoquer, nous rappelons d'abord la notion de décomposition arborescente de graphe [16].

Définition 14 (décomposition arborescente) *Une **décomposition arborescente** d'un graphe $G = (X, E)$ est une paire (N, T) où $T = (I, F)$ est un arbre avec des nœuds I et des arêtes F et $N = \{N_i : i \in I\}$ est une famille de sous-ensembles de X, telle que chaque sous-ensemble N_i est un nœud de T et vérifie (i) $\cup_{i \in I} N_i = X$, (ii) pour chaque arête $\{x, y\} \in E$, il existe $i \in I$ avec $\{x, y\} \subseteq N_i$, et (iii) pour tout $i, j, k \in I$, si k est un chemin de i à j dans T, alors $N_i \cap N_j \subseteq N_k$.*

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

Cette définition est souvent étendue aux hypergraphes en se référant au *graphe primal* d'un hypergraphe (X, E), qui est le graphe (X, E') où $E' = \{\{x, y\} | \exists e \in E \text{ tel que } x, y \in e\}$.

Définition 15 (largeur arborescente bornée) *Une instance de CSP possède une* ***largeur arborescente bornée*** *si sa largeur arborescente est bornée par une constante. Nous notons BTW l'ensemble de ces instances, et qui constitue une classe polynomiale.*

Théorème 20 $BTW \cap DBTP \neq \emptyset$ *et* $BTW \perp DBTP$.

Preuve : Tout CSP binaire acyclique CSP possède une largeur arborescente bornée par 1 et vérifie DBTP. Donc, l'intersection n'est pas vide. Maintenant, considérons une instance ayant n variables avec $n \gg 3$, dont la largeur arborescente est bornée par une constante $k \geq 3$ et qui contient le sous-problème présenté dans la figure 7. Cette instance possède une largeur arborescente bornée mais ne vérifie pas DBTP. Inversement, une instance ayant n variables et une contrainte d'arité n est DBTP mais n'a pas une largeur arborescente bornée. □

5 (D)BTP et l'(Hyper-)k-Cohérence

Dans cette partie, nous étudions les liens existant entre (D)BTP et l'(Hyper-)k-cohérence orientée. Cette étude nous semble naturelle dans la mesure où, comme évoqué dans [5], BTP est une propriété moins forte que l'hyper-3-cohérence [14].

Comme l'hyper-3-cohérence qui implique BTP, est trop forte, nous définissons une forme relâchée en prenant en compte un ordre sur les contraintes :

Définition 16 (Hyper-k-Cohérence Orientée) *Étant donnés un CSP* $P = (X, D, C)$*, un ordre sur les contraintes* $\prec$ *et un entier* k *tel que* $1 \leq k \leq e$*, P vérifie l'****Hyper-k-cohérence orientée*** *si pour tout sous-ensemble de* k *contraintes tel que* $c_1 \prec c_2 \prec \ldots \prec c_{k-1} \prec c_k$*, on a* $\bowtie_{i=1}^{k-1} R(c_i)[(\bigcup_{i=1}^{k-1} S(c_i)) \cap S(c_k)] \subseteq R(c_k)[(\bigcup_{i=1}^{k-1} S(c_i)) \cap S(c_k)]$

Théorème 21 *Si un CSP P vérifie DBTP par rapport à un ordre* $\prec$ *et qu'il est Hyper-k-cohérent orienté par rapport à* $\prec$ *pour* $2 \leq k < e$*, alors P est Hyper-*$(k+1)$*-cohérent orienté par rapport à* $\prec$*.*

Preuve : Supposons que P ne soit pas Hyper-$(k+1)$-cohérent. Alors, il existe un sous-ensemble de $k+1$ contraintes tel que $c_1 \prec \ldots \prec c_k \prec c_{k+1}$, $\exists(t_1, \ldots, t_k) \in R(c_1) \times \ldots \times R(c_k), \forall t_{k+1} \in R(c_{k+1})$, $\bowtie_{i=1}^{k} t_i[(\bigcup_{i=1}^{k} S(c_i)) \cap S(c_{k+1})] \neq t_{k+1}[(\bigcup_{i=1}^{k} S(c_i)) \cap S(c_{k+1})]$. Considérons les k sous-ensembles de k contraintes $\{c_1, \ldots c_{j-1}, c_{j+1}, \ldots c_k, c_{k+1}\}$, pour $1 \leq j \leq k$. Comme P est Hyper-k-cohérent orienté, pour $1 \leq j \leq k$, il existe un tuple t_{k+1}^j of $R(c_{k+1})$ tel que $\bowtie_{i=1, i\neq j}^{k} t_i[(\bigcup_{i=1, i\neq j}^{k} S(c_i)) \cap S(c_{k+1})] = t_{k+1}^j[(\bigcup_{i=1, i\neq j}^{k} S(c_i)) \cap S(c_{k+1})]$. Considérons $1 \leq j < j' \leq k$. Nous avons deux cas :

(1) $t_{k+1}^j = t_{k+1}^{j'}$. Alors $t_{j'}[S(c_{j'}) \cap S(c_{k+1})] = t_{k+1}^j[S(c_{j'}) \cap S(c_{k+1})]$ et $t_j[S(c_j) \cap S(c_{k+1})] = t_{k+1}^{j'}[S(c_j) \cap S(c_{k+1})]$. Donc $\bowtie_{i=1}^{k} t_i[(\bigcup_{i=1}^{k} S(c_i)) \cap S(c_{k+1})] = t_{k+1}^j[(\bigcup_{i=1}^{k} S(c_i)) \cap S(c_{k+1})]$ et nous avons une contradiction.

(2) $t_{k+1}^j \neq t_{k+1}^{j'}$. Nous avons $t_j[S(c_j) \cap S(c_{j'})] = t_{j'}[S(c_j) \cap S(c_{j'})]$, $t_{j'}[S(c_{j'}) \cap S(c_{k+1})] = t_{k+1}^j[S(c_{j'}) \cap S(c_{k+1})]$, $t_j[S(c_j) \cap S(c_{k+1})] = t_{k+1}^{j'}[S(c_j) \cap S(c_{k+1})]$, $t_{j'}[S(c_{j'}) \cap S(c_{k+1})] \neq t_{k+1}^{j'}[S(c_{j'}) \cap S(c_{k+1})]$ et $t_j[S(c_j) \cap S(c_{k+1})] \neq t_{k+1}^j[S(c_j) \cap S(c_{k+1})]$. Par conséquent P ne vérifie pas DBTP par rapport à $\prec$ et nous avons encore une contradiction.

Donc, P est Hyper-$(k+1)$-cohérent orienté par rapport à l'ordre $\prec$. □

Comme l'intercohérence correspond à l'hyper-2-cohérence, nous pouvons en déduire ce corollaire :

Corollaire 1 *Si un CSP P vérifie DBTP par rapport à un ordre* $\prec$ *sur les contraintes et qu'il est intercohérent orienté par rapport à* $\prec$*, alors P est Hyper-*$(k+1)$*-cohérent orienté par rapport à* $\prec$ *pour* $1 \leq k < e$*.*

Donc, un CSP qui est à la fois DBTP et intercohérent orienté par rapport à un ordre donné sur les contraintes est cohérent. Il s'ensuit également qu'un tel CSP vérifie l'hyper-3-cohérence orientée.

De plus, comme l'Hyper-$(k+1)$-cohérence correspond à la k-cohérence sur le problème dual, nous pouvons également en déduire le théorème et le corollaire suivants en appliquant un raisonnement similaire.

Théorème 22 *Si un CSP binaire P vérifie BTP par rapport à un ordre* $<$ *sur les variables et qu'il est k-cohérent orienté par rapport à* $<$ *pour* $2 \leq k < n$*, alors P est* $(k+1)$*-cohérent orienté par rapport à* $<$*.*

Corollaire 2 *Si un CSP binaire P vérifie BTP par rapport à un ordre* $<$ *sur les variables et qu'il est k-cohérent orienté par rapport à* $<$*, alors P est* $(k+1)$*-cohérent orienté par rapport à* $<$ *pour* $1 \leq k < n$*.*

Comme conséquence, un CSP binaire qui est à la fois BTP et DAC par rapport à un ordre donné sur les variables est cohérent.

En ce qui concerne le positionnement de l'Hyper-3-cohérence orientée par rapport à BTP, nous pouvons prouver que l'hyper-3-cohérence n'implique pas nécessairement BTP. Par exemple, nous pouvons considérer une instance binaire avec 6 variables $\{x_a, x_b, \ldots, x_f\}$. Nous définissons cette instance en reproduisant plusieurs fois un même motif tel que chaque valeur

apparaissant dans une occurrence du motif n'apparaisse dans aucune autre de ses occurrences. Ce motif consiste en un triangle cassé sur une variable z pour un triplet (x, y, z) (i.e. ce qui impose la condition $z < max(x, y)$ sur $<$) et chaque valeur des variables x, y et z est liée à une valeur donnée d'une variable qui n'est pas impliqué dans ce triplet. Nous reproduisons ce motif six fois de telles sorte que les conditions suivantes soient vérifiées : $x_a < max(x_b, x_c)$, $x_b < max(x_d, x_e)$, $x_c < max(x_e, x_f)$, $x_d < max(x_a, x_b)$, $x_e < max(x_a, x_b)$ et $x_f < max(x_a, x_b)$. La figure 8(b) présente ce motif pour les triplets (x_a, x_b, x_e), un triangle cassé sur x_e (correspondant à la condition $x_e < max(x_a, x_b)$) et sur les variables indépendantes x_c, x_d et x_f. En faisant cela, la microstructure de notre CSP binaire possède six composantes connexes. On peut noter que cette instance n'est pas BTP parce que les six conditions rendent impossible la construction d'un ordre approprié sur les variables. Néanmoins, il est hyper-3-cohérent orienté ainsi qu'arc-cohérent.

6 Conclusion et perspectives

Dans cet article, nous avons étudié une classe polynomiale hybride dont les instances peuvent être résolues en temps polynomial par des algorithmes de type MAC. Nous avons prouvé qu'elle est incomparable avec plusieurs classes polynomiales connues (notamment BTP) et qu'elle inclut des classes polynomiales à la fois structurelles et relationnelles (à savoir les CSP β-acycliques et les CSP Triangulaires). Enfin, nous avons mis en évidence les liens existant entre (D)BTP et la k-cohérence ainsi que l'hyper-k-cohérence orientée.

Une première extension devrait consister en l'étude des liens entre DBTP et d'autres classes polynomiales non prises en compte dans cette contribution. Puis, dans le même esprit, nous pourrions explorer la possibilité de définir de nouvelles classes polynomiales basées sur d'autres codages de CSP non binaires.

Références

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

[2] C. Bessière, K. Stergiou, and T. Walsh. Domain filtering consistencies for non-binary constraints. *Artificial Intelligence*, 172 :800–822, 2008.

[3] D. Cohen, M. Cooper, M. Green, and D. Marx. On guaranteeing polynomially bounded search tree size. In *Proc. of CP*, pages 160–171, 2011.

[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] D. Duris. Some characterizations of γ and β-acyclicity of hypergraphs. *Information Processing Letters*, 112 (16) :617–620, 2012.

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

[9] M. H Graham. On the universal relation. Technical report, University of Toronto, 1979.

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

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

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

[13] P. Jégou. Decomposition of Domains Based on the Micro-Structure of Finite Constraint Satisfaction Problems. In *Proc. of AAAI*, pages 731–736, 1993.

[14] P. Jégou. On the Consistency of General Constraint-Satisfaction Problems. In *Proc. of AAAI*, pages 114–119, 1993.

[15] A. El Mouelhi, P. Jégou, C.Terrioux, and B. Zanuttini. Some New Tractable Classes of CSPs and their Relations with Backtracking Algorithms. In *Proc. of CP-AI-OR*, 2013.

[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 *Proc. of ECAI*, pages 125–129, 1994.

[18] P. van Beek and R. Dechter. On the minimality and decomposability of row-convex constraint networks. *J. ACM*, 42(3) :543–561, 1995.