2013 IEEE 25th International Conference on Tools with Artificial Intelligence

A Hybrid Tractable Class for Non-Binary CSPs

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*—Find new islands of tractability, that is classes of CSPs for which polytime algorithms exist, is a fundamental task in the study of constraint satisfaction problems. The concept of hybrid tractable class, which allows to deal simultaneously with the restrictions of languages and, for example, the satisfaction of structural properties, is an approach which has already shown its interest in this domain. Here we study a hybrid class for non-binary CSPs. With this aim in view, we consider the tractable class BTP introduced in [1]. While this class has been defined for binary CSPs, the authors have suggested to extend it to CSPs with constraints of arbitrary arities, using the dual representation of such CSPs. We develop this idea by proposing a new definition without exploiting the dual representation, but using a semantic property associated to the compatibility relations of the constraints. This class, called DBTP for Dual BTP, is firstly shown to be tractable. Then it is compared to some known classes. In particular, we prove that DBTP is incomparable with BTP and that it includes some well known classes of CSPs such as β-acyclic CSPs.**

I. Introduction

A *CSP instance* $P = (X, D, C)$ is defined by a set X of n variables (denoted $x_1, ..., x_n$), a set of domains $D = \{d_1, ..., d_n\}$ (d_i is the set of the possible values for the variable x_i) and a set C of e constraints (denoted $c_1, ..., c_e$). Each constraint c_i involves a set of variables called the *scope* of c_i and denoted $S(c_i)$. A constraint c_i allows a set of tuples over $\prod_{x_j \in S(c_i)} d_j$ defined by the relation $R(c_i)$. $r_i = |S(c_i)|$ is the *arity* of the constraint c_i while r denotes the largest arity and $\rho = max\{|R(c_i)|\}$ the size of the largest relation. Usually, we distinguish binary constraints whose arity is equal to 2 from non-binary ones (also called n-ary). Likewise, binary CSPs (CSPs where the constraints are binary) are considered separately from CSPs with constraints of arbitrary arities. For binary CSPs, we will denote by c_{ij} the constraints involving x_i and x_j. For both binary CSPs and CSPs of arbitrary arity, the problem of finding a solution (i.e. an assignment of a value to each variable which satisfies all the constraints) is NP-Complete.

Although the problem CSP is NP-complete, there exist classes of instances that can be solved (and often recognized) in polynomial time. These classes are called "tractable classes" and rely on some properties that can be verified by the instances. There are two main kinds of such properties. The first one concerns the structural properties of the constraint network. For example, we know that tree-structured binary CSPs can be solved in linear time [2]. Another kind of properties is related to restrictions on the language defining the constraints. These restrictions concern the domains and/or the compatibility relations associated with the constraints. For example, it is the case for the class of "0-1-all constraints" [3]. More recently, some tractable classes have been proposed which are related to these two kinds of properties, such as the BTP class [1]. Their interest is that they are able to take into account both language and structure restrictions. They are thus sometimes called "hybrid classes".

In this paper, we study a hybrid tractable class called *DBTP* for *Dual Broken Triangle Property*. So, this class is based on the concept of "Broken Triangle" which is the basis of BTP. While BTP is only defined for binary constraints, DBTP is defined for CSPs whose constraints have arbitrary arities. Using the dual representation of CSPs, we can consider that this class has been firstly (and briefly) proposed in [1], as a non-binary version of BTP. However, we can also define DBTP by a semantic property related to the compatibility of tuples appearing in triples of relations associated to constraints, without an explicit link to the dual representation. But we show that these two definitions are equivalent (see Theorem 3). Nevertheless, DBTP is a tractable class quite different from BTP. For example, we prove that DBTP is not a generalization of BTP to constraints of arbitrary arity since in the case of binary CSPs, BTP and DBTP are formally different (see Theorem 11). Another example of these differences is related to the fact that DBTP is a conservative property for the filtering of domains and for the filtering of relations while BTP is conservative only for the filtering of domains. Moreover, we show that this tractable class includes simultaneously, structural classes such as β-$acyclic$ CSPs but also classes defined by language restrictions. We also establish that DBTP is incomparable with many well known tractable classes (e.g. ZOA [3], row-convex [4] or max-closed [5]).

As mentioned above, we prove that DBTP is a conservative property for many classical filterings like arc-consistency or pairwise consistency. It ensues that DBTP

1082-3409/13 $31.00 © 2013 IEEE
DOI 10.1109/ICTAI.2013.144

seems to have a real practical interest since any instance satisfying DBTP can be solved in polytime using algorithms similar to MAC [6].

This paper is organized as follows. In section II, we introduce the class DBTP and provide its main features. Then in section III, we study the relationship between BTP and DBTP and show that DBTP includes β-acyclic CSPs. Section IV examines the relationship between DBTP and other well known tractable classes. Finally, we conclude and give some perspectives in section V.

II. DBTP: Definition and Properties

First, we recall the BTP property on which the DBTP property relies:

Definition 1 (Broken Triangle Property [1]): A CSP instance (X, D, C) satisfies the **Broken Triangle Property** (BTP) w.r.t. the variable ordering $<$ if, for all triples of variables (x_i, x_j, x_k) s.t. $x_i < x_j < x_k$, s.t. $(v_i, v_j) \in R(c_{ij})$, $(v_i, v_k) \in R(c_{ik})$ and $(v_j, v'_k) \in R(c_{jk})$, then either $(v_i, v'_k) \in R(c_{ik})$ or $(v_j, v_k) \in R(c_{jk})$. If neither of these two tuples exist, (v_i, v_j), (v_i, v_k) and (v_j, v'_k) is called a **Broken Triangle on** x_k. Let BTP be the set of the instances for which BTP holds w.r.t. some variable ordering.

The BTP property is relative to the compatibility between the values of domains which can be graphically visualized on the micro-structure graph[1]. As each of these compatibilities involves as many values as the arity of the considered constraint, such a property cannot be easily generalized to non-binary CSPs. So a natural alternative[2] consists in considering the compatibilities between the relations through the notion of dual of a CSP instance. The **dual** of the CSP $P = (X, D, C)$ is the binary CSP $P^d = (X^d, D^d, C^d)$ where each constraint c_i of C is associated to the variable x_i^d of X^d whose domain d_i^d is defined by the tuples t_i of $R(c_i)$ s.t. $\forall x_j \in S(c_i), t_i[\{x_j\}] \in d_j$ (where $t[Y]$ denotes the restriction of the tuple t to the variables of the subset $Y \subseteq X$), and a constraint c_{ij}^d of C^d links two variables x_i^d and x_j^d of X^d if the corresponding constraints c_i and c_j of C share at least a variable (i.e. $S(c_i) \cap S(c_j) \neq \emptyset$). The relation $R(c^d)$ is defined by the tuples $(t_i, t_j) \in d_i^d \times d_j^d$ s.t. $t_i[S(c_i) \cap S(c_j)] = t_j[S(c_i) \cap S(c_j)]$. It is well known that, for any CSP P, P has a solution iff P^d has a solution.

We now define the DBTP property:

Definition 2 (Dual Broken-Triangle Property): A CSP $P = (X, D, C)$ satisfies the **Dual Broken Triangle Property** (DBTP) w.r.t. the constraint ordering $\prec$ iff the dual of P satisfies BTP w.r.t. $\prec$. Let $DBTP$ be the set of the instances for which the DBTP property holds for some constraint ordering.

[1]The micro-structure [7] of a binary CSP $P = (X, D, C)$ is the undirected graph $\mu(P) = (V, E)$ where $V = \{(x_i, v_i) : x_i \in X, v_i \in d_i\}$ and $E = \{ \{(x_i, v_i), (x_j, v_j)\} : i \neq j, c_{ij} \notin C$ or $(v_i, v_j) \in R(c_{ij})\}$

[2]Such an idea has already been introduced in [1] but it was just mentioned briefly and thus, it was not studied in depth.

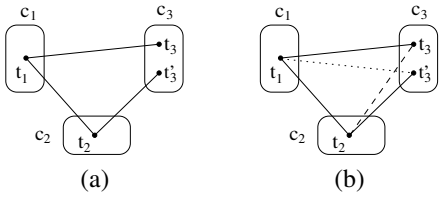

Figure 1. Illustration of DBTP on the constraints c_1, c_2 and c_3.

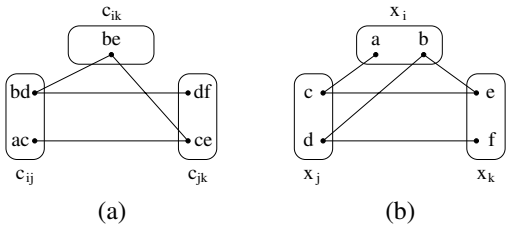

Figure 2. An instance which satisfies DBTP (a) but not BTP (b).

We can observe graphically the DBTP property on the micro-structure of the dual of the original instance. For instance, Figure 1(a) represents the micro-structure of the dual instance of a CSP P with three constraints c_1, c_2 and c_3. In this example, we consider four tuples, $t_1 \in R(c_1)$, $t_2 \in R(c_2)$ and $t_3, t'_3 \in R(c_3)$ s.t. $t_1[S(c_1) \cap S(c_2)] = t_2[S(c_1) \cap S(c_2)]$, $t_1[S(c_1) \cap S(c_3)] = t_3[S(c_1) \cap S(c_3)]$, $t_2[S(c_2) \cap S(c_3)] = t'_3[S(c_2) \cap S(c_3)]$, $t_1[S(c_1) \cap S(c_3)] \neq t'_3[S(c_1) \cap S(c_3)]$ and $t_2[S(c_2) \cap S(c_3)] \neq t_3[S(c_2) \cap S(c_3)]$. If we consider the ordering $c_1 \prec c_2 \prec c_3$, P does not satisfy DBTP w.r.t. $\prec$. Now, if we have P' (see Figure 1(b)) s.t. either t_1 and t'_3 (dotted edge) or t_2 and t_3 (dashed edge) are compatible, then P' satisfies DBTP according to $\prec$.

The class $DBTP$ differs necessarily from the class BTP since $DBTP$ may contain non-binary instances while BTP is restricted to binary instances. It follows a natural question about the comparison of these two classes in the particular case of binary CSPs. In particular, a binary instance may satisfy DBTP while not satisfying BTP. For instance, Figure 2(b) depicts the micro-structure of a binary instance which is DBTP w.r.t. the ordering $c_{ij} \prec c_{jk} \prec c_{ik}$ but not BTP. Figure 2(a) represents the micro-structure of its dual. Conversely, a binary instance can satisfy BTP but not DBTP. This case is illustrated in Figure 3 (where the broken triangles in broken lines prove that DBTP does not hold). Theorem 1 is deduced from these examples.

Theorem 1: Let $P = (X, D, C)$ be a binary CSP.

- P satisfies DBTP $\not\Rightarrow$ P satisfies BTP,
- P satisfies BTP $\not\Rightarrow$ P satisfies DBTP.

This first theorem shows that DBTP is then not a generalization of BTP to non-binary CSPs.

We now prove that the class of CSPs which satisfy DBTP is tractable, thanks to the two next lemmas, whose proofs exploit the approach proposed in [1].

Lemma 1: Any CSP $P = (X, D, C)$ which satisfies

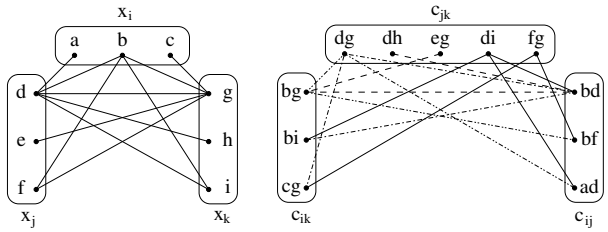

Figure 3. An instance satisfying BTP (a) but not DBTP (b).

DBTP w.r.t. the constraint ordering $\prec$ can be solved in $O(e^2.r.\rho^2)$.

Proof: The first step consists in building the dual of P, what can be achieved in $O(e^2.r.\rho^2)$. Then, as the dual of P is BTP, we know that it can be solved in $O(e^2.\rho^2)$ [1]. Hence, the overall complexity is $O(e^2.r.\rho^2)$. □

Lemma 2 expresses that the constraint ordering $\prec$ related to DBTP may be computed (if any) in polynomial time.

Lemma 2: Given any CSP $P = (X, D, C)$, determining if a constraint ordering $\prec$ s.t. P is DBTP w.r.t. $\prec$ exists (and finding it if any) can be achieved in polynomial time.

Proof: A possible algorithm consists in computing first the dual of P and then determining if an ordering $\prec$ s.t. the dual of P is BTP exists like in [1]. Both steps are polynomial (see the previous proof and [1]). Hence, the overall complexity is polynomial. □

The two previous lemmas allow to establish the tractability of DBTP.

Theorem 2: $DBTP$ is a tractable class.

We now present an alternative and equivalent characterization of DBTP:

Theorem 3: A CSP $P = (X, D, C)$ satisfies the DBTP property w.r.t. the constraint ordering $\prec$ iff for all triples of constraints (c_i, c_j, c_k) s.t. $c_i \prec c_j \prec c_k$, for all $t_i \in R(c_i)$, $t_j \in R(c_j)$ and $t_k, t'_k \in R(c_k)$ s.t.

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

then

- either $t'_k[S(c_i) \cap S(c_k)] = t_i[S(c_i) \cap S(c_k)]$
- or $t_j[S(c_j) \cap S(c_k)] = t_k[S(c_j) \cap S(c_k)]$.

Proof: P satisfies DBTP w.r.t. $\prec$

$\Leftrightarrow$ P^d satisfies BTP w.r.t. $\prec$

$\Leftrightarrow$ for all triples of variables (x_i^d, x_j^d, x_k^d) s.t. $x_i^d \prec x_j^d \prec x_k^d$, for all $t_i \in d_i^d$, $t_j \in d_j^d$ and $t_k, t'_k \in d_k^d$ s.t. $(t_i, t_j) \in R(c_{ij}^d)$, $(t_i, t_k) \in R(c_{ik}^d)$ and $(t_j, t'_k) \in R(c_{jk}^d)$ then either $(t_i, t'_k) \in R(c_{ik}^d)$ or $(t_j, t_k) \in R(c_{jk}^d)$

$\Leftrightarrow$ for all triples of constraints (c_i, c_j, c_k) s.t. $c_i \prec c_j \prec c_k$, for all $t_i \in R(c_i)$, $t_j \in R(c_j)$ and $t_k, t'_k \in R(c_k)$ s.t. $t_i[S(c_i) \cap S(c_j)] = t_j[S(c_i) \cap S(c_j)]$, $t_i[S(c_i) \cap S(c_k)] = t_k[S(c_i) \cap S(c_k)]$ and $t'_k[S(c_j) \cap S(c_k)] = t_j[S(c_j) \cap S(c_k)]$ then either $t'_k[S(c_i) \cap S(c_k)] = t_i[S(c_i) \cap S(c_k)]$ or $t_j[S(c_j) \cap S(c_k)] = t_k[S(c_j) \cap S(c_k)]$. □

We can note that this characterization makes possible the recognition of DBTP instances directly by exploiting the tuples of relations without building the dual instance.

At present, we wonder what the DBTP property becomes when applying a filtering algorithm. A class of CSP $\mathcal{C}$ instances is said **conservative** w.r.t. a filtering consistency ϕ if it is closed under ϕ, that is, if the problem obtained after the application of ϕ belongs to the class $\mathcal{C}$. A property is said conservative if it defines a conservative class of instances.

Property 1: DBTP is conservative for any filtering consistency which only removes values from domains or tuples from relations.

Proof: Let us consider a CSP P satisfying DBTP w.r.t. a given constraint ordering. The removal of a value from the domain of a variable x of P induces the deletion of some tuples for the constraints whose scope contains x. In other words, it implies the deletion of some values for the variables of the dual of P. On the other part, the deletion of some tuples is equivalent to remove some values from domains of some dual variables. Therefore, in both cases, the deletions of values or tuples in the original instance lead to remove values of the dual variables. As BTP is conservative w.r.t. domain filtering consistencies, the dual of P after these removals still satisfies BTP. Hence, P is still DBTP. □

For instance, this property holds for any domain filtering consistency (e.g (Generalized) Arc-Consistency or Path Inverse Consistency [8]) applied on the original instances or their dual. In particular, it is the case for the pairwise-consistency [9] (introduced in the field of Relational Databases Theory [10]) which is equivalent to applying arc-consistency on the dual instance [9].

Definition 3 (Pairwise-Consistency [9]): A CSP $P = (X, D, C)$ is **pairwise-consistent** iff $\forall\ 1 \leq i \leq e$, $R(c_i) \neq \emptyset$ and $\forall\ 1 \leq i < j \leq e$, $R(c_i)[S(c_i) \cap S(c_j)] = R(c_j)[S(c_i) \cap S(c_j)]$.

As MAC [6] maintains arc-consistency (denoted AC) at each step of the search, we define MPWC as the algorithm corresponding to maintain the pairwise-consistency.

Theorem 4: If $P = (X, D, C)$ satisfies DBTP, then MPWC solves P in polynomial time w.r.t. any ordering.

Proof: As the pairwise-consistency on P is equivalent to the arc-consistency on the dual of P [9], the application of MPWC on P is equivalent to MAC on the dual of P. Moreover, as P is DBTP, P^d is BTP and so, according to theorem 7.6 of [1], MPWC solves P in polynomial time. □

Finally, we derive a similar result for MAC in the particular case where any pair of constraints share at most one variable. Before, we need to recall two results about arc-consistency and pairwise consistency.

Lemma 3 (Prop. 8.1, p. 146 in [11]): Let $P = (X, D, C)$ be a CSP s.t. $\forall c_i, c_j \in C, |S(c_i) \cap S(c_j)| \leq 1$. If the problem P is arc-consistent, then it is pairwise-consistent.

Lemma 4: Let $P = (X, D, C)$ be an arc-consistent CSP s.t. $\forall c_i, c_j \in C, |S(c_i) \cap S(c_j)| \leq 1$. If the problem P' obtained from P by ~~deleting some values~~ and enforcing AC has no empty domain, then its dual is arc-consistent.

Proof: Consider P, and P' obtained from P by deleting some values and enforcing AC such that it has no empty domain. Since P' is arc-consistent, by lemma 3, it is also pairwise-consistent. Thus, as the pairwise-consistency on P is equivalent to the arc-consistency on the dual of P [9], the dual of P' is arc-consistent. □

Theorem 5: If $P = (X, D, C)$ s.t. $\forall c_i, c_j \in C, |S(c_i) \cap S(c_j)| \leq 1$ is arc-consistent and satisfies DBTP, then MAC can solve P in polynomial time.

Proof: If, after having enforced arc-consistency, no domain, neither relation is empty, then P is pairwise-consistent and has a solution. According to lemma 4, the problem obtained after deleting some values and enforcing arc-consistency still remains pairwise-consistent. Therefore, when applying MAC on the original problem, we also maintain the pairwise-consistency. Moreover, as pairwise-consistency is equivalent to arc-consistency on the dual problem [9], theorem 7.6 of [1] implies that MAC can solve P in polytime since the dual is BTP. □

Of course, this theorem holds for binary CSPs.

III. DBTP vs BTP

We saw with Theorem 1, that even in the case of binary CSPs, BTP and DBTP classes are different. Such results were foreseeable since, even if the original instance and its dual represent the same problem, their structure and micro-structure are quite different. This result relies on the presence of broken triangles in the micro-structure of the instance or of its dual instance. In both cases, these broken triangles often involve values which would be deleted by some filtering consistency like arc-consistency. So, as DBTP and BTP are conservative w.r.t. domain filtering consistencies, we focus our study on binary instances which satisfy arc-consistency and thus pairwise-consistency (by lemma 4). Under these assumptions, we can prove the following lemma:

Lemma 5: Given an arc-consistent binary CSP $P = (X, D, C)$, if for a triple (x_i, x_j, x_k) of variables, we have a broken triangle on x_k, then we have a broken triangle on c_{ik} and one on c_{jk} for the triple (c_{ij}, c_{ik}, c_{jk}) in the dual.

Proof: Let x_i, x_j, $x_k \in X$ s.t. $(v_i, v_j) \in R(c_{ij})$, $(v_i, v_k) \in R(c_{ik})$, $(v_j, v'_k) \in R(c_{jk})$, $(v_i, v'_k) \notin R(c_{ik})$ and $(v_j, v_k) \notin R(c_{jk})$. As P is pairwise-consistent, there are some values $v'_i \in d_i$ and $v'_j \in d_j$ s.t. $v_i \neq v'_i$, $v_j \neq v'_j$, $(v'_i, v'_k) \in R(c_{ik})$ and $(v'_j, v_k) \in R(c_{jk})$. So, $((v_i, v_j), (v_i, v_k))$, $((v_i, v_j), (v_j, v'_k))$ and $((v_i, v_k), (v'_j, v_k))$ forms a broken triangle on c_{jk} for the triple (c_{ij}, c_{ik}, c_{jk}). Likewise for $((v_i, v_j), (v_j, v'_k))$, $((v_i, v_j), (v_i, v_k))$ and $((v_j, v'_k), (v'_i, v'_k))$ on c_{ik}. □

The presence of a broken triangle on x_k for a triple (x_i, x_j, x_k) imposes the condition $x_k < max(x_i, x_j)$ on the variable ordering $<$ (see the proof of theorem 3.2 of [1]). Consequently, according to lemma 5, it corresponds to impose the two conditions $c_{jk} \prec max(c_{ij}, c_{ik})$ and $c_{ik} \prec max(c_{ij}, c_{jk})$ for the triple (c_{ij}, c_{ik}, c_{jk}) on the constraint ordering $\prec$. It ensues that any arc-consistent and pairwise-consistent binary instance which satisfies BTP and has two broken triangles for two different variables of a same triple of variables cannot satisfy DBTP since we will obtain all the possible broken triangles for the corresponding triple of constraints.

Conversely, we show now that a binary instance can be arc-consistent and DBTP but not BTP. For this purpose, let us consider a binary instance with 9 variables $\{x_a, x_b, \ldots, x_i\}$. We define this instance by reproducing several times a same pattern s.t. each value appearing in an instance of the pattern does not appear in any other instance. This pattern consists in a broken triangle on a variable z for a triple (x, y, z) (i.e. which imposes the condition $z < max(x, y)$ on $<$) and each value of the variables x, y and z is linked to a given value of any variable which is not involved in this triple. We reproduce this pattern 9 times s.t. the following conditions are imposed: $x_a < max(x_b, x_c)$, $x_b < max(x_e, x_h)$, $x_c < max(x_e, x_g)$, $x_d < max(x_a, x_g)$, $x_e < max(x_a, x_i)$, $x_f < max(x_d, x_e)$, $x_g < max(x_h, x_i)$, $x_h < max(x_b, x_d)$ and $x_i < max(x_c, x_f)$. Figure 4(b) depicts this pattern for the triple (x_a, x_b, x_c), a broken triangle on x_a (corresponding to the condition $x_a < max(x_b, x_c)$) and an independent variable x_e while Figure 4 (a) describes the corresponding part of the dual instance. By doing this, the micro-structure of our binary CSP or one of its dual instance have 9 connected components. We can note that this instance is not BTP because the 9 conditions make impossible the construction of a suitable variable ordering. In contrast, it is DBTP (w.r.t the ordering $c_{ab} \prec c_{ac} \prec c_{ad} \prec c_{bf} \prec c_{bh} \prec c_{ci} \prec c_{df} \prec c_{dh} \prec c_{ef} \prec c_{ei} \prec c_{gh} \prec c_{gi} \prec c_{af} \prec c_{bc} \prec c_{bd} \prec c_{ce} \prec c_{cg} \prec c_{de} \prec c_{dg} \prec c_{fi} \prec c_{hi} \prec c_{ae} \prec c_{ag} \prec c_{be} \prec c_{bg} \prec c_{cd} \prec c_{cf} \prec c_{di} \prec c_{fh} \prec c_{ai} \prec c_{bi} \prec c_{eg} \prec c_{eh} \prec c_{fg} \prec c_{ah} \prec c_{ch}$), arc-consistent and pairwise-consistent.

Now, we focus our study on acyclic CSPs. [1] has already proved that such binary CSPs satisfy BTP. We are going to show that this is also true for DBTP. Let $TREE$ be the set of binary CSPs whose constraint graph is acyclic.

Theorem 6: $TREE \subsetneq DBTP$.

Proof: Let $DUAL\text{-}TREE$ be the set of binary CSPs which are the dual of instances from $TREE$. As shown in [1], $DUAL\text{-}TREE \subsetneq BTP$. Hence $TREE \subsetneq DBTP$. □

This result can be extended to CSPs of arbitrary arity. For this, we must consider the notion of cyclicity in hy-

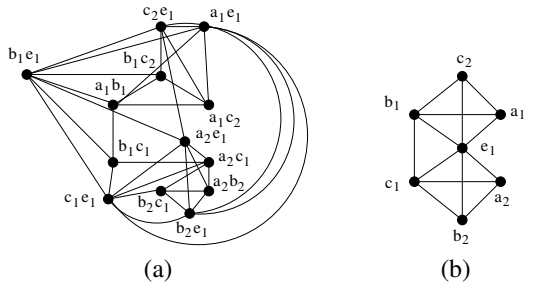

Figure 4. Part of an instance satisfying DBTP, arc-consistency and pairwise-consistency but not BTP.

pergraphs, for which different degrees of cyclicity has been defined [10]. Here, we are interested by α-acyclicity and β-acyclicity. We will first show that β-acyclic CSPs satisfy DBTP and later, we will show it is not the case for α-acyclic CSPs. We first recall the definition of β-acyclicity of (constraint) hypergraph.

Definition 4 ([12]): $H = (X, C)$ is a β-acyclic hypergraph iff it has no Graham cycle. A sequence $(c_1, ..., c_m, c_{m+1})$ with $m \geq 3$ s.t. $(c_1, ..., c_m)$ are distinct and $c_1 = c_{m+1}$ is a Graham cycle if each $\Delta_i = S(c_i) \cap S(c_{i+1})$ $(1 \leq i \leq m)$ is nonempty, and whenever $i \neq j$, Δ_i and Δ_j are incomparable (i.e. $\Delta_i \not\subseteq \Delta_j$ and $\Delta_j \not\subseteq \Delta_i$).

It has been recently shown in [13] that β-acyclic hypergraphs can be defined by applying the two following rules that yield the empty hypergraph:

(1) If a hyperedge is empty, we remove it from C.
(2) If a vertex is a nest point (i.e. the set of hyperedges containing it is a chain for the inclusion relation), then we remove it from H (i.e. from X and from the hyperedges that contain it).

Theorem 7 ([13]): A hypergraph H is β-acyclic if and only if, after applying the two rules successively until none can be applied, we obtain the empty hypergraph.

Using these definitions, we can now establish the next theorem:

Theorem 8: Given a CSP (X, D, C), there exists a constraint ordering $\prec$, s.t. $\forall c_i, c_j, c_k \in C$ s.t. $c_i \prec c_j \prec c_k$, we have $S(c_i) \cap S(c_k) \subseteq S(c_j) \cap S(c_k)$ or $S(c_j) \cap S(c_k) \subseteq S(c_i) \cap S(c_k)$ if and only if (X, D, C) has a β-acyclic constraint hypergraph.

Proof: $(\Rightarrow)$ By contraposition. So we show that if the constraint hypergraph (X, C) is β-cyclic, then, there is no constraint ordering.

Consider a constraint hypergraph (X, C) which is β-cyclic. So, it has a Graham cycle, denoted by the sequence of hyperedges $(c_1, ..., c_m, c_{m+1})$. Consider an arbitrary constraint ordering $\prec$. Necessarily, among the constraints of this cycle, there is a maximum constraint c_k w.r.t. the ordering $\prec$. Consider its two neighbors in the cycle, denoted c_i and c_j (with $c_i, c_j \prec c_k$). By definition of Graham cycles, we know that $S(c_i) \cap S(c_k)$ and $S(c_j) \cap S(c_k)$ are incomparable. So, we have neither $S(c_i) \cap S(c_k) \subseteq S(c_j) \cap S(c_k)$ nor $S(c_j) \cap S(c_k) \subseteq S(c_i) \cap S(c_k)$, and then no suitable constraint ordering $\prec$ exists.

$(\Leftarrow)$ Here, we use the Theorem 7. So, given a β-acyclic CSP (X, D, C) of hypergraph H which admits a constraint ordering $\prec$, we will show that :

(1) A hyperedge $S(c_i)$ is empty iff H without $S(c_i)$ admits an ordering and is β-acyclic.
(2) A vertex x of H is a nest point such H admits an ordering iff H without x admits an ordering and is β-acyclic.

It is immediate to see that the property holds by applying the rule (1). So, consider the rule (2). Assume that for a hypergraph H we have an ordering $\prec$. So, $\forall c_i, c_j, c_k \in C$ s.t. $c_i \prec c_j \prec c_k$, we have $S(c_i) \cap S(c_k) \subseteq S(c_j) \cap S(c_k)$ or $S(c_j) \cap S(c_k) \subseteq S(c_i) \cap S(c_k)$. We have five cases to consider:

1) $x \notin S(c_i) \cup S(c_j) \cup S(c_k)$: thus after the deletion of x, neither $S(c_i) \cap S(c_k)$ nor $S(c_j) \cap S(c_k)$ have changed. So, the property holds.
2) $x \in S(c_i) \cap S(c_j) \cap S(c_k)$: thus after the deletion of x, it disappears from each intersection $S(c_i) \cap S(c_k)$ and $S(c_j) \cap S(c_k)$, and thus, the property holds.
3) x belongs to only one set $S(c_i)$ or $S(c_j)$ or $S(c_k)$: so x belongs to no intersection and thus, the property holds after the deletion.
4) $x \in S(c_i) \cap S(c_j)$ and $x \notin S(c_k)$: so x belongs neither to $S(c_i) \cap S(c_k)$, nor to $S(c_i) \cap S(c_k)$ and thus, the property holds after the deletion.
5) $x \in S(c_i) \cap S(c_k)$ and $x \notin S(c_j)$ (or symmetrically $x \in S(c_j) \cap S(c_k)$ and $x \notin S(c_i)$) : so before the deletion, we have necessarily $S(c_i) \cap S(c_k) \not\subseteq S(c_j) \cap S(c_k)$ and $S(c_j) \cap S(c_k) \subsetneq S(c_i) \cap S(c_k)$. Thus, after the deletion of x, we have at least $S(c_j) \cap S(c_k) \subseteq S(c_i) \cap S(c_k)$

So we have shown that if we can delete the whole hypergraph, which does not contradict the property on the ordering, necessarily, the first hypergraph is β-acyclic and admits a suitable constraint ordering. □

We can note that this theorem explains why the condition enunciated in lemma 4.6 of [1] holds independently from the scope of the constraints. This lemma and the previous theorem allow us to obtain Theorem 9 where $\beta\text{-}ACYCLIC$ is the set of CSPs whose constraint (hyper)graph is β-acyclic.

Theorem 9: $TREE \subsetneq \beta\text{-}ACYCLIC \subsetneq DBTP$.

Proof: According to Theorem 8, we know that for any β-acyclic CSP (X, D, C), there exists a constraint ordering $\prec$, s.t. $\forall c_i, c_j, c_k \in C$ s.t. $c_i \prec c_j \prec c_k$, we have $S(c_i) \cap S(c_k) \subseteq S(c_j) \cap S(c_k)$ or $S(c_j) \cap S(c_k) \subseteq S(c_i) \cap S(c_k)$. Moreover, according to lemma 4.6 of [1], any CSP satisfying the latter condition has a BTP dual. Hence $\beta\text{-}ACYCLIC \subsetneq DBTP$. □

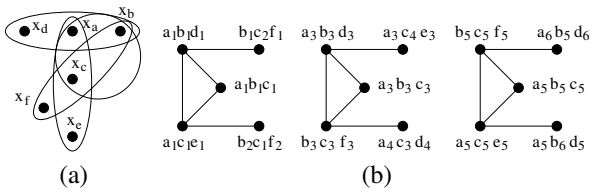

Figure 5. An α-acyclic CSP (a) but which does not satisfy DBTP (b).

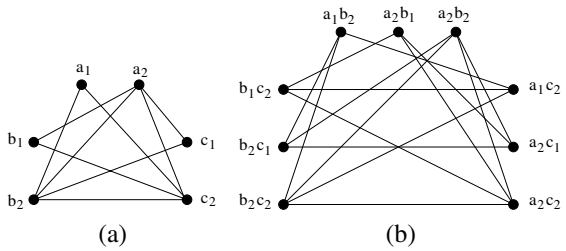

Figure 6. A CSP which isRC, ZOA and RRM (a) but not DBTP (b).

However, the equivalence of lemma 4.6 of [1] only holds for the reverse direction (what does not endanger the proof of Theorem 9). Clearly, it suffices to consider a binary instance with 3 monovalent variables (i.e. variables whose domain contains a single value) pairwise connected (each constraint allows the single tuple). Its dual satisfies BTP while the constraint graph is not β-acyclic.

Now, we show that if α-$ACYCLIC$ is the set of CSPs whose constraint (hyper)graph is α-acyclic, then the sets α-$ACYCLIC$ and $DBTP$ are incomparable. So, we recall that α-acyclicity of (constraint) hypergraphs can be defined using the "running intersection property" [10], namely:

Definition 5: (X, C) is an α-acyclic hypergraph iff there exists an ordering $(c_1, ..., c_e)$ s.t. $\forall k, 1 < k \leq e, \exists j < k, (S(c_k) \cap \bigcup_{i=1}^{k-1} S(c_i)) \subseteq S(c_j)$.

Let us consider a CSP with six variables $x_a, \ldots x_f$ and four constraints whose scope are respectively $\{x_a, x_b, x_c\}$, $\{x_a, x_b, x_d\}$, $\{x_a, x_c, x_e\}$ and $\{x_b, x_c, x_f\}$. Figure 5 depicts its constraint hypergraph (a) and the micro-structure of its dual (b). We can note that this instance is α-acyclic but does not satisfy DBTP since no suitable constraint ordering exist. Moreover, it is well known that β-$ACYCLIC \subsetneq \alpha$-$ACYCLIC$ [10]. Hence, if we denote $A \perp B$ two tractable classes which are incomparable (i.e. neither $A \subseteq B$, nor $B \subseteq A$), we obtain the following theorem:

Theorem 10: α-$ACYCLIC \cap DBTP \neq \emptyset$ and α-$ACYCLIC \perp DBTP$.

Through this section, we have established:

Theorem 11: $BTP \cap DBTP \neq \emptyset$ and $BTP \perp DBTP$.

In the next section, we study the link between DBTP and some other tractable classes.

IV. DBTP VS SOME TRACTABLE CLASSES

A. For binary CSPs

As DBTP and BTP are two different classes, we first focus on some tractable classes included in BTP. These classes whose definitions are recalled below rely on restricted constraint languages.

Definition 6 (Row-convex [4]): A binary CSP $P = (X, D, C)$ is said **row-convex** w.r.t. a variable ordering $<$ and a value ordering, if, for each constraint c_{ij} of C with $x_i < x_j$, $\forall v_i \in d_i$, $\{v_j \in d_j | (v_i, v_j) \in R(c_{ij})\} = [a_j..b_j]$ for some $a_j,\ b_j \in d_j$ where $[a_j..b_j]$ denotes the values belonging to d_j between a_j and b_j w.r.t. the value ordering. We denote RC the set of row-convex instances.

Definition 7 (0-1-all [3]): A binary CSP $P = (X, D, C)$ is said **0-1-all** if for each constraint c_{ij} of C, for each value $v_i \in d_i$, c_{ij} satisfies one of the following conditions:

- (ZERO) for any value $v_j \in d_j, (v_i, v_j) \notin R(c_{ij})$,
- (ONE) there is a unique value $v_j \in d_j$ such as $(v_i, v_j) \in R(c_{ij})$,
- (ALL) for any value $v_j \in d_j, (v_i, v_j) \in R(c_{ij})$.

We denote ZOA the set of instances which are 0-1-all.

Definition 8 (Renamable right monotone [1]): A binary CSP $P = (X, D, C)$ is said **renamable right monotone** w.r.t. a variable ordering $<$ if, for $2 \leq j \leq n$, each domain d_j can be ordered by $\preccurlyeq_j$ s.t. for each constraint c_{ij} of C with $x_i < x_j$, $\forall v_i \in d_i, v_j, v'_j \in d_j$, if $(v_i, v_j) \in R(c_{ij})$ and $v_j \preccurlyeq_j v'_j$ then $(v_i, v'_j) \in R(c_{ij})$. We denote RRM the set of these instances.

The next theorem shows that these tractable classes share some instances with DBTP but are different.

Theorem 12: $RC \cap DBTP \neq \emptyset$ and $RC \perp DBTP$.
$ZOA \cap DBTP \neq \emptyset$ and $ZOA \perp DBTP$.
$RRM \cap DBTP \neq \emptyset$ and $RRM \perp DBTP$.

Proof: If we consider the binary CSP of Figure 6(a), it is 0-1-all, row-convex, and renamable right monotone w.r.t. the lexicographic value and variable orderings. However, as shown in Figure 6(b), this instance is not DBTP. Conversely, any non-binary DBTP instance cannot belong to RC, ZOA or RRM.

In order to prove that $DBTP$ intersects RC, ZOA and RRM, it is sufficient to consider a monovalent and consistent binary CSP with three variables and three constraints since such an instance satisfies both $DBTP$, RC, ZOA and RRM. □

The next class relies on the number of maximal cliques of the micro-structure:

Definition 9 (Maximal clique bounded [14]): A CSP $P = (X, D, C)$ is said **maximal clique bounded** if the number of maximal cliques in its micro-structure is polynomial w.r.t the size of P. We denote CL the set of these instances.

Theorem 13: $CL \cap DBTP \neq \emptyset$ and $CL \perp DBTP$.

Proof: Any monovalent and consistent binary CSP has a single maximal clique and is DBTP. So the intersection is

not empty.

Consider any binary CSP s.t. its micro-structure has a polynomial number of maximal cliques. We add to such a CSP additional variables with additional values and additional constraints corresponding to the instance depicted in figure 1, s.t. these values are not compatible to the one of the first part of this CSP. So, it has a polynomial number of maximal cliques in its micro-structure but is not DBTP. Conversely, any non-binary DBTP instance cannot belong to CL. □

Regarding classes based on restricted structures, we have proved in theorem 9 that $TREE \subsetneq DBTP$.

B. For CSPs of arbitrary arity

We first consider some known tractable classes based on restricted constraint languages like the max-closed class.

Definition 10 (Max-closed [5]): A CSP $P = (X, D, C)$ is said *max-closed* if for each constraint c of arity r_c, $\forall (v_1, v_2, \ldots, v_{r_c}), (v'_1, v'_2, \ldots, v'_{r_c}) \in R(c)$, $(max(v_1, v'_1), max(v_2, v'_2), \ldots, max(v_{r_c}, v'_{r_c})) \in R(c)$. We denote MC the set of max-closed instances.

Theorem 14: $MC \cap DBTP \neq \emptyset$ and $MC \perp DBTP$.

Proof: The proof of $MC \cap DBTP \neq \emptyset$ and $MC \not\subseteq DBTP$ is similar to one of theorem 12. Regarding $DBTP \not\subseteq MC$, any CSP having two variables and one binary constraint is DBTP but not necessarily max-closed. □

Definition 11 (incrementally functional [15]): A CSP $P = (X, D, C)$ is said *incrementally functional* if there exists a variable ordering $<$ s.t. for $1 \leq i < n$, each solution of $P[\{x_1, \ldots, x_i\}]$ extends to at most one solution of $P[\{x_1, \ldots, x_{i+1}\}]$ where, for $X' \subseteq X, P[X']$ denotes the CSP (X', D', C') where $D' = \{d_i | x_i \in X'\}$ and $C' = \{c' | c \in C \; s.t. \; S(c) \cap X' \neq \emptyset, S(c') = S(c) \cap X' \; and \; R(c') = \{t[S(c')] | t \in R(c)\}\}$. We denote $IFUN$ the set of these instances.

Theorem 15: $IFUN \cap DBTP \neq \emptyset$ and $IFUN \perp DBTP$.

Proof: In order to prove that the intersection is not empty, we consider a CSP with four monovalent variables $x_1, \ldots, x_4$ and three ternary constraints c_1, c_2 and c_3 s.t. $S(c_1) = \{x_1, x_2, x_3\}$, $R(c_1) = \{(v_1, v_2, v_3)\}$, $S(c_2) = \{x_1, x_2, x_4\}$, $R(c_2) = \{(v_1, v_2, v_4)\}$, $S(c_3) = \{x_2, x_3, x_4\}$ and $R(c_3) = \{(v_2, v_3, v_4)\}$. This instance is incrementally functional (using the numeration of variables as ordering) and DBTP.

The instance of figure 7(a) is incrementally functional but not DBTP (b). Conversely, any DBTP instance having several solutions cannot be incrementally functional. □

Definition 12 (Dual CB [14]): A CSP $P = (X, D, C)$ is said **dual maximal clique bounded (DMCB)** if the number of maximal cliques in the micro-structure of its dual instance is polynomial w.r.t. the size of P. We denote DCL the set of these instances.

Theorem 16: $DCL \cap DBTP \neq \emptyset$ and $DCL \perp DBTP$.

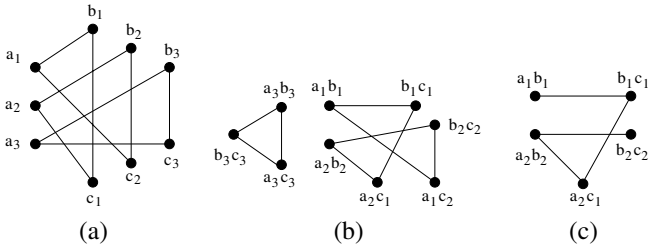

Figure 7. An incrementally functional instance (a) but which does not satisfy DBTP (b). An instance which is DBTP but not triangular (c).

Proof: Let us consider the first instance defined in the proof of theorem 15. The micro-structure of its dual instance has a single maximal clique and the instance is DBTP. So, the intersection is not empty. Regarding the instance depicted in figure 7(b), it has a polynomial number of maximal cliques in the micro-structure of its dual instance but is not DBTP. Conversely, a binary instance whose constraint graph is a star and for which each domain has several values is DBTP but has an unbounded number of maximal cliques in the micro-structure of its dual. □

Now we introduce a new tractable class based on a constraint language restriction.

Definition 13 (Triangular): A CSP $P = (X, D, C)$ is said **triangular** w.r.t. a constraint ordering $\prec$ iff $\forall c_i, c_j, c_k$, $c_i \prec c_j \prec c_k$, $\forall t_i \in R(c_i), t_j \in R(c_j), t_k \in R(c_k)$, if $t_i[S(c_i) \cap S(c_j)] = t_j[S(c_i) \cap S(c_j)]$ and $t_i[S(c_i) \cap S(c_k)] = t_k[S(c_i) \cap S(c_k)]$ then, $t_j[S(c_j) \cap S(c_k)] = t_k[S(c_j) \cap S(c_k)]$. We denote TR the set of triangular instances.

Theorem 17: If a CSP is triangular w.r.t. $\prec$, then it satisfies DBTP w.r.t. $\prec$.

Proof: Assume that P is triangular but not DBTP. So, there exist three constraints c_i, c_j and c_k, $c_i \prec c_j \prec c_k$, $t_i \in R(c_i)$, $t_j \in R(c_j)$ and $t_k, t'_k \in R(c_k)$ s.t. $t_i[S(c_i) \cap S(c_j)] = t_j[S(c_i) \cap S(c_j)]$, $t_i[S(c_i) \cap S(c_k)] = t_k[S(c_i) \cap S(c_k)]$, $t'_k[S(c_j) \cap S(c_k)] = t_j[S(c_j) \cap S(c_k)]$, $t'_k[S(c_i) \cap S(c_k)] \neq t_i[S(c_i) \cap S(c_k)]$ and $t_j[S(c_j) \cap S(c_k)] \neq t_k[S(c_j) \cap S(c_k)]$. As P is triangular w.r.t. $\prec$, we must have $t'_k[S(c_i) \cap S(c_k)] = t_i[S(c_i) \cap S(c_k)]$ and $t_j[S(c_j) \cap S(c_k)] = t_k[S(c_j) \cap S(c_k)]$, what is not possible since P does not satisfy DBTP. □

Theorem 18: $TR \subsetneq DBTP$.

Proof: Theorem 17 shows that $TR \subseteq DBTP$. The instance depicted in figure 7(c) satisfies DBTP but is not triangular. □

Regarding classes based on restricted structures, we have proved in Theorems 9 and 10 that $\beta\text{-}ACYCLIC \subsetneq DBTP$, $\alpha\text{-}ACYCLIC \cap DBTP \neq \emptyset$ and $\alpha\text{-}ACYCLIC \perp DBTP$. Another important tractable class based on restricted structure is related to the tree-width. We first recall the notion of tree-decomposition of graphs [16].

Definition 14 (tree-decomposition): A **tree-decomposition** of a graph $G = (X, E)$ is a pair (N, T) where

$T = (I, F)$ is a tree with nodes I and edges F and $N = \{N_i : i \in I\}$ is a family of subsets of X, s.t. each subset N_i is a node of T and verifies (i) $\cup_{i \in I} N_i = X$, (ii) for each edge $\{x, y\} \in E$, there exists $i \in I$ with $\{x, y\} \subseteq N_i$, and (iii) for all $i, j, k \in I$, if k is in a path from i to j in T, then $N_i \cap N_j \subseteq N_k$.

The width w of a tree-decomposition (N, T) is equal to $max_{i \in I}|N_i| - 1$. The **tree-width** w^* of G is the minimal width over all the tree-decompositions of G.

Classically, this definition is extended to hypergraphs by considering the notion of primal graphs. The primal graph of a hypergraph (X, E) is the graph (X, E') where $E' = \{\{x, y\} | \exists e \in E \ s.t. \ x, y \in e\}$.

Definition 15 (bounded tree-width): Let k be a fixed positive integer. The class BTW_k is the set of the instances whose tree-width is bounded by k.

Theorem 19: $BTW_1 \subsetneq DBTP$.
For $k > 1$, $BTW_k \cap DBTP \neq \emptyset$ and $BTW_k \perp DBTP$.
Proof: It is well known that BTW_1 is the set of tree-structured binary CSPs. So according to theorem 9, we have $BTW_1 \subsetneq DBTP$. For $k > 1$, as $BTW_1 \subsetneq BTW_k$, the intersection $BTW_k \cap DBTP$ is not empty. Now, let us consider an instance having n variables with $n \geq 3$, whose tree-width is bounded by a constant $k \geq 2$ and which contains the subproblem depicted in figure 7(a). This instance has a bounded tree-width but does not satisfy DBTP. Conversely, any instance having n variables and one constraint of arity n is DBTP but has an unbounded tree-width. Hence $BTW_k \perp DBTP$. □

V. Conclusion and future works

In this paper, we have studied a hybrid tractable class whose instances can be solved in polynomial time by MAC-like algorithms. We have then proved that it is incomparable with several known tractable classes (notably BTP) and that it captures both structural and relational tractable classes (namely β-acyclic CSPs and Triangular CSPs).

A first extension consists in studying the link between DBTP and other tractable classes we have not mentioned in this paper. Another one consists in considering other properties and then in extending other tractable classes to non-binary CSPs using a similar approach, using the dual representation. One interesting candidate could be the *min-of-max extendable* property also introduced in [1].

Then, in the same spirit, we can also explore the possibility of defining new tractable classes, taking properties like BTP (or some others) and exploiting other encodings of non-binary CSPs.

Acknowledgments

This work was supported by the French National Research Agency under grant TUPLES (ANR-2010-BLAN-0210).

The authors would like to thank Martin C. Cooper for his useful comments.

References

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

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

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

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

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

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

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

[9] 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.

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

[11] 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.

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

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

[14] 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.

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

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