Artificial Intelligence 174 (2010) 570–584

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Generalizing constraint satisfaction on trees: Hybrid tractability and variable elimination

Martin C. Cooper [a], Peter G. Jeavons [b,*], András Z. Salamon [b,c]

[a] IRIT, University of Toulouse III, 31062 Toulouse, France
[b] Computing Laboratory, University of Oxford, Oxford, OX1 3QD, UK
[c] Oxford-Man Institute of Quantitative Finance, 9 Alfred Street, Oxford, OX1 4EH, UK

ARTICLE INFO

Article history:
Received 17 August 2009
Received in revised form 13 February 2010
Accepted 24 March 2010
Available online 27 March 2010

Keywords:
Constraint satisfaction
Tractability
Computational complexity
Arc consistency
Variable ordering
Variable elimination

ABSTRACT

The Constraint Satisfaction Problem (CSP) is a central generic problem in artificial intelligence. Considerable progress has been made in identifying properties which ensure tractability in such problems, such as the property of being tree-structured. In this paper we introduce the broken-triangle property, which allows us to define a novel tractable class for this problem which significantly generalizes the class of problems with tree structure. We show that the broken-triangle property is conservative (i.e., it is preserved under domain reduction and hence under arc consistency operations) and that there is a polynomial-time algorithm to determine an ordering of the variables for which the broken-triangle property holds (or to determine that no such ordering exists). We also present a non-conservative extension of the broken-triangle property which is also sufficient to ensure tractability and can also be detected in polynomial time.

We show that both the broken-triangle property and its extension can be used to eliminate variables, and that both of these properties provide the basis for preprocessing procedures that yield unique closures orthogonal to value elimination by enforcement of consistency. Finally, we also discuss the possibility of using the broken-triangle property in variable-ordering heuristics.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

The Constraint Satisfaction Problem (CSP) is a central generic problem in artificial intelligence where each instance consists of a collection of variables which must be assigned values subject to specified constraints. Each CSP instance has an underlying undirected graph, known as its *constraint network*, whose nodes are the variables of the instance, and whose edges connect precisely those pairs of variables which are related by some specified constraint. Such a graph is sometimes called the *structure* of the instance.

There is a well-known efficient algorithm for solving any CSP instance whose underlying constraint network is a *tree* [1,2]. If establishing arc consistency leads to a domain wipe-out, then no solution exists; otherwise a solution exists and can be found by a backtrack-free search if the variables are ordered from any designated root to the leaves.

However, having tree structure is a very restrictive property. It is therefore worthwhile exploring more general problem classes, to identify more widely-applicable properties which still allow efficient solution algorithms. Any subclass of the general CSP which can be solved in polynomial time, and also can be identified in polynomial time, is called a *tractable* subclass.

* Corresponding author.
E-mail addresses: cooper@irit.fr (M.C. Cooper), Peter.Jeavons@comlab.ox.ac.uk (P.G. Jeavons), Andras.Salamon@comlab.ox.ac.uk (A.Z. Salamon).

0004-3702/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.artint.2010.03.002

There has been a considerable research effort in identifying tractable subclasses of the CSP over the past two decades. Most of this work has focused on one of two general approaches: either identifying forms of constraint which are sufficiently restrictive to ensure tractability no matter how they are combined [3,4], or else identifying structural properties of constraint networks which ensure tractability no matter what forms of constraint are imposed [2,5].

The first approach has had considerable success in characterizing precisely which forms of constraint ensure tractability no matter how they are combined. A set of constraint types with this property is called a tractable *constraint language*. In general it has been shown that *any* tractable constraint language must have certain kinds of algebraic properties known as polymorphisms [6]. A complete characterization of all possible tractable constraint languages has been established in the following cases: *conservative* constraint languages (i.e., constraint languages containing all unary constraints) [7], and constraint languages over a 2-element domain [8] or a 3-element domain [9].

The second approach has also had considerable success in characterizing precisely which structures of constraint networks ensure tractability no matter what constraints are imposed. For the class of problems where the arity of the constraints is bounded above by some fixed constant (such as binary constraint problems) it has been shown that (subject to certain technical assumptions) the *only* class of structures which ensure tractability are structures of *bounded tree-width* [10–12]. This result significantly extends the class of tree-structured CSPs while retaining tractability.

In practice, constraint satisfaction problems usually do not possess a sufficiently restricted structure or use a sufficiently restricted constraint language to fall into any of these tractable classes. They may still have properties which ensure they can be solved efficiently, but these properties concern both the structure of the constraint network and the form of the constraints. Such properties have sometimes been called *hybrid* reasons for tractability [13–16], and they are less widely-studied and much less well-understood than the language properties and structural properties described above.

A classical approach to tractability of CSPs is to identify conditions on the class of CSP instances that can be used to construct an ordering of variables which allows the instance to be solved efficiently. Freuder introduced a condition that allows a variable ordering to be found in polynomial time, such that this variable ordering provides a backtrack-free search procedure [1]. The condition amounts to requiring that a level of consistency has been enforced that is at least as great as a measure he called the *width* of the constraint graph. More generally, the amount of backtracking can be bounded in terms of the relationship between the level of consistency and the width of the constraint graph [17]. We note that Freuder's notion of width is equivalent to the notion of *tree-width* which is now widely used in graph theory [18].

The basic property described in this paper, which we call the Broken-Triangle Property (BTP), is a polynomial-time detectable property which defines a novel hybrid tractable class of binary CSP instances. The BTP can be viewed as forbidding the occurrence of certain subproblems of a fixed size within a CSP instance. A number of other properties of subproblems of bounded size that guarantee tractability have previously been identified in the literature [19,20], but the BTP is unusual in that it also incorporates variable ordering information.

The class of CSP instances that have the BTP with respect to some ordering is tractable: for all such instances there is a polynomial-time procedure to determine a variable ordering which guarantees backtrack-free search. Our class is not contained in the classes considered by Freuder, as we do not require a fixed relationship between tree-width and consistency [1,17]. We show that all tree-structured CSP instances satisfy the BTP, as well as many other instances that are not tree-structured (including some with unbounded tree-width).

We also show that the BTP, and certain generalizations, can be used to define a variable-elimination strategy which can be applied to any binary CSP. Even when no variables can be eliminated by this strategy, we show that it can still provide a basis for a new form of variable-ordering heuristic. For example, if the BTP is satisfied on a subset S of the variables, then these variables should be placed at the end of the variable ordering. This guarantees that a search algorithm which maintains arc consistency during search will not backtrack on the variables in S.

The paper is structured as follows. Sections 2 and 3 introduce the broken-triangle property and prove the tractability of binary CSP instances satisfying the BTP even in the case when the variable ordering is unknown *a priori*. Section 4 shows that the BTP defines a tractable class that properly includes several other known tractable classes. Section 5 gives an alternative characterization of instances which have the BTP, while Section 6 defines a non-conservative generalization of the BTP. Variable elimination by means of the BTP and its extension are discussed in Sections 7 and 8. Finally, in Section 9 we discuss the possible use of the BTP in variable-ordering heuristics and prove the intractability of finding a maximum subset of the variables on which a CSP instance has the BTP.

2. The broken-triangle property

In this paper we focus on binary constraint satisfaction problems. A ***binary relation*** over domains D_i and D_j is a subset of $D_i \times D_j$. For a binary relation R, the reverse relation $\text{rev}(R)$ is defined as $\{(v,u) \mid (u,v) \in R\}$.

A binary CSP ***instance*** consists of a set of ***variables*** (where each variable is denoted by a number $i \in \{1,\ldots,n\}$); for each variable i, a ***domain*** D_i containing possible ***values*** for variable i; and a set of ***constraints***. Each constraint is of the form $\langle (i,j), R\rangle$, where i and j are variables, the pair (i,j) is called the ***scope*** of the constraint, and R is a relation such that $R \subseteq D_i \times D_j$, specifying the allowed combinations of values for the variables in the scope.

To simplify notation we introduce the notion of a ***canonical constraint relation*** which combines all of the specified information about a pair of variables i, j. In the following definition, we use the notation $\bigcap S$, where S is a set of relations, to denote the intersection of all the relations in S.

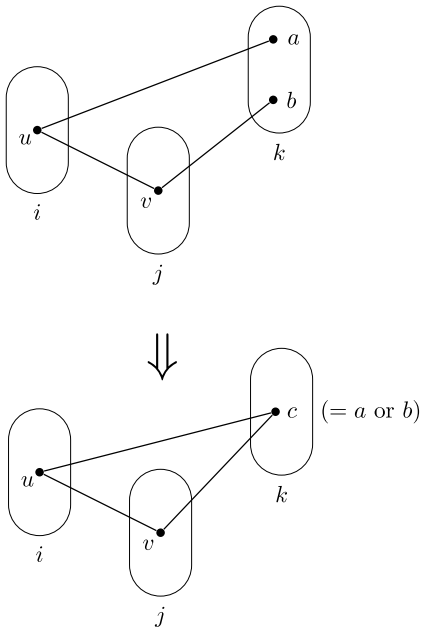

Fig. 1. The broken-triangle property on variables i, j, k.

Definition 2.1. Suppose i and j are variables of a CSP instance. Denote by U_{ij} the set of constraint relations specified for the (ordered) pair of variables (i, j). The ***canonical constraint relation*** between variables i and j will be denoted R_{ij} and is defined as

$$R_{ij} = \bigcap \big(U_{ij} \cup \{ \mathrm{rev}(R) \mid R \in U_{ji} \} \big).$$

The canonical constraint relation R_{ij} contains precisely the pairs of values that are allowed for the variables i and j by all the constraints on i and j. Note that $R_{ij} = \mathrm{rev}(R_{ji})$. If there are no specified constraints on the pair of variables i and j, then all pairs of values are allowed, so R_{ij} is defined to be the ***complete*** relation $D_i \times D_j$. (This can be viewed as defining the value of an empty intersection in Definition 2.1 to be $D_i \times D_j$.)

Throughout this paper, whenever we consider a binary CSP instance, we will use the notation R_{ij} to refer to the canonical constraint relation between variables i and j.

Definition 2.2. A binary CSP instance satisfies the ***broken-triangle property*** (***BTP***) with respect to the variable ordering $<$, if, for all triples of variables i, j, k such that $i < j < k$, if $(u, v) \in R_{ij}$, $(u, a) \in R_{ik}$ and $(v, b) \in R_{jk}$, then either $(u, b) \in R_{ik}$ or $(v, a) \in R_{jk}$.

The broken-triangle property can be represented in diagrammatic form by the implication shown in Fig. 1. In the figure, each vertex represents an assignment of a value to a variable, and for each variable an oval is drawn containing all its possible values. Each line represents a consistent assignment of values for a pair of variables. A line joins element $u \in D_i$ and element $v \in D_j$ if $(u, v) \in R_{ij}$. The BTP on i, j, k simply says that for any "broken-triangle" $a - u - v - b$, as illustrated in Fig. 1, there is always a true triangle $u - v - c$ (where c is either equal to a or b).

It is important to note that the BTP must be satisfied for *all* triples $i < j < k$, even if the description of the instance does not specify a constraint between all pairs of these variables. As noted above, if there is no specified constraint between i and j, then R_{ij} allows all pairs of values.

We remark that the definition of the BTP is similar to the standard definition of *directional path consistency* [2], but a little stronger. To be directional path consistent requires that for all triples of variables i, j, k such that $i < j < k$, if $(u, v) \in R_{ij}$, then there exists *some* $c \in D_k$ such that $(u, c) \in R_{ik}$ and $(v, c) \in R_{jk}$ (but the implication shown in Fig. 1 may not be satisfied, because c is not necessarily equal to a or b).

We also remark that the definition of the BTP is similar to the definition of *hyper-3-consistency* given in [21], but a little weaker. To be hyper-3-consistent requires that for all triples of variables i, j, k, if $(u, v) \in R_{ij}$, $(u, a) \in R_{ik}$ and $(v, b) \in R_{jk}$, then *both* $(u, b) \in R_{ik}$ and $(v, a) \in R_{jk}$.

A simple reformulation of Definition 2.2 shows that the BTP is equivalent to forbidding the existence of subproblems of a certain form.

Lemma 2.3. *A binary CSP instance satisfies the broken-triangle property with respect to the variable ordering $<$, if, and only if, it does not contain an ordered triple of variables i, j, k with $i < j < k$, and values $a, b \in D_k$, $u \in D_i$ and $v \in D_j$, such that $(u, v) \in R_{i,j}$, $(u, a) \in R_{ik}$, $(v, b) \in R_{jk}$, $(u, b) \notin R_{ik}$ and $(v, a) \notin R_{jk}$.*

For an element $a \in D_i$, we write $R_{ij}(a)$ to represent $\{b \in D_j : (a, b) \in R_{ij}\}$, the ***image*** of a in the relation R_{ij}. The next result shows that the BTP is equivalent to having certain inclusion relations between certain image sets.

Lemma 2.4. *A binary CSP instance satisfies the broken-triangle property with respect to the variable ordering $<$ if, and only if, for all triples of variables $i < j < k$, and for all $(u, v) \in R_{ij}$,*

$$\big(R_{ik}(u) \subseteq R_{jk}(v)\big) \vee \big(R_{jk}(v) \subseteq R_{ik}(u)\big). \tag{1}$$

Proof. The condition that either $R_{ik}(u) \subseteq R_{jk}(v)$ or $R_{jk}(v) \subseteq R_{ik}(u)$ is equivalent to stating that there do not exist elements a of $R_{ik}(u)$ and b of $R_{jk}(v)$ such that $a \notin R_{jk}(v)$ and $b \notin R_{ik}(u)$. By the definition of the image of an element in a relation, this in turn is equivalent to the statement that there do not exist $a, b \in D_k$ such that $(u, a) \in R_{ik}$, $(v, b) \in R_{jk}$, $(u, b) \notin R_{ik}$ and $(v, a) \notin R_{jk}$. Hence the result follows by Lemma 2.3. □

Definition 2.5. A class of CSP instances is called ***conservative*** if it is closed under domain restrictions (i.e., the addition of arbitrary unary constraints). A property is called conservative if it defines a conservative class of instances.

Lemma 2.6. *The broken-triangle property with respect to any fixed variable ordering is conservative.*

Proof. Lemma 2.3 states that the broken-triangle property holds for a binary CSP instance if and only if it does not contain certain subproblems. Removing values from the domain of any variable in an instance cannot create new subproblems, and hence cannot cause the broken-triangle property to become false. □

This result implies that the broken-triangle property is invariant under domain reduction operations such as arc consistency: if a binary CSP instance satisfies the broken-triangle property, then so does its closure under arc consistency. Indeed, any pre-processing operation which only performs domain reductions, such as arc consistency, path-inverse consistency [22], or neighbourhood substitution [23,24], can be applied before looking for a variable ordering for which the broken-triangle property is satisfied; these reduction operations cannot destroy the broken-triangle property, but they can make it more likely to hold.

A set of CSP instances may satisfy the broken-triangle property due to the structure of the constraint graph, due to the language of the constraint relations, or due to a combination of these.

Example 2.7. Let I be a binary CSP instance whose constraint relations are all *zero/one/all* (ZOA) relations, as defined in [25]. Such relations have the property that, for each pair of variables i, j, each $u \in D_i$ is compatible with either zero, one or all possible values for j. ZOA relations notably include all relations specified by 2SAT clauses. An important property of instances where all constraint relations are ZOA is that this remains true after establishing arc consistency and path consistency. Therefore we may assume without loss of generality that I is both arc consistent and path consistent. By arc consistency, and the definition of ZOA relations, $R_{ij}(u)$ is either a singleton or equal to D_j, whenever i and j are distinct variables and $u \in D_i$. The BTP could then only be violated for $u \in D_i$, $v \in D_j$ if $R_{ik}(u)$, $R_{jk}(v)$ were both singletons with $R_{ik}(u) \neq R_{jk}(v)$. However, if this were the case, then $\langle u, v\rangle$ would have been deleted from R_{ij} by path consistency. Therefore I satisfies the BTP.

Further examples of some broad classes of CSP instances which satisfy the BTP are discussed in Section 4 (see Theorem 4.8).

3. Tractability of BTP instances

In this section we show that if a CSP instance has the broken-triangle property with respect to some fixed variable ordering, then finding a solution is tractable. Moreover, we show that the problem of finding a suitable ordering if it exists is also tractable.

For a binary CSP instance with n variables, let $d = \max\{|D_1|, \ldots, |D_n|\}$ and let e be the number of constraints. Note that an assignment of values $\langle u_1, \ldots, u_k\rangle$ to the first k variables of a binary CSP instance is called ***consistent*** if $u_i \in D_i$ whenever $1 \leqslant i \leqslant k$, and $(u_i, u_j) \in R_{ij}$ whenever $1 \leqslant i < j \leqslant k$ [26].

Theorem 3.1. *For any binary CSP instance which satisfies the BTP with respect to some known variable ordering $<$, it is possible to find a solution in $O(d^2 e)$ time (or determine that no solution exists).*

Proof. By the discussion in Section 2, if an instance has the BTP with respect to $<$, then establishing arc consistency preserves the BTP. Furthermore, it is known that arc consistency can be established in $O(d^2e)$ time [27]. If this results in an empty domain, then the instance has no solutions. Therefore, we assume in the following that the CSP instance is arc consistent and has non-empty domains.

We can assign some value $u_1 \in D_1$ to the first variable, since D_1 is non-empty. To prove the result it is sufficient to show, for all $k = 2, \ldots, n$, that any consistent assignment $\langle u_1, \ldots, u_{k-1}\rangle$ for the first $k-1$ variables can be extended to a consistent assignment $\langle u_1, \ldots, u_k\rangle$ for the first k variables. The case $k = 2$ follows from arc consistency.

By Lemma 2.4, if $i < j < k$ then either $R_{ik}(u_i) \subseteq R_{jk}(u_j)$ or $R_{jk}(u_j) \subseteq R_{ik}(u_i)$. For any fixed $k > 2$, we can therefore apply Lemma 2.4 to every pair of variables occurring before k in the ordering to establish that the set $\{R_{ik}(u_i) \mid i < k\}$ is totally ordered by subset inclusion, and hence has a minimal element

$$R_{i_0k}(u_{i_0}) = \bigcap_{i<k} R_{ik}(u_i) \tag{2}$$

for some $i_0 < k$. Since the instance is arc consistent, $R_{i_0k}(u_{i_0}) \neq \emptyset$. By the definition of $R_{ik}(u_i)$, it follows from Eq. (2) that $\langle u_1, \ldots, u_k\rangle$ is a consistent assignment for the first k variables, for any choice of $u_k \in R_{i_0k}(u_{i_0})$.

The time taken to calculate the intersections in (2) is at most $O(ed)$ overall, since each value in the domain must be checked against each relevant constraint. □

Theorem 3.2. *Given a binary CSP instance I, there is a polynomial-time algorithm to find a variable ordering $<$, such that I satisfies the broken-triangle property with respect to $<$ (or to determine that no such ordering exists).*

Proof. Given a binary CSP instance I, we define an associated CSP instance P_I that has a solution precisely when there exists a suitable variable ordering for I.

To construct P_I, let $O_1, \ldots, O_n$ be variables taking values in $\{1, \ldots, n\}$ representing positions in the ordering. We impose the ternary constraint

$$O_k < \max\{O_i, O_j\} \tag{3}$$

for all triples of variables i, j, k in I such that the broken-triangle property fails to hold for some $u \in D_i$, $v \in D_j$, and $a, b \in D_k$ when the variables are ordered $i < j < k$. The instance P_I then has a solution precisely if there is an ordering of the variables $1, \ldots, n$ of I which satisfies the broken-triangle property. Note that if the solution obtained represents a partial order (for instance, if O_i and O_j are assigned the same value for some $i \neq j$), then it can be extended to a total order which still satisfies all the constraints by using a linear-time topological sort.

For each triple of variables in I, the construction of the corresponding constraints in P_I requires $O(d^4)$ steps to check which constraints to add. There are $O(n^3)$ such triples, so constructing instance P_I takes $O(n^3d^4)$ steps, which is polynomial in the size of I.

The constraints in P_I are all of the form (3), and such constraints have the property that they are ***max-closed***[1] [28]. Max-closed constraints are a tractable constraint language [28]: any CSP instance with max-closed constraints can be solved by establishing generalized arc consistency [29] and then choosing the maximum element which remains in each variable domain. Since the size of P_I is polynomial in the size of I, it follows that the instance P_I can be solved in time polynomial in the size of I. □

In Section 7 we describe an alternative, more efficient, approach to finding a suitable variable ordering, or determining that no such ordering exists (see Corollary 7.5).

4. Related tractable classes

In this section we will show that the broken-triangle property generalizes several other known tractable classes, some of which have unbounded tree-width.

Definition 4.1. A binary relation R_{ij} on the sets D_i and D_j, where D_j is totally ordered, is said to be ***right monotone*** if $\forall a \in D_i$, $\forall b, c \in D_j$,

$$(a, b) \in R_{ij} \wedge b < c \quad \Rightarrow \quad (a, c) \in R_{ij}.$$

A commonly-used right monotone constraint is the inequality constraint: "value of variable $i \leqslant$ value of variable j". The complete relation, which allows every combination of values, is also right monotone.

[1] To verify that a constraint is max-closed we need to show that if we take any 2 tuples which satisfy the constraint, and then compute the maximum of the 2 values in each co-ordinate position of these tuples, then we always obtain a tuple that satisfies the constraint [28]. To verify that this holds for all the constraints in P', we simply note that if $p_1 < \max\{q_1, r_1\}$ and $p_2 < \max\{q_2, r_2\}$ then $\max(p_1, p_2) < \max\{\max(q_1, q_2), \max(r_1, r_2)\}$.

Definition 4.2. A binary CSP instance is ***renamable right monotone*** with respect to a variable ordering $<$ if each set D_k, for $k \in \{2, \ldots, n\}$, can be ordered (separately), such that R_{ik} is right monotone for every $i < k$.

Proposition 4.3. *If a binary CSP instance is renamable right monotone with respect to a variable ordering $<$, then it satisfies the broken-triangle property with respect to $<$.*

Proof. Suppose the CSP instance is renamable right monotone with respect to variable ordering $<$, and let k be any variable. Since the instance is renamable right monotone with respect to $<$, there is an ordering of D_k such that whenever $i < k$ then R_{ik} is right monotone. Now suppose $i < j < k$ are variables in this ordering. Then each of R_{ik} and R_{jk} is right monotone. Now suppose that $(u, v) \in R_{ij}$, $(u, a) \in R_{ik}$ and $(v, b) \in R_{jk}$. If $a < b$, then $(u, b) \in R_{ik}$ (since R_{ik} is right monotone); if $a > b$, then $(v, a) \in R_{jk}$ (since R_{jk} is right monotone). Hence, by Definition 2.2, the broken-triangle property is satisfied for the triple i, j, k. Since the choice of k was arbitrary, it follows that the instance satisfies the BTP. □

Using Lemma 2.4 we can obtain another simple sufficient condition for the broken-triangle property to hold.

Lemma 4.4. *A binary CSP instance satisfies the broken-triangle property with respect to a variable ordering $<$ if, for all triples of variables $i < j < k$, either R_{ik} or R_{jk} is a complete relation.*

Proof. If R_{ik} is a complete relation, then $R_{ik}(u) = D_k$, while if R_{jk} is a complete relation, then $R_{jk}(v) = D_k$. In either case, by Lemma 2.4, the instance satisfies the BTP. □

Proposition 4.5. *If a binary CSP instance has a tree structure, then it satisfies the broken-triangle property with respect to any variable ordering in which each node occurs before its children.*

Proof. If we order the nodes of a tree from any designated root to the leaves, then each node is connected to at most one node earlier in the ordering. Hence, if a CSP instance has a tree structure, then any variable ordering $<$ of this kind has the property that, for each variable k, there is at most one variable i, with $i < k$, such that R_{ik} is not a complete relation. Hence, by Lemma 4.4, the instance satisfies the BTP with respect to any such ordering. □

To obtain a further class of examples we note that with any CSP instance P (of arbitrary arity) we can associate a binary CSP instance I_P which is called the ***dual*** of P [30]: the instance I_P has a variable for each constraint of P, and a binary constraint between each pair of variables associated with overlapping constraints of P. The binary constraints in I_P are defined to ensure that the variables in I_P can only take values which agree on the corresponding shared variables of P (see [30] for details).

Lemma 4.6. *Let P be a CSP instance (of arbitrary arity) with constraint scopes $E_1, E_2, \ldots, E_m$, where the constraints allow all combinations of values from some fixed domain D. The dual instance I_P, with corresponding variables $1, 2, \ldots, m$, has the BTP with respect to some ordering $<$ if, and only if, for all triples E_i, E_j, E_k with $i < j < k$ we have*

$$(E_i \cap E_k \subseteq E_j \cap E_k) \ \vee \ (E_j \cap E_k \subseteq E_i \cap E_k). \tag{4}$$

Moreover, if this condition holds, then the dual of any instance P' with the same constraint scopes also has the BTP with respect to $<$.

Proof. By Lemma 2.4, I_P has the BTP with respect to $<$ if and only if for all triples $i < j < k$, and for all $(u, v) \in R_{ij}$, $(R_{ik}(u) \subseteq R_{jk}(v)) \vee (R_{jk}(v) \subseteq R_{ik}(u))$. But, by the definition of the dual [30], for any fixed pair u, v, the range $R_{ik}(u)$ is determined purely by the overlap of E_i and E_k, and the range $R_{jk}(v)$ is determined purely by the overlap of E_j and E_k. Hence, I_P has the BTP with respect to $<$ if and only if the given condition holds on $E_i \cap E_k$ and $E_j \cap E_k$.

Finally, we note that the dual of any other instance P' with the same constraint scopes can be obtained from the dual of such an instance P, for an appropriate choice of D, by imposing restrictions on the domains. Hence, by Lemma 2.6, the dual of P' will also have the BTP with respect to $<$. □

The edges of any tree can be ordered so that they satisfy the condition given in Lemma 4.6, so we obtain the following corollary of this result.

Corollary 4.7. *The dual of any tree-structured binary CSP instance has the BTP with respect to some ordering.*

Now let ***TREE*** be the constraint satisfaction problem consisting of all instances that have tree structure, ***RRM*** be the CSP consisting of all instances that are renamable right monotone with respect to some variable ordering, and ***DUAL-TREE*** be the CSP consisting of all instances which are duals of instances with tree structure. Note that these three classes are incomparable, for example an instance consisting of a single equality constraint belongs to TREE and DUAL-TREE but not to

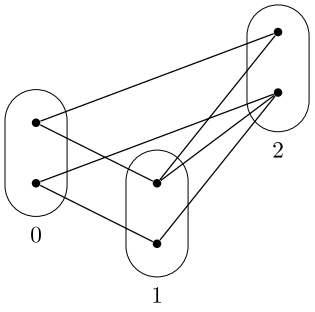

Fig. 2. An instance in BTP that is not in RRM, TREE or DUAL-TREE.

RRM. Moreover, both RRM and DUAL-TREE contain instances of arbitrary tree-width; for example, RRM contains instances where the constraint structure is a grid of arbitrary size, and DUAL-TREE contains instances where the constraint structure is a clique of arbitrary size (such as the dual of any instance whose structure is a star). Finally, let ***BTP*** be the CSP consisting of all instances which have the broken-triangle property with respect to some variable ordering. We now show that the class BTP properly includes the other 3 classes.

Theorem 4.8. *RRM* $\subsetneq$ *BTP, TREE* $\subsetneq$ *BTP, and DUAL-TREE* $\subsetneq$ *BTP.*

Proof. The inclusions follow from Propositions 4.3, 4.5 and Corollary 4.7; the instance shown in Fig. 2 establishes the strict separations. □

5. An alternative characterization of the BTP

In this section we consider the broad class of properties which are both conservative and preserved by taking subproblems. We show that the broken-triangle property is the *only* such property which ensures that the following desirable behaviour can be guaranteed simply by achieving a certain level of arc consistency:

Definition 5.1. A CSP instance is ***universally backtrack-free*** with respect to an ordering $<$ of its n variables if $\forall k \in \{2, \ldots, n\}$, any consistent assignment for the first $k-1$ variables can be extended to a consistent assignment for the first k variables.

Definition 5.2. Given a CSP instance I on variables $1, \ldots, n$, the induced subproblem $I(\{i_1, \ldots, i_m\})$, where $1 \leqslant i_1 < i_2 < \cdots < i_m \leqslant n$, is the m-variable CSP instance with domains $D_{i_1}, \ldots, D_{i_m}$ and exactly those constraints of I whose scopes are subsets of $\{i_1, \ldots, i_m\}$.

Definition 5.3. A set Σ of CSP instances is ***inclusion-closed*** if $\forall I \in \Sigma$, all subproblems $I(M)$ on subsets M of the variables of I also belong to Σ.

Definition 5.4. (See [2].) A binary CSP instance is ***directional arc consistent*** with respect to a variable ordering $<$, if for all pairs of variables $i < j$, for every a in D_i, there exists some $b \in D_j$ such that $(a, b) \in R_{ij}$.

Given any binary CSP instance I, we can remove values from the domains of the variables to achieve directional arc consistency, without changing the set of solutions [2]. The result of applying such an algorithm is unique, and is called the directional arc-consistency closure of I.

Proposition 5.5. *Let* Σ *be a conservative inclusion-closed set of binary CSP instances. The directional arc-consistency closure* DAC(I) *of every* $I \in \Sigma$ *with respect to a variable ordering* $<$ *is universally backtrack-free with respect to* $<$ *if, and only if,* $\forall I \in \Sigma$, DAC(I) *satisfies the broken-triangle property with respect to* $<$.

Proof. The argument used in the proof of Theorem 3.1 shows that if any binary CSP instance satisfies the broken-triangle property then its directional arc-consistency closure is universally backtrack-free.

To prove the converse, suppose that Σ is a conservative inclusion-closed set of CSP instances and consider any $I \in \Sigma$. Since Σ is conservative, DAC(I) also belongs to Σ, since it is obtained from I by a sequence of domain reductions. In the following, we let D_i denote the domain of variable i in DAC(I). Consider three variables $i < j < k$ and four domain values $u \in D_i$, $v \in D_j$, $a, b \in D_k$ such that $(u, v) \in R_{ij}$, $(u, a) \in R_{ik}$ and $(v, b) \in R_{jk}$. Denote by I' the induced subproblem of DAC(I) on variables i, j, k and with reduced domain $\{a, b\}$ for variable k. Establishing directional arc consistency in I' may reduce the domains of variables i and j, but cannot delete v from the domain of variable j (since it has a support, namely b,

at k) nor can it delete u from the domain of variable i (since it has supports at variables j and k). If DAC(I') is universally backtrack-free, then the consistent assignment $\langle u, v\rangle$ for the variables $\langle i, j\rangle$ can be extended to a consistent assignment for $\langle i, j, k\rangle$, which must be either $\langle u, v, a\rangle$ or $\langle u, v, b\rangle$. This corresponds exactly to the definition of the broken-triangle property, and so DAC(I) satisfies the BTP. □

6. Generalizing the BTP

In this section we show that a weaker form of the broken-triangle property also implies backtrack-free search. This leads to a strictly larger, but non-conservative, tractable class of CSP instances. Throughout this section, we assume that all variable domains are totally ordered.

Definition 6.1. A binary CSP instance is ***min-of-max extendable*** **(*MME*)** with respect to the variable ordering $<$, if for all triples of variables i, j, k such that $i < j < k$, if $(u, v) \in R_{ij}$, then $\langle u, v, c\rangle$ is a consistent assignment for $\langle i, j, k\rangle$, where

$$c = \min\big(\max\big(R_{ik}(u)\big), \max\big(R_{jk}(v)\big)\big).$$

The symmetrically equivalent property max-of-min extendability is defined similarly, but with $c = \max(\min(R_{ik}(u)), \min(R_{jk}(v)))$.

Lemma 6.2. *A binary CSP instance satisfies the broken-triangle property with respect to a variable ordering $<$ if, and only if, it is min-of-max extendable with respect to $<$ for all possible domain orderings.*

Proof. Suppose that a CSP instance satisfies the broken-triangle property with respect to $<$, and consider an arbitrary ordering of each of the domains. To prove min-of-max extendability, it suffices to apply the broken-triangle property to $a = \max(R_{ik}(u))$ and $b = \max(R_{jk}(v))$. If $a = b$ then clearly $\langle u, v, \min(a, b)\rangle$ is a consistent assignment for $\langle i, j, k\rangle$. Otherwise, since a and b are maximal, $\langle u, v, \max(a, b)\rangle$ is not a consistent assignment; hence by the BTP, $\langle u, v, \min(a, b)\rangle$ is a consistent assignment.

To prove the converse, suppose that a CSP instance is min-of-max extendable for all possible domain orderings. For any distinct $a, b \in D_k$, consider an ordering of D_k for which $a = \max(D_k)$ and $b = \max(D_k - \{a\})$. The broken-triangle property then follows from the definition of min-of-max extendability. □

Theorem 6.3. *If a binary CSP instance is min-of-max extendable with respect to some known variable ordering $<$ and some (possibly unknown) domain orderings, and is also directional arc consistent with respect to $<$, then it is universally backtrack-free with respect to $<$, and hence can be solved in polynomial time.*

Proof. Suppose that $\langle u_1, \ldots, u_{k-1}\rangle$ is a consistent assignment for the variables $\langle 1, \ldots, k-1\rangle$. By directional arc consistency, $\forall i < k$, $R_{ik}(u_i) \neq \emptyset$. This means that

$$c = \min\big\{\max\big(R_{ik}(u_i)\big)\colon\ 1 \leqslant i \leqslant k-1\big\}$$

is well-defined. Let $j \in \{1, \ldots, k-1\}$ be such that $c = \max(R_{jk}(u_j))$. Let i be any variable in $\{1, \ldots, k-1\} - \{j\}$. Applying the definition of min-of-max extendability to variables i, j, k allows us to deduce that $(u_i, c) \in R_{ik}$. It follows that $\exists u_k \in D_k$ (namely $u_k = c$) such that $\langle u_1, \ldots, u_k\rangle$ is a consistent assignment for the variables $\langle 1, \ldots, k\rangle$. This establishes that the instance is universally backtrack-free.

Note that we used the ordering of domain D_k only to prove the existence of at least one consistent extension $\langle u_1, \ldots, u_k\rangle$ of $\langle u_1, \ldots, u_{k-1}\rangle$. A search algorithm can find some consistent value for u_k, without any information about the domain orderings, simply by checking each possible value from D_k. Hence the instance can be solved in polynomial time, even when the domain orderings are unknown. □

Theorem 6.4. *The problem of finding a variable ordering for a binary CSP instance with ordered domains such that it is min-of-max extendable with respect to that ordering (or determining that no such ordering exists) is solvable in polynomial time.*

Proof. The requirements for the ordering are a subset of the requirements for establishing the broken-triangle property. Hence the result can be proved exactly as in the proof of Theorem 3.2. □

We can use Theorem 6.4 in the following way: given a CSP instance with ordered domains, compute its arc consistency closure, and then test (in polynomial time) whether this reduced instance is min-of-max extendable for some ordering of its variables. If we find such an ordering, then the instance can be solved in polynomial-time, by Theorem 6.3.

However, this approach is not guaranteed to find all possible useful variable orderings achieving min-of-max extendability, because min-of-max extendability is not conservative, as the following example shows.

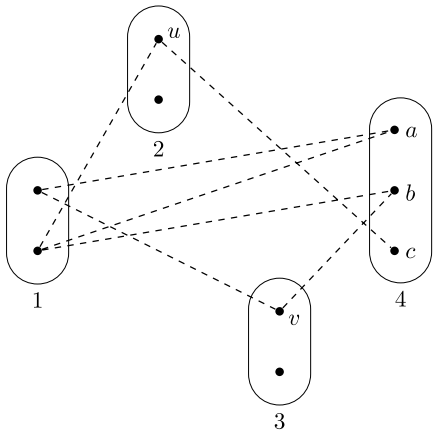

Fig. 3. An example of a CSP instance which is directional arc consistent and min-of-max extendable, but which fails to be min-of-max extendable when made arc consistent.

Example 6.5. Fig. 3 shows a binary CSP instance in which a dashed line between values $p \in D_i$ and $q \in D_j$ represents the fact that $\langle p, q\rangle \notin R_{ij}$. All pairs of values not joined by a dashed line are consistent. This instance is directional arc consistent and min-of-max extendable with respect to the variable ordering $1 < 2 < 3 < 4$ and the ordering $a > b > c$ of domain D_4 (and any orderings of the other domains). However, this instance does not satisfy the BTP (consider the "broken triangle" $b - u - v - c$).

If we establish arc consistency, the value a is deleted from D_4 since it has no support in D_1. The resulting CSP instance is arc consistent but no longer min-of-max extendable, since $\langle u, v\rangle \in R_{23}$ but cannot be extended to a consistent assignment for variables $2, 3, 4$.

Since min-of-max extendability is not a conservative property, it can be the case that, for some variable orderings, the directional arc-consistency closure is min-of-max extendable but the full arc-consistency closure is not, as in Example 6.5.

Finally, we show that min-of-max extendability is a strict generalization of a previously-identified hybrid tractable class.

Definition 6.6. (See [31].) A CSP instance is ***row-convex*** (with respect to a fixed variable ordering and fixed domain orderings) if for all pairs of variables $i < j$, $\forall u \in D_i$, $R_{ij}(u)$ is the interval $[a, b]$ for some $a, b \in D_j$.

It is known that a directional path-consistent row-convex binary CSP instance is universally backtrack-free and hence tractable [31]. (However, it should be noted that establishing directional path consistency may destroy row-convexity.) Our interest in this hybrid tractable class is simply to demonstrate that it is a special case of min-of-max extendability.

Proposition 6.7. *If a binary CSP instance is directional path-consistent and row-convex, then it is min-of-max extendable* (*and max-of-min extendable*).

Proof. Consider the triple of variables $i < j < k$ and suppose that $(u, v) \in R_{ij}$. By directional path consistency, $\exists c \in D_k$ such that $(u, c) \in R_{ik}$ and $(v, c) \in R_{jk}$. By row-convexity, $R_{ik}(u)$ and $R_{jk}(v)$ are intervals in the ordered domain D_k. The existence of c means that these intervals overlap. Both end-points of this overlap provide extensions of (u, v) to a consistent assignment for the variables $\langle i, j, k\rangle$. One end-point is given by $\min(\max(R_{ik}(u)), \max(R_{jk}(v)))$ which ensures min-of-max extendability. (The other ensures max-of-min extendability.) □

On the other hand, it is perfectly possible for a binary CSP instance to be min-of-max extendable without being row-convex, as shown by Example 6.5 (consider $R_{34}(v)$).

7. Variable elimination using the BTP

Classical techniques for reducing the search space of a CSP are based on domain reduction via value elimination. In this section we will show that the BTP can be used to define a novel strategy for *variable* elimination in binary CSPs. We will investigate the time complexity of applying this variable elimination strategy, and explore the interaction between variable and value eliminations.

In contrast to bucket elimination, which requires space exponential in the tree-width of the constraint network [32], our scheme for variable elimination using the BTP can be applied in polynomial time even to classes of CSP instances with unbounded tree-width.

Existing techniques for choosing good variable orderings have focused on variants of the smallest domain heuristic [33], probabilistic estimates of the most likely assignments [34], and dynamic variable orderings together with a local search criterion [35]. Our scheme could be used as part of input preprocessing, or even during search, together with the usual enforcement of arc or path consistency. Existing variable ordering heuristics can then be applied during search after some variables have been eliminated.

We first define a *local* broken-triangle property for a single variable k.

Definition 7.1. A variable k in a binary CSP instance satisfies the ***local broken-triangle property* (*lBTP*)** if for all pairs of distinct variables $i, j \neq k$, $\forall a, b \in D_k$, $\langle u, v\rangle \in R_{ij}$, $\langle u, a\rangle \in R_{ik}$ and $\langle v, b\rangle \in R_{jk}$ implies that $\langle u, b\rangle \in R_{ik}$ or $\langle v, a\rangle \in R_{jk}$.

Let $[n]^{-k}$ denote the set $\{1, \ldots, n\} \setminus \{k\} = \{1, \ldots, k-1, k+1, \ldots, n\}$.

Proposition 7.2. *Let I be an arc-consistent binary CSP instance on variables $[n]$ such that variable k satisfies the lBTP, and let I' be the induced subproblem of I on variables $[n]^{-k}$. Then I' has a solution if and only if I has a solution. Indeed, the set of solutions of I' is exactly the projection of the set of solutions of I onto the variables $[n]^{-k}$.*

Proof. Since I' is a subproblem of I, we only need to prove that every solution of I' can be extended to a solution of I. Let $\langle a_1, \ldots, a_{k-1}, a_{k+1}, \ldots, a_n\rangle$ be a solution of I'. By arc consistency, $\forall i \neq k$, $R_{ik}(a_i) \neq \emptyset$. Since k satisfies the lBTP, by the same argument as in the proof of Lemma 2.4, any two elements of $\{R_{ik}(a_i) \mid i \in [n]^{-k}\}$ are ordered by subset inclusion, so this set is totally ordered by subset inclusion, and hence has a non-empty intersection. This implies that every solution of I' can be extended to a solution of I, which completes the proof. □

Proposition 7.2 tells us that the solvability of any CSP instance is invariant under the elimination of any variables that satisfy the lBTP.

The following proposition gives a concrete example of the application of this form of variable elimination.

Proposition 7.3. *In a path consistent binary CSP instance, all variables k such that $|D_k| = 2$ satisfy the lBTP and hence can be eliminated.*

Proof. Consider a variable k such that $|D_k| = 2$. Suppose for a contradiction that for some $i, j \neq k$, we have $\langle u, v\rangle \in R_{ij}$, $\langle u, a\rangle \in R_{ik}$, $\langle v, b\rangle \in R_{jk}$, $\langle u, b\rangle \notin R_{ik}$ and $\langle v, a\rangle \notin R_{jk}$. Then $a \neq b$, and we can deduce that $D_k = \{a, b\}$. But then $\langle u, v\rangle \in R_{ij}$ has no support at variable k, which contradicts the assumption of path consistency. □

Theorem 7.4. *The closure of a binary CSP instance under the elimination of all variables that satisfy the lBTP is unique and can be found in $O(ned^3)$ time.*

Proof. By Definition 7.1, if variable k satisfies the lBTP in a binary CSP instance I, then k satisfies the lBTP in any subproblem of I that includes k. Thus, if variables j and k both satisfy the lBTP, then variable j will still satisfy the lBTP after elimination of variable k. It follows that the result of eliminating all variables that satisfy the lBTP is unique, since a variable elimination cannot invalidate another variable elimination. However, as with value eliminations by arc consistency, variable eliminations can provoke new variable eliminations; if variable i did not satisfy the lBTP before elimination of k, it may satisfy the lBTP after the elimination of k.

A variable k satisfies the lBTP and can be eliminated if, for all distinct $i, j \in [n]^{-k}$, $(u, v) \in R_{ij}$ implies that

$$R_{ik}(u) \subseteq R_{jk}(v) \ \vee \ R_{ik}(u) \supseteq R_{jk}(v). \tag{5}$$

If for some $u \in D_i$ and some $v \in D_j$, Eq. (5) does not hold, then we say that $\{i, j\}$ is an ***lBTP obstruction-pair*** for k. Let Ob_k be the set of lBTP obstruction-pairs $\{i, j\}$ for k. We can assume that the set $R_{ik}(u)$ (for each $i \in [n]^{-k}$ and each $u \in D_i$) is stored in a direct access data structure, so that the test $a \in R_{ik}(u)$ is an $O(1)$ operation. Furthermore, Eq. (5) trivially holds if there is no constraint with scope $\{i, k\}$ or no constraint with scope $\{j, k\}$. It follows that Ob_k can be determined by checking whether Eq. (5) holds for each pair of constraints whose scopes contain k, and each possible choice of the values u and v, by checking each possible $a \in D_k$. This can be completed in $O(e_k^2 d^3)$ time, where d is the maximum domain size and e_k is the number of binary constraints whose scope contains k.

Hence, the sets Ob_k (for all $k \in [n]$) can be calculated in $O(ned^3)$ time since $\sum_{k=1}^{n} e_k^2 d^3 \leqslant \sum_{k=1}^{n} ne_k d^3 = 2ned^3$.

If any set Ob_k is empty, then variable k satisfies the lBTP and can be eliminated. When a variable k is eliminated, the sets Ob_i ($i \neq k$) must be updated by deleting all obstruction-pairs containing k. If any other Ob_i now becomes empty, then variable i can also be eliminated, and the same updating procedure must be applied again. Since each obstruction-pair $\{i, j\}$ is deleted at most once from each Ob_k, the total number of deletions is bounded above by $\sum_{k=1}^{n} |Ob_k| \leqslant \sum_{k=1}^{n} e_k^2 \leqslant \sum_{k=1}^{n} ne_k = 2ne$. Therefore, using appropriate indexing, the total time complexity to determine the variable elimination closure by lBTP is $O(ned^3 + ne) = O(ned^3)$. □

Corollary 7.5. *The problem of finding a variable ordering for which a binary CSP instance I satisfies the BTP (or determining that no such order exists) can be solved in $O(ned^3)$ time.*

Proof. To determine whether there exists a variable ordering for which I satisfies the BTP, we can simply test whether its closure under the elimination of all variables that satisfy the lBTP is empty. If so, then the reverse of the order in which variables were eliminated is a variable ordering for which I satisfies the BTP. Otherwise, no such ordering exists. □

Theorem 7.6. *If the binary CSP instance I satisfies the BTP for some possibly unknown variable order $\prec$, then MAC (Maintaining Arc Consistency during search) solves I in $O(ned^3)$ time whatever the instantiation order $<$ of the variables during search.*

Proof. Since I satisfies the BTP, and the BTP is a conservative property, any restriction I' of I produced by domain reductions (whether the result of instantiation of variables or arc consistency operations) also satisfies the BTP. By the proof of Theorem 3.1, any arc consistent CSP instance that satisfies the BTP either has a solution or an empty domain. It follows that MAC will be backtrack-free when applied to I, provided an assignment of a value to a variable is only accepted if the resulting arc-consistency closure has non-empty domains. In the worst case, for each variable i, MAC will have to try all values in D_i. This makes a total of $O(nd)$ times that arc consistency will have to be established. The complexity of $O(ned^3)$ follows assuming that MAC uses an $O(ed^2)$ arc consistency algorithm such as AC-2001 [27]. □

Corollary 7.5 shows that the class of binary CSP instances satisfying the BTP for some possibly unknown variable ordering can be recognized in $O(ned^3)$ time. Combining this with Theorem 7.6, it is possible to detect whether a binary CSP instance I satisfies the BTP, and if the result is positive, to actually solve I, in $O(ned^3)$ time. The advantage of this approach, rather than simply running MAC, is that we know in advance (in polynomial time) whether the BTP is satisfied and hence that search will terminate in polynomial time.

In general, we would expect that some but not all variables can be eliminated using lBTP. An important question is how variable elimination by lBTP interacts with classical value elimination techniques. Eliminating values (for example, by arc consistency, path inverse consistency or neighbourhood substitution) cannot invalidate a variable elimination by lBTP but can provoke new variable eliminations by lBTP. Similarly, a variable elimination by lBTP cannot invalidate value eliminations by any of these techniques but can provoke a new value elimination by neighbourhood substitution. However, a variable elimination by lBTP cannot destroy any form of consistency, and hence cannot provoke new value eliminations by consistency operations. It is known that eliminating values by any convergent sequence of neighbourhood substitution or consistency operations produces a CSP instance which is unique up to isomorphism [24]. It follows from the above discussion that a similar result holds for any convergent sequence of variable eliminations by lBTP and value eliminations by neighbourhood substitution or consistency operations.

8. Variable elimination using min-of-max extendability

As with the BTP, min-of-max extendability can also be used to eliminate variables. The important difference is that min-of-max extendability is defined relative to an ordering of the domains. In this section, we first assume that orderings are given for all domains, which allows us to generalize in a straightforward way the results of Section 7. Finally, we study the case in which suitable domain orderings must be calculated.

We first define a local form of min-of-max extendability for a single variable k.

Definition 8.1. A variable k in a binary CSP instance is ***locally min-of-max extendable* (*lMME*)** for some ordering of D_k, if for all pairs of distinct variables $i, j \neq k$, $(u, v) \in R_{ij}$ implies that $\langle u, v, c\rangle$ is a consistent assignment for $\langle i, j, k\rangle$, where

$$c = \min\bigl(\max\bigl(R_{ik}(u)\bigr), \max\bigl(R_{jk}(v)\bigr)\bigr).$$

Proposition 8.2. *Let I be an arc-consistent binary CSP instance on variables $\{1, \ldots, n\}$ such that variable k is lMME for some ordering of D_k, and let I' be the induced subproblem of I on variables $[n]^{-k}$. Then I' has a solution if and only if I has a solution. Indeed, the set of solutions to I' is exactly the projection of the set of solutions to I onto variables $[n]^{-k}$.*

Proof. Since I' is a subproblem of I we only need to prove that every solution of I' can be extended to a solution of I. Let $\langle a_1, \ldots, a_{k-1}, a_{k+1}, \ldots, a_n\rangle$ be a consistent labelling of $[n]^{-k}$, i.e., a solution of I'. By arc consistency, $\forall i \neq k$, $R_{ik}(a_i) \neq \emptyset$. Let $c = \min\{\max(R_{ik}(a_i))\colon\ i \in [n]^{-k}\}$ and suppose that $c = \max(R_{i_0k}(a_{i_0}))$ where $i_0 \in [n]^{-k}$. Since variable k is lMME, $\forall j \notin \{i_0, k\}$, $\langle a_{i_0}, a_j, c\rangle$ is a consistent assignment for $\langle i_0, j, k\rangle$. Hence $\langle a_1, \ldots, a_{k-1}, c, a_{k+1}, \ldots, a_n\rangle$ is a consistent assignment for variables $\{1, \ldots, n\}$, i.e., a solution of I. □

An important, although obvious, property of any form of variable elimination is that it preserves arc consistency, since eliminating a variable cannot invalidate arc consistency.

Theorem 8.3. *Given orderings of all domains, the closure of an arc-consistent binary CSP under the elimination of all variables that are lMME is unique and can be found in $O(ned^2)$ time.*

Proof. By Definition 8.1, if variables j and k are both lMME, then variable j will still be lMME after elimination of variable k. It follows that the result of eliminating all variables that are lMME is unique, and can be obtained by eliminating such variables (in any order) until convergence.

Call $\{i, j\}$ an ***lMME obstruction-pair*** for k if $\exists(u, v) \in R_{ij}$ such that $\langle u, v, c\rangle$, where $c = \min(\max(R_{ik}(u)), \max(R_{jk}(v)))$, is not a consistent assignment for $\langle i, j, k\rangle$. The maximum value in each $R_{ij}(u)$, for all i, j, u, can be precomputed in $O(ed^2)$ time. Using these precomputed values we can calculate Ob_k, the set of lMME obstruction-pairs for k, in $O(e_k^2 d^2)$ time, where e_k is the number of binary constraints whose scope includes k. Summing over all variables k, we obtain a complexity of $O(\sum_{k=1}^{n} e_k d^2) = O(ned^2)$.

As in the proof of Theorem 7.4, we require $O(ne)$ additional time to propagate deletions of variables. Hence the total time complexity to calculate the closure under eliminations of variables that are lMME is $O(ed^2 + ned^2 + ne) = O(ned^2)$. □

If a variable satisfies the lBTP, then it is lMME (but the converse is not always true). It follows that eliminating variables that are lMME is both stronger and (asymptotically) faster than eliminating variables that satisfy the lBTP.

Corollary 8.4. *For given domain orderings, the problem of finding a variable ordering for which an arc-consistent binary CSP instance is min-of-max extendable (or determining that no such variable ordering exists) can be solved in $O(ned^2)$ time.*

Proof. Similar to the proof of Corollary 7.5. An arc-consistent binary CSP instance is min-of-max extendable if, and only if, each of its variables can be eliminated in turn (in some order) because they are lMME. □

For a given ordering of D_k, we can clearly test whether the variable k is lMME in polynomial time. We call such an ordering of the values of k a **min-of-max ordering**. If $|D_k|$ is bounded by a constant, then we can test all possible orderings of D_k in polynomial time. An obvious question is, in the case that $|D_k|$ is not bounded by a constant, whether determining the existence of a min-of-max ordering of D_k is tractable or not. We complete our study of min-of-max extendability by showing below that this problem is, in fact, NP-complete.

MIN-OF-MAX ORDERING (MMO)	
Input:	a binary CSP instance I and a variable k
Question:	does there exist a min-of-max ordering of D_k in I?

Theorem 8.5. *The MMO problem for the class of CSP instances with finite but unbounded domain size is NP-complete.*

Proof. MMO $\in$ NP since checking that k is lMME for a given ordering of D_k is polynomial-time. To prove NP-completeness we will construct a polynomial reduction from SAT to MMO. Consider an instance I_{SAT} of SAT with n variables $v_1, \ldots, v_n$. We will construct a 3-variable binary CSP instance I in which $D_3 = \{1, \ldots, 2n+2\}$ and for which there exists a min-of-max ordering of D_k if, and only if, I_{SAT} is satisfiable.

Let $D_1 = D_2$ and place an equality constraint on variables $\langle 1, 2\rangle$. For each $i \in \{1, \ldots, n\}$, we add distinct values a_i, b_i, c_i to $D_1 = D_2$ and tuples to the constraints with scopes $\langle 1, 3\rangle$, $\langle 2, 3\rangle$ such that $R_{13}(a_i) = \{i, 2n+1\}$, $R_{23}(a_i) = \{n+i, 2n+1\}$, $R_{13}(b_i) = \{i, 2n+2\}$, $R_{23}(b_i) = \{n+i, 2n+2\}$, $R_{13}(c_i) = \{i, n+i, 2n+1\}$, $R_{23}(c_i) = \{i, n+i, 2n+2\}$ (for $i \in \{1, \ldots, n\}$). This is illustrated in Fig. 4. This construction ensures that, in any min-of-max ordering $\prec$ of D_3, for each $i \in \{1, \ldots, n\}$, exactly one of i, $n+i$ is less than both of $2n+1$, $2n+2$: the assignment $\langle a_i, a_i\rangle$ (respectively $\langle b_i, b_i\rangle$) to variables $\langle 1, 2\rangle$ ensures that $\min(i, n+i)$ is less than $2n+1$ (respectively $2n+2$), while the assignment $\langle c_i, c_i\rangle$ to variables $\langle 1, 2\rangle$ ensures that at most one of i, $n+i$ is less than both of $2n+1$, $2n+2$.

For any min-of-max ordering $\prec$ of D_3, we associate the truth value `false` with v_i precisely when $i \prec 2n+1$ and $i \prec 2n+2$. Clearly, $n+i$ can be associated with $\neg v_i$, since by the construction in Fig. 4 exactly one of i, $n+i$ is less than both of $2n+1$, $2n+2$. For each clause $C_j = v_{i_1} \vee \cdots \vee v_{i_r}$ in I_{SAT}, we add a distinct value d_j to $D_1 = D_2$ and tuples to the constraints with scopes $\langle 1, 3\rangle$, $\langle 2, 3\rangle$ such that $R_{13}(d_j) = \{i_1, \ldots, i_r, 2n+1\}$, $R_{23}(d_j) = \{i_1, \ldots, i_r, 2n+2\}$. This is illustrated in Fig. 4 for the clause $v_{i_1} \vee v_{i_2} \vee v_{i_3}$. In any min-of-max ordering of D_3, not all of $i_1, \ldots, i_r$ can be less than both $2n+1$ and $2n+2$; this corresponds to imposing the constraint that one of the variables $v_{i_1}, \ldots, v_{i_r}$ is true. (If the clause involves a negated variable $\neg v_i$, then we replace the value i by $n+i$ in $R_{13}(d_j)$ and $R_{23}(d_j)$.) It follows that there is a min-of-max ordering of D_3 if, and only if, I_{SAT} is satisfiable. This reduction can clearly be completed in polynomial time. □

9. Variable ordering heuristics based on the BTP

Even when it is not possible to eliminate any variable from an instance I using the lBTP, as described in Section 7, it may still be true that some induced *subproblem* of I, on some subset of variables S, satisfies the BTP for some ordering of

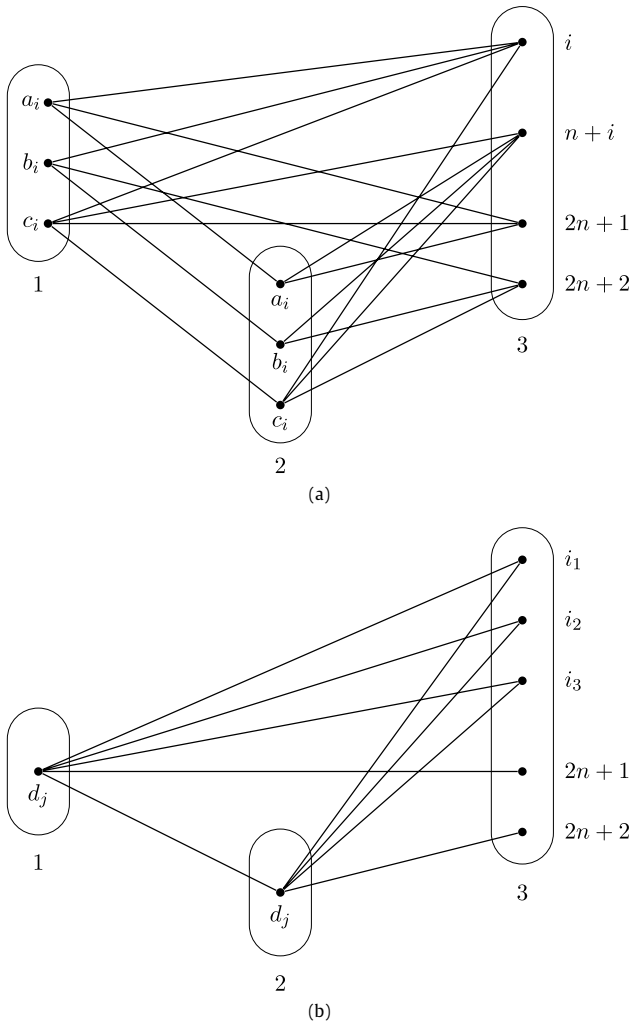

Fig. 4. (a) The construction which codes $v_{n+i} = \neg v_i$, since it implies that exactly one of the values i, $n+i$ is less than both of $2n+1$, $2n+2$ in an MMEP ordering of D_3. (b) The construction which codes the clause $v_{i_1} \vee v_{i_2} \vee v_{i_3}$ since it implies that not all of i_1, i_2, i_3 are less than both of $2n+1$, $2n+2$ in an MMEP ordering of D_3.

its variables. In this case, by ordering the whole set of variables X of I so that the variables in S all occur after the variables in $X - S$, the subproblem on variables S obtained after the instantiation of the variables in $X - S$ satisfies the BTP, since the BTP is conservative. In this case, we know by Theorem 7.6 that MAC will be backtrack-free on the variables S. This implies that the maximum number of leaf nodes visited in the MAC search tree will be $d^{n-|S|}$. (Note that the same argument does not apply to min-of-max extendability, as it is not conservative.)

Hence, to limit the size of the search tree explored by the MAC algorithm, it is desirable to find a large set of variables S on which the BTP is satisfied. Once we have found such a set of variables S, then standard variable-ordering heuristics [33–35] can be used to order the variables in $X - S$.

Unfortunately, finding the largest possible set of variables on which the BTP is satisfied is an NP-hard problem, as we now show.

Theorem 9.1. *Given a binary CSP instance I on variables X, it is* NP*-hard to determine the maximum size of a subset $S \subseteq X$ such that the induced subproblem of I on S satisfies the BTP.*

Proof. The problem MAX-ONES is the problem of finding a solution to a problem SAT which maximizes the number of boolean variables assigned the value `true`. It is known [36] that MAX-ONES is NP-hard (in fact, APX-complete) even when restricted to instances whose clauses are all of the form $\phi(x_i, x_j, x_k)$, where $\phi(x, y, z) = \neg x \vee \neg y \vee \neg z$. We will show that any instance I_1 of MAX-ONES whose clauses are all of this form can be expressed as a problem of finding the maximum size of a subset of variables of a binary CSP instance I_2 such that the induced subproblem of I_2 on S satisfies the BTP.

We say that there is a ***BTP obstruction*** (i, j, k, a, u, v, b) with respect to variable ordering $<$ if there exist $i < j < k$ such that $u \in D_i$, $v \in D_j$, $a, b \in D_k$ and $(u, a) \in R_{ik}$, $(u, v) \in R_{ij}$, $(v, b) \in R_{jk}$, $(u, b) \notin R_{ik}$, and $(v, a) \notin R_{jk}$.

For each variable x_i in I_1, we create a variable i in I_2. For each clause $\phi(x_i, x_j, x_k)$ in I_1, we create three BTP obstructions which imply $O_i < \max(O_j, O_k)$, $O_j < \max(O_k, O_i)$ and $O_k < \max(O_i, O_j)$. For example, creating a BTP obstruction which implies $O_k < \max(O_i, O_j)$ means adding values a, b to D_k, u to D_i and $v \in D_j$ satisfying $(u, v) \in R_{ij}$, $(u, a) \in R_{ik}$, $(v, b) \in R_{jk}$, $(u, b) \notin R_{ik}$ and $(v, a) \notin R_{jk}$. The inequality constraints implied by the three BTP obstructions are inconsistent, but any subset of two of these constraints is consistent. Therefore at most two of i, j, k can belong to S. Associating $x_i = \texttt{true}$ with $i \in S$, these three BTP obstructions therefore impose the constraint $\phi(x_i, x_j, x_k)$ on the corresponding variables in I_1. This reduction is polynomial-time (and in fact log-space), which demonstrates that finding the maximum induced subproblem of I_2 which satisfies the BTP is NP-hard. □

10. Conclusion

We have described new tractable classes of binary CSP instances which significantly generalize tree-structured problems as well as previously-identified language-based classes. These new classes are obtained by imposing requirements on the sets of constraints imposed on all ordered triples of variables. Moreover, we have shown that the problem of determining a variable ordering for which these properties hold is solvable in polynomial time.

We see this work as a first step towards a complete characterization of all hybrid tractable classes of constraint satisfaction problems, that is, all tractable classes which are obtained by restricting the combined properties of both the constraint relations and the way in which they interact.

From a practical point of view, we have shown that the properties which define the tractable classes introduced in this paper can also be used to eliminate variables in binary CSPs. This provides us with a reduction technique which is orthogonal and complementary to classical value elimination techniques such as arc consistency.

Furthermore, one of the local properties (the BTP) can be used to improve variable-ordering heuristics by instantiating last a subset of variables on which the BTP holds. An interesting area for future research would be to investigate variable ordering heuristics which try to minimize the number of times the BTP is not satisfied.

Acknowledgements

We would like to thank Chris Jefferson for pointing out Theorem 7.6.

References

[1] E.C. Freuder, A sufficient condition for backtrack-free search, Journal of the ACM 29 (1) (1982) 24–32, doi:10.1145/322290.322292.
[2] R. Dechter, J. Pearl, Network-based heuristics for constraint-satisfaction problems, Artificial Intelligence 34 (1) (1987) 1–38, doi:10.1016/0004-3702(87)90002-6.
[3] A. Bulatov, P. Jeavons, A. Krokhin, Classifying the complexity of constraints using finite algebras, SIAM Journal on Computing 34 (3) (2005) 720–742, doi:10.1137/S0097539700376676.
[4] T. Feder, M.Y. Vardi, The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory, SIAM Journal on Computing 28 (1) (1998) 57–104, doi:10.1137/S0097539794266766. Available from: http://link.aip.org/link/?SMJ/28/57/1.
[5] D. Cohen, P. Jeavons, M. Gyssens, A unified theory of structural tractability for constraint satisfaction problems, Journal of Computer and System Sciences 74 (5) (2008) 721–743, doi:10.1016/j.jcss.2007.08.001.
[6] P. Jeavons, On the algebraic structure of combinatorial problems, Theoretical Computer Science 200 (1–2) (1998) 185–204, doi:10.1016/S0304-3975(97)00230-2.
[7] A.A. Bulatov, Tractable conservative constraint satisfaction problems, in: Proceedings of 18th IEEE Symposium on Logic in Computer Science (LICS 2003), Ottawa, Canada, 22–25 June 2003, IEEE Computer Society, 2003, pp. 321–330. Available from: http://csdl.computer.org/comp/proceedings/lics/2003/1884/00/18840321abs.htm.
[8] T.J. Schaefer, The complexity of satisfiability problems, in: STOC '78: Proceedings of the Tenth Annual ACM Symposium on Theory of Computing, 1978, pp. 216–226.
[9] A.A. Bulatov, A dichotomy theorem for constraint satisfaction problems on a 3-element set, Journal of the ACM 53 (1) (2006) 66–120, doi:10.1145/1120582.1120584.
[10] M. Grohe, The structure of tractable constraint satisfaction problems, in: Proceedings of the 31st Symposium on Mathematical Foundations of Computer Science, in: Lecture Notes in Computer Science, vol. 4162, Springer-Verlag, 2006, pp. 58–72.
[11] V. Dalmau, P.G. Kolaitis, M.Y. Vardi, Constraint satisfaction, bounded treewidth, and finite-variable logics, in: CP '02: Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming, in: Lecture Notes in Computer Science, vol. 2470, Springer-Verlag, 2002, pp. 310–326.
[12] M. Grohe, The complexity of homomorphism and constraint satisfaction problems seen from the other side, Journal of the ACM 54 (1) (2007) 1–24, doi:10.1145/1206035.1206036.
[13] R. Dechter, Constraint Processing, Morgan Kaufmann, 2003.
[14] D. Cohen, P. Jeavons, The complexity of constraint languages, in: F. Rossi, P. van Beek, T. Walsh (Eds.), Handbook of Constraint Programming, Elsevier, 2006, pp. 245–280.
[15] D.A. Cohen, M.J. Green, Typed guarded decompositions for constraint satisfaction, in: F. Benhamou (Ed.), CP '06: Proceedings of the 12th International Conference on Principles and Practice of Constraint Programming, in: Lecture Notes in Computer Science, vol. 4204, Springer-Verlag, 2006, pp. 122–136.
[16] T.K. Kumar, A framework for hybrid tractability results in boolean weighted constraint satisfaction problems, in: CP '08: Proceedings of the 14th International Conference on Principles and Practice of Constraint Programming, in: Lecture Notes in Computer Science, vol. 5202, Springer-Verlag, 2008, pp. 282–297.
[17] E.C. Freuder, A sufficient condition for backtrack-bounded search, Journal of the ACM 32 (4) (1985) 755–761, doi:10.1145/4221.4225.

[18] H.L. Bodlaender, A linear-time algorithm for finding tree-decompositions of small treewidth, SIAM Journal on Computing 25 (6) (1996) 1305–1317, doi:10.1137/S0097539793251219. Available from: http://link.aip.org/link/?SMJ/25/1305/1.
[19] J. Pearson, P. Jeavons, A survey of tractable constraint satisfaction problems, Tech. Rep. CSD-TR-97-15, Royal Holloway, University of London, July 1997. Available from: ftp://ftp.cs.rhul.ac.uk/pub/constraints/survey.ps.
[20] P. van Beek, R. Dechter, Constraint tightness and looseness versus local and global consistency, Journal of the ACM 44 (4) (1997) 549–566, doi:10.1145/263867.263499.
[21] P. Jégou, On the consistency of general constraint-satisfaction problems, in: AAAI, 1993, pp. 114–119. Available from: http://www.aaai.org/Library/AAAI/1993/aaai93-018.pdf.
[22] E.C. Freuder, C.D. Elfe, Neighborhood inverse consistency preprocessing, in: Proc. AAAI/IAAI-96, Portland, OR, vol. 1, 1996, pp. 202–208. Available from: http://www.aaai.org/Library/AAAI/1996/aaai96-030.php.
[23] E.C. Freuder, Eliminating interchangeable values in constraint satisfaction problems, in: Proc. AAAI-91, Anaheim, CA, 1997, pp. 227–233. Available from: http://www.aaai.org/Library/AAAI/1991/aaai91-036.php.
[24] M.C. Cooper, Fundamental properties of neighbourhood substitution in constraint satisfaction problems, Artificial Intelligence 90 (1–2) (1997) 1–24, doi:10.1016/S0004-3702(96)00018-5.
[25] M. Cooper, D. Cohen, P. Jeavons, Characterising tractable constraints, Artificial Intelligence 65 (1994) 347–361, doi:10.1016/0004-3702(94)90021-3.
[26] C. Bessiere, Constraint propagation, in: F. Rossi, P. van Beek, T. Walsh (Eds.), Handbook of Constraint Programming, Elsevier, 2006, pp. 29–83.
[27] C. Bessière, J.-C. Régin, Refining the basic constraint propagation algorithm, in: Proc IJCAI'01, Seattle, WA, 2001, pp. 309–315. Available from: http://www.lirmm.fr/%7Ebessiere/stock/ijcai01.ps.
[28] P.G. Jeavons, M.C. Cooper, Tractable constraints on ordered domains, Artificial Intelligence 79 (2) (1995) 327–339, doi:10.1016/0004-3702(95)00107-7.
[29] R. Mohr, G. Masini, Good old discrete relaxation, in: Y. Kodratoff (Ed.), Proceedings of the 8th European Conference on Artificial Intelligence—ECAI'88, Pitman, 1988, pp. 651–656.
[30] R. Dechter, J. Pearl, Tree clustering for constraint networks (research note), Artificial Intelligence 38 (3) (1989) 353–366, http://dx.doi.org/10.1016/0004-3702(89)90037-4.
[31] P. van Beek, R. Dechter, On the minimality and decomposability of row-convex constraint networks, Journal of the ACM 42 (3) (1995) 543–561, doi:10.1145/210346.210347.
[32] R. Dechter, Bucket elimination: A unifying framework for reasoning, Artificial Intelligence 113 (1–2) (1999) 41–85, doi:10.1016/S0004-3702(99)00059-4.
[33] B.M. Smith, The Brélaz heuristic and optimal static orderings, in: CP '99: Proceedings of the 5th International Conference on Principles and Practice of Constraint Programming, in: Lecture Notes in Computer Science, vol. 1713, Springer-Verlag, 1999, pp. 405–418. Available from: http://www.springerlink.com/content/3dgkv9kgvc4h340w/.
[34] J.C. Beck, P. Prosser, R.J. Wallace, Toward understanding variable ordering heuristics for constraint satisfaction problems, in: Proceedings of the Fourteenth Irish Artificial Intelligence and Cognitive Science Conference, 2003, pp. 11–16. Available from: http://tidel.mie.utoronto.ca/pubs/promise.aics.pdf.
[35] C. Bessière, A. Chmeiss, L. Saïs, Neighborhood-based variable ordering heuristics for the constraint satisfaction problem, in: CP '01: Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming, in: Lecture Notes in Computer Science, vol. 2239, Springer-Verlag, 2001, pp. 565–569.
[36] N. Creignou, S. Khanna, M. Sudan, Complexity Classification of Boolean Constraint Satisfaction Problems, SIAM Monographs on Discrete Mathematics and Applications, vol. 7, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2001.