ELSEVIER

Artificial Intelligence 79 (1995) 327–339

Artificial Intelligence

Research Note

Tractable constraints on ordered domains

Peter G. Jeavons [a,*], Martin C. Cooper [b]

[a] *Department of Computer Science, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK*

[b] *IRIT, University of Toulouse III, Toulouse, France*

Received December 1994; revised October 1995

Abstract

Finding solutions to a constraint satisfaction problem is known to be an NP-complete problem in general, but may be tractable in cases where either the set of allowed constraints or the graph structure is restricted. In this paper we identify a restricted set of contraints which gives rise to a class of tractable problems. This class generalizes the notion of a Horn formula in propositional logic to larger domain sizes. We give a polynomial time algorithm for solving such problems, and prove that the class of problems generated by *any* larger set of constraints is NP-complete.

1. Introduction

Combinatorial problems abound in artificial intelligence. Examples include planning, temporal reasoning, line-drawing labelling and circuit design. The Constraint Satisfaction Problem (CSP) [14] is a generic combinatorial problem which is widely studied in the AI community because it allows all of these problems to be expressed in a natural and direct way. Reduction operations [10, 12] and intelligent search methods [4, 18] developed for the CSP have been applied with success to many different practical problems.

Finding solutions to a constraint satisfaction problem is known to be an NP-complete problem in general [12] even when the constraints are restricted to binary constraints. However, many of the problems which arise in practice have special properties which allow them to be solved efficiently.

The question of identifying restrictions to the general problem which are sufficient to ensure tractability has been discussed by a number of authors. Such restrictions may either involve the structure of the constraints, in other words which variables may be

* Corresponding author. E-mail: pete@dcs.rhbnc.ac.uk.

SSDI 0004-3702(95)00107-7

constrained by which other variables, or they may involve the nature of the constraints, in other words which combinations of values may be allowed for variables which are mutually constrained. Examples of the first approach may be found in [4, 6, 8, 14, 15] (see [3] for a survey) and examples of the second approach may be found in [1, 11, 14, 19–21].

In this paper we take the second approach, and carefully consider what restrictions must be imposed on the nature of the constraints in order to ensure that all the problems involving those constraints can be solved efficiently. We identify a particular class of constraints, $\mathcal{M}$, which has the following properties:

(1) Any constraint satisfaction problem with the constraints chosen from $\mathcal{M}$ can be solved in polynomial time. (Section 4.)

(2) When the domain of each variable in the problem is {False, True}, $\mathcal{M}$ is precisely the set of constraints which are definable by conjunctions of Horn clauses. (Section 5.)

(3) For any constraint C not in $\mathcal{M}$, the class of problems generated by $\mathcal{M} \cup \{C\}$ is NP-complete. (Section 6.)

In summary, this paper identifies a generalization of the notion of Horn formulae to larger domain sizes, which provides a new maximal class of tractable problems. The relationship between this new class and earlier tractable classes which have been identified in the literature is described in Section 3.

2. Definitions

Definition 2.1. A *constraint satisfaction problem*, P, consists of

- a finite set of variables, N, identified by the natural numbers $1, 2, \ldots, n$,
- a domain of values, D,
- a list of constraints $(C(S_1), C(S_2), \ldots, C(S_e))$; each S_i is an ordered subset of the variables, and each constraint $C(S_i)$ is a set of tuples indicating the mutually consistent values for the variables in S_i.

A *solution* to a constraint satisfaction problem is an assignment of values to the variables which is consistent with all of the constraints.

The length of the tuples in a given constraint will be called the "arity" of that constraint. In particular, unary constraints specify the allowed values for a single variable, and binary constraints specify the allowed combinations of values for a pair of variables.

It is convenient to make use of the following operations from relational algebra [13].

Definition 2.2. Let S be any ordered set of r variables and let $C(S)$ be a constraint on S.

For any ordered subset $S' \subseteq S$, let $(i_1, i_2, \ldots, i_k)$ be the indices of the elements of S' in S. Define the *projection* of $C(S)$ onto S', denoted $\pi_{S'}(C(S))$, as follows

$$\pi_{S'}(C(S)) = \{(x_{i_1}, x_{i_2}, \ldots, x_{i_k}) \mid \exists (x_1, x_2, \ldots, x_r) \in C(S)\}.$$

Definition 2.3. For any constraints $C(S_1)$ and $C(S_2)$, the *join* of $C(S_1)$ and $C(S_2)$, denoted $C(S_1) \bowtie C(S_2)$ is the constraint on $S_1 \cup S_2$ containing all tuples t such that $\pi_{S_1}(\{t\}) \subseteq C(S_1)$ and $\pi_{S_2}(\{t\}) \subseteq C(S_2)$.

We shall assume, for simplicity, that each variable is subject to at least one constraint. Hence, the set of all solutions to a constraint satisfaction problem P, denoted $\mathrm{Sol}(P)$, is simply the join of all the constraints [8]

$$\mathrm{Sol}(P) = C(S_1) \bowtie C(S_2) \bowtie \cdots \bowtie C(S_c).$$

The decision problem for a constraint satisfaction problem is to determine whether or not this join is non-empty.

The class of problems in which the constraints all belong to some class $\mathcal{C}$ will be denoted $\mathrm{CSP}(\mathcal{C})$. In an earlier paper [1] we characterize precisely the classes of constraints $\mathcal{C}$ which give rise to tractable problem classes $\mathrm{CSP}(\mathcal{C})$, in those cases where the constraint set $\mathcal{C}$ is closed under permutations of the domain D.

In what follows, we shall assume that the domain D is a totally-ordered set. This assumption is not unreasonable, since in many applications the domain may be considered to be a subset of the natural numbers, or the real numbers. By making use of this additional ordering property of the domain we are able to define new sets of constraints which give rise to tractable problems.

As a consequence of assuming that the domain is ordered, we may define the following operation on the elements of any constraint.

Definition 2.4. Let C be a constraint and let $t = (x_1, x_2, \ldots, x_r)$ and $t' = (x'_1, x'_2, \ldots, x'_r)$ be elements of C.

The *maximum* of t and t', denoted $t \sqcup t'$ is defined as follows:

$$t \sqcup t' = (\max(x_1, x'_1), \max(x_2, x'_2), \ldots, \max(x_r, x'_r)).$$

The *minimum* of t and t', denoted $t \sqcap t'$ is defined as follows:

$$t \sqcap t' = (\min(x_1, x'_1), \min(x_2, x'_2), \ldots, \min(x_r, x'_r)).$$

Using these operations on tuples, we now define the following property of constraints.

Definition 2.5. A constraint C is said to be *max-closed* if, for all $t, t' \in C$,

$$t \sqcup t' \in C.$$

Similarly, C is said to be *min-closed* if, for all $t, t' \in C$,

$$t \sqcap t' \in C.$$

Lemma 2.6. *All unary constraints are max-closed.*

Proof. Since the domain D is assumed to be totally ordered, we know that for any unary constraint C, and any $(x), (x') \in C$, $(x) \sqcup (x') = (\max(x, x')) = (x)$ or (x'), so $(x) \sqcup (x') \in C$. Hence, C is max-closed. □

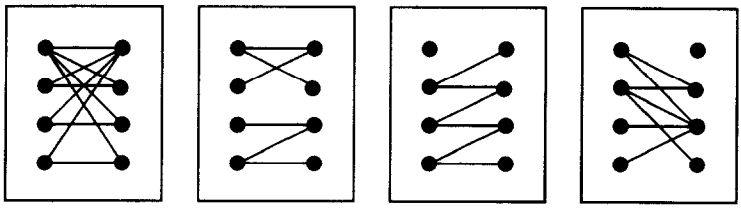

Fig. 1. Examples of binary max-closed constraints.

Example 2.7. Fig. 1 shows some examples of binary max-closed constraints. In this figure the constraints are shown diagrammatically by drawing a single point for each possible value for one variable on the left, and a single point for each possible value for the other variable on the right, and then connecting two points if the corresponding combination of values is allowed by the constraint. (The values are drawn in ascending order from bottom to top in the figure.)

Example 2.8. The constraint programming language CHIP incorporates a number of constraint solving techniques for arithmetic and other constraints. In particular, it provides a constraint solver for a restricted class of constraints over natural numbers, referred to as *basic constraints* [21]. These basic constraints are of two kinds, which are referred to as "domain constraints" and "arithmetic constraints". The domain constraints described in [21] are unary constraints which restrict the value of a variable to some specified finite subset of the natural numbers. The arithmetic constraints described in [21] are binary constraints which have one of the following forms:

$$aX \neq b,$$
$$aX = bY + c,$$
$$aX \leqslant bY + c,$$
$$aX \geqslant bY + c,$$

where variables are represented by upper-case letters, and constants by lower case letters, all constants are positive and a is non-zero.

All of these constraints are max-closed (and also min-closed). Hence, any system of constraints of this restricted type can be solved efficiently using the algorithm described below.

Other (non-binary) arithmetic constraints which are also max-closed, and could therefore be added to this set without losing this property, include

$$a_1X_1 + a_2X_2 + \cdots + a_rX_r \geqslant bY + c,$$
$$aX_1X_2 \cdots X_r \geqslant bY + c,$$
$$(a_1X_1 \geqslant b_1) \vee (a_2X_2 \geqslant b_2) \vee (a_rX_r \geqslant b_r) \vee (aY \leqslant b) \quad \text{(see Theorem 5.2)}.$$

Further important examples of non-binary max-closed constraints will be discussed in Section 5.

3. Related work

Van Beek [19] introduced the notion of "row-convex" constraints, which can be defined as follows:

Definition 3.1. A binary constraint C over an ordered domain D is said to be *row-convex* if, for all w, x, y, z in D,

$$[((w,x) \in C) \wedge ((w,z) \in C) \wedge (x \leqslant y \leqslant z)] \Rightarrow ((w,y) \in C).$$

Row-convex constraints include all "functional" constraints (which are binary constraints where each value for either variable is compatible with at most one value for the other variable), and all monotonic constraints, which are defined as follows:

Definition 3.2. A binary constraint C over domain D is said to be *monotonic* if, for all w, x, y, z in D,

$$[((x,z) \in C) \wedge (w \geqslant x) \wedge (y \leqslant z)] \Rightarrow ((w,y) \in C).$$

It is shown in [19] that any constraint satisfaction problem which is path-consistent and contains only row-convex constraints can be solved in polynomial time.

In general, however, the class of problems CSP(C) may be NP-complete, even when C contains only row-convex constraints, since the transposition of a row-convex constraint is not necessarily row-convex, and the composition of two row-convex constraints is not necessarily row-convex.

The class of 0/1/all constraints introduced in [1] (and independently in [11]) is a subclass of the row-convex constraints (which includes all functional constraints). It is shown in [1] that CSP(C) is solvable in polynomial time whenever C contains only 0/1/all constraints. It is also shown that CSP(C) is NP-complete whenever C contains all 0/1/all constraints over some fixed domain together with *any* constraint which is not 0/1/all.

The max-closed constraints introduced in this paper are *not* a subclass of the row-convex constraints, even when restricted to binary constraints, as the following example illustrates.

Example 3.3. Consider the binary constraint C over domain $D = \{a, b, c, d\}$ defined by

$$C = \{(a,a), (a,b), (a,c), (a,d), (b,a), (b,b), (c,a), (c,c), (d,a), (d,d)\}.$$

This constraint corresponds to the first constraint shown in Fig. 1 (Example 2.7).

If the domain is ordered such that $a > b > c > d$, then C is max-closed but *not* row-convex. Furthermore, it cannot be made row-convex by reordering the domain values in any way.

Note also that problems containing only binary max-closed constraints can be solved by establishing arc-consistency, as indicated by Theorem 4.2 below, whereas problems

involving row-convex constraints require path-consistency in general [19].

On the other hand, there is considerable overlap between the two classes of constraints, as the following result indicates.

Proposition 3.4.

- *Any binary constraint which is* both *max-closed and min-closed is row-convex.*
- *Any binary constraint which is monotonic is both max-closed and min-closed.*

The binary constraints described in Example 2.8 are all both max-closed and min-closed, hence they are also row-convex.

The converses of the statements in Proposition 3.4 do not hold. For example, the second binary constraint shown in Fig. 1 is row-convex and max-closed but not min-closed. The third binary constraint shown in Fig. 1 is both max-closed and min-closed but not monotonic.

4. Solving problems with max-closed constraints

The class of all max-closed constraints over some fixed domain D will be denoted $\mathcal{M}_D$. The following properties of $\mathcal{M}_D$ follow directly from the above definitions.

Proposition 4.1.

- *$\mathcal{M}_D$ is closed under the join operation. In other words, for any pair of constraints $C(S_1), C(S_2) \in \mathcal{M}_D$,*

$$C(S_1) \bowtie C(S_2) \in \mathcal{M}_D.$$

- *$\mathcal{M}_D$ is closed under projection. In other words, for any constraint $C(S) \in \mathcal{M}_D$ and any subset $S' \subseteq S$,*

$$\pi_{S'}(C(S)) \in \mathcal{M}_D.$$

The next result is a generalization of Theorem 35 from [21]. A constraint satisfaction problem is said to be "pair-wise consistent" [9, 13] if for any pair of constraints $C(S_1)$ and $C(S_2)$, $\pi_{S_1 \cap S_2}(C(S_1)) = \pi_{S_1 \cap S_2}(C(S_2))$.

Theorem 4.2. *Any $P \in \mathrm{CSP}(\mathcal{M}_D)$ which is pair-wise consistent either has a solution or has an empty constraint.*

Proof. Assume that P has no empty constraints. For each variable i, let x_i be the maximum value allowed for that variable by some constraint $C(S_j)$ such that $i \in S_j$. We claim that $(x_1, x_2, \ldots, x_n)$ is a solution to P.

To establish this claim, consider any constraint $C(S)$ of P, where $S = (i_1, i_2, \ldots, i_r)$, and any $i_j \in S$. By the choice of the x_i, and pair-wise consistency, we must have some tuple $t_{i_j} \in C(S)$ whose jth coordinate is x_{i_j}. Since $C(S)$ is max-closed, the maximum of all these tuples must belong to $C(S)$, but again by the choice of x_i, this must be $(x_{i_1}, x_{i_2}, \ldots, x_{i_r})$. Hence $(x_1, x_2, \ldots, x_n)$ satisfies $C(S)$, and the claim follows. □

Corollary 4.3. *The time complexity of solving any* $P \in \mathrm{CSP}(\mathcal{M}_D)$ *which has* c *constraints of maximum cardinality* t *is* $O(c^2t^2)$.

Proof. Pair-wise consistency may be achieved in $O(c^2t^2)$ time by using an associated binary representation [9].

Since the resulting problem may also be obtained by performing a succession of join and projection operations, it is still an element of $\mathrm{CSP}(\mathcal{M}_D)$ by Proposition 4.1. Hence, by Theorem 4.2 it either has an empty constraint or a solution.

This solution may be obtained in $O(ct)$ time, by inspecting each remaining tuple in each constraint and recording the maximum value allowed for each variable. □

Equivalent results clearly also hold for min-closed constraints.

For a given problem instance over a domain D which is *not* in $\mathrm{CSP}(\mathcal{M}_D)$, it will sometimes be possible to reorder D in such a way that all of the constraints become max-closed. To check whether a particular reordering achieves this, it is sufficient to check every pair of tuples in every constraint, to ensure that Definition 2.5 is satisfied. The time required to check an ordering is therefore $O(ct^2)$, and for any fixed domain size, the number of possible reorderings is constant.

There exists an even broader class of problem instances in which it is possible to choose a separate ordering of the domain at each variable in such a way that all of the constraints become max-closed. Whether there is an efficient algorithm to find such a combination of orderings when it exists is currently an open question.

5. Horn clauses

A Horn clause is a logical expression consisting of a disjunction of propositional variables, at most one of which is unnegated. A Horn formula is a conjunction of Horn clauses.

The importance of Horn formulae is illustrated by the fact that the satisfiability problem for Horn formulae (HORNSAT) is P-complete [16]. Hence every polynomially solvable decision problem may be reduced to HORNSAT in logarithmic space.

In this section we give an alternative description of min-closed constraints and show that Horn clauses are a special case of such constraints.

In order to express a constraint $C(S)$ as a formula in first-order logic we introduce variable symbols $v_1, v_2, \ldots, v_r$ to denote the variables in S, and constant symbols $a_1, a_2, \ldots$ and $b_1, b_2, \ldots$ to denote fixed elements of the domain D.

Lemma 5.1. *Any constraint may be expressed as the conjunction of expressions of the form*

$$\neg[(a_1 \leqslant v_1) \wedge (a_2 \leqslant v_2) \wedge \cdots \wedge (a_r \leqslant v_r) \wedge (v_1 \leqslant b_1) \wedge (v_2 \leqslant b_2) \wedge \cdots \wedge (v_r \leqslant b_r)]. \quad (1)$$

Proof. Straightforward, since each constraint $C(S)$, where $S = \{1, 2, \ldots, r\}$ is the conjunction of the negations

$$\neg [(a_1 \leqslant v_1) \wedge (a_2 \leqslant v_2) \wedge \cdots \wedge (a_r \leqslant v_r) \\ \wedge (v_1 \leqslant a_1) \wedge (v_2 \leqslant a_2) \wedge \cdots \wedge (v_r \leqslant a_r)]$$

for each tuple $(a_1, a_2, \ldots, a_r) \notin C(S)$. □

Theorem 5.2. *A constraint is min-closed if and only if it is logically equivalent to a conjunction of disjunctions of the following form*

$$(v_1 < a_1) \vee (v_2 < a_2) \vee \cdots \vee (v_r < a_r) \vee (v_i > b_i). \tag{2}$$

Proof. ($\Rightarrow$) If $C(S)$ is min-closed, then by Lemma 5.1 we may express $C(S)$ as a conjunction of expressions of the form given in Eq. (1).

Now consider any conjunct, γ, in this expression

$$\gamma = \neg [(a_1 \leqslant v_1) \wedge (a_2 \leqslant v_2) \wedge \cdots \wedge (a_r \leqslant v_r) \\ \wedge (v_1 \leqslant b_1) \wedge (v_2 \leqslant b_2) \wedge \cdots \wedge (v_r \leqslant b_r)].$$

In order to satisfy γ, every tuple in $C(S)$ must falsify at least one of the conjuncts in γ. In other words, every tuple in $C(S)$ must break at least one of the bounds specified by γ.

Let $A \subseteq C(S)$ be the set of tuples in $C(S)$ which satisfy all of the lower bounds in γ.

For any tuple $t \in A$ let $B(t) \subseteq \{b_1, b_2, \ldots, b_r\}$ be the upper bounds which t breaks. By the argument above, $B(t)$ must be non-empty.

Since $C(S)$ is min-closed, if A contains any pair of tuples t, t', then it also contains the tuple $t \sqcap t'$. But $B(t \sqcap t') = B(t) \cap B(t')$ so we have, for all $t, t' \in A$

$$B(t) \cap B(t') \neq \emptyset.$$

Repeating this argument we obtain

$$\bigcap_{t \in A} B(t) \neq \emptyset.$$

Hence there is at least one upper bound b_i in γ which is broken by all tuples t satisfying the lower bounds. This implies that the conjunct γ may be replaced with

$$\neg [(a_1 \leqslant v_1) \wedge (a_2 \leqslant v_2) \wedge \cdots \wedge (a_r \leqslant v_r) \wedge (v_i \leqslant b_i)]$$

without allowing any additional elements in $C(S)$. This expression is clearly equivalent to the expression in the theorem.

($\Leftarrow$) Conversely, we shall now show that any constraint which satisfies an expression of the form given in the theorem must be min-closed.

Assume, for contradiction, that $t_1 = (x_1, x_2, \ldots, x_r)$ and $t_2 = (y_1, y_2, \ldots, y_r)$ satisfy a conjunction of expressions of the form given in the theorem, but $t_1 \sqcap t_2$ does not.

This implies that $t_1 \sqcap t_2$ fails to satisfy some conjunct γ of this expression which is logically equivalent to the following

$$\neg[(a_1 \leqslant v_1) \wedge (a_2 \leqslant v_2) \wedge \cdots \wedge (a_r \leqslant v_r) \wedge (v_i \leqslant b_i)].$$

But, by definition of $t_1 \sqcap t_2$ this means that $a_j \leqslant \min(x_j, y_j)$, $j = 1, 2, \ldots, r$, and $\min(x_i, y_i) \leqslant b_i$. Hence either t_1 or t_2 also fail to satisfy γ, which contradicts the choice of t_1 and t_2. □

Corollary 5.3. *If the domain of values for the variables is* {True, False}, *with* False < True, *then a constraint is min-closed if and only if it is logically equivalent to a conjunction of Horn clauses.*

Proof. When the domain of values is {True, False}, with False < True, then the values of the a_i's and b_i's in the expression given in Eq. (2) must be replaced with True or False. Simplifying the resulting expression therefore gives an expression of the form

$$\neg v_{i_1} \vee \neg v_{i_2} \vee \cdots \vee \neg v_{i_t} \vee v_i$$

or

$$\neg v_{i_1} \vee \neg v_{i_2} \vee \cdots \vee \neg v_{i_t},$$

which are precisely the forms allowed for Horn clauses. □

Corollary 5.3 is equivalent to the well-known result that a theory in propositional logic may be expressed in Horn formulae if and only if its set of models is closed under intersection (see Lemma 4.5 of [5]).

By a similar argument we may show that max-closed constraints over this Boolean domain are precisely those which may be expressed by conjunctions of disjunctions with at most one negated variable.

6. Intractability of extensions

In this section we shall demonstrate that any superset of the set of max-closed (or min-closed) constraints can generate intractable problems. Hence these sets of constraints are both maximal sets of tractable constraints.

We begin by characterizing max-closed constraints using the following property.

Definition 6.1. A constraint $C(S)$ is said to be *"crossover-closed"* if, for all $i, j \in S$,

$$[(x_i, x_j) \in \pi_{(i,j)}(C(S))] \wedge [(y_i, y_j) \in \pi_{(i,j)}(C(S))] \wedge (x_i > y_i) \wedge (x_j < y_j)$$
$$\Rightarrow [(x_i, y_j) \in \pi_{(i,j)}(C(S))].$$

For binary constraints, this is equivalent to being max-closed.

Lemma 6.2. *A binary constraint is max-closed if and only if it is crossover-closed.*

However, for constraints of higher arity, it is possible to be crossover-closed without being max-closed, as the following example illustrates.

Example 6.3. Consider the following constraint, C, consisting of three tuples

$$C = \{(T,T,F),(T,F,T),(F,T,T)\}.$$

If the domain D is ordered such that $F < T$, then C is crossover-closed, because the projection of C onto any pair of variables is $\{(T,T),(T,F),(F,T)\}$.

However, C is not max-closed because the maximum of any pair of tuples is (T,T,T), which is not an element of C.

We now establish the precise relationship between these properties for constraints of any arity.

Lemma 6.4. *A constraint $C(S)$ is max-closed if and only if every intersection of $C(S)$ with max-closed constraints is crossover-closed.*

Proof. ($\Rightarrow$) If $C(S)$ is max-closed then, for all $i, j \in S$,

$$[(x_i,x_j) \in \pi_{(i,j)}(C(S))] \wedge [(y_i,y_j) \in \pi_{(i,j)}(C(S))]$$
$$\Rightarrow [(\max(x_i,y_i),\max(x_j,y_j)) \in \pi_{(i,j)}(C(S))].$$

Hence, $C(S)$ is crossover-closed.

Furthermore, the join of $C(S)$ with any max-closed constraint remains max-closed, by Proposition 4.1. Hence, $C(S)$ remains crossover-closed no matter what further restrictions are imposed by max-closed constraints.

($\Leftarrow$) If $C(S)$ is not max-closed, then there exist $t_1, t_2 \in C(S)$, such that $t = t_1 \sqcup t_2 \notin C(S)$. Let $t_1 = (x_1, x_2, \ldots, x_r)$ and $t_2 = (y_1, y_2, \ldots, y_r)$ and let $t = (z_1, z_2, \ldots, z_r)$ where $z_i = \max(x_i, y_i)$.

Now impose a further constraint on S which restricts each variable $i \in S$ to values less than or equal to z_i. This additional constraint is clearly max-closed, and the intersection of $C(S)$ with this constraint results in a new constraint, which will be denoted $C'(S)$.

Choose a minimal subset $M = \{i_1, \ldots, i_m\} \subseteq S$ such that $\pi_M(\{t\}) \notin \pi_M(C'(S))$. By the choice of t, we have $2 \leqslant |M| \leqslant r$. Since M is minimal, for any $i_j \in M$ we have $\pi_{M\setminus\{i_j\}}(\{t\}) \in \pi_{M\setminus\{i_j\}}(C'(S))$. In other words, for any $i_j \in M$, $C'(S)$ must contain a tuple $(z_{i_1}, \ldots, z_{i_{j-1}}, z'_{i_j}, z_{i_{j+1}}, \ldots, z_{i_m})$ where $z'_{i_j} \neq z_{i_j}$. Since variable i_j is constrained by $C'(S)$ to take values less than or equal to z_{i_j}, it follows that $z'_{i_j} < z_{i_j}$.

Now choose any two distinct variables $i_j, i_k \in M$ and impose a further constraint on the variables in $M \setminus \{i_j, i_k\}$ (if any) which requires each variable i_p to take the value z_{i_p}. The intersection of $C'(S)$ with this additional (max-closed) constraint results in a new constraint $C''(S)$ such that

$$[(z_{i_j}, z'_{i_k}) \in \pi_{(i_j,i_k)}(C''(S))] \wedge [(z'_{i_j}, z_{i_k}) \in \pi_{(i_j,i_k)}(C''(S))]$$
$$\wedge (z_{i_j} > z'_{i_j}) \wedge (z'_{i_k} < z_{i_k}) \wedge [(z_{i_j}, z_{i_k}) \notin \pi_{(i_j,i_k)}(C''(S))].$$

Hence $C''(S)$ is not crossover-closed. □

Using this lemma, we are able to prove the main results of this section.

Theorem 6.5. *For any domain D, with $|D| \geqslant 3$, and any constraint C not in $\mathcal{M}_D$,* $\mathrm{CSP}(\mathcal{M}_D \cup \{C\})$ *is* NP*-complete.*

Furthermore, it remains NP*-complete even when $\mathcal{M}_D$ is restricted to* binary *max-closed constraints.*

Proof. Any CSP clearly belongs to NP since a solution may be checked against all of the constraints of the problem in polynomial time.

To demonstrate that $\mathrm{CSP}(\mathcal{M}_D \cup \{C\})$ is NP-complete we shall provide a polynomial time reduction from the NP-complete problem GRAPH 3-COLORABILITY [7].

To carry out this reduction, we first note that, by Lemma 6.4, since C is not max-closed, we may compose C with max-closed constraints to form a constraint C' which is not crossover-closed. In other words, on some pair of coordinate positions, there exist values x_1, x_2, y_1, y_2, with $x_1 > y_1$ and $x_2 < y_2$, such that C' allows the combinations (x_1, x_2) and (y_1, y_2) but does not allow the combination (x_1, y_2). Without loss of generality, we may assume that this holds in the first and last coordinate positions.

Now let r be the length of the tuples in C and consider the constraint satisfaction problem P with variables $\{1, 2, 3, \ldots, 2r, 2r+1, 2r+2\}$, domain $D = \{a, b, c, \ldots\}$ where $(a > b > c)$, and the following constraints:

$$
\begin{aligned}
&C(1, 2, \ldots, r) = C(r+1, r+2, \ldots, 2r) = C',\\
&C(2r+1, 1) = \{(a, x_1), (b, x_1), (c, x_1), (b, y_1), (c, y_1)\},\\
&C(2r+2, r) = \{(a, y_2), (b, y_2), (c, y_2), (b, x_2), (c, x_2)\},\\
&C(2r+1, r+1) = \{(a, x_1), (b, x_1), (c, x_1), (a, y_1), (c, y_1)\},\\
&C(2r+2, 2r) = \{(a, y_2), (b, y_2), (c, y_2), (a, x_2), (c, x_2)\},\\
&C(2r+1, 2r+2) = \{(a, a), (a, b), (b, a), (b, b), (c, a), (a, c), (c, b), (b, c)\}.
\end{aligned}
$$

The problem P is illustrated in Fig. 2. Note that the additional constraints used in P are all max-closed, hence $P \in \mathrm{CSP}(\mathcal{M}_D \cup \{C\})$.

By explicitly constructing all possible solutions to P, we may show that all possible combinations of a, b and c are allowed for the variables $2r+1$ and $2r+2$ except for the pairs $(a, a), (b, b)$ and (c, c).

But this means that we may reduce any instance of GRAPH 3-COLORABILITY to a problem in $\mathrm{CSP}(\mathcal{M}_D \cup \{C\})$ in polynomial time, by replacing each edge in the graph with P and identifying the vertices of the edge with the variables corresponding to $2r+1$ and $2r+2$.

Since the above construction uses only binary constraints, this result remains true even when $\mathcal{M}_D$ is restricted to binary constraints. □

For the case of $|D| = 2$, it is possible to obtain a similar result by constructing a polynomial time reduction from the NP-complete problem NOT-ALL-EQUAL-SATISFIABILITY [7].

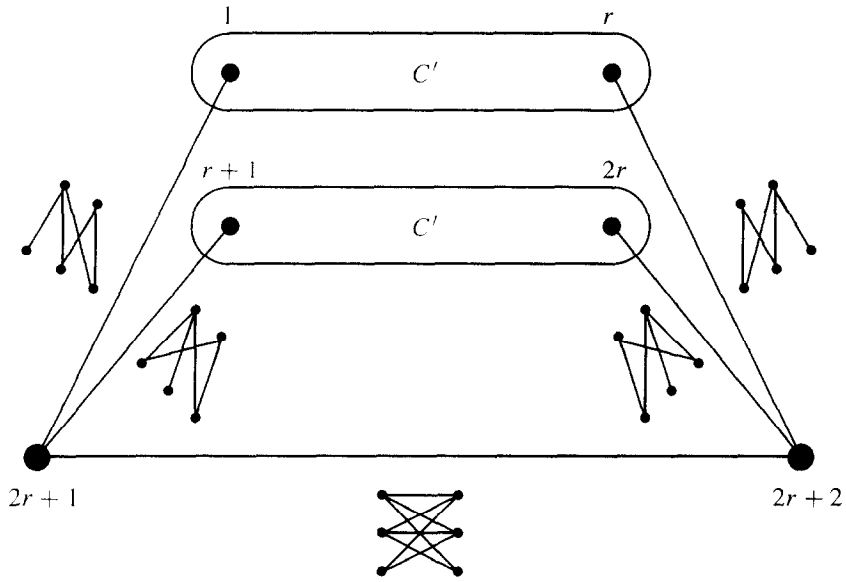

Fig. 2. The CSP P used to construct a $\neq$-constraint.

Theorem 6.6. *For any domain D, with $|D| = 2$, and any constraint C not in $\mathcal{M}_D$,* CSP($\mathcal{M}_D \cup \{C\}$) *is* NP*-complete.*

Furthermore, it remains NP*-complete even when $\mathcal{M}_D$ is restricted to* ternary *max-closed constraints.*

Theorem 6.6 may also be proved by using Corollary 5.3, together with Schaefer's characterization of tractable subproblems of the SATISFIABILITY problem, given in [17].

Corresponding results may of course be obtained for min-closed constraints.

7. Conclusion

This paper has demonstrated the significance of the class of constraints which have been designated as max-closed constraints.

These results provide efficient techniques for solving any constraint satisfaction problem where the constraints lie within this class. This may be known *a priori* from some feature of the original problem (such as the fact that the constraints are expressed in Horn formulae), or it may be achievable by suitable pre-processing. In order words, we have identified another possible class of "target" problems, for which an efficient algorithm is known, which can be used to provide a possible goal for a general problem solving scheme, as in [4].

We have also shown that any class of problems allowing a larger set of constraints is NP-complete, so it is unlikely that efficient general solution techniques exist.

Acknowledgements

The authors wish to acknowledge the support of the British Council for this research.

References

[1] M.C. Cooper, D.A. Cohen and P.G. Jeavons, Characterizing tractable constraints, *Artif. Intell.* **65** (1994) 347–361.
[2] R. Dechter, From local to global consistency, *Artif. Intell.* **55** (1992) 87–107.
[3] R. Dechter, Constraint networks, in: S.C. Shapiro, ed., *Encyclopedia of Artificial Intelligence* (Wiley, New York, 2nd ed., 1992) 276–285.
[4] R. Dechter and J. Pearl, Network-based heuristics for constraint-satisfation problems, *Artif. Intell.* **34** (1988) 1–38.
[5] R. Dechter and J. Pearl, Structure identification in relational data, *Artif. Intell.* **58** (1992) 237–270.
[6] E.C. Freuder, A sufficient condition for backtrack-bounded search, *J. ACM* **32** (1985) 755–761.
[7] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to NP-Completeness* (Freeman, San Francisco, CA, 1979).
[8] M. Gyssens, P. Jeavons and D. Cohen, Decomposing constraint satisfaction problems using database techniques, *Artif. Intell.* **66** (1994) 57–89.
[9] P. Janssen, P. Jegou, B.Nouguier and M.C. Vilarem, A filtering process for general constraint satisfaction problems: achieving pair-wise consistency using an associated binary representation, in: *Proceedings IEEE Workshop on Tools for Artificial Intelligence* (1989) 420–427.
[10] P.G. Jeavons, D.A. Cohen and M.C. Cooper, A substitution operation for constraints, in: *Proceedings of PPCP94*, Lecture Notes in Computer Science **874** (Springer-Verlag, Berlin, 1994) 1–9.
[11] L. Kirousis, Fast parallel constraint satisfaction, *Artif. Intell.* **64** (1993) 147–160.
[12] A.K. Mackworth, Consistency in networks of relations, *Artif. Intell.* **8** (1977) 99–118.
[13] D. Maier, *The Theory of Relational Databases* (Computer Science Press, Rockville, MD, 1983).
[14] U. Montanari, Networks of constraints: fundamental properties and applications to picture processing, *Information Sci.* **7** (1974) 95–132.
[15] U. Montanari and F. Rossi, Constraint relaxation may be perfect, *Artif. Intell.* **48** (1991) 143–170.
[16] C.H. Papadimitriou, *Computational Complexity* (Addison-Wesley, Reading, MA, 1994).
[17] T.J. Schaefer, The complexity of satisfiability problems, in: *Proceedings 10th ACM Symposium on Theory of Computing (STOC)* (1978) 216–226.
[18] E. Tsang, *Foundations of Constraint Satisfaction* (Academic Press, New York, 1993).
[19] P. van Beek, On the minimality and decomposability of row-convex constraint networks, in: *Proceedings AAAI-92*, San Jose, CA (1992) 447–452.
[20] P. van Beek and R. Dechter, Constraint tightness versus global consistency, in: *Proceedings 4th International Conference on Principles of Knowledge Representation and Reasoning (KR94)*, Berlin (1994) 572–582.
[21] P. Van Hentenryck, Y. Deville and C.-M. Teng, A generic arc-consistency algorithm and its specializations, *Artif. Intell.* **57** (1992) 291–321.