ELSEVIER

Operations Research Letters 30 (2002) 9–16

Operations Research Letters

www.elsevier.com/locate/dsw

An efficient algorithm for a class of constraint satisfaction problems

Gerhard J. Woeginger[a,b,*,1]

[a] *Department of Mathematics, University of Twente, P.O. Box 217, 7500 AE Enschede, Netherlands*

[b] *Institut für Mathematik, Technische Universität Graz, Steyrergasse 30, A-8010 Graz, Austria*

Received 24 July 2001; received in revised form 29 September 2001; accepted 31 October 2001

Abstract

We define the class of the so-called monotone constraint satisfaction problems (MON-CSP). MON-CSP forms a subclass of the class of min-closed (respectively, max-closed) constraint satisfaction problems of Jeavons and Cooper (Artificial Intelligence 79 (1995) 327). We prove that for all problems in the class MON-CSP there exists a very fast and very simple algorithm for testing feasibility.

We then show that a number of well-known results from the literature are special cases of MON-CSP: (1) Satisfiability of Horn formulae; (2) graph homomorphisms to directed graphs with an $\underline{X}$-numbering; (3) monotone integer programming with two variables per inequality; (4) project scheduling under AND/OR precedence constraints. Our results provide a unified algorithmic approach to all these problems.

Keywords: Constraint satisfaction; Feasibility checking; Efficient algorithm; Satisfiability; Horn formula; Graph coloring; Graph homomorphism; Monotone integer programming; AND/OR project scheduling

1. Introduction

We consider constraint satisfaction problems (CSP) of the following type. Let $\mathscr{D}$ be a domain that is totally ordered by '$\preccurlyeq$', and let '$\prec$' be the underlying strict total order. Let $-\infty \notin \mathscr{D}$ and $+\infty \notin \mathscr{D}$ be special elements such that $-\infty \prec x \prec +\infty$ holds for all $x \in \mathscr{D}$. An instance I of the considered constraint satisfaction problem consists of

- a finite set $X = \{x_1, \ldots, x_n\}$ of variables;
- a finite domain $D_i \subseteq \mathscr{D}$ for every variable x_i ($i = 1, \ldots, n$);
- a finite set C^- of lower bound constraints of the form "$f(x_1, \ldots, x_n) \preccurlyeq x_i$", where f is some function $\mathscr{D}^n \to \mathscr{D} \cup \{-\infty, +\infty\}$;
- a finite set C^+ of upper bound constraints of the form "$x_i \preccurlyeq g(x_1, \ldots, x_n)$", where g is some function $\mathscr{D}^n \to \mathscr{D} \cup \{-\infty, +\infty\}$.

We assume that the functions f and g in the lower and upper bound constraints are given by a simple oracle algorithm that can be evaluated in polynomial time. The goal is to decide whether there exists a

* Corresponding author. Department of Mathematics, University of Twente, P.O. Box 217, 7500 AE Enschede, Netherlands.

E-mail address: g.j.woeginger@math.utwente.nl (G.J. Woeginger).

[1] Supported by the START program Y43-MAT of the Austrian Ministry of Science.

0167-6377/02/$ - see front matter
PII: S0167-6377(01)00114-6

feasible solution, i.e., values $\bar{x}_i \in D_i$ for $1 \leqslant i \leqslant n$, such that all constraints $f(\bar{x}_1,\ldots,\bar{x}_n) \preccurlyeq \bar{x}_i$ in C^- and all constraints $\bar{x}_i \preccurlyeq g(\bar{x}_1,\ldots,\bar{x}_n)$ in C^+ are satisfied simultaneously. Note that in a feasible solution, none of the variables x_i may take the values $\pm\infty$.

In general, the above defined constraint satisfaction problem is a strongly NP-complete problem (see e.g., Theorem 6 in Section 3). In this note we will mainly deal with MON-CSP, a special case of CSP that is based on the so-called monotone functions. A function $h:\mathscr{D}^n \to \mathscr{D} \cup \{-\infty,+\infty\}$ is called *monotone*, if for any $x'_1,\ldots,x'_n \in \mathscr{D}$ and any $x''_1,\ldots,x''_n \in \mathscr{D}$ with $x'_i \preccurlyeq x''_i$ for $1 \leqslant i \leqslant n$, we have $h(x'_1,\ldots,x'_n) \preccurlyeq h(x''_1,\ldots,x''_n)$.

Theorem 1 (Monotone constraint satisfaction problems, MON-CSP)**.** *For an instance of the above defined class of constraint satisfaction problems, we denote* $n=|X|$ *and* $m=|C^-|+|C^+|$. *Furthermore, we denote by* $d=\sum_{i=1}^{n}|D_i|$ *the total size of all variable domains, and by* $T(n,m)$ *the time needed for checking whether a given assignment for the n variables in X satisfies all m constraints. The following two special cases—that are called monotone constraint satisfaction problems—can be solved within time* $O(dT(n,m))$, *i.e., within time polynomially bounded in d and* $T(n,m)$.

(P1) *All functions f in the lower bound constraints in* C^- *are monotone.*
The set C^+ *of upper bound constraints is empty.*

(P2) *All functions g in the upper bound constraints in* C^+ *are monotone.*
The set C^- *of lower bound constraints is empty.*

Theorem 1 will be proved in Section 2. The underlying algorithm is quite simple, and the underlying ideas apparently have been used many times before in the literature. The main contribution of this note is to identify this polynomially solvable special case MON-CSP and this algorithm, to point out its wide applicability, and to illustrate that it is at the heart of many algorithmic results in discrete applied mathematics, mathematical programming, and theoretical computer science.

Now let us discuss the relationship between MON-CSP and other classes of constraint satisfaction problems from the literature. Whereas our special case MON-CSP only allows lower and upper bound constraints on the variables, the constraint satisfaction problems considered by Jeavons and Cooper [8] allow arbitrary constraints on subsets of the variables in X. Every such constraint is specified by explicitly listing the mutually consistent values for the variables in the corresponding subset. A constraint satisfaction problem is called *min-closed*, if with any two feasible solutions $x'_1,\ldots,x'_n$ and $x''_1,\ldots,x''_n$ also the element-wise minimum $\min\{x'_1,x''_1\},\ldots,\min\{x'_n,x''_n\}$ is a feasible solution. *Max-closed* problems are defined analogously. Jeavons and Cooper [8] show that min-closed and max-closed constraint satisfaction problems can be solved in polynomial time; here polynomial means polynomial in the size of the listings of all mutually consistent values for all constraints. See [7,6] for several related results.

Example 2. Consider a constraint satisfaction problem with variables $\{x_1,x_2,x_3,x_4,x_5\}$ that all have the same domain $\mathscr{D}=\{0,1\}$ with $0 \prec 1$. There are two constraints $x_1+x_2+x_3 \leqslant 1$ and $x_3+x_4+x_5 \leqslant 2$. In our oracle representation, these constraints are specified by writing down these two inequalities. Checking whether a given solution is feasible can be done by plugging in the corresponding values into the inequalities. In the representation of Jeavons and Cooper [8], these constraints are specified by explicitly listing the allowed values

$$(0,0,0),\ (1,0,0),\ (0,1,0),\ (0,0,1)$$

for the triple (x_1,x_2,x_3), and the allowed values

$$(0,0,0),\ (1,0,0),\ (0,1,0),\ (0,0,1),\ (1,1,0),\ (1,0,1),\ (0,1,1)$$

for the triple (x_3,x_4,x_5). Checking whether a given solution is feasible can be done by comparing it against the listed triples.

The following facts are easily verified: This instance is a min-closed constraint satisfaction problem. The constraints cannot be rewritten as monotone lower or monotone upper bound constraints, and hence this instance does not belong to the class MON-CSP.

In Lemma 5 we will show that MON-CSP of type (P1) forms a special case of the min-closed constraint satisfaction problems, and that MON-CSP of type (P2)

```
1  For i = 1 to n do
2      Set x_i := min(D_i)
3  While (there exists a violated constraint) and (all x_i ≠ +∞) do
4      Let x_i be a violating variable
5      Set x_i := next(x_i, D_i)
6  If (all x_i ≠ +∞) then output 'Feasible' else output 'Infeasible'
```

Fig. 1. The algorithm that solves MON-CSP of type (P1).

forms a special case of the max-closed constraint satisfaction problems. Still, our main result in Theorem 1 is completely independent of the results in Jeavons and Cooper [8]. What we prove is a polynomial time result for the case where the constraints are represented in a short and compact way via an oracle function; this yields a small input size. What they prove is a polynomial time result for the case where the constraints are given as lists of feasible tuples; this may yield an exponentially larger input size. Also the algorithmic approaches are quite different: The approach in [8] is based on pair-wise consistency, and hence repeatedly intersects pairs of constraints. Our approach looks at the constraints one by one, and keeps cutting down the domains of single variables.

Organization of the note: In Section 2 we prove Theorem 1. In Section 3, we prove some complementary negative results that draw a sharp borderline between tractable and intractable cases of CSP. Moreover we illustrate that in a certain oracle model, the time complexity stated in Theorem 1 is essentially best possible. Sections 4–7 contain applications of Theorem 1 to various areas. Section 4 deals with the satisfiability of logical formulae, and with Horn formulae. Section 5 discusses homomorphisms to directed graphs, Section 6 considers monotone integer programming with two variables per inequality. Finally, Section 7 deals with project scheduling under AND/OR precedence constraints. The four problems discussed in Sections 4–7 all are shown to form special cases of MON-CSP, and Theorem 1 provides a unified algorithmic treatment of these four problems.

2. Positive results

In this section, we will prove Theorem 1. In fact, we will only prove the polynomial time result for MON-CSP of type (P1). The proof for MON-CSP of type (P2) can be done by completely symmetric arguments.

Hence, let us assume that the instance I of the constraint satisfaction problem has $C^+ = \emptyset$ and that all functions f in the given lower bound constraints in C^- are monotone. Let $D \subseteq \mathscr{D}$ be finite. We write $\min(D)$ for the minimum element in D with respect to $\preccurlyeq$. For $x \in D$, we write $\mathrm{next}(x, D)$ short for $\min(D \cap \{y \in D\colon x \prec y\} \cup \{+\infty\})$; in other words, $\mathrm{next}(x, D)$ is the smallest element in $D \cup \{+\infty\}$ that is strictly greater than x. If a constraint $f(x_1, \ldots, x_n) \preccurlyeq x_i$ in C^- is not satisfied for a certain setting of the variables, then it is *violated* and x_i is a *violating* variable. Our algorithm is depicted in Fig. 1.

Lemma 3. *The algorithm in Fig.* 1 *terminates after at most* $d = \sum_{i=1}^{n} |D_i|$ *executions of the while-loop in lines* 3–5.

Proof. Every time the while-loop in lines 3–5 is executed, the value of one of the variables in X is increased. There are at most d possibilities for increasing a variable. □

Lemma 4. *Assume that* $x_1^*, \ldots, x_1^n$ *is a feasible solution for an instance* I of *MON-CSP of type* (P1). *Then for* $i = 1, \ldots, n$ *the invariant* $x_i \preccurlyeq x_i^*$ *is true throughout the execution of the algorithm in Fig.* 1.

Proof. Suppose for the sake of contradiction that the claimed inequality does not hold throughout the execution. Consider the first moment t in time when $x_k \succ x_k^*$ holds for some index k. This moment t cannot occur during the execution of the for-loop in lines 1–2, since there $x_k = \min(D_k) \preccurlyeq x_k^*$ holds throughout.

Hence, moment t must occur during the execution of the while-loop in lines 3–5 when variable x_k is updated. This means that at the moment t^- just before moment t, some constraint $f(x_1, \ldots, x_n) \preccurlyeq x_k$ in C^-

was violated. At that time t^-, we then had

$$x_k \prec f(x_1,\ldots,x_n). \tag{1}$$

Moreover up to time t^- we had $x_i \preccurlyeq x_i^*$ for all $i = 1,\ldots,n$, and hence by the monotonicity of f

$$f(x_1,\ldots,x_n) \preccurlyeq f(x_1^*,\ldots,x_1^n). \tag{2}$$

Finally, since $x_1^*,\ldots,x_n^*$ is a feasible solution we have

$$f(x_1^*,\ldots,x_n^*) \preccurlyeq x_k^*. \tag{3}$$

Putting (1), (2), and (3) together, we conclude that $x_k \prec x_k^*$ at time t^-. At time t, the value of variable x_k is increased to $\text{next}(x_k, D_k) \preccurlyeq x_k^*$. This contradiction completes the proof. □

Proof of Theorem 1. Now let us complete the proof of our main theorem. We start with the analysis of the time complexity. The for-loop in lines 1–2 takes at most $O(d)$ time. By Lemma 3, the while-loop in lines 3–5 is executed at most d times. Every single execution checks feasibility of the current solution in $T(n,m)$ time and then possibly updates a variable. It is easy to preprocess the data such that afterwards every variable update can be done in constant time. Hence, the run time indeed is $O(dT(n,m))$.

Now let us turn to correctness. If the instance is infeasible, then the while-loop in lines 3–5 will eventually increase one of the variables to $+\infty$. Then the final check in line 6 gives the correct answer. If the instance is feasible, then by Lemma 4 the while-loop in lines 3–5 can never increase a variable to $+\infty$. Therefore, the only possibility for leaving the while-loop is to have no more violated constraints. Again, the final check in line 6 gives the correct answer. □

The following lemma observes that feasible solutions for MON-CSP of type (P1) are closed under element-wise minimum taking. This implies that (unless the instance is infeasible) there exists a smallest feasible solution, and it is easy to see that our algorithm always ends up with this smallest feasible solution. By symmetry, feasible solutions for MON-CSP of type (P2) are closed under element-wise maximum taking.

Lemma 5. *Let $x_1',\ldots,x_n'$ and $x_1'',\ldots,x_n''$ be feasible solutions for an instance I of MON-CSP of type* (P1). *Then also the element-wise minimum $\min\{x_1',x_1''\},\ldots,\min\{x_n',x_n''\}$ is a feasible solution for instance I.*

Proof. Consider an arbitrary constraint $f(x_1,\ldots,x_n) \preccurlyeq x_i$ in C^-. We have

$$f(\min\{x_1',x_1''\},\ldots,\min\{x_n',x_n''\}) \preccurlyeq f(x_1',\ldots,x_n') \preccurlyeq x_i',$$

where the first inequality follows from the monotonicity of f and the second inequality follows from the feasibility of $x_1',\ldots,x_n'$. Analogously we get

$$f(\min\{x_1',x_1''\},\ldots,\min\{x_n',x_n''\}) \preccurlyeq f(x_1'',\ldots,x_n'') \preccurlyeq x_i''.$$

Combining these two statements yields $f(\min\{x_1',x_1''\}, \ldots,\min\{x_n',x_n''\}) \preccurlyeq \min\{x_i',x_i''\}$. □

Finally, let us briefly compare our algorithm to the algorithm of Jeavons and Cooper [8] for min-closed satisfaction problems. The algorithm in [8] is based on pair-wise consistency testing (see also [13]), and it repeatedly intersects pairs of constraints, and computes and lists all mutually consistent values for these intersected constraints. This approach clearly cannot be applied to the case where the constraints are specified as oracles: The listing of these mutually consistent values would exponentially blow up the worst case running time.

The algorithm in Fig. 1 behaves quite differently: It is variable-oriented, and it keeps shrinking the domains of single violating variables little by little. The algorithm is heavily based on the asymmetry in the constraints of MON-CSP of type (P1): Whenever a lower bound constraint $f(x_1,\ldots,x_n) \preccurlyeq x_i$ is violated, the algorithm can identify x_i as the guilty variable, and it then punishes x_i by shrinking its domain. The algorithm in Fig. 1 does not seem to generalize to general min-closed constraint satisfaction problems. It heavily exploits the structure of MON-CSP, and it does not work for completely symmetric constraints as, e.g., in Example 2. In other words, the algorithm exploits properties of MON-CSP that are not available anymore in the general min-closed CSP.

3. Negative results

In this section, we prove some negative results that complement the positive results in Theorem 1 for MON-CSP. The (difficult) special cases (H1), (H2),

(H3) in Theorem 6 are fairly close to the (easy) special cases (P1) and (P2) in Theorem 1, and hence indicate the exact borderline between tractable and intractable cases of CSP.

Theorem 6. *The following three special cases* (H1)–(H3) *of the constraint satisfaction problem all are NP-complete in the strong sense*:

(H1) *The set* C^- *of lower bound constraints is empty.*
(H2) *The set* C^+ *of upper bound constraints is empty.*
(H3) *All functions* f *in the lower bound constraints in* C^- *and all functions* g *in the upper bound constraints in* C^+ *are monotone.*

Proof. The proof is done by slightly modifying corresponding arguments from [8]. All reductions are from the NP-complete ONE-IN-THREE-3SAT problem [2]: Given a set of Boolean variables $X = \{x_1, \ldots, x_n\}$ and a set C of clauses over X such that every clause consists of exactly three, unnegated variables. Does there exist a truth assignment of X such that each clause in C has exactly one true literal?

Proof for (H1): We use the domain $\mathscr{D} = \{\text{TRUE}, \text{FALSE}\}$ with FALSE $\leqslant$ TRUE. For every variable $x_i \in X$ we set $D_i = \mathscr{D}$. For every clause $c \in C$, we introduce a function $g_c : \mathscr{D}^n \to \mathscr{D} \cup \{-\infty, +\infty\}$ such that $g_c(x_1, \ldots, x_n) = \text{TRUE}$ if under the truth assignment $(x_1, \ldots, x_n)$ the clause c contains exactly one true literal, and $g_c(x_1, \ldots, x_n) = -\infty$ otherwise. For every clause $c \in C$, the set C^+ contains the upper bound constraint $x_1 \leqslant g_c(x_1, \ldots, x_n)$. The set C^- is empty. It can be verified that this constraint satisfaction instance has a feasible solution if and only if the ONE-IN-THREE-3SAT instance has answer YES.

Proof of (H2): This can be done symmetrically to the above proof of (H1).

Proof of (H3): We use the domain $\mathscr{D} = \{0, 1\}$ with $0 \leqslant 1$. We set $D_i = \mathscr{D}$ for every $x_i \in X$. There is one special variable x_0 with $D_0 = \{1\}$. For every clause $c = (x_i \vee x_j \vee x_k)$ in C, the set C^+ contains the upper bound constraint $x_0 \leqslant x_i + x_j + x_k$, and the set C^- contains the lower bound constraint $x_i + x_j + x_k \leqslant x_0$. Note that all involved functions are monotone. It can be verified that this constraint satisfaction instance has a feasible solution if and only if the ONE-IN-THREE-3SAT instance has answer YES. □

The stated time complexity in Theorem 1 depends linearly on the total size $d = \sum_{i=1}^{n} |D_i|$ of all variable domains. Since all involved functions are monotone, one might hope to get a speed-up to $O(\log d)$ or to $O(n \log d)$ by applying some kind of binary search. However, the following example illustrates that such a speed-up can only be possible if the constraints carry a strong additional structure and are easy to handle.

Example 7. Consider a constraint satisfaction problem with domain $\mathscr{D} = \{1, \ldots, d\}$ and with the natural ordering $\leqslant$ on the integers; we set $+\infty = d + 1$. Let $X = \{x_1\}$, let $D_1 = \mathscr{D}$, and let there be a single monotone constraint $f(x_1) \leqslant x_1$ where the function f is given as an oracle.

Assume that the function f is either of the form $f_{+\infty}(x) = x + 1$ for all $x \in \mathscr{D}$, or of the following form f_e with $e \in \mathscr{D}$: $f_e(x) = x + 1$ for $x \neq e$ and $f_e(e) = e$. In the case $f = f_{+\infty}$ the instance is infeasible, whereas in the case $f = f_e$ the instance is feasible. It is impossible to distinguish between these two cases without (in the worst case) evaluating the function f at all d points in $\mathscr{D}$. All these cases are in MON-CSP.

4. Application: Horn formulae

Propositional Horn formulae form an interesting special case of the SATISFIABILITY problem: Let $X = \{x_1, \ldots, x_n\}$ be a set of Boolean variables. A clause over X is a *Horn clause*, if it contains at most one positive literal. That is, all its literals except possibly for one are negations of variables, like

$$(\neg x_1 \vee \neg x_2 \vee \neg x_3), \quad (\neg x_4 \vee x_5), \quad (x_6),$$

$$(\neg x_7), \quad (x_8 \vee \neg x_9 \vee \neg x_{10} \vee \neg x_{11}).$$

The HORN-SAT problem consists in deciding for a given set C of Horn clauses, whether there exists a truth assignment of X that satisfies all clauses in C. It is well-known that HORN-SAT is polynomially solvable; see e.g. [1] or Chapter 4.2 in the book of Papadimitriou [11].

In this section, we show that HORN-SAT can be formulated as a MON-CSP of type (P1). We use the domain $\mathscr{D} = \{\text{TRUE}, \text{FALSE}\}$ with FALSE $\leqslant$ TRUE.

For every Boolean variable $x_i \in X$ we set $D_i = \mathscr{D}$. Moreover, there are two special variables y_F with domain {FALSE}, and y_T with domain {TRUE}. For every clause $c \in C$, we will introduce one corresponding constraint in C^-.

- If c consists of a single positive literal x_i, then we introduce the lower bound constraint $y_T \preccurlyeq x_i$ on x_i.
- If c only consists of negative literals, say $\neg x_{j_1}, \ldots, \neg x_{j_k}$, then we introduce the lower bound constraint $x_{j_1} \wedge \cdots \wedge x_{j_k} \preccurlyeq y_F$ on y_F.
- If c consists of the positive literal x_i and the negative literals $\neg x_{j_1}, \ldots, \neg x_{j_k}$, then we introduce the lower bound constraint $x_{j_1} \wedge \cdots \wedge x_{j_k} \preccurlyeq x_i$ on x_i.

In any feasible solution for the constraint satisfaction problem, $y_F = \text{FALSE}$ and $y_T = \text{TRUE}$. Moreover, it is easily verified that a constraint is satisfied if and only if the corresponding clause is satisfied. All the functions in the lower bounds are monotone functions, and C^+ is empty. The total size d of all variable domains is $O(n)$, and the time $T(n,m)$ needed for checking whether a given truth assignment satisfies all constraints is $O(nm)$. With this, Theorem 1 implies polynomial time solvability of HORN-SAT.

In fact, applying our algorithm from Section 2 to HORN-SAT yields exactly the same algorithm as described in Chapter 4.2 of [11] for HORN-SAT.

5. Application: graph homomorphisms

A *homomorphism* of a directed graph G to a directed graph H is a mapping $f : V(G) \to V(H)$ which preserves the arcs, i.e., such that $(x,y) \in E(G)$ implies $(f(x), f(y)) \in E(H)$. For a fixed directed graph H, the *H-coloring problem* is the decision problem in which we are given an arbitrary directed graph G and are to decide whether or not there is a homomorphism from G to H. The name is due to the fact that for undirected graphs the K_n-coloring problem simply asks whether or not G is n-colorable. It was shown by Hell and Nešetřil [4] that for *undirected* graphs H-coloring is polynomial when H is bipartite and NP-complete otherwise. The H-coloring problem for *directed* graphs has received much recent attention, but the boundary between easy and hard cases is still not understood.

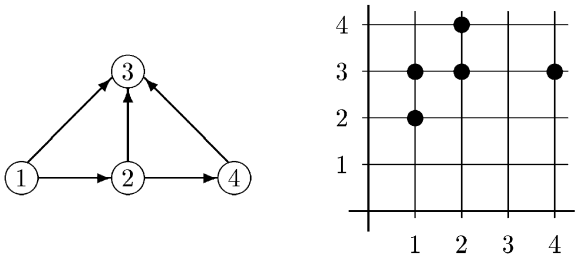

Fig. 2. The $\underline{X}$-numbering of a graph H together with a geometric visualization.

Gutjahr et al. [3] proved that for directed graphs H with a so-called $\underline{X}$-numbering (read: 'X underbar numbering'), H-coloring is polynomial. An *$\underline{X}$-numbering* of H is a numbering $1, \ldots, |V(H)|$ of its vertices with the following property: Whenever (a,b) and (c,d) are arcs in $E(H)$, then also $(\min\{a,c\}, \min\{b,d\})$ is an arc in $E(H)$. See Fig. 2 for an example of an $\underline{X}$-numbering. A very instructive way of looking at $\underline{X}$-numberings is to interpret the arcs (x,y) as geometric points with coordinates x and y in the Euclidean plane. In the standard domination ordering of two-dimensional vectors, $(x_1, y_1) \leqslant (x_2, y_2)$ holds if and only if $x_1 \leqslant x_2$ and $y_1 \leqslant y_2$. With respect to this partial order, the point $(\min\{a,c\}, \min\{b,d\})$ is the greatest common lower bound for the two points (a,b) and (c,d). It can be verified that in an $\underline{X}$-numbering, for any a, b with $1 \leqslant a, b \leqslant |V(H)|$ the set

$$E(a,b) := \{(i,j) \in E(H) : i \geqslant a \text{ and } j \geqslant b\}$$

is either empty or has a unique smallest element with respect to the domination partial order. If the set $E(a,b)$ is non-empty, then we denote by $x(a,b)$ the x-coordinate and by $y(a,b)$ the y-coordinate of its smallest element. And if the set $E(a,b)$ is empty, then we set $x(a,b) = +\infty$ and $y(a,b) = +\infty$. We observe that $x(a,b)$ and $y(a,b)$ both are monotone functions on $\{1, \ldots, |V(H)|\}$.

Now we will show that for a graph H with an $\underline{X}$-numbering, the H-coloring problem can be formulated as a MON-CSP of type (P1). We use the domain $\mathscr{D} = \{1, \ldots, |V(H)|\}$ with the natural ordering of the integers. For every vertex $v \in V(G)$, there is a corresponding variable v with domain $D_v = \mathscr{D}$. For every arc $(u,v) \in E(G)$, we introduce the two lower bound constraints $u \geqslant x(u,v)$ and $v \geqslant y(u,v)$. By the above

observation, all functions in the lower bounds are monotone. Moreover, the set C^+ is empty. The total size of all variable domains is $|V(G)| \cdot |V(H)| = O(|V(G)|)$, and the time needed for checking whether a given coloring satisfies all constraints is $O(|E(G)|)$.

Lemma 8. *Let (u,v) be an arc in $E(G)$. If the variables u and v satisfy the constraints $u \geqslant x(u,v)$ and $v \geqslant y(u,v)$, then $u = x(u,v)$ and $v = y(u,v)$ holds.*

As a consequence of this lemma, the variables u and v satisfy the two constraints if and only if the arc (u,v) is legally colored. With this, Theorem 1 implies a polynomial time algorithm for H-coloring. In fact the algorithm resulting from Section 2 is structurally different and slightly simpler than the algorithm in [3].

6. Application: monotone integer programming

Integer programming and many of its special cases are NP-complete problems. Two of its most restricted special cases are: IP2, integer programming with two variables per inequality. And MON-IP2, integer programming with two variables per inequality where every inequality is *monotone*, i.e., of the form

$$c \leqslant ax_i - bx_j, \tag{4}$$

where the coefficients a and b both are non-negative. Moreover, for every variable x_i there are prespecified lower and upper bounds

$$\ell_i \leqslant x_i \leqslant u_i. \tag{5}$$

Lagarias [9] has proved that deciding feasibility of MON-IP2 systems over the integers is NP-complete. Hochbaum and Naor [5] observe that IP2 polynomially reduces to MON-IP2. Moreover, [5] presents a pseudo-polynomial time algorithm for MON-IP2 with time complexity $O(md)$; here m is the number of constraints of type (4), and $d = \sum_i (u_i - \ell_i)$ is the total size of all variable domains.

Interestingly, MON-IP2 can be formulated both, as a MON-CSP of type (P1) and as a MON-CSP of type (P2). As domain $\mathscr{D}$ we use the integers $\mathscr{Z}$ with the natural ordering. For every variable x_i we set $D_i = \{z \in \mathscr{Z} \mid \ell_i \leqslant z \leqslant u_i\}$ according to (5). To get a formulation of type (P1), we put for every inequality (4) the lower bound constraint $x_i \geqslant (bx_j + c)/a$ on x_i into C^-. And to get a formulation of type (P2), we put the upper bound constraint $x_j \leqslant (ax_i - c)/b$ on x_j into C^+. The resulting two MON-CSP formulations clearly are equivalent to the inequality systems in (4) and (5). All the functions in the lower (respectively, upper) bounds are monotone functions, and the set C^+ (respectively, the set C^-) is empty. The total size of all variable domains is d, and the time $T(n,m)$ needed for checking whether a given vector satisfies all inequalities is $O(m)$.

With this, Theorem 1 implies the pseudo-polynomial time solvability of MON-IP2 in $O(md)$ time. In fact, the algorithm resulting from our proof in Section 2 is exactly the same algorithm as described in [5] for MON-IP2.

7. Application: AND/OR project scheduling

In AND/OR project scheduling, an instance consist of a set $P = \{1,\ldots,n\}$ of projects with durations $p_1,\ldots,p_n$ together with a set of precedence constraints. Every precedence constraint c is specified by a time lag d_c, and by a pair (Q,i) where $Q \subseteq P$ is a subset of the projects and where $i \in P \setminus Q$ is some fixed project. If c is an AND precedence constraint, then project i can only be started d_c time units after *all* of the projects in Q have been completed. If we denote by S_i the starting time of project i, then an AND precedence constraint c may be formulated as

$$S_i \geqslant \max_{q \in Q} \{S_q + p_q + d_c\}. \tag{6}$$

And in case c is an OR precedence constraint, then project i can only be started d_c time units after *one* of the projects in Q has been completed. This translates into

$$S_i \geqslant \min_{q \in Q} \{S_q + p_q + d_c\}. \tag{7}$$

The problem is to decide whether all projects can be implemented so that the AND/OR precedence constraints all are obeyed. The potentially cyclic structure of the OR precedence constraints makes this problem very difficult. Moehring et al. [10] present a pseudo-polynomial time algorithm for this problem, and they also give polynomial time algorithms for some special cases. It is an outstanding open question whether this project scheduling problem can be solved

in polynomial time. The problem is known to be polynomial time equivalent to certain problems for mean payoff games [14] and to feasibility problems for dynamic min–max systems [12], but the computational complexity of all these problems is not understood.

Lemma 9. *Let $\Delta = \sum d_c$ denote the sum of the time lags over all precedence constraints. If there exists a feasible solution to the AND/OR project scheduling problem, then there exists a feasible solution in which all projects start between time* 0 *and time* Δ.

We now show that the AND/OR project scheduling problem can be formulated as a MON-CSP of type (P1). We use the domain $\mathscr{D} = \{0, \ldots, \Delta\}$ with the natural ordering of the integers (where Δ is defined as in Lemma 9). For every project $i \in P$, there is one variable S_i with domain $D_i = \mathscr{D}$. The constraints are just as specified in (6) and (7). All these constraints are lower bound constraints with monotone functions. The set C^+ is empty. The total size of all variable domains is $O(n\Delta)$, and the time $T(n,m)$ needed for checking whether a given vector satisfies all inequalities is $O(mn)$.

With this, Theorem 1 implies the pseudo-polynomial time solvability of AND/OR project scheduling in $O(mn^2\Delta)$ time.

References

[1] W.F. Dowling, J.H. Gallier, Linear time algorithms for testing satisfiability of propositional Horn formulae, J. Logic Programming 1 (1984) 267–284.

[2] M.R. Garey, D.S. Johnson, Computers and Intractability: a Guide to the Theory of NP-Completeness, Freeman, San Francisco, 1979.

[3] W. Gutjahr, E. Welzl, G.J. Woeginger, Polynomial graph-colorings, Discrete Appl. Math. 35 (1992) 29–45.

[4] P. Hell, J. Nešetřil, On the complexity of H-coloring, J. Combin. Theory (B) 48 (1990) 92–110.

[5] D.S. Hochbaum, J. Naor, Simple and fast algorithms for linear and integer programs with two variables per inequality, SIAM J. Comput. 23 (1994) 1179–1192.

[6] P.G. Jeavons, D. Cohen, M.C. Cooper, Constraints, consistency, and closure, Artificial Intelligence 101 (1998) 251–265.

[7] P.G. Jeavons, D. Cohen, M. Gyssens, Closure properties of constraints, J. ACM 44 (1997) 527–548.

[8] P.G. Jeavons, M.C. Cooper, Tractable constraints on ordered domains, Artificial Intelligence 79 (1995) 327–339.

[9] J.C. Lagarias, The computational complexity of simultaneous diophantine approximation problems, SIAM J. Comput. 14 (1985) 196–209.

[10] R.H. Moehring, M. Skutella, F. Stork, Scheduling with AND/OR precedence constraints, Report No. 689/2000, TU Berlin, Germany, 2000.

[11] C.H. Papadimitriou, Computational Complexity, Addison-Wesley, Reading, MA, 1994.

[12] U. Schwiegelshohn, L. Thiele, Dynamic min–max problems, Discrete Event Dynamic Systems 9 (1999) 111–134.

[13] P. van Hentenryck, Y. Deville, C.-M. Teng, A generic arc-consistency algorithm and its specializations, Artificial Intelligence 57 (1992) 291–321.

[14] U. Zwick, M. Paterson, The complexity of mean payoff games on graphs, Theoret. Comput. Sci. 158 (1996) 343–359.