Hybrid tractability of valued constraint problems✩

Martin C. Coopera, Stanislav Živnýb,c,*

a IRIT, University of Toulouse III, 31062 Toulouse, France
b University College, University of Oxford, Oxford OX1 4BH, UK
c Computing Laboratory, University of Oxford, Oxford OX1 3QD, UK

A R T I C L E   I N F O

Article history:
Received 24 August 2010
Received in revised form 6 December 2010
Accepted 19 February 2011
Available online 24 February 2011

Keywords:
Constraint optimisation
Computational complexity
Tractability
Soft constraints
Valued constraint satisfaction problems
Graphical models
Forbidden substructures

A B S T R A C T

The constraint satisfaction problem (CSP) is a central generic problem in computer science and artificial intelligence: it provides a common framework for many theoretical problems as well as for many real-life applications. Valued constraint problems are a generalisation of the CSP which allow the user to model optimisation problems. Considerable effort has been made in identifying properties which ensure tractability in such problems. In this work, we initiate the study of hybrid tractability of valued constraint problems; that is, properties which guarantee tractability of the given valued constraint problem, but which do not depend only on the underlying structure of the instance (such as being tree-structured) or only on the types of valued constraints in the instance (such as submodularity). We present several novel hybrid classes of valued constraint problems in which all unary constraints are allowed, which include a machine scheduling problem, constraint problems of arbitrary arities with no overlapping nogoods, and the SoftAllDiff constraint with arbitrary unary valued constraints. An important tool in our investigation will be the notion of forbidden substructures.

1. Introduction

An instance of the constraint satisfaction problem (CSP) consists of a collection of variables which must be assigned values subject to specified constraints. Each CSP instance has an underlying hypergraph, known as its constraint hypergraph, whose vertices are the variables of the instance and whose hyperedges are the scopes of the constraints. Such a hypergraph is also known as the structure of the instance.

An important line of research on the CSP is to identify all tractable cases which are recognisable in polynomial time. Most of this work has been focused on one of the two general approaches: either identifying forms of constraint which are sufficiently restrictive to ensure tractability no matter how they are combined [1,2], or else identifying structural properties of constraint networks which ensure tractability no matter what forms of constraint are imposed [3].

The first approach has led to identifying certain algebraic properties known as polymorphisms [4] which are necessary for a set of constraint types to ensure tractability. A set of constraint types which ensures tractability is called a tractable constraint language. The second approach has been used to characterise all tractable cases of bounded-arity CSPs (such as binary CSPs): the only class of structures which ensures tractability (subject to certain complexity theory assumptions) are essentially structures of bounded tree-width [5,6].

✩ A preliminary version of part of this work appeared in Proceedings of the 16th International Conference on Principles and Practice of Constraint Programming (CP), LNCS, vol. 6308, 2010, pp. 152–166. This work was supported by EPSRC grant EP/F01161X/1. Stanislav Živný was supported by Junior Research Fellowship at University College, Oxford.
* Corresponding author at: Computing Laboratory, University of Oxford, Wolfson Building, Parks Road, Oxford OX1 3QD, UK. Tel: +44 (0)1865 273884.
E-mail addresses: cooper@irit.fr (M.C. Cooper), standa.zivny@comlab.ox.ac.uk (S. Živný).

© 2011 Elsevier B.V. All rights reserved.
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. Nevertheless, they may still have properties which ensure they can be solved efficiently, but these properties concern both the structure and the form of the constraints. Such properties have sometimes been called *hybrid* reasons for tractability [7–11].

Since in practice many constraint satisfaction problems are over-constrained, and hence have no solution, or are under-constrained, and hence have many solutions, soft constraint satisfaction problems have been studied [7]. Several very general soft CSP frameworks have been proposed in the literature [12,13]. In this paper we focus on one of the very general frameworks, the valued constraint satisfaction problem (VCSP) [12]. In an instance of the valued CSP, every constraint is associated with a cost function (rather than a relation as in the CSP) which allows the user to express preferences among different partial assignments, and the goal is to find an optimal assignment (i.e. an assignment of smallest total cost).

Similarly to the CSP, an important line of research on the VCSP is to identify tractable cases which are recognisable in polynomial time. It is well known that structural reasons for tractability generalise to the VCSP [7]. In the case of language restrictions, only a few conditions are known to guarantee tractability of a given set of valued constraints [14–16,22].

Up until now there have been very few results on hybrid tractability for the VCSP, that is, tractability of subproblems of the VCSP defined by properties which are not exclusively language-based or structure-based. For instance, Kumar defines an interesting framework for hybrid tractability for the Boolean weighted CSP [10]. However, to the best of our knowledge, this framework has so far not provided any new hybrid classes. In fact, all tractable classes presented in [10] are not hybrid and are already known.

**Contributions.** The main contribution of the paper is the study of hybrid tractability of VCSPs and the introduction of novel hybrid tractable classes of VCSPs. As a first step, we start with binary VCSPs.

We introduce the class defined by the *joint-winner property* (JWP). This class generalises the SoftAllDiff constraint with arbitrary unary valued constraints. Moreover, we generalise this class to a larger class of non-binary VCSPs including CSP and MAX-CSP instances with no overlapping nogoods.

The rest of the paper is organised as follows. In Section 2, we define binary constraint satisfaction problems (CSPs), valued constraint satisfaction problems (VCSPs) and other necessary definitions needed throughout the paper. In Section 3, we study binary VCSPs whose only soft constraints are unary. A connection between these VCSPs and the maximum weight independent set problem in certain graph classes leads in a straightforward manner to corresponding hybrid tractable classes of VCSPs. In Section 4, we define the joint-winner property and give several examples of studied problems that satisfy the joint-winner property. In Section 5, we study important properties of VCSP instances satisfying the joint-winner property, which allow us, in Section 6, to present a polynomial-time algorithm for solving binary VCSPs satisfying the joint-winner property. In Section 7, we prove that this new tractable class is maximal. In Section 8, we extend the class of tractable VCSPs defined by the joint-winner property. Finally, in Section 9, we summarise our work and finish with some open problems.

We remark that even though our results are formulated as valued constraint satisfaction problems, it is clear that these results apply to various other optimisation frameworks that are equivalent to valued constraint problems such as Gibbs energy minimisation, Markov Random Fields and other graphical models [17,18].

2. Preliminaries

In this paper we firstly focus on binary valued constraint satisfaction problems before generalising to problems with cost functions of arbitrary arity. We denote by \( \mathbb{Q}_+ \) the set of all non-negative rational numbers. We denote \( \mathbb{Q}_+ = \mathbb{Q}_+ \cup \{\infty\} \) with the standard addition operation extended so that for all \( a \in \mathbb{Q}_+ \), \( a + \infty = \infty \). Members of \( \mathbb{Q}_+ \) are called costs.

A unary cost function over domain \( D_i \) is a mapping \( c_i : D_i \rightarrow \mathbb{Q}_+ \). A binary cost function over domains \( D_i \) and \( D_j \) is a mapping \( c_{ij} : D_i \times D_j \rightarrow \mathbb{Q}_+ \). If the range of \( c_i \) (\( c_{ij} \) respectively) lies entirely within \( \mathbb{Q}_+ \), then \( c_i \) (\( c_{ij} \) respectively) is called a *finite-valued* cost function.

If the range of \( c_i \) (\( c_{ij} \) respectively) is \( [\alpha, \infty) \), for some \( \alpha \in \mathbb{Q}_+ \), then \( c_i \) (\( c_{ij} \) respectively) is called a *crisp* cost function. Note that crisp cost functions are just relations; that is, subsets of \( D_i \) (in the unary case) or \( D_i \times D_j \) (in the binary case) corresponding to the set of finite-cost tuples. If \( c_i \) (\( c_{ij} \) respectively) is not a crisp cost function, it is called soft.

A binary VCSP instance [12] consists of a set of variables (denoted as \( v_i \), where \( i \in \{1, \ldots, n\} \); for each variable \( v_i \) a domain \( D_i \) containing possible values for variable \( v_i \); and a set of valued constraints. Each valued constraint is either of the form \( (v_i, c_i) \), where \( v_i \) is a variable and \( c_i \) is a unary cost function (constraints of this form are called *unary* constraints), or of the form \( (\langle v_i, v_j \rangle, c_{ij}) \), where \( v_i \) and \( v_j \) are variables with \( i < j \), the pair \( \langle v_i, v_j \rangle \) is called the *scope* of the constraint, and \( c_{ij} \) is a binary cost function (constraints of this form are called *binary* constraints). A constraint is called crisp if its associated cost function is crisp, and similarly a constraint is called soft if its associated cost function is soft. For notational convenience, throughout this paper we assume that there is a unique valued constraint on any given scope. In particular, in the absence of an explicit constraint between two variables \( v_i, v_j \) \( (i < j) \), we assume that the corresponding cost function is the constant function \( c_{ij} = 0 \).

A solution to aVCSP instance is an assignment of values from the domains to the variables with the minimum total cost given by

\[
\sum_{i=1}^{n} c_i(v_i) + \sum_{1 \leq i < j \leq n} c_{ij}(v_i, v_j).
\]
A VCSP instance with only crisp constraints is called a CSP instance. In the CSP, the task of finding an optimal assignment amounts to deciding whether there is a consistent assignment (in which all constraints are satisfied), i.e., an assignment with finite cost. Hence, each CSP instance is equivalent to a normalised version in which all finite costs are replaced by 0.

The micro-structure of a binary CSP instance $\mathcal{P}$ is a graph where the set of vertices corresponds to the set of possible assignments of values to variables: a vertex $(v_i, a)$ represents the assignment of value $a$ to variable $v_i$ [19]. The edges of the micro-structure connect all pairs of variable-value assignments $\{(v_i, a), (v_j, b)\}$ such that $i < j$ and $c_{ij}(a, b) < \infty$. The micro-structure of a binary VCSP instance is defined similarly. In the weighted micro-structure of a binary VCSP instance, edges and vertices are assigned the corresponding binary and unary costs. For CSPs (and VCSPs), the micro-structure complement is the complement of the micro-structure: its edges represent pairs of variable-value assignments that are incompatible according to the constraints. The micro-structure complement thus contains all edges $\{(v_i, a), (v_j, b)\}$ such that $i < j$ and $c_{ij}(a, b) = \infty$, but also, for every variable $v_i$, all edges of the form $\{(v_i, a), (v_i, b)\}$ for $a \neq b \in D_i$ as every variable can be assigned only one value.

A clique in a graph is a set of vertices which are pairwise adjacent. An independent set in a graph is a set of vertices which are pairwise non-adjacent. It is well known that solving an $n$-variable CSP instance $\mathcal{P}$ is equivalent to finding a clique of size $n$ in the micro-structure of $\mathcal{P}$, and to finding an independent set of size $n$ in the micro-structure complement of $\mathcal{P}$ [19]. Therefore, tractability results on the maximum independent set problem for various classes of graphs can be straightforwardly used to obtain tractable CSP classes [8,20].

Given a graph $G$, we denote by $V(G)$ the set of vertices of $G$ and by $E(G)$ the set of edges of $G$. A coloured graph $G = (G, c_G)$ is a graph $G$ and a colouring $c_G : V(G) \rightarrow \{1, \ldots, n\}$ of the vertices of $G$. The coloured micro-structure (complement) of an $n$-variable binary CSP or VCSP instance $\mathcal{P}$ is $\mathcal{G} = (G, c_G)$ where $G$ is the micro-structure (complement) of $\mathcal{P}$ and $c_G$ is a colouring $c_G : V(G) \rightarrow \{1, \ldots, n\}$ of the vertices of $G$, such that the colour of vertex $(v_i, a)$ is $i$. Unlike the micro-structure, the coloured micro-structure contains all the information necessary to reconstruct a CSP instance $\mathcal{P}$. Similarly, the weighted version of the coloured micro-structure of a VCSP instance, in which edges and vertices are assigned costs, contains enough information to reconstruct a VCSP instance.

A graph $H$ is an induced subgraph of $G$ if there is an injective mapping $f : V(H) \rightarrow V(G)$ such that $\{u, v\} \in E(H)$ if, and only if, $\{f(u), f(v)\} \in E(G)$. We now extend the notion of induced subgraphs to that of induced and forbidden substructures of the coloured micro-structure and coloured micro-structure complement. Let $\mathcal{G} = (G, c_G)$ be the coloured micro-structure or coloured micro-structure complement of a CSP instance $\mathcal{P}$. Now let $\mathcal{H} = (H, c_H)$ be a coloured graph. We say that $\mathcal{H}$ is an induced substructure of $\mathcal{G}$ if $H$ is an induced subgraph of $G$ and vertices of the same colour are mapped to vertices of the same colour; that is, if there is an injective mapping $f : V(H) \rightarrow V(G)$ such that $\{1\} \{u, v\} \in E(H)$ if, and only if, $\{f(u), f(v)\} \in E(G)$; and (2) $c_H(u) = c_H(v)$ if, and only if, $c_G(f(u)) = c_G(f(v))$. In this paper we will present classes of VCSPs that will be defined by forbidding a certain substructure (called a pattern) in the coloured micro-structure or the coloured micro-structure complement of the VCSP instance. In figures, we will often draw ovals around vertices that are assigned the same colour (corresponding to the assignments to the same variable).

3. Conservative VCSPs with crisp binary constraints

In this section we restrict our attention to binary VCSP instances with crisp binary constraints. There are no restrictions on unary constraints; hence both crisp and soft unary constraints are allowed.

Definition 1. A class of VCSP instances is conservative if it is closed under the addition of arbitrary unary valued constraints.

It follows from a result of Takhanov [21] that there is a P/NP-hard dichotomy for conservative binary VCSPs in which all binary constraints are crisp and belong to a particular language. Furthermore, it follows from a recent result of Kolmogorov and Živný [22] that there is a P/NP-hard dichotomy for all conservative binary VCSPs in which all binary (not necessarily crisp) constraints belong to a particular language. It is an open question whether a similar dichotomy exists for hybrid classes.

3.1. Maximum weight independent set

We show how tractability results on the maximum weight independent set in perfect, fork-free and apple-free graphs can be used to obtain hybrid tractable classes of conservative VCSPs.

Let $G$ be a graph $G = (V, E)$ with weights $w : V \rightarrow \mathbb{Q}_+$ on the vertices of $G$. The weight of an independent set $S$ in $G$, denoted $w(S)$, is the sum of weights of the vertices in $S$: $w(S) = \sum_{v \in S} w(v)$. It is easy to show that given a binary VCSP instance $\mathcal{P}$ where only unary constraints can be soft, $\mathcal{P}$ can be solved by finding a maximum weight independent set in the micro-structure complement of $\mathcal{P}$ with weights given by $w((v_i, a)) = M - c_i(a)$, where $M$ is strictly greater than the maximum finite unary cost $c_i(a)$. Indeed, independent sets of weight strictly greater than $Mn(n - 1)$ are in one-to-one correspondence with consistent assignments to $n$ variables in $\mathcal{P}$ (since $\forall i, a, 0 \leq c_i(a) < M$ implies that the weight of an

[1] We remark that both dichotomy results [21,22] hold for non-binary VCSPs, and thus are not restricted to only binary VCSPs.
independent set of size \( n \) is strictly greater than \( n(Mn - M) = Mn(n - 1) \) and the weight of an independent set of size \( n - 1 \) is at most \( Mn(n - 1) \).

Several well-studied classes of graphs admit a polynomial-time algorithm for the maximum weight independent set problem (MWIS). For example, it is known that MWIS in perfect graphs is solvable in polynomial time [23]. Moreover, perfect graphs can be recognised in polynomial time [24]. Similar results hold for fork-free [25] and apple-free graphs [26]. The combination of these results gives:

**Theorem 2.** The class of VCSP instances with crisp binary and arbitrary unary constraints whose micro-structure complement is either perfect, fork-free or apple-free is tractable.

The tractability of VCSPs with perfect micro-structure and soft unary constraints was also independently pointed out by Takhanov [21] and Jebara [27].

It is not possible to express all properties of VCSP or CSP instances uniquely in terms of properties of the micro-structure, since the micro-structure is a graph which does not contain the important information telling us which of its vertices correspond to assignments to the same variables. This information is, however, contained in the coloured micro-structure. This is why in the rest of the paper we consider properties of the (weighted) coloured micro-structure in order to define novel tractable hybrid classes.

### 3.2. Broken-triangle property

We have seen that forbidding certain patterns in the micro-structure guarantees the tractability of certain classes of VCSPs. The recently discovered broken-triangle property (BTP) [11], which is equivalent to forbidding a pattern in the coloured micro-structure of a CSP, guarantees that a feasible solution can be found in polynomial time. In this section, we show that the hybrid tractable class of CSPs defined by the broken-triangle property is not extendible to VCSPs with arbitrary soft unary constraints. A binary CSP instance \( \mathcal{P} \) satisfies the broken-triangle property with respect to the variable ordering \( < \), and only if, for all triples of variables \( v_i, v_j, v_k \) such that \( i < j < k \), if \( c_{ij}(u, v) < \infty \), \( c_{ik}(u, a) < \infty \) and \( c_{jk}(v, b) < \infty \), then either \( c_{ik}(u, b) < \infty \) or \( c_{jk}(v, a) < \infty \). (In other words, every “broken” triangle \( a - u - v - b \) can be closed.) This is equivalent to forbidding the pattern from Fig. 1 as an induced substructure in the coloured micro-structure of \( \mathcal{P} \).

**Theorem 3.** Assuming \( P \neq NP \), there is no tractable conservative class of VCSP instances which includes all CSP instances satisfying the broken-triangle property.

**Proof.** Let \((G, k)\) be an instance of the decision version of the maximum independent set problem which consists in deciding whether there is an independent set of size at least \( k \) in graph \( G \). This problem is known to be NP-complete [28]. We now transform this instance into a binary VCSP instance with soft unary constraints that satisfies the broken-triangle property.

Every vertex of \( G \) is represented by a Boolean variable \( v_i \) where \( D_i = \{0, 1\} \). We impose the constraint \( (\langle v_i, v_j \rangle, \{0, 0\}, \{0, 1\}, \{1, 0\}) \) if the vertices corresponding to \( v_i \) and \( v_j \) are adjacent in \( G \). Now the assignments satisfying all constraints are in one-to-one correspondence with independent sets \( I \) in \( G \), where vertex \( i \in I \) if, and only if, \( v_i = 1 \). We also impose the soft unary constraints \( (v_i, c_i) \), where \( c_i(x) = 1 - x \) for \( x \in \{0, 1\} \). The unary constraints ensure that the goal is to maximise the number of variables assigned value 0, which is the same as maximising the number of variables assigned value 1. Therefore, the constructed VCSP instance is equivalent to the given maximum independent set problem. It remains to show that the resulting VCSP instance satisfies the broken-triangle property with respect to some ordering. In fact, we show that it is satisfied with respect to any ordering. Take any three variables \( v_i, v_j, v_k \) such that \( i < j < k \). If either of the pairs of variables \( \langle v_i, v_k \rangle, \langle v_j, v_k \rangle \) are not constrained, then the broken-triangle property is trivially satisfied. Assume therefore that these two constraints are present. The situation is illustrated in Fig. 2. It can be easily checked that the broken-triangle property is indeed satisfied whether the constraint on \( \langle v_i, v_j \rangle \) is \( \{0, 0\}, \{0, 1\}, \{1, 0\} \) (as shown in Fig. 2) or the complete constraint. \( \square \)
4. Joint-winner property

In this section we define the joint-winner property (see Fig. 3), which is the key concept in this paper. We present several examples of known tractable problems that are generalised by the joint-winner property. In Section 5 we study basic properties of VCSPs satisfying the joint-winner property as these will be important in designing a polynomial-time algorithm in Section 6.

Recall that we assume that in a binary VCSP there is a unique cost function on all unordered pairs of distinct variables \( \{v_i, v_j\} \). For simplicity of presentation, we can therefore write \( c_{ij}(a, b) \), even if the order of \( i \) and \( j \) is unknown, to represent the unique binary cost of assigning \( (a, b) \) to \( (v_i, v_j) \).

**Definition 4 (Joint-winner property).** A triple of variables \( \langle v_i, v_j, v_k \rangle \) satisfies the joint-winner property (JWP) if \( c_{ij}(a, b) \geq \min(c_{ik}(a, c), c_{jk}(b, c)) \) for all domain values \( a \in D_i, b \in D_j, c \in D_k \).

A binary VCSP instance satisfies the joint-winner property if every triple of distinct variables of the instance satisfies the joint-winner property.

Note that the joint-winner property places no restrictions on the unary constraints \( c_i \) and is hence conservative.

**Example 5.** The joint-winner property in a normalised binary CSP instance amounts to forbidding the multiset of binary costs \([0, \infty, \infty]\) in any triangle formed by three assignments to distinct variables. Recall that a normalised CSP is a VCSP with only zero or infinite costs. Equivalently (given that the cost 0 corresponds to compatible assignments and thus non-edges in the micro-structure complement), a CSP instance \( \mathcal{P} \) satisfies the joint-winner property if, and only if, the coloured micro-structure complement of \( \mathcal{P} \) forbids the structure in Fig. 4 as an induced substructure. Since this combination can never occur on triples of variables \( \langle v_i, v_j, v_k \rangle \) constrained by the three binary constraints \( v_i \neq v_j \neq v_k \neq v_i \), the class of CSPs satisfying the joint-winner property generalises the AllDifferent constraint with arbitrary unary valued constraints. This generalisation is equivalent to allowing at most one assignment from each of a set of disjoint sets of (variable,value) assignments.
Example 6. Consider the (unweighted) MAX-2SAT problem with no repeated clauses, which is a well-known NP-complete problem [28,29]. An instance \( \phi \) of MAX-2SAT, where \( \phi \) is a 2-CNF formula, can be seen as a Boolean binary \([0, 1]\)-valued VCSP instance. The joint-winner property is equivalent to the following condition on \( \phi \): if \((l \lor l_1) \in \phi \) and \((l \lor l_2) \in \phi \), then \((l_1 \lor l_2) \in \phi \), where \( l, l_1, l_2 \) are three literals of distinct variables.

It is well known that the set of VCSP instances comprising a single SoftAllDiff constraint and arbitrary unary valued constraints is tractable. Furthermore, this class of instances is hybrid in that tractability does not follow uniquely from the language of cost functions nor functions uniquely from the structure of the instance. As the next example shows, this class satisfies the JWP, thus showing that the JWP defines a hybrid tractable class (the tractability of the JWP being shown later in Section 6).

Example 7. One of the most commonly used global constraints is the AllDifferent constraint [30]. Petit et al. introduced a soft version of the AllDifferent constraint, SoftAllDiff [31]. They proposed two types of costs to be minimised: graph- and variable-based costs. The former counts the number of equalities, whilst the latter counts the minimum number of variables that need to change value in order to satisfy the AllDifferent constraint. The algorithms for filtering these constraints, introduced in the same paper, were then improved by van Hoeve et al. [32].

It is easy to check that the graph-based variant of the SoftAllDiff constraint satisfies the joint-winner property. In this case for every \( i \) and \( j \), the cost function \( c_{ij} \) is defined as \( c_{ij}(a, b) = 1 \) if \( a = b \), and \( c_{ij}(a, b) = 0 \) otherwise. Take any three variables \( v_i, v_j, v_k \) and \( a \in D_i, b \in D_j, c \in D_k \). If \( c_{ij}(a, b) = c_{jk}(b, c) = c_{ik}(a, c) \) (which means that the domain values \( a, b, c \) are all equal or all different), then the joint-winner property is satisfied trivially. If only one of the costs is 1, then the joint-winner property is satisfied as well. Observe that due to the transitivity of equality it cannot happen that only one of the binary costs shown in the triangle of Fig. 3 is zero, from which it follows immediately that the joint-winner property is satisfied in \( P \).

We remark that the class defined by the JWP is strictly bigger than SoftAllDiff, and hence our generic algorithm is not as efficient as tailor-made algorithms for SoftAllDiff [32,33]. Several generalisations of the AllDifferent constraints have been studied [34,35], but these do not satisfy the JWP property.

Example 8. Suppose that \( n \) jobs must be assigned to \( d \) machines. Let \( l_i(m) \) be the length of time required for machine \( m \) to complete job \( i \). If machine \( m \) cannot perform job \( i \), then \( l_i(m) = \infty \). We use variable \( v_i \) to represent the machine to which job \( i \) is assigned. The set of jobs (which we denote by \( S_m \)) assigned to the same machine \( m \) are performed in series in increasing order of their length \( l_i(m) \). The aim is to minimise \( T \) the sum, over all jobs, of their time until completion. If jobs \( i \) and \( j \) are assigned to the same machine, and \( l_i(m) < l_j(m) \), then job \( j \) will have to wait during the execution of job \( i \), contributing a time of \( l_i(m) \) to the sum \( T \). This means that

\[
T = \sum_{m=1}^{d} \left( \sum_{i \in S_m} l_i(m) + \sum_{i \neq j \in S_m} \min(l_i(m), l_j(m)) \right).
\]

In other words, optimal assignments of jobs to machines are given by solutions to the binary VCSP with unary constraints \( c_i(m) = l_i(m) \) (representing the execution times of jobs) and binary constraints

\[
c_{ij}(m, m') = \begin{cases} \min(l_i(m), l_j(m)) & \text{if } m = m', \\ 0 & \text{otherwise} \end{cases}
\]

( representing the waiting times of jobs).

The joint-winner property \( c_{ij}(a, b) \geq \min(c_{ik}(a, c), c_{jk}(b, c)) \) is trivially satisfied in this VCSP instance if \( a, b, c \) are not all equal, since in this case one of \( c_{ik}(a, c), c_{jk}(b, c) \) is zero. It is also satisfied when \( a = b = c = m \) since \( \min(l_i(m), l_j(m)) \geq \min(l_i(m), l_k(m)), \min(l_j(m), l_k(m)) \).

This problem has been shown solvable in polynomial time in [36,37].

5. Structure of problems satisfying the JWP

The next lemma explains the reason for the name of the joint-winner property: in every triangle there is no unique smallest cost.
Lemma 9. A binary VCSP instance satisfies the joint-winner property if, and only if, for all triples of distinct variables \((v_1, v_j, v_k)\) and for all \(a \in D_i, b \in D_j, c \in D_k\), two of the costs \(c_{ij}(a, b), c_{ik}(a, c), c_{jk}(b, c)\) are equal and less than or equal to the third.

Proof. Assume that the joint-winner property is satisfied on the triples of variables \(\{v_1, v_j, v_k\}, \{v_j, v_k, v_i\}\), and \(\{v_k, v_i, v_j\}\). We write \(\alpha = c_{ij}(a, b), \beta = c_{ik}(a, c), \text{ and } \gamma = c_{jk}(b, c)\). Without loss of generality, let \(\alpha = \min(\alpha, \beta, \gamma)\). From \(\alpha \leq \min(\beta, \gamma)\), we can deduce that \(\alpha = \min(\beta, \gamma)\) and hence that two of \(\alpha, \beta, \gamma\) are equal and less than or equal to the third.

On the other hand, if two of \(\alpha, \beta, \gamma\) are equal and less than or equal to the third, then \(\min(\beta, \gamma) = \min(\alpha, \beta, \gamma) \leq \alpha\) and the JWP is satisfied. □

We say that the binary constraint on variables \(v_j, v_j\) has a Z-configuration on sub-domains \(\{a, b\} \subseteq D_i, \{c, d\} \subseteq D_j\) if \(\min(c_{ij}(a, c), c_{ij}(b, c), c_{ij}(b, d)) > c_{ij}(a, d)\). Since the JWP imposes a condition on exactly three variables, it imposes no condition on individual binary constraints. In particular, Z-configurations may occur. However, we will show that all Z-configurations can be eliminated by a simple pre-processing step. This will greatly simplify the structure of the instance allowing us to apply a min-cost max-flow algorithm to solve JWP instances.

We say that the pair of sub-domains \((S_i, S_j)\) (where \(S_i \subseteq D_i, S_j \subseteq D_j\)) is independent of the other variables if \(\forall a, b \in S_i, \forall c, d \in S_j, \forall k \notin \{i, j\}, \forall e \in D_k, c_{ik}(a, e) = c_{ik}(b, e) = c_{jk}(c, e) = c_{jk}(d, e)\).

Lemma 10. If a binary VCSP instance satisfying the JWP has a Z-configuration on sub-domains \([a, b], [c, d]\) is independent of the other variables.

Proof. Without loss of generality, the presence of a Z-configuration means that \(\min(c_{ij}(a, c), c_{ij}(b, c), c_{ij}(b, d)) > \beta\) where \(c_{ij}(a, d) = \beta\). Consider some \(k \notin \{i, j\}\) and some \(e \in D_k\). By the JWP on the triple of assignments \(\{v_1, a\}, \{v_j, d\}, \{v_k, e\}\), we have either (1) \(c_{ik}(a, e) \leq \beta\) or (2) \(c_{jk}(d, e) \leq \beta\). In case (1), by repeated application of the JWP on the triple of assignments \(\{v_1, a\}, \{v_j, c\}, \{v_k, e\}\), we can deduce that \(c_{ik}(a, e) = c_{ik}(b, e) = c_{jk}(d, e)\). In case (2), applying the JWP on the same triple, but in the reverse order, again allows us to deduce \(c_{ik}(a, e) = c_{ik}(c, e) = c_{jk}(d, e)\). □

Lemma 11. In a binary VCSP instance satisfying the JWP, if \((S_i, S_j)\) (where \(S_i \subseteq D_i, S_j \subseteq D_j\)) is independent of the other variables, and \(f \in D_i - S_i\) is such that \(\exists c, d \in S_j\) such that \(c_{ij}(f, c) \neq c_{ij}(f, d)\), then \((S_i \cup f, S_j)\) is independent of the other variables.

Proof. Consider any variable \(k \notin \{i, j\}\) and any \(e \in D_k\). Since \((S_i, S_j)\) is independent of the other variables, \(c_{ik}(c, e) = c_{jk}(d, e)\). By applying the JWP on the triple of assignments \(\{v_1, f\}, \{v_j, c\}, \{v_k, e\}\) and \(\{v_1, f\}, \{v_j, d\}, \{v_k, e\}\), we can deduce that either \(c_{ik}(c, e) = c_{ik}(d, e)\) or \(c_{jk}(c, e) = c_{jk}(d, e)\). This shows that \((S_i \cup f, S_j)\) is independent of the other variables. □

We say that \((S_i, S_j)\) is independent of other domain-values if \(\forall c, d \in S_j, \forall f \in D_i - S_i, c_{ij}(f, c) = c_{ij}(f, d)\) and \(\forall a, b \in S_i, \forall g \in D_j - S_j, c_{ij}(a, g) = c_{ij}(b, g)\).

Given a Z-configuration on \([a, b] \subseteq D_i, [c, d] \subseteq D_j\), under the JWP assumption we can use a simple algorithm to build a maximal pair of sub-domains \((S_i, S_j)\) which is independent of the other variables (such that \([a, b] \subseteq S_i, [c, d] \subseteq S_j\)). From Lemma 10, we can initialise \((S_i, S_j)\) to \([a, b], [c, d]\). Then, by Lemma 11 we can simply keep adding \(f \in D_i - S_i\) to \(S_i\) if \(\exists c, d \in S_j\) such that \(c_{ij}(f, c) \neq c_{ij}(f, d)\), and, by symmetry, we can keep adding \(g \in D_j - S_j\) to \(S_j\) if \(\exists a, b \in S_i\) such that \(c_{ij}(a, g) \neq c_{ij}(b, g)\). When this process converges, \((S_i, S_j)\) is not only independent of the other variables but also independent of other domain-values.

Now let \(p_0 \in S_i\) minimise \(c_{ij}(a)\), where \(a \in S_i\). Similarly, let \(q_0 \in S_j\) minimise \(c_{ij}(a)\), where \(a \in S_j\). Furthermore, let \(p_1 \in S_i\) and \(q_1 \in S_j\) be the pair that minimises \(c_{ij}(a) + c_{ij}(b) + c_{ij}(d)\) for \(a \in S_i\) and \(b \in S_j\). We replace the sub-domains \(S_i, S_j\) by the single domain values \(p, q\) where \(c_{ij}(p) = c_{ij}(p_0), c_{ij}(q) = c_{ij}(q_0)\), and \(c_{ij}(p, q) = c_{ij}(p_1, q_1) + (c_{ij}(p_1) - c_{ij}(p_0)) + (c_{ij}(q_1) - c_{ij}(q_0))\) so that \(c_{ij}(p) + c_{ij}(q) - c_{ij}(p, q) = c_{ij}(p_1, q_1) + c_{ij}(p_1) + c_{ij}(q_1)\). For all \(f \in D_i - S_i, \text{ we set } c_{ij}(f, q) = c_{ij}(f, q_0)\) and \(c_{ij}(p, f) = c_{ij}(p_0, f)\). Also \(\forall k \notin \{i, j\}, \forall u \in D_k, c_{ik}(p, u) = c_{ik}(p_0, u) = c_{ik}(p_1, u)\) and \(c_{ik}(q, u) = c_{ik}(q_0, u) = c_{ik}(q_1, u)\). The pair of domain values \(p, q\) simulate either \(p_0\) or \(q_0\) if only one of the values \(p, q\) is used but simulates \(p_1, q_1\) if both values are used. We observe that this substitution preserves the JWP property. This is trivially true in all triangles involving only one of the assignments \(p, q\), since the binary costs are identical to those for \(p_0, q_0\). Since \((S_i, S_j)\) is independent of other domain-values, any triangle of costs involving both assignments \(p_1, q_1\) is isosceles. In particular, for any \(u \in D_k\), where \(k \notin i\) and \(k \neq j\), \(c_{ik}(p_1, u) = c_{ik}(q_1, u) = \alpha\) and \(c_{ik}(p_1, q_1) \geq \alpha\) (by the JWP). Since \(c_{ik}(p, u) = c_{ik}(p_1, u) = c_{ik}(q, u) = c_{ik}(p, q) \geq c_{ik}(p_1, q_1)\), the JWP property is preserved.

This substitution operation is guaranteed to preserve at least one optimal solution. This is because, by our choice of \(p_0, q_0, p_1, q_1\), (1) in any assignment \(x\) (to all \(n\) variables) in which \(x_i \in S_i, x_j \notin S_j\), replacing \(x_i\) by \(p_0\) (or by \(p\) which is equivalent) cannot increase its cost, (2) in any assignment \(x\) in which \(x_i \notin S_i, x_j \in S_j\), replacing \(x_j\) by \(q_0\) (or by \(q\)) cannot increase its cost, and (3) in any assignment \(x\) in which \(x_i \in S_i, x_j \in S_j\), simultaneously replacing \(x_i\) by \(p_1\) and \(x_j\) by \(q_1\) (or replacing \(x_i\) by \(p\) and \(x_j\) by \(q\)) cannot increase its cost.
We say that a VCSP instance satisfying the JWP is Z-free it contains no Z-configurations. We know that a VCSP satisfying the JWP can be made Z-free by applying the above pre-processing step to eliminate all Z-configurations. This pre-processing takes polynomial time. A naive algorithm requires \( O(n^2d^4) \) time to detect all Z-configurations. For a given pair \((S_i,S_j)\), when a new domain value \( g \) is added to \( S_j \) we have to test for all \( f \in D_1 - S_i \) whether \( \exists c \in S_j \) such that \( c_{ij}(f,c) \neq c_{ij}(f, g) \). This is an \( O(d^2) \) operation. By a simple counting argument, we can see that the total number of times some domain value is added to some set \( S_i \) is less than the total number of \((\text{variable},\text{value})\) assignments which are eliminated by the substitution operation described above, which, in turn, is bounded above by \( nd \). Thus the total complexity of the pre-processing operation is \( O(n^2d^4 + nd^3) \) and hence \( O(n^2d^4) \).

The assignment graph of a binary VCSP instance \( \mathcal{P} \) is the graph \((V_P, E_P)\) with vertices \( V_P = \{ \langle v_i, a \rangle : 1 \leq i \leq n, a \in D_i \}\) and edges \( E_P = \{ \langle (v_i, a), (v_j, b) \rangle : i \neq j \} \). The complete assignment graph is simply the complete graph with vertices \( V_P \).

**Definition 12.** Let \( \mathcal{P} \) be a binary VCSP instance. A subgraph \( C = (V_C, E_C) \) of the assignment graph of \( \mathcal{P} \) is an assignment-clique if \( C \) forms a clique in the complete assignment graph of \( \mathcal{P} \) when all edges between assignments to the same variable are added to \( E_C \), i.e. \((V_C, E_C \cup S_C)\) is a clique, where \( S_C = \{ \langle v_i, a \rangle, \langle v_j, b \rangle : \langle v_i, a \rangle, \langle v_j, b \rangle \in V_C \text{ and } a \neq b \}\).

**Lemma 13.** Let \( \mathcal{P} \) be a Z-free binary VCSP instance satisfying the JWP. Then, for a fixed \( \alpha \), the edges of the assignment graph of \( \mathcal{P} \) corresponding to binary costs of at least \( \alpha \) intersect with the corresponding vertices, form non-intersecting assignment-cliques.

**Proof.** We first show that if \( \mathcal{P} \) is Z-free then within a single binary valued constraint any connected set \( E \) of edges of the assignment graph of \( \mathcal{P} \) corresponding to binary costs of at least \( \alpha \) do form an assignment-clique. This follows from a simple inductive argument on the number \( m_i \) of assignments to \( v_i \), where \( v_i, v_j \) are the two variables in the scope of the constraint. It is trivially true for \( m_i = 1 \). Let \( A_j \) denote the assignments to variable \( v_j \) covered by \( E \). Consider what happens when a new edge is added to \( E \), corresponding to a new assignment to \( v_i \), to produce a new connected set of edges \( E' \) covering \( m_i + 1 \geq 2 \) assignments to \( v_i \). Since \( E' \) is connected, the new assignment \( \langle v_i, a \rangle \) is connected by an edge in \( E' \) to some assignment \( \langle v_j, c \rangle \in A_j \). By the inductive hypothesis, all assignments \( \langle v_j, d \rangle \in A_j \) (including \( \langle v_j, c \rangle \)) are connected by edges in \( E \subset E' \) to all other assignments \( \langle v_i, b \rangle (b \neq a) \). Since \( \mathcal{P} \) is Z-free, this implies that \( \langle v_i, a \rangle \) is also connected in \( E' \) to all assignments \( \langle v_j, d \rangle \neq \langle v_j, c \rangle \in A_j \).

For a contradiction let us assume that the edges of the assignment graph of \( \mathcal{P} \) corresponding to binary costs of at least \( \alpha \) do not form non-intersecting assignment-cliques. Hence there are three vertices \( \langle v_i, a \rangle, \langle v_j, b \rangle, \langle v_k, c \rangle \) (where \( i, j, k \) are distinct) of the assignment graph such that \( c_{ij}(a, b) \geq \alpha \), \( c_{ik}(a, c) \geq \alpha \), and \( c_{jk}(b, c) < \alpha \). But this is in contradiction with Lemma 9. \( \square \)

In the rest of the paper, when we refer to an assignment-clique \( C_\alpha \) in the assignment graph corresponding to binary costs of at least \( \alpha \), we implicitly assume that \( C_\alpha \) is maximal. Since an assignment-clique \( C_\alpha \) is uniquely defined by its set of vertices, we can identify \( C_\alpha \) with its set of vertices.

**Lemma 14.** Let \( \mathcal{P} \) be a Z-free binary VCSP instance satisfying the JWP. Let \( C_\alpha \) be an assignment-clique in the assignment graph of \( \mathcal{P} \) corresponding to binary costs of at least \( \alpha \), and \( C_\beta \) an assignment-clique in the assignment graph of \( \mathcal{P} \) corresponding to binary costs of at least \( \beta \). If \( C_\alpha \) intersects \( C_\beta \) and \( \alpha \leq \beta \), then \( C_\alpha \supseteq C_\beta \).

**Proof.** Suppose that \( C_\alpha \) and \( C_\beta \) intersect and \( \alpha \leq \beta \). If \( \alpha = \beta \), the claim is satisfied trivially by Lemma 13, so we can suppose that \( \alpha < \beta \). For a contradiction, assume that \( C_\alpha \nsubseteq C_\beta \). By our assumptions, \( \exists \langle v_i, a \rangle \in C_\alpha \cap C_\beta \) and \( \exists \langle v_j, b \rangle \in C_\beta \setminus C_\alpha \). Since \( C_\beta \) is an assignment-clique, we must have \( c_{ij}(a, b) \geq \beta > \alpha \). Thus, by Lemma 13, the edge \( \langle v_j, b \rangle, \langle v_i, a \rangle \) is part of an assignment-clique \( C'_\alpha \) of edges of cost at least \( \alpha \) (but not \( C_\alpha \) since \( \langle v_j, b \rangle \notin C_\alpha \)). But then \( C_\alpha \) and \( C'_\alpha \) intersect at \( \langle v_i, a \rangle \) which contradicts Lemma 13. \( \square \)

**6. Algorithm for JWP**

In this section we present a polynomial-time algorithm for solving Z-free binary VCSPs satisfying the joint-winner property. The algorithm is an extension of a reduction to the standard max-flow/min-cut problem that has been used for flow-based soft global constraints [38,32,39].

First we review some basics on flows in graphs. We refer the reader to the standard textbook [40] for more details. Here we present only the notions and results needed for our purposes. In particular, we deal with only integral flows. We denote by \( \mathbb{N} \) the set of positive integers with zero. Let \( G = (V, A) \) be a directed graph with vertex set \( V \) and arc set \( A \). To each arc \( a \in A \) we assign a demand/capacity function \( d(a), c(a) \) and a weight function \( w(a) \), where \( d(a), c(a) \in \mathbb{N} \) and \( w(a) \in \mathbb{Q}_+ \). Let \( s,t \in V \). A function \( f: A \to \mathbb{N} \) is called an \( s-t \) flow (or just a flow) if for all \( v \in V \setminus \{s,t\} \),

\[
\sum_{a=(u,v) \in A} f(a) = \sum_{a=(v,u) \in A} f(a) \quad \text{(flow conservation)}.
\]
We say that a flow is feasible if \( d(a) \leq f(a) \leq c(a) \) for each \( a \in A \). We define the value of flow \( f \) as \( \text{val}(f) = \sum_{a \in (s,v) \in A} f(a) - \sum_{a \in (v,s) \in A} f(a) \). We define the cost of flow \( f \) as \( \sum_{a \in A} w(a) f(a) \). A minimum-cost flow is a feasible flow with minimum cost.

Algorithms for finding the minimum-cost flow of a given value are well known [41, 40]. Given a network \( G \) with integer demand and capacity functions, the successive shortest path algorithm [41] can be used to find a feasible \( s-t \) flow with value \( \alpha \in \mathbb{N} \) (if such a flow exists) and minimum cost in time \( O(\alpha \cdot SP(n,m)) \), where \( SP(n,m) \) is the time to compute a shortest directed path in \( G \) with \( n \) vertices and \( m \) edges. If there is no feasible flow with value \( \alpha \), then the successive shortest path algorithm outputs “no flow” in time \( O(\alpha \cdot SP(n,m)) \).

Given a Z-free binary VCSP instance \( P \) satisfying the JWP, we construct a directed graph \( G_P \) whose minimum-cost flows of value \( n \) are in one-to-one correspondence with the solutions to \( P \). Apart from the source node \( s \), \( G_P \) has three types of node:

1. A variable node \( v_i \) (\( i = 1, \ldots, n \)) for each variable of \( P \);
2. An assignment node \( \langle v_i, a \rangle \) (\( a \in D_i, i = 1, \ldots, n \)) for each possible variable-value assignment in \( P \);
3. An assignment-clique node \( C_\alpha \) for each maximal assignment-clique of edges in the assignment graph of \( P \) corresponding to binary costs of at least \( \alpha \). (The subscript \( \alpha \) is equal to the minimum cost of edges in the assignment-clique and, where necessary, we use \( C_\alpha, C_\beta, \ldots \) to denote the distinct non-intersecting assignment-cliques corresponding to the same value of \( \alpha \).)

In \( G_P \) there is an arc \((s, v_i)\) for each variable \( v_i \) of \( P \). For all variables \( v_i \) and for each \( a \in D_i \), there is an arc \((v_i, \langle v_i, a \rangle)\) and also an arc from \( \langle v_i, a \rangle \) to the assignment-clique \( C_\alpha \) containing \( \langle v_i, a \rangle \) such that \( \alpha \) is maximal (\( C_\alpha \) is unique by Lemma 14).

We say that assignment-clique \( C_\beta \) is the father of assignment-clique \( C_\alpha \) if it is the minimal assignment-clique which properly contains \( C_\alpha \), i.e. \( C_\alpha \subset C_\beta \) (and hence \( \alpha > \beta \)) and \( b \notin C_\gamma \) such that \( C_\alpha \subset C_\gamma \subset C_\beta \) (\( C_\beta \) is unique by Lemma 14). In \( G_P \), for each assignment-clique \( C_\alpha \) with father \( C_\beta \), there is a bundle of arcs from \( C_\alpha \) to \( C_\beta \) consisting of \( r \) arcs \( e_i \) (\( i = 1, \ldots, r \)), where \( r \) is the number of vertices in the assignment-clique \( C_\alpha \). The weight of arc \( e_i \) from \( C_\alpha \) to \( C_\beta \) is \( w(e_i) = (i-1)(\alpha-b) \) (We use the convention 0\( \infty = 0 \), so that if \( \alpha = \infty \) then there is a single arc of weight 0; the arcs of weight \( \infty \) can simply be omitted.) We identify the sink node \( t \) with the assignment-clique \( C_0 \) consisting of all edges in the assignment graph (since all binary costs are at least 0). Each arc has demand 0 and capacity 1 except for arcs \((s, v_i)\) which have both demand 1 and capacity 1 (this forces a flow of exactly 1 through each variable node \( v_i \)). Weights of all arcs are zero except for arcs going from an assignment-clique node to its father assignment-clique node, as described above, and arcs from a variable node \( v_i \) to an assignment node \( \langle v_i, a \rangle \) which have a weight of \( c_i(a) \).

We show below that flows in the constructed network are in one-to-one correspondence with assignments in \( P \), but first we give an example.

**Example 15.** We show the general construction on a simple example. Let \( P \) be a VCSP instance with variables \( v_1, v_2, v_3 \) and \( D_1 = D_2 = \{a, b\}, D_3 = \{a\} \). The weighted coloured micro-structure of \( P \) is illustrated in Fig. 5. Missing edges have cost 0. There are two assignment-cliques corresponding to cost at least 1: \( C_1 = \{(v_1, a), (v_2, a), (v_3, a)\} \) (in solid red in Fig. 5) and \( C_1' = \{(v_1, b), (v_2, b)\} \) (in dotted blue in Fig. 5); and one assignment-clique corresponding to cost at least 2: \( C_2 = \{(v_1, a), (v_2, a)\} \) (in dashed green in Fig. 5). The network corresponding to instance \( P \) is shown in Fig. 6: demands and capacities are in square brackets, and weights of arcs without numbers are 0. The bold red edges represent flow \( f \) corresponding to the assignment \( v_1 = v_2 = v_3 = a \) with total cost 4, which is the same as the cost of \( f \).

We now prove that flows \( f \) in \( G_P \) are in one-to-one correspondence with assignments in the VCSP \( P \) and, furthermore, that the cost of \( f \) is equal to the cost in \( P \) of the corresponding assignment.

All feasible flows have value \( n \) since all \( n \) arcs \((s, v_i)\) leaving the source have both demand and capacity equal to 1. Flows in \( G_P \) necessarily correspond to the assignment of a unique value \( a_i \) to each variable \( v_i \) since the flow of 1 through node \( v_i \) must traverse a node \( \langle v_i, a_i \rangle \) for some unique \( a_i \in D_i \). It remains to show that for every assignment \( \langle a_1, \ldots, a_n \rangle \) to
From Definition 4, recognition can be achieved in Proof.

Lemma 16. A Z-free binary VCSP instance \( \mathcal{P} \) satisfying the joint-winner property has at most 2nd \(-\) 1 different maximal assignment-cliques \( C_\alpha \).

Proof. Consider Z-free binary VCSP instances \( \mathcal{P} \) satisfying the joint-winner property and whose assignment graph has \( N = nd \) vertices. Let \( A_N \) be the maximum number of assignment-cliques \( C_\alpha \) in such instances \( \mathcal{P} \). We prove by induction that \( A_N \leq 2N - 1 \). Clearly, \( A_1 = 1 \). Consider an assignment graph of with \( N \) vertices. Consider any assignment-clique \( C_\alpha \) of \( \mathcal{P} \) which does not include all (variable,value) assignments of \( \mathcal{P} \) but which is maximal in the sense that there is no other such assignment-clique \( C_\beta \supset C_\alpha \). Let \( r = |C_\alpha| \). By Lemma 14 and by maximality of \( C_\alpha \), no other assignment-clique \( C_\beta \) can intersect both \( C_\alpha \) and its complement except an assignment-clique containing all the vertices of the assignment graph of \( \mathcal{P} \). Hence we can partition the set of variable-value assignments of \( \mathcal{P} \) (creating two VCSP instances whose assignment graphs have \( r \) and \( N - r \) vertices) and obtain the following recurrence: \( A_N \leq 1 + A_r + A_{N-r} \). From the inductive hypothesis, \( A_r \leq 2r - 1 \) and \( A_{N-r} \leq 2(N - r) - 1 \). Since \( 1 + 2r - 1 + 2(N - r) - 1 = 2N - 1 \), it follows that \( A_N \leq 2N - 1 \).

We show that \( n \)-variable instances satisfying the joint-winner property can be recognised and solved in time which is cubic in \( n \).

Theorem 17. A binary VCSP satisfying the joint-winner property is solvable in \( O(n^2d^3(n + d)) \) time.

Proof. From Definition 4, recognition can be achieved in \( O(n^3d^3) \) time. We have seen that a VCSP satisfying the JWP can be made Z-free in \( O(n^2d^3) \) time.
To solve a Z-free VCSP instance satisfying the JWP, we create a vertex for each of the assignment-cliques corresponding to binary costs of at least $\alpha$. By Lemma 16, there are at most $O(nd)$ different assignment-cliques. So our network has $n' = O(nd + nd + n + 2) = O(nd)$ vertices and $m' = O(n^2d^2)$ edges. The network can be built in $O(n^2d^3)$ time by $O(nd)$-times invoking depth-first search on the assignment graph of the instance, which is of size $O(n^2d^2)$.

The successive shortest path algorithm can be used to find a feasible flow with value $n$ and minimum cost in a network with $n'$ vertices and $m'$ edges in time $O(n \cdot SP(n', m'))$, where $SP(n', m')$ is the time to compute a shortest directed path in the network with $n'$ vertices and $m'$ edges [41,40]. Using Fibonacci heaps, this is $O(n(m' + n \log n')) = O(n(n^2d^2 + nd \log(ndef))) = O(n^3d^2)$. Hence, the complexity to solve a Z-free JWP instance is $O(n^3d^2)$. The total complexity, including pre-processing is thus $O(n^2d^2(n + d))$. \(\square\)

7. Maximality of JWP

Tractable classes defined by structural or language restrictions are often shown to be maximal. That is, any extension of the class is NP-hard. We consider that a hybrid tractable class defined by a set $S$ of possible combinations of $k$ costs within a subproblem is $k$-maximal if extending $S$ to include any other single combination of costs renders the problem NP-hard. In particular, a tractable class such as JWP, defined by a rule on the three costs in a 3-variable subproblem, is 3-maximal if extending it to include any other single combination of costs on 3 variables renders the problem NP-hard. The main result of this section is that the joint-winner property defines a 3-maximal tractable class when there is no bound on the number of possible distinct cost values. This result holds for any domain size $d \geq 2$. But we also show that the JWP defines a 3-maximal tractable class in almost all cases when we impose a bound of two on the number of possible distinct cost values (which corresponds to interesting subproblems of the VCSP such as CSP or MAX-CSP).

We first consider the special case in which all costs belong to $\{\alpha, \beta\}$ (for some fixed distinct costs $\alpha < \beta$). If $\beta = \infty$ this corresponds to CSP, and if $\beta < \infty$ this corresponds to MAX-CSP. When all costs belong to $\{\alpha, \beta\}$, the joint-winner property defines a maximal tractable class, except for the case when the domain size is 2 and all costs belong to $\{\alpha, \infty\}$, where $\alpha < \infty$. This latter case is just the well-known tractable class of Boolean binary CSPs [42].

**Theorem 18.** If all costs belong to $\{\alpha, \beta\}$ (for some fixed distinct costs $\alpha < \beta$), then the joint-winner property defines a maximal tractable class provided $d \geq 2$ or $(d \geq 2) \land (\beta < \infty)$, where $d$ is the maximum domain size.

**Proof.** To prove maximality we have to show the NP-hardness of the set of instances defined by the fact that in each triangle the triple of costs either satisfies the joint-winner property or is just one other fixed combination. Since all costs belong to $\{\alpha, \beta\}$ where $\alpha < \beta$, from Definition 4, the only situation forbidden by the JWP is that there are 3 variables $v_1, v_j, v_k$ and domain values $a \in D_i, b \in D_j, c \in D_k$ such that $c_{ij}(a, b) = \alpha$ and $c_{jk}(a, c) = c_{jk}(b, c) = \beta$. Hence extending the JWP means allowing all combinations of costs from $\{\alpha, \beta\}$ in all triangles.

If $\beta = \infty$, allowing all combinations of costs means that our instance allows all binary relations (corresponding to the set of finite-cost tuples) and hence we can encode any binary CSP. This is NP-complete if $d > 2$ [29].

If $\beta < \infty$, allowing all combinations of costs in $\{\alpha, \beta\}$ is equivalent to the set of instances of binary MAX-CSP in which no two constraints can have the same scope. The NP-hardness of this latter problem for $d > 2$ follows from the following reduction from binary MAX-CSP, which is a well-known NP-complete problem [28,29]. A polynomial-time reduction of an instance $I$ of binary MAX-CSP into an equivalent instance $I'$ in which no two constraints have the same scope can be achieved by replacing each variable $v_i$ in $I$ by $M$ variables $v_i^j$ ($j = 1, \ldots, M$) in $I'$ constrained by a clique of equality constraints, where $M$ is greater than the total number of constraints in $I$. If the $j$th constraint in $I$ has scope $\{v_i, v_j\}$, then it is replaced in $I'$ by the same constraint on scope $\{v_i^j, v_j^j\}$. In the optimal solution to $I'$, variables $v_i^j$ ($j = 1, \ldots, M$) are necessarily assigned the same value (otherwise a cost of at least $M$ would be incurred). \(\square\)

We now prove our main theorem concerning 3-maximality of the joint-winner property in the general case in which costs are not restricted to just two values $\alpha, \beta$. The essential differences compared to the proof of Theorem 18 are that (1) we now also have to prove maximality relative to the addition of any combination of costs involving three distinct values $\alpha, \beta, \gamma$, and (2) even when $|(\alpha, \beta, \gamma)| = 2$ the JWP allows triangles involving three distinct costs.

**Theorem 19.** The joint-winner property defines a 3-maximal tractable class for any domain size $d \geq 2$.

**Proof.** Let $\alpha < \beta \leq \gamma$ be any combination of costs which does not satisfy the JWP. To prove 3-maximality we have to show the NP-hardness of the set of instances defined by the fact that in each triangle the triple of costs either satisfies the joint-winner property or is $\{\alpha, \beta, \gamma\}$. We firstly consider the case $\gamma < \infty$.

It is well known that MAX-CSP is NP-hard [29]. Firstly, we show that binary MAX-CSP (coded as a binary VCSP with $\{0, 1\}$-valued cost functions) is NP-hard even on bipartite graphs. This follows from the fact that we can easily convert any instance of binary MAX-CSP into an instance of binary MAX-CSP on a bipartite graph, as follows. For each $c_{ij}(a, b) = 1$, introduce an extra Boolean variable $w_{ijab}$, and replace the cost of 1 for $(a, b)$ in $c_{ij}$ by costs of 1 for $(x_i, w_{ijab}) = (a, 0)$ and
Let notational convenience, in this section we interpret a solution of clauses in which variable variable variable variable triangle the triple of costs either satisfies the JWP or is the new \( \text{wijab} \) be written as the following operation to all cost functions: multiply by \( fi \) s, \( g i \) t, \( h i \) m. In the resulting VCSP, all triangles of costs are can add the singleton \( v \) h, \( v \) i, \( v \) j occurs. We construct a VCSP instance \( P \) containing \( m \) copies \( v \) i ( \( j = 0, \ldots, m - 1 \) ) of each variable \( v \) i. For each \( i = 1, \ldots, n \), the \( v \) i variables \( v \) i ( \( j = 0, \ldots, m - 1 \) ) are joined together by a cycle of \( \{ 0, \infty \} \)-valued crisp constraints coding the inequalities \( v \) i \( j \leq v \) i \( j+1 \) (where the addition in the superscript is modulo \( m \)). Since these crisp constraints form a cycle, they are equivalent to \( v \) i = \( \cdots = v \) i \( m-1 \). It is easy to verify that this cycle of crisp constraints satisfies the JWP since no triangle of binary costs contains two infinite costs.

The 2SAT clauses of \( I \) are replaced by corresponding \( \{ 0, 1 \} \)-valued constraints in \( P \) such that the \( j \)th occurrence of \( v \) i in \( I \) is replaced by \( v \) i \( j \) in \( P \). By this construction, every triangle of variables in \( P \) involves at most one \( \{ 0, 1 \} \)-valued constraint corresponding to a clause of \( I \). It follows that the triple of costs \( \{ 0, 1, 1 \} \) does not occur in any triangle in \( P \). Since infinite cost can only occur in the non-intersecting cycles of crisp constraints, the triples of costs \( \{ 0, \infty, \infty \} \) and \( \{ 1, \infty, \infty \} \) also do not occur in any triangle in \( P \). Thus, in each triangle in \( P \) the triple of costs either satisfies the JWP or is \( \{ 0, 1, 1 \} \). Since \( P \) is equivalent to \( I \), and the reduction is clearly polynomial, this completes the proof of NP-hardness.

It remains to consider the case \( \alpha < \beta = \gamma = \infty \). Analogously to the case \( \alpha < \beta < \gamma = \infty \), it suffices to give a polynomial-time reduction from MAX-2SAT to MAX-2SAT for each \( v \) i \( j \) \( k \) \( l \) \( m \) \( n \) \( i \) \( j \) \( k \) \( l \) \( m \) \( n \) such that the costs in each triangle either satisfy the JWP or are \( \{ 0, \infty, \infty \} \). Since \( \{ 0, \infty, \infty \} \) are sets of \( \{ 0, 1 \} \)-valued constraints, for all \( \alpha \in \{ 0, \infty, \infty \} \) do not occur in any triangle in \( P \). Thus, in each triangle in \( P \) the triple of costs either satisfies the JWP or is \( \{ 0, 1, 1 \} \). Since \( P \) is equivalent to \( I \), and the reduction is clearly polynomial, this completes the proof of NP-hardness.

For the following definition, we require the notion of a non-decreasing derivative of a discrete function. The derivative of the function \( f : [0, \ldots, s] \rightarrow \mathbb{Q}_+ \) is non-decreasing if \( f(m+2) - f(m+1) \geq f(m+1) - f(m) \) for all \( m \in [0, \ldots, s-2] \) (where subtraction is extended to \( \mathbb{Q}_+ \) by defining \( \infty - \infty = \infty \) for all \( \alpha \in \mathbb{Q}_+ \)). Two sets \( S, T \) are said to be non-overlapping if they are either disjoint or one is a subset of the other (i.e. \( S \cap T = \emptyset, S \subseteq T \text{ or } T \subseteq S \)).

**Definition 20.** Let \( P \) be a VCSP instance. Let \( C_1, \ldots, C_r \) be sets of \( \{ \text{variable, value} \} \) assignments of \( P \) such that for all \( i, j, C_i \) and \( C_j \) are non-overlapping. Instance \( P \) satisfies the non-overlapping convexity property if the objective function of \( P \) can be written as \( f_1(N(x, C_1)) \cdots + f_r(N(x, C_r)) \) such that each \( f_i : [0, \ldots, s_i] \rightarrow \mathbb{Q}_+ \) \( (i = 1, \ldots, r) \) is a non-decreasing function with non-decreasing derivative, where \( N(x, C_i) = |x \cap C_i| \) is the number of \( \{ \text{variable, value} \} \) assignments in the solution \( x \) which lie in \( C_i \) and \( s_i \) is the number of distinct variables occurring in the set of \( \{ \text{variable, value} \} \) assignments \( C_i \).

Observe that this definition allows any unary valued constraints, since for each \( \{ \text{variable, value} \} \) assignment \( (v, a) \) we can add the singleton \( C_i = \{ (v, a) \} \) which is necessarily either disjoint or a subset of any other set \( C_k \) (and furthermore the corresponding function \( f_i : [0, 1] \rightarrow \mathbb{Q}_+ \) is trivially non-decreasing with a non-decreasing derivative).
Remark 21. To express a binary Z-free VCSP satisfying the JWP in the more general form given in Definition 20, we add a set of (variable, value) assignments \( C_i \) corresponding to each assignment-clique \( C_a \) in the assignment graph, with \( \phi_i(m) = \binom{\alpha}{m}(\alpha - \beta) \) for \( m \geq 0 \), where \( C_\beta \) is the father assignment-clique of \( C_a \). It is easy to check that \( \phi_i \) is non-decreasing and has a non-decreasing derivative. To solve a VCSP instance satisfying the non-overlapping convexity property, we will build a network (described, below, in the proof of Theorem 22) in which the weight of the \( m \)th arc leaving the \( C_i \) node is \( f_i(m) - f_i(m-1) \). Since \( \binom{\alpha}{m}(\alpha - \beta) - \binom{\alpha-1}{m}(\alpha - \beta) = (m-1)(\alpha - \beta) \), this means that, in the case of a binary VCSP satisfying the JWP, we will build the same network as described in Section 6.

Note that in this case, to avoid counting binary costs more than once, the functions \( f_i \) are proportional to the marginal costs \( \alpha - \beta \). When solving a VCSP instance satisfying the non-overlapping convexity property in which the functions \( f_i \) are given as input, no management of marginal costs is required.

Theorem 22. Any VCSP instance \( \mathcal{P} \) satisfying the non-overlapping convexity property can be solved in polynomial time.

Proof. We can consider each set of (variable, value) assignments \( C_i \) as an assignment-clique: since \( C_i \) is a subset of the vertices of the assignment graph of \( \mathcal{P} \), we just need to add all edges between those vertices \( (v_j, a), (v_k, b) \) of \( C_i \) representing assignments to distinct variables \( (j \neq k) \).

We build the same network as described in Section 6, but have to adjust the weights on edges between the assignment-cliques \( C_i \). Recall that \( s_i \) represents the number of distinct variables in \( C_i \). Without loss of generality, we can assume that \( f_i(0) = 0 \). (If this is not the case, we can replace \( f_i \) by \( f'_i \) where \( f'_i(m) = f_i(m) - f_i(0) \) to produce an equivalent problem with \( f'_i(0) = 0 \) and in which \( f'_i \) is non-decreasing and with a non-decreasing derivative.) Let \( \Delta_i(m) = f_i(m) - f_i(m-1) \) for \( m = 1, \ldots, s_i \). Since the \( f_i(1 \leq i \leq r) \) are non-decreasing functions with a non-decreasing derivative, \( \Delta_i \) is non-negative and non-decreasing. In the network, there are \( s_i \) arcs \( e_i^k \) \((k = 1, \ldots, s_i)\) from \( C_i \) to its (unique) father assignment-clique. The weight (cost) of arc \( e_i^k \) is \( \Delta_i(k) \).

Similarly to the construction in Section 6, arcs of weight \( \infty \) can be omitted. The same argument and calculation as in Section 6 proves that the algorithm is correct. In particular, we need to show that given any feasible assignment \( x \) (that is, of finite cost) to a VCSP instance \( \mathcal{P} \) that satisfies the non-overlapping convexity property, there is a corresponding minimum-cost feasible flow \( f \) in \( G_P \) of cost \( f_1(N(x, C_1)) + \cdots + f_r(N(x, C_r)) \). The same construction of \( f \) as in Section 6 gives us a flow \( f \) of the following cost:

\[
\sum_i \sum_{m=1}^{N(C_i)} \Delta_i(m) = \sum_i \left[ f_i(1) + (f_i(2) - f_i(1)) + \cdots + (f_i(N(C_i)) - f_i(N(C_i) - 1)) \right]
\]

which corresponds precisely to the cost \( f_1(N(x, C_1)) + \cdots + f_r(N(x, C_r)) \) of the assignment \( x \). To see that this flow is a minimum cost, among flows corresponding to the assignment \( x \), observe that the only choices to be made concern the arcs \( e_i^k \). Since the weights \( \Delta_i(k) \) of these arcs are non-decreasing, a flow of value \( v \) which takes the first \( v \) arcs \( e_i^k \) \((k = 1, \ldots, v)\) is necessarily of minimum cost. \( \square \)

Example 23. An example of the non-overlapping convexity property is the set of non-binary (i.e. arbitrary but fixed maximum arity) MAX-CSP instances with no overlapping nogoods, which we will now describe. A CSP is traditionally defined by a set of constraints each of which is given in the form of a scope and a relation containing the set of possible assignments to the variables in the scope of the constraint. Another way of specifying a CSP instance is by listing its nogoods, where a nogood is a set of (variable, value) assignments which cannot simultaneously be made. The nogoods are easily obtained from the complement of each constraint relation. A MAX-CSP instance with no overlapping nogoods is such that for all nogoods \( N_i, N_j \) in the corresponding CSP instance, either \( N_i \cap N_j = \emptyset \), or \( N_i \subset N_j \) or \( N_j \subset N_i \).

To see that any MAX-CSP instance \( I \) with no overlapping nogoods \( N_1, \ldots, N_r \) satisfies the non-overlapping convexity property, for \( i = 1, \ldots, r \), define \( f_i \) by \( f_i([N_i]) = 1 \) and \( f_i(m) = 0 \) for \( m < |N_i| \). Clearly each \( f_i \) is a non-decreasing function with a non-decreasing derivative and the MAX-CSP instance \( I \) is equivalent to the VCSP instance with objective function \( f_i([x \cap N_i]) + \cdots + f_r([x \cap N_r]) \).

Example 24. By an entirely similar argument, it follows that another example of the non-overlapping convexity property is the set of non-binary CSP instances with non-overlapping nogoods \( N_1, \ldots, N_r \). It suffices to define \( f_i \) by \( f_i([N_i]) = \infty \) and \( f_i(m) = 0 \) for \( m < |N_i| \) (for each \( i = 1, \ldots, r \)).

This class can be extended to include all CSP instances with a set of nogoods \( S_k(N_1) \cup \cdots \cup S_k(N_r) \), with \( S_k(N_i) \) the set of all cardinality-\( k_i \) subsets of \( N_i \), where the sets of (variable, value) assignments \( N_i (i = 1, \ldots, r) \) are pairwise non-overlapping. Each \( k_i \) is an integer in \([1, \ldots, |N_i|]\). To express such a CSP instance as a VCSP instance satisfying the non-overlapping convexity property, it suffices to define \( f_i \) by \( f_i(m) = \infty \) if \( m \geq k_i \) and \( f_i(m) = 0 \) for \( m < k_i \) (for each \( i = 1, \ldots, r \).
Clearly each $f_j$ is a non-decreasing function with a non-decreasing derivative and the CSP instance $I$ is equivalent to the VCSP instance with objective function $f_I(x \cap N_1) + \cdots + f_I(x \cap N_t)$.

**Example 25.** An example of an optimisation problem satisfying the non-overlapping convexity property is the following office assignment problem. Each of $n$ staff members, represented by $n$ variables, must be assigned an office. There are $m$ offices. At most $u_j$ people can be assigned office $j$. Unary cost functions can be used to express personal preferences of each staff member for each office. There are also non-overlapping groups of people $G_1, \ldots, G_s$ who we would prefer to assign to different offices (such as married couples, for example).

For each office there is a set of (variable,value) assignments $C_j$ consisting of all possible assignments of value $j$ to any variable. For each group $G_i$ and for each office $j$, there is a set of assignments $C_{ij} \subseteq C_j$ of members of this group to office $j$. Clearly, the sets of assignments $C_j$ ($j \in \{1, \ldots, m\}$) and $C_{ij}$ ($i \in \{1, \ldots, s\}, j \in \{1, \ldots, m\}$) are all non-overlapping. The function $f_j$ which imposes the capacity constraint for office $j$, given by $f_j(k) = \infty$ if $k > u_j$ and $f_j(k) = 0$ otherwise, is non-decreasing and has a non-decreasing derivative. A cost function $f_{ij}$ such as $f_{ij}(t) = t - 1$ (which is non-decreasing and has a non-decreasing derivative) can be used to code the fact that we prefer that staff members from group $G_i$ do not share the same office.

**Example 26.** A commonly occurring problem in academia is the allocation of courses to teachers subject to timetabling constraints and the personal preferences of the teachers along with a criterion to avoid allocating too many hours to any of the teachers. In a simple version of this problem, the weekly timetable has been divided up into non-intersecting time-slots, each course involves giving only one lecture a week and one-hour time-slots have already been allocated to each course.

The courses, numbered from 1 to $n$, correspond to the variables of the problem and must be assigned one of the $m$ teachers.

For each teacher $j \in \{1, \ldots, m\}$, let $C_j$ represent all assignments of teacher $j$ to any course. For each time-slot $s \in \{1, \ldots, t\}$ and each teacher $j \in \{1, \ldots, m\}$, let $C_{js}$ represent the set of assignments of teacher $j$ to courses allocated the time-slot $s$. Clearly, the sets of assignments $C_j$ ($j \in \{1, \ldots, m\}$) and $C_{js}$ ($j \in \{1, \ldots, m\}$ and $s \in \{1, \ldots, t\}$) are all non-overlapping. For each teacher $j$, the function $f_j$ corresponding to the set of assignments $C_j$ and given by $f_j(k) = \max(0, p_j (k - u_j))$ (where $u_j$ is the maximum number of courses teacher $j$ can give before they have to be paid overtime and $p_j$ is the amount teacher $j$ earns per hour of overtime) represents the total cost in overtime payments of teacher $j$. The function $f_{js}$, which codes the incompatibility of allocating teacher $j$ to two different courses during the same time-slot $s$, is given by $f_{js}(0) = f_{js}(1) = 0$ and $f_{js}(k) = \infty$ if $k > 1$. The functions $f_j$ and $f_{js}$ are non-decreasing and have a non-decreasing derivative. Thus this simple course-allocation problem can be expressed as a VCSP with the non-overlapping convexity property.

### 9. Conclusions

We have studied hybrid tractability of valued CSPs. In particular, we have studied the tractability of sets of instances defined by properties of subproblems of size $k$. For $k = 2$, such properties can only define language classes. We have shown in this paper that it is possible to define a non-trivial tractable hybrid class by a property on subproblems of size $k = 3$. We have studied the tractable class of VCSPs defined by this property, known as the joint-winner property (JWP), as a necessary first step towards a general theory of tractability of optimisation problems which will eventually cover structural, language and hybrid reasons for tractability. Moreover, we have also presented other novel hybrid tractable classes of VCSPs.

The JWP is interesting in its own right since it is a proper extension to known tractable classes (such as VCSPs consisting of arbitrary unary constraints and non-intersecting SoftAllDiff constraints of arbitrary arity, as well as a machine scheduling problem).

We have demonstrated 3-maximality of the tractable class defined by the JWP. However, the existence of a larger tractable class subsuming JWP and defined by a rule on $k$-variable subproblems (for $k > 3$) is an interesting open question. Indeed, we have introduced a generalisation of the JWP which is solved by a similar algorithm but which allows soft constraints of arbitrary arity. The most interesting open question brought out by this work is whether other tractable hybrid classes can be defined by properties of $k$-variable subproblems for $k > 3$.

### References


