A new hybrid tractable class of soft constraint problems*

Martin C. Cooper[1] and Stanislav Živný[2]

[1] IRIT, University of Toulouse III, 31062 Toulouse, France
cooper@irit.fr
[2] Computing Laboratory, University of Oxford, OX1 3QD Oxford, UK
standa.zivny@comlab.ox.ac.uk

Abstract. The constraint satisfaction problem (CSP) is a central generic problem in artificial intelligence. Considerable effort has been made in identifying properties which ensure tractability in such problems. In this paper we study hybrid tractability of soft constraint problems; that is, properties which guarantee tractability of the given soft constraint problem, but properties which do not depend only on the underlying structure of the instance (such as being tree-structured) or only on the types of soft constraints in the instance (such as submodularity).

We firstly present two hybrid classes of soft constraint problems defined by forbidden subgraphs in the structure of the instance. These classes allow certain combinations of binary crisp constraints together with arbitrary unary soft constraints.

We then introduce the joint-winner property, which allows us to define a novel hybrid tractable class of soft constraint problems with soft binary and unary constraints. This class generalises the SoftAllDiff constraint with arbitrary unary soft constraints. We show that the joint-winner property is easily recognisable in polynomial time and present a polynomial-time algorithm based on maximum-flows for the class of soft constraint problems satisfying the joint-winner property. Moreover, we show that if cost functions can only take on two distinct values then this class is maximal.

1 Introduction

Background 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 undirected graph, known as its *constraint network*, whose vertices are the variables of the instance, and two vertices are adjacent if corresponding variables are related by some constraint. Such a graph is also known as the *structure* of the instance.

* Stanislav Živný gratefully acknowledges the support of EPSRC grant EP/F01161X/1, EPSRC PhD+ Award, and Junior Research Fellowship at University College, Oxford.

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 [4,15], or else identifying structural properties of constraint networks which ensure tractability no matter what forms of constraint are imposed [13].

The first approach has led to identifying certain algebraic properties known as polymorphisms [20] which are necessary for a set of constraint types to ensure tractability. A set of constraint types with this property 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 structures of *bounded tree-width* [17].

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 [12,10,9,22,11].

Since in practice many constraint satisfaction problems are over-constrained, and hence have no solution, *soft* constraint satisfaction problems have been studied [12]. In an instance of the soft CSP, every constraint is associated with a function (rather than a relation as in the CSP) which represents preferences among different partial assignments, and the goal is to find the best assignment. Several very general soft CSP frameworks have been proposed in the literature [30,2]. In this paper we focus on one of the very general frameworks, the *valued* constraint satisfaction problem (VCSP) [30].

Similarly to the CSP, an important line of research on the VCSP is to identify tractable cases which are recognisable in polynomial time. Is is well known that structural reasons for tractability generalise to the VCSP [12]. In the case of language restrictions, only a few conditions are known to guarantee tractability of a given set of valued constraints [8,7].

Up until now there have been very few results on hybrid tractability for the VCSP. For instance, Kumar defines an interesting framework for hybrid tractability for the Boolean weighted CSP [22]. 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 [22] are not hybrid and are already known.

Contributions In this paper we study hybrid tractability of the VCSP. As a first step, we start with binary VCSPs. We will demonstrate two hybrid classes defined by forbidding certain graphs as induced subgraphs in the structure of the VCSP instance.[3] However, these tractable classes are not entirely satisfactory as a first step towards a general theory of hybrid tractable classes of VCSP instances since the only soft constraints they allow are unary.

[3] More precisely, in the *micro-structure complement* of the instance.

As the main contribution of the paper, we introduce the *joint-winner property* (JWP), which allows us to define a *novel* hybrid tractable class of VCSPs.

We now describe the joint-winner property. Let v_i and v_j be two variables, and let $c_{ij}(a,b)$ denote the cost of the assignment $v_i = a$ and $v_j = b$ given by the valued constraint c_{ij} between the variables v_i and v_j. We denote by D_i the domain of variable v_i. A VCSP instance satisfies the joint-winner property if for every triple of distinct variables v_i, v_j, v_k and all domain values $a \in D_i, b \in D_j, c \in D_k$: $c_{ij}(a,b) \geq \min(c_{ik}(a,c), c_{jk}(b,c))$ (see Figure 1).

The joint-winner property is preserved by all unary constraints and hence conservative, and also easily recognisable in polynomial time. The polynomial-time algorithm for solving instances satisfying JWP is based on maximum flows.

As the next example shows, the well-known hybrid tractable class SoftAllDiff satisfies the JWP property, thus showing that JWP defines a hybrid tractable class.

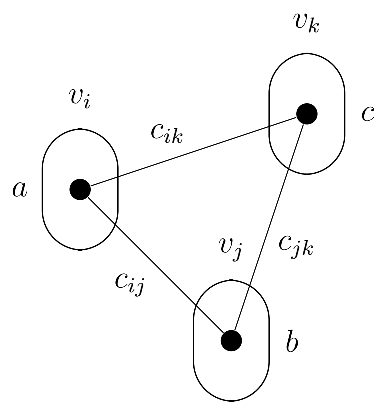

Fig. 1. The joint-winner property: $c_{ij}(a,b) \geq \min(c_{ik}(a,c), c_{jk}(b,c))$.

Example 1. One of the most commonly used global constraints is the AllDifferent constraint [28]. Petit et al. introduced a soft version of the AllDifferent constraint, SoftAllDiff [27]. 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 number of variables violating an AllDifferent constraint. The algorithms for filtering these constraints, introduced in the same paper, were then improved by van Hoeve et al. [33].

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 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 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 costs is 0.

In order to code the variable-based SoftAllDiff constraint as a binary VCSP $\mathcal{P}$, we can create n variables v_i' with domains $D_i \times \{1,2\}$. The assignment $v_i' = (a,1)$ means that v_i is the first variable of the original CSP to be assigned the value a, whereas $v_i' = (a,2)$ means that v_i is assigned a but it is not the first such variable. In $\mathcal{P}$ there is a crisp constraint which disallows $v_i' = v_j' = (a,1)$ (for any value $a \in D_i \cap D_j$) for each pair of variables $i < j$ together with a soft unary constraint $c_i(a,k) = k-1$ (for $k = 1,2$) for each $i \in \{1,\ldots,n\}$. Hence at most one variable can be the first to be assigned a, and each assignment of the value a to a variable (apart from the first) incurs a cost of 1. Again due to the transitivity of equality, it cannot happen that only one of the binary costs shown in the triangle of Figure 1 is zero, from which it follows immediately that the joint-winner property is satisfied in $\mathcal{P}$. We remark that the class defined by JWP is strictly bigger than SoftAllDiff, and hence our generic algorithm is not as efficient as tailor-made algorithms for SoftAllDiff.

When restricted to the standard CSP, the JWP property gives a set of disjoint AllDifferent constraints on subdomains, which is a proper (although not very surprising) generalisation of disjoint AllDifferent constraints.

Example 2. 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

$$T \;=\; \sum_{m=1}^{d} \Big(\sum_{i \in S_m} l_i(m) \;+\; \sum_{\substack{i,j \,\in\, S_m \\ i<j}} \min(l_i(m), l_j(m)) \Big)$$

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 $\min(l_i(m), l_j(m)) \geq \min(\min(l_i(m), l_k(m)), \min(l_j(m), l_k(m)))$.

This problem has been shown solvable in polynomial time in [19,3].

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. Using a connection between these VCSPs and the maximum weighted independent set problem in certain graph classes, we identify hybrid tractable classes of VCSPs. In Section 4, we define the joint-winner property. In Section 5, we present a polynomial-time algorithm for solving binary VCSPs satisfying the joint-winner property. In Section 6, we present an important case in which this new tractable class is maximal.

2 Preliminaries

In this paper we focus on binary valued constraint satisfaction problems. We denote by $\mathbb{Q}_+$ the set of all non-negative rational numbers. We denote $\overline{\mathbb{Q}}_+ = \mathbb{Q}_+ \cup \{\infty\}$ with the standard addition operation extended so that for all $a \in \overline{\mathbb{Q}}_+$, $a + \infty = \infty$. Members of $\overline{\mathbb{Q}}_+$ are called *costs*.

A unary cost function over domain D_i is a mapping $c_i : D_i \to \overline{\mathbb{Q}}_+$. A binary cost function over domains D_i and D_j is a mapping $c_{ij} : D_i \times D_j \to \overline{\mathbb{Q}}_+$. For notational convenience, throughout this paper we assume that there is a unique valued constraint on any given scope. In particular, $c_{ij}(a,b) = c_{ji}(b,a)$, since they are simply two different ways of representing the unique cost of simultaneously assigning $\langle a, b\rangle$ to variables $\langle i, j\rangle$. 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 [30] 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 $\langle v_i, c_i\rangle$, 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 $\langle\langle v_i, v_j\rangle, c_{ij}\rangle$, where v_i and v_j are variables, 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.

A *solution* to a VCSP 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 solution amounts to deciding whether there is a solution with finite cost (all constraints are satisfied).

The *microstructure* 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 $\langle v_i, a\rangle$ represents the assignment of value a to variable v_i [21]. The edges of the microstructure connect all pairs of variable-value assignments that are allowed by the constraints. (Note that if there is no explicit constraint between two variables, then it is considered to be the complete constraint.) The microstructure of a binary VCSP instance is defined similarly, but both vertices and edges of the graph are assigned costs. For CSPs, the *microstructure complement* is the complement of the microstructure: its edges represent pairs of variable-value assignments that are disallowed by the constraints. Hence for every variable v_i, the microstructure complement contains all edges of the form $\{\langle v_i, a\rangle, \langle v_i, b\rangle\}$ 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 a CSP instance $\mathcal{P}$ is equivalent to finding a clique of size n in the microstructure of $\mathcal{P}$, and to finding an independent set of size n in the microstructure complement of $\mathcal{P}$ [21]. Therefore, tractability results on the maximum independent set problem for various classes of graphs can be easily used to obtain tractable CSP classes [10].

3 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. First we show how tractability results on the maximum weighted independent set in graphs can be used to obtain tractable classes of VCSPs. Next we show that the recently discovered hybrid CSP class defined by the broken-triangle property [11] is not extendible to soft unary constraints.

Let G be a graph $G = \langle V, E\rangle$ with weights $w : V \to \mathbb{Q}_+$ 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 see that given a binary VCSP instance $\mathcal{P}$ where only unary constraints can be soft, $\mathcal{P}$ can be solved by finding a maximum weighted independent set in the microstructure complement of $\mathcal{P}$ with weights given by $w(\langle v_i, a\rangle) = Mn - c_i(a)$, 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 assignments to n variables in $\mathcal{P}$.

Two well-studied classes of graphs for which the maximum weighted independent set problem (WIS) is known to be solvable in polynomial time are perfect graphs and claw-free graphs.

A graph G is *perfect* if for every induced subgraph H of G, the chromatic number (the smallest number of colours needed to colour vertices of H such that adjacent vertices are coloured differently) of H is equal to the size of the largest clique in H. The Strong Perfect Graph Theorem states that a graph G is perfect if, and only if, G does not contain any cycle of odd length ≥ 5 (known as a hole) nor any complement of a cycle of odd length ≥ 5 (known as an antihole) as an induced subgraph [6]. It is known that WIS in perfect graphs is solvable in polynomial time [18]. Moreover, perfect graphs can be recognised in polynomial time [5].

A graph G is *claw-free* if it does not contain a claw as an induced subgraph, where a claw is a complete bipartite graph $K_{1,3}$ with 1 vertex in one group and 3 vertices in the other group. It is obvious that claw-free graphs are recognisable in polynomial time. Extending Edmond's algorithm for maximum matchings in graphs [14], Minty designed an algorithm for the independent set problem in claw-free graphs [24]. Minty's algorithm was later corrected and extended to the maximum weighted independent set problem in claw-free graphs [25]. The combination of these results gives:

Theorem 1. *The class of VCSP instances (with crisp binary and arbitrary unary constraints) whose microstructure complement is either perfect or claw-free is tractable.*

The tractability of VCSPs with perfect microstructure (complement) and soft unary constraints was also pointed out by Takhanov [32].

Next we show that the recently discovered hybrid class of tractable CSPs defined by the broken-triangle property [11] is *not* extendible to VCSPs with soft unary constraints. A binary CSP instance $\mathcal{P}$ satisfies the broken-triangle property with respect to the variable ordering $<$ if, 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.)

Let $\langle G, k\rangle$ 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 [16]. 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 $\langle\langle v_i, v_j\rangle, \{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle\}\rangle$ 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 $\langle v_i, c_i\rangle$, where $c_i(x) = 1 - x$. The unary constraints ensure that the goal is to minimise 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. 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 Figure 2. It can be checked easily that the broken-triangle property is indeed satisfied whether the constraint on $\langle v_i, v_j\rangle$ is $\{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle\}$ (as shown in Figure 2) or the complete constraint. This gives us the following result:

Theorem 2. *Assuming P≠NP, the broken-triangle property cannot be extended to a tractable class including soft unary constraints.*

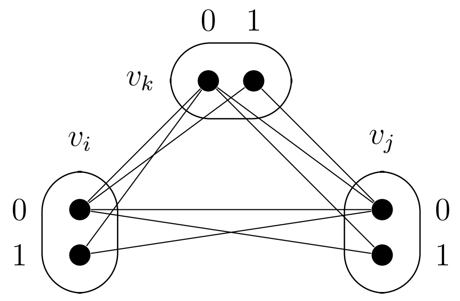

Fig. 2. VCSP encoding maximum independent set.

4 Joint-winner property

In this section we define the joint-winner property (see Figure 1), which is the key concept in this paper. We also study basic properties of VCSPs satisfying the joint-winner property as these will be important in designing a polynomial-time algorithm in Section 5.

Definition 1 (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.*

The joint-winner property places no restrictions on the unary soft constraints c_i. Note that the JWP on a CSP instance amounts to forbidding the multiset of binary costs $\{\alpha, \infty, \infty\}$ (for $\alpha < \infty$) in any triangle formed by three assignments to distinct variables. 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 soft unary constraints.

The next lemma explains the reason for the name of the joint-winner property: in every triangle there is no unique smallest cost.

Lemma 1. *A VCSP instance satisfies the joint-winner property if and only if, for all triples of distinct variables $\langle v_i, v_j, v_k\rangle$ 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 $\langle v_i, v_j, v_k\rangle$, $\langle v_j, v_k, v_i\rangle$ and $\langle v_k, v_i, v_j\rangle$, and write $\alpha = c_{ij}(a, b)$, $\beta = c_{ik}(a, c)$ and $\gamma = c_{jk}(b, c)$. Without loss of generality, let $\alpha = \min(\alpha, \beta, \gamma)$. From $\alpha \geq \min(\beta, \gamma)$, we can deduce that $\alpha = \min(\beta, \gamma)$ and hence that two of α, β, γ are equal and less than or equal to the third.

On the other hand, if two of α, β, γ 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. $\square$

Lemma 2. *Let $\mathcal{P}$ be a binary VCSP instance. Then, for a fixed α, the edges of the microstructure of $\mathcal{P}$ corresponding to binary costs of at least α, together with the corresponding vertices, form non-intersecting cliques.*

Proof. For a contradiction let us assume that the edges of the microstructure of $\mathcal{P}$ corresponding to binary costs of at least α do not form non-intersecting cliques. This means that there are three vertices $\langle v_i, a\rangle, \langle v_j, b\rangle, \langle v_k, c\rangle$ of the microstructure 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 1. $\square$

Lemma 3. *Let $\mathcal{P}$ be a binary VCSP instance. Let C_α be a clique in the microstructure of $\mathcal{P}$ corresponding to binary costs of at least α, and C_β a clique in the microstructure of $\mathcal{P}$ corresponding to binary costs of at least β. If C_α intersects C_β and $\alpha \leq \beta$, then $C_\alpha \supseteq C_\beta$.*

Proof. Suppose that C_α and C_β intersect and $\alpha \leq \beta$. If $\alpha = \beta$, the claim is satisfied trivially by Lemma 2, so we can suppose that $\alpha < \beta$. For a contradiction, assume that $C_\alpha \not\supseteq C_\beta$. 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_β is a clique, we must have $c_{ij}(a, b) \geq \beta > \alpha$. Thus, by Lemma 2, the edge $\{\langle v_j, b\rangle, \langle v_i, a\rangle\}$ is part of a clique C'_α of edges of cost at least α (but not C_α since $\langle v_j, b\rangle \notin C_\alpha$). But then C_α and C'_α intersect at $\langle v_i, a\rangle$ which contradicts Lemma 2. $\square$

5 Algorithm

In this section we present a polynomial-time algorithm for solving 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 [29,33,23].

First we review some basics on flows in graphs (see [1] for more details). 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 $w(a)$. Let $s, t \in V$. A function $f : A \to \mathbb{Q}$ is called an $s - t$ *flow* (or a flow) if

- $f(a) \geq 0$ for all $a \in A$;
- for all $v \in V \backslash \{s, t\}$, $\sum_{a=(u,v)\in A} f(a) = \sum_{a=(v,u)\in A} f(a)$ (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 $val(f) = \sum_{a=(s,v)\in A} f(a) - \sum_{a=(v,s)\in A} f(a)$. 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 described in [31, Chapter 12] and [1]. Given a network G with integer demand and capacity functions, the *successive shortest path algorithm* [31], can be used to find a feasible $s - t$ flow with value α and minimum cost in time $O(\alpha \cdot SP)$, where SP is the time to compute a shortest directed path in G.

Given a VCSP $\mathcal{P}$ satisfying the JWP, we construct a directed graph $G_{\mathcal{P}}$ whose minimum-cost integral flows of value n are in one-to-one correspondence with the solutions to $\mathcal{P}$. Apart from the source node s, $G_{\mathcal{P}}$ has three types of node:

1. a variable node v_i $(i = 1, \ldots, n)$ for each variable of $\mathcal{P}$;
2. an assignment node $\langle v_i, a\rangle$ $(a \in D_i,\ i = 1, \ldots, n)$ for each possible variable-value assignment in $\mathcal{P}$;
3. a clique node C_α for each clique of edges in the microstructure of $\mathcal{P}$ corresponding to binary costs of at least α. (The subscript α is equal to the minimum cost of edges in the clique and, where necessary, we use C_α, $C'_\alpha, \ldots$ to denote the distinct non-intersecting cliques corresponding to the same value of α.)

In $G_{\mathcal{P}}$ there is an arc (s, v_i) for each variable v_i of $\mathcal{P}$. 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 clique C_α containing $\langle v_i, a\rangle$ such that α is maximal (C_α is unique by Lemma 3).

We say that clique C_β is the *father* of clique C_α if it is the minimal clique which properly contains C_α, i.e. $C_\alpha \subset C_\beta$ (and hence $\alpha > \beta$) and $\nexists C_\gamma$ such that $C_\alpha \subset C_\gamma \subset C_\beta$ (C_β is unique by Lemma 3). In $G_{\mathcal{P}}$, for each clique C_α with father C_β, there is a bundle of arcs from C_α to C_β consisting of r arcs e_i $(i = 1, \ldots, r)$, where r is the number of vertices in the clique C_α. The weight of arc e_i from C_α to C_β is $w(e_i) = i(\alpha - \beta)$. (If $\alpha = \infty$ then there is a single arc of weight 0; the arcs of weight ∞ can simply be omitted.) We identify the sink node t with the clique C_0 consisting of all edges in the microstructure (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 0 except for arcs going from a clique node to its father 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 integral flows in the constructed network are in one-to-one correspondence with assignments in $\mathcal{P}$, but first we give an example.

Example 3. We show the general construction on a simple example. Let $\mathcal{P}$ be a VCSP instance with variables v_1, v_2, v_3 and $D_1 = D_2 = \{a, b\}$, $D_3 = \{a\}$. The microstructure of $\mathcal{P}$ is shown in Figure 3. Missing edges have cost 0. There are two cliques corresponding to cost at least 1: $C_1 = \{\langle v_1, a\rangle, \langle v_2, a\rangle, \langle v_3, a\rangle\}$ and $C'_1 = \{\langle v_1, b\rangle, \langle v_2, b\rangle\}$; and one clique corresponding to cost at least 2: $C_2 = \{\langle v_1, a\rangle, \langle v_2, a\rangle\}$ (see Figure 3). The network corresponding to instance $\mathcal{P}$ is shown in Figure 4: demands and capacities are in square brackets, and weights of arcs without numbers are 0. The bold 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 integral flows f in $G_{\mathcal{P}}$ are in one-to-one correspondence with assignments in the VCSP $\mathcal{P}$ and, furthermore, that the cost of f is equal to the cost in $\mathcal{P}$ 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. Integral flows in $G_{\mathcal{P}}$ 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 $\langle v_1, \ldots, v_n\rangle$ which is feasible (i.e. whose cost in $\mathcal{P}$ is finite), there is a corresponding minimum-cost integral feasible flow f in $G_{\mathcal{P}}$ of cost $\sum_i c_i(a_i) + \sum_{i<j} c_{ij}(a_i, a_j)$.

For each arc a which is incoming to or outgoing from $\langle v_i, u\rangle$ in $G_{\mathcal{P}}$, let $f(a) = 1$ if $u = a_i$ and 0 otherwise. We denote the number of assignments $\langle v_i, a_i\rangle$ in clique C_α by $N(C_\alpha) = |\{\langle v_i, a_i\rangle \in C_\alpha : 1 \le i \le n\}|$. By construction, each clique node C_α in $G_{\mathcal{P}}$ only has outgoing arcs to its father clique. For the outgoing arc a of weight i from C_α to its father clique, let $f(a) = 1$ if $N(C_\alpha) > i$ and 0 otherwise. This simply means that the outgoing arcs (each of capacity 1) from C_α are used in increasing order of their weight, one per assignment $\langle v_i, a_i\rangle \in C_\alpha$. This is clearly a minimum-cost flow corresponding to the assignment $\langle a_1, \ldots, a_n\rangle$.

Let $cf(C_\alpha)$ denote the cost β of the father clique C_β of C_α. The cost of flow f is given by

$$\begin{aligned}
&\sum_{i=1}^{n} c_i(a_i) \;+\; \sum_{C_\alpha} \sum_{i=1}^{N(C_\alpha)-1} i(\alpha - cf(C_\alpha)) \\
=&\sum_{i=1}^{n} c_i(a_i) \;+\; \sum_{C_\alpha} \frac{(N(C_\alpha)-1)N(C_\alpha)}{2}(\alpha - cf(C_\alpha))
\end{aligned}$$

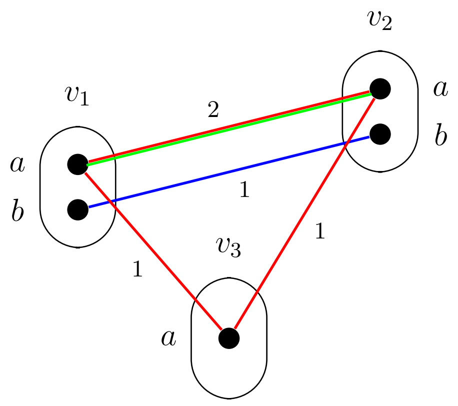

Fig. 3. Microstructure of $\mathcal{P}$ described in Example 3.

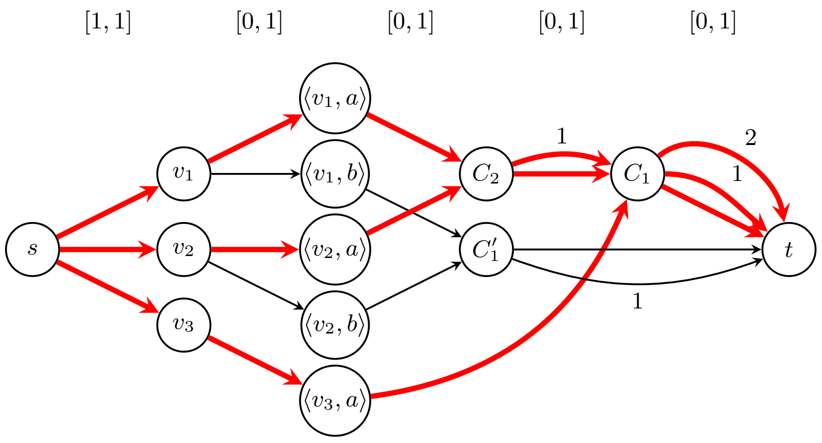

Fig. 4. Network $G_{\mathcal{P}}$ corresponding to the VCSP $\mathcal{P}$ of Example 3.

This corresponds precisely to the cost of the assignment $\langle a_1, \ldots, a_n \rangle$ in $\mathcal{P}$, since in a clique C_α with father clique C_β, each of the $(N(C_\alpha) - 1)N(C_\alpha)/2$ binary constraints contributes a cost of $\alpha - \beta$ over and above the cost of β for each of the edges in C_β.

Theorem 3. *VCSPs satisfying the joint-winner property are recognisable and solvable in polynomial time.*

Proof. From Definition 1, recognition can be achieved in $O(n^3 d^3)$ time, where $d = \max_{1 \leq i \leq n} |D_i|$ is the size of the largest domain.

To solve a VCSP satisfying the JWP, we create a vertex for each of the cliques corresponding to binary costs of at least α. There are at most $|D_i| \times |D_j|$ different costs in the cost function c_{ij}. Hence in total there are at most $O(n^2 d^2)$ different cliques. So our network has $O(n^2 d^2 + nd + n + 2) = O(n^2 d^2)$ vertices. The result follows from the fact that a polynomial-time algorithm exists for determining a minimum-cost maximum flow in a network. In particular, using the successive shortest path algorithm, the running time is $O(n \cdot SP)$, where SP is the time to compute a shortest directed path in the network [31,1]. Using Fibonacci heaps, this is $O(n(n^4 d^4 + n^2 d^2 \log(n^2 d^2))) = O(n^5 d^4)$. $\square$

6 Maximality

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 of possible combinations of costs within a subproblem is maximal if extending it to include any other single combination of costs renders the problem NP-hard. In particular, since JWP is defined on 3-variable subproblems, we call an instance *3-maximal* if extending it to include any other single combination of costs on 3 variables renders the problem NP-hard. 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.

In this section we show a special case for which the joint-winner property is 3-maximal, namely when all binary cost functions take on only two possible costs $\alpha < \beta$.

Theorem 4. *If all costs belong to $\{\alpha, \beta\}$ (for some fixed distinct costs $\alpha < \beta$), then the joint-winner property defines a 3-maximal tractable class provided $d > 2$ or $(d \geq 2) \wedge (\beta < \infty)$, where d is the maximum domain size.*

Proof. 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 just one other fixed combination. Since all costs belong to $\{\alpha, \beta\}$ where $\alpha < \beta$, from Definition 1, the only situation forbidden by the JWP is that there are 3 variables v_i, 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 $c_{ik}(a, c) = c_{jk}(b, c) = \beta$. 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 $d > 2$.

If $\beta < \infty$, allowing all combinations of costs in $\{\alpha, \beta\}$ is equivalent to the set of instances of MAX-CSP in which no two constraints can have the same scope. The NP-hardness of this latter problem for $d \geq 2$ follows from the following reduction from MAX-CSP [26]. A polynomial reduction of an instance I of 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. 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). $\square$

7 Conclusions

We consider the tractable class of VCSPs defined by 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.

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).

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B. Network Flows: Theory, Algorithms, and Applications. Prentice Hall/Pearson (2005)
2. Bistarelli, S., Montanari, U., Rossi, F.: Semiring-based Constraint Satisfaction and Optimisation. Journal of the ACM **44**(2) (1997) 201–236
3. Bruno, J.L., Coffman, E.G.,Sethi, R.: Scheduling Independent Tasks to Reduce Mean Finishing Time. Communications of the ACM **17**(7) (1974) 382–387
4. Bulatov, A., Krokhin, A., Jeavons, P.: Classifying the Complexity of Constraints using Finite Algebras. SIAM Journal on Computing **34**(3) (2005) 720–742
5. Chudnovsky, M., Cornuéjols, G., Liu, X., Seymour, P.D., Vušković, K.: Recognizing Berge graphs. Combinatorica **25**(2) (2005) 143–186
6. Chudnovsky, M., Robertson, N., Seymour, P., Thomas, R.: The strong perfect graph theorem. Annals of Mathematics **164**(1) (2006) 51–229
7. Cohen, D.A., Cooper, M.C., Jeavons, P.G.: Generalising submodularity and Horn clauses: Tractable optimization problems defined by tournament pair multimorphisms. Theoretical Computer Science **401**(1-3) (2008) 36–51
8. Cohen, D.A., Cooper, M.C., Jeavons, P.G., Krokhin, A.A.: The Complexity of Soft Constraint Satisfaction. Artificial Intelligence **170**(11) (2006) 983–1016
9. Cohen, D., Jeavons, P.: The complexity of constraint languages. In: Handbook of Constraint Programming. Rossi, F., van Beek, P., Walsh, T. (eds) Elsevier (2006)
10. Cohen, D.A.: A New Class of Binary CSPs for which Arc-Constistency Is a Decision Procedure. In: CP (2003) 807–811

11. Cooper, M.C., Jeavons, P.G., Salamon, A.Z.: Generalizing constraint satisfaction on trees: hybrid tractability and variable elimination. Artificial Intelligence (2010)
12. Dechter, R.: Constraint Processing. Morgan Kaufmann (2003)
13. Dechter, R., Pearl, J.: Network-based Heuristics for Constraint Satisfaction Problems. Artificial Intelligence **34**(1) (1988) 1–38
14. Edmonds, J.: Paths, trees, and flowers. Canad. J. Math **17** (1965) 449–467
15. Feder, T., Vardi, M.: The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: A Study through Datalog and Group Theory. SIAM Journal on Computing **28**(1) (1998) 57–104
16. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman (1979)
17. Grohe, M.: The complexity of homomorphism and constraint satisfaction problems seen from the other side. Journal of the ACM **54**(1) (2007)
18. Grötschel, M., Lovasz, L., Schrijver, A.: The ellipsoid method and its consequences in combinatorial optimization. Combinatorica **1**(2) (1981) 169–198
19. Horn, W.A.: Minimizing Average Flow Time with Parallel Machines. Operations Research **21**(3) (1973) 846–847
20. Jeavons, P.: On the Algebraic Structure of Combinatorial Problems. Theoretical Computer Science **200**(1-2) (1998) 185–204
21. Jégou, P.: Decomposition of Domains Based on the Micro-Structure of Finite Constraint-Satisfaction Problems. In: AAAI (1993) 731–736
22. Kumar, T.K.S.: A framework for hybrid tractability results in boolean weighted constraint satisfaction problems. In: CP (2008) 282–297
23. Lee, J.H.M., Leung, K.L.: Towards efficient consistency enforcement for global constraints in weighted constraint satisfaction. In: IJCAI (2009) 559–565
24. Minty, G.J.: On maximal independent sets of vertices in claw-free graphs. J. Comb. Theory, Ser. B **28**(3) (1980) 284–304
25. Nakamura, D., Tamura, A.: A revision of Minty's algorithm for finding a maximum weighted stable set of a claw-free graph. J. Oper. Res. Soc. **44**(2) (2001) 194–204
26. Papadimitriou, C.H. Computational Complexity. Addison-Wesley, 1994.
27. Petit, T., Régin, J.C., Bessière, C. Specific Filtering Algorithms for Over-Constrained Problems. In: CP (2001) 451–463
28. Régin, J.C.: A filtering algorithm for constraints of difference in CSPs. In: AAAI (1994) 362–367
29. Régin, J.C.: Cost-based arc consistency for global cardinality constraints. Constraints **7**(3-4) (2002) 387–405
30. Schiex, T., Fargier, H., Verfaillie, G.: Valued Constraint Satisfaction Problems: Hard and Easy Problems. In: IJCAI (1995) 631–637
31. Schrijver, A.: Combinatorial Optimization: Polyhedra and Efficiency. Volume 24 of Algorithms and Combinatorics. Springer (2003)
32. Takhanov, R.: A Dichotomy Theorem for the General Minimum Cost Homomorphism Problem. In: STACS (2010) 657–668
33. van Hoeve, W.J., Pesant, G., Rousseau, L.M.: On global warming: Flow-based soft global constraints. J. Heuristics **12**(4-5) (2006) 347–373