The complexity of valued constraint models*

Stanislav Živný and Peter G. Jeavons

Computing Laboratory, University of Oxford, Oxford, UK
{stanislav.zivny,peter.jeavons}@comlab.ox.ac.uk

Abstract. The Valued Constraint Satisfaction Problem (VCSP) is a general framework encompassing many optimisation problems. We discuss precisely what it means for a problem to be *modelled* in the VCSP framework. Using our analysis, we show that some optimisation problems, such as (s,t)-Min-Cut and Submodular Function Minimisation, can be modelled using a restricted set of valued constraints which are tractable to solve regardless of how they are combined. Hence, these problems can be viewed as special cases of more general problems which include all possible instances using the same forms of valued constraint. However, other, apparently similar, problems such as Min-Cut and Symmetric Submodular Function Minimisation, which also have polynomial-time algorithms, can only be naturally modelled in the VCSP framework by using valued constraints which can represent NP-complete problems. This suggests that the reason for tractability in these problems is more subtle; it relies not only on the form of the valued constraints, but also on the precise structure of the problem. Furthermore, our results suggest that allowing constant constraints can significantly alter the complexity of problems in the VCSP framework, in contrast to the CSP framework.

1 Introduction

The study of combinatorial optimisation traditionally considers a range of specific problem types, including integer programming problems, problems on graphs and networks, and Boolean problems [2], such as submodular function minimisation [14]. An important issue for any combinatorial optimisation problem is how to choose an effective representation, which can be crucial to the efficiency of solving the problem.

The valued constraint satisfaction problem (VCSP) is a single generic framework, for modelling a wide range of optimisation problems [1,11,12].

Our aim in this paper is to investigate which standard combinatorial optimisation problems can be modelled in the VCSP framework, and whether finding such models sheds new light on the complexity of these problems. We will focus on Boolean problems (i.e., each variable can take one of two possible values), which are equivalent to minimising functions defined on sets.

We need to be a little careful in defining what it means for a problem to be modelled in the VCSP framework: we clearly need to exclude modelling procedures that simply obliterate the structure of the problem we are attempting to model. For example, simply finding a solution to each given instance (using some algorithm) and then creating a VCSP instance which allows precisely that

* Stanislav Živný is supported by EPSRC grant EP/F01161X/1.

solution is not a useful approach to modelling. The standard way of excluding such pathological approaches is to limit the computational resources allowed to transform the problem from one representation to another. However, when dealing with tractable problems, we also need a suitably tight definition of what it means for a problem to be *modelled* in the VCSP framework. (More on modelling via constraints can be found in [15].)

In this paper we shall say that we have a VCSP *model* for a given combinatorial optimisation problem if the entire function to be minimised in that problem is *expressible* using some collection of valued constraints (in a precise sense defined below; note that this notion of expressibility was a major tool in the complexity analysis of a wide variety of Boolean constraint problems carried out by Creignou et al. [5], where it was referred to as *implementation*). We show that many standard problems can be modelled in this way. Moreover, some problems, including for example the (s,t)-Min-Cut problem and the problem of Submodular Function Minimisation, can be modelled using very restricted forms of constraints. In fact the forms of constraints needed to model these problems are sufficiently restricted that they can be solved in polynomial time regardless of how they are combined. Hence, these problems can be viewed as special cases of more general problems which include all possible instances using the same forms of valued constraint.

On the other hand, we show that other apparently similar problems, which also have polynomial-time algorithms, can only be modelled using forms of constraint which are powerful enough to represent NP-complete problems. Our examples include the standard Min-Cut problem. This result indicates that the reason for the tractability of such problems relies on the precise structure of the problem and not just the form of the individual constraints. Such problems provide a fresh incentive to develop the theoretical analysis of the complexity of valued constraint problems, which currently has very little to say about such "hybrid" reasons for tractability.

2 Background

Given some fixed set D, a function from D^k to $\overline{\mathbb{Q}}_+$, where $\overline{\mathbb{Q}}_+$ is the set of all positive rational numbers together with infinity will be called a *cost function*.

Definition 1. *An instance $\mathcal{P}$ of* VCSP *is a triple $\langle V, D, \mathcal{C}\rangle$, where V is a finite set of* variables*, which are to be assigned values from the set D, and $\mathcal{C}$ is a set of* valued constraints*. Each constraint $c \in \mathcal{C}$ is a pair $c = \langle \sigma, \phi\rangle$, where σ is a tuple of variables of length $|\sigma|$, called the* scope *of c, and $\phi : D^{|\sigma|} \to \overline{\mathbb{Q}}_+$ is a cost function. An* assignment *for the instance $\mathcal{P}$ is a mapping s from V to D. The* cost *of an assignment s is defined as follows:*

$$Cost_{\mathcal{P}}(s) = \sum_{\langle\langle v_1, v_2, \ldots, v_m\rangle, \phi\rangle \in \mathcal{C}} \phi(\langle s(v_1), s(v_2), \ldots, s(v_m)\rangle).$$

A solution *to $\mathcal{P}$ is an assignment with minimum cost.*

The VCSP is a very general framework which allows us to describe many optimisation problems, including many NP-hard problems [11]. Any set, Γ, of cost functions is called a *valued constraint language*. The class VCSP(Γ) is

defined to be the class of all VCSP instances where the cost functions of all valued constraints lie in Γ. The complexity of a valued constraint language Γ is defined as the complexity of VCSP(Γ). A valued constraint language Γ is called *tractable* if VCSP(Γ') is solvable in polynomial time for every finite $\Gamma' \subseteq \Gamma$, and Γ is called *intractable* if VCSP(Γ') is NP-hard for some finite subset $\Gamma' \subseteq \Gamma$. Many examples of tractable valued constraint languages have now been identified [4].

Definition 2. *For any* VCSP *instance* $\mathcal{P} = \langle V, D, \mathcal{C} \rangle$*, and any list* $l = \langle v_1, \ldots, v_m \rangle$ *of variables of* $\mathcal{P}$*, the* projection *of* $\mathcal{P}$ *onto* l*, denoted* $\pi_l(\mathcal{P})$*, is the* m*-ary cost function defined as follows:*

$$\pi_l(\mathcal{P})(x_1, \ldots, x_m) \;=\; \min_{\{s : V \to D \mid \langle s(v_1), \ldots, s(v_m) \rangle = \langle x_1, \ldots, x_m \rangle\}} Cost_{\mathcal{P}}(s).$$

We say that a cost function ϕ *is* expressible *over a valued constraint language* Γ *if there exists an instance* $\mathcal{P} \in$ VCSP(Γ) *and a list* l *of variables of* $\mathcal{P}$ *such that* $\pi_l(\mathcal{P}) = \phi$*.*

We denote by $\langle \Gamma \rangle$ the *expressive power* of Γ, which is the set of all cost functions expressible over Γ up to additive and multiplicative constants.

Theorem 3 ([4]). *For any valued constraint language* Γ *and any cost function* ϕ *expressible over* Γ*,* VCSP(Γ) *and* VCSP($\Gamma \cup \{\phi\}$) *are linear-time equivalent.*

3 Boolean optimisation problems

In this section we recall some standard Boolean optimisation problems.

(s,t)-Min-Cut: For a directed graph $G = \langle V, E \rangle$ with weights $w : E \to \mathbb{Q}_+$, $s, t \in V$, C is an (s,t)-*cut* if $C \subseteq V$, $s \in C$ and $t \notin C$. The weight of C is $\sum_{(u,v) \in E, u \in C, v \notin C} w(u,v)$. The (s,t)-Min-Cut problem consists in finding the minimum-weight (s,t)-cut. A cubic-time algorithm (in the number of vertices) based on network flows is known for this problem [6].

Min-Cut: For an undirected graph $G = \langle V, E \rangle$ with weights $w : E \to \mathbb{Q}_+$, C is a *cut* if $C \subseteq V$. The weight of C is defined as above. The Min-Cut problem consists in finding the minimum-weight cut C such that $C \neq \emptyset$ and $C \neq V$. Using the cubic-time algorithm for the (s,t)-Min-Cut problem [6], one can easily construct an algorithm for the Min-Cut problem of order $O(n^4)$. A purely combinatorial cubic-time algorithm which is not based on network flows is also known for this problem [16].

Submodular Function Minimisation (SFM): A function ψ defined on subsets of a set V is called a *submodular function* [14] if, for all subsets S and T of V, $\psi(S \cap T) + \psi(S \cup T) \leq \psi(S) + \psi(T)$. The problem of Submodular Function Minimisation (SFM) consists in finding a subset S of V for which the value of $\psi(S)$ is minimal. It is a central problem in discrete optimisation, with links to many different areas [14]. The time complexity of the fastest published algorithm for SFM is $O(n^5 \mathrm{EO} + n^6)$ where $n = |V|$ and EO is the time to evaluate $\psi(S)$ for some $S \subseteq V$ [9,7].

(s,t)-SFM: Given a submodular function ψ defined on subsets of a set V, and two elements $s,t \in V$, the problem of (s,t)-Submodular Function Minimisation consists in finding a nonempty subset S of V such that $s \in S$, $t \notin S$, and the value of $\psi(S)$ is minimal.

Proposition 4. (s,t)-SFM *is linear-time equivalent to* SFM.

Proof. First we show that (s,t)-SFM is reducible to SFM. Clearly, $S \subset V$ is a solution to an instance $\langle V, \psi \rangle$ of (s,t)-SFM if and only if $S \setminus \{s\}$ is a solution to the instance $\langle V \setminus \{s,t\}, \overline{\psi} \rangle$ of SFM where $\overline{\psi}(U) = \psi(U \cup \{s\})$. On the other hand, $S \subseteq V$ is a solution to an instance $\langle V, \psi \rangle$ of SFM if and only if $S \cup \{s\}$ is a solution to the instance $\langle V \cup \{s,t\}, \overline{\psi} \rangle$ of (s,t)-SFM, where $\overline{\psi}(U) = \psi(U \setminus \{s\})$. □

Symmetric SFM (SSFM): Given a submodular function ψ defined on subsets of a set V, we say that ψ is *symmetric* if for every $U \subseteq V$, $\psi(U) = \psi(V \setminus U)$. Note that $\psi(\emptyset) = \psi(V) \leq \psi(U)$ for every $U \subseteq V$. Symmetric Submodular Function Minimisation consists in finding a nonempty proper subset S of V for which the value of $\psi(S)$ is minimal. Queyranne extended the cubic-time algorithm for the Min-Cut problem mentioned above to obtain a purely combinatorial cubic-time algorithm for SSFM [10].

(s,t)-SSFM: Given a symmetric submodular set function ψ on subsets of V and two elements $s,t \in V$, the problem of (s,t)-Symmetric Submodular Function Minimisation consists in finding a proper nonempty subset S of V, where $s \in S$ and $t \notin S$, for which the value of $\psi(S)$ is minimal.
Similarly to the proof of Proposition 4, it can be easily shown that SSFM is reducible to (s,t)-SSFM.

Proposition 5 ([10]). SFM *is linear-time reducible to* (s,t)-SSFM.

However, using the same proof idea as Proposition 4 does *not* show that (s,t)-SSFM is reducible to SSFM (as it seems difficult to preserve two properties, namely being submodular and symmetric, at the same time).[1] Moreover, Proposition 5 shows that (s,t)-SSFM is as hard as SFM, so a time-complexity-preserving reduction from (s,t)-SSFM to SSFM would make SFM equivalent to SSFM, which would yield a cubic-time algorithm for SFM. This would be a major advance in discrete optimisation.

4 Modelling in the VCSP framework

It is easy (and standard) to see that any set function ψ defined on subsets of $V = \{v_1, \ldots, v_n\}$ can be associated with a function $\phi : \{0,1\}^n \to \overline{\mathbb{Q}}_+$ defined as follows: for each tuple $t \in \{0,1\}^n$, set $\phi(t) = \psi(T)$, where $T = \{v_i \mid t[i] = 1\}$ (moreover, if $U \subseteq V$ is forbidden, then set $\psi(T) = \infty$).

Note that the submodularity condition on a set function ψ is equivalent to the following condition on the associated Boolean function ϕ: for every two tuples $s,t \in \{0,1\}^n$, $\phi(\textsc{Min}(s,t)) + \phi(\textsc{Max}(s,t)) \leq \phi(s) + \phi(t)$, where both Min and Max are applied coordinate-wise. We therefore call a cost function ϕ satisfying

[1] More on the relationships between SFM, SSFM and (s,t)-SSFM can be found in [8].

this condition submodular. We now define a precise notion of what it means to model a problem in the VCSP framework. This notion is designed to rule out pathological cases and ensure that the models we allow do provide some insight into the nature of the problem being modelled.

Definition 6. *Let $\mathcal{P}$ be a problem which consists in minimising a given function ψ defined on the subsets of a given set V, and let ϕ be the associated Boolean cost function, as defined above. We say that $\mathcal{P}$ can be* e-modelled *by* VCSP(Γ) *if ϕ can be expressed over Γ.*

In other words, a problem $\mathcal{P}$ can be *e-modelled* by VCSP(Γ) if, for any instance $\langle\{v_1, v_2, \ldots, v_n\}, \psi\rangle$ of $\mathcal{P}$, there is an instance $\mathcal{I} = \langle W, \{0,1\}, \mathcal{C}\rangle$ of VCSP(Γ), and a list of variables $\langle w_{v_1}, w_{v_2}, \ldots, w_{v_n}\rangle \subseteq W$, such that for any $S \subseteq V$, the minimal cost over all assignments for $\mathcal{I}$ which assign each variable w_{v_i} the value 0 or 1 according to whether or not $v_i \in S$, is equal to $\psi(S)$.

Theorem 7.
- (s,t)-Min-Cut, SFM, (s,t)-SFM, *and* (s,t)-SSFM *can be e-modelled by* VCSP(Γ)*, by a suitable choice of tractable valued constraint language Γ.*
- Min-Cut *and* SSFM *can be e-modelled by* VCSP(Γ)*, but only by using an intractable language Γ.*

Proof. Consider first the (s,t)-Min-Cut problem. We have to prove that there exists a tractable valued constraint language Γ_{cut} such that (s,t)-Min-Cut can be e-modelled by VCSP(Γ_{cut}). For any $w \in \mathbb{Q}_+$, we define the binary cost function λ^w as $\lambda^w(x,y) = w$ if $\langle x,y\rangle = \langle 0,1\rangle$, and 0 otherwise. For each $d \in \{0,1\}$ and each $c \in \overline{\mathbb{Q}}_+$, we define the unary cost function μ_d^c as $\mu_d^c(x) = c$ if $x \neq d$, and 0 otherwise. Now let Γ_{cut} consist of all λ^w and μ_u^c for $w \in \mathbb{Q}_+$, $c \in \overline{\mathbb{Q}}_+$ and $d \in \{0,1\}$.

Now consider any instance of (s,t)-Min-Cut with graph $G = \langle V, E\rangle$ and weight function $w : E \to \mathbb{Q}_+$. Define a corresponding instance $\mathcal{I}$ of VCSP(Γ_{cut}) as $\mathcal{I} = \langle V, \{0,1\}, \{\langle\langle i,j\rangle, \lambda^{w(i,j)}\rangle \mid \langle i,j\rangle \in E\} \cup \{\langle s, \mu_0^\infty\rangle, \langle t, \mu_1^\infty\rangle\}\rangle$. Note that in any solution to $\mathcal{I}$ the source and target nodes, s and t, must take the values 0 and 1, respectively. Moreover, the weight of any cut containing s and not containing t is equal to the cost of the corresponding assignment to $\mathcal{I}$. Hence we have shown that (s,t)-Min-Cut can be e-modelled by VCSP(Γ_{cut}).

On the other hand, we claim that VCSP(Γ_{cut}) can be reduced to (s,t)-Min-Cut in linear time as follows: any unary constraint on variable v with cost function μ_0^c (respectively μ_1^c) is represented by an edge of weight c from the source node s to node v (respectively, from node v to the target node t). Any binary constraint on variables v_1, v_2 with cost function λ^w is represented by an edge of weight w from nodes v_1 to v_2. Hence VCSP(Γ_{cut}) has the same time complexity as (s,t)-Min-Cut.

Next we consider the SFM problem. Let Γ_{sub} be the valued constraint language which consists of all submodular cost functions. Because submodular cost functions may take infinite values, instances of VCSP(Γ_{sub}) cannot be simply solved by standard submodular function minimisation algorithms for finite-valued submodular functions. However, Cohen et al. showed [4] that VCSP(Γ_{sub})

is polynomial-time reducible to the problem of SFM over a ring family[2] which is known to be equivalent to SFM [13], so Γ_{sub} is tractable.

Now we consider the (s,t)-SFM problem. A *constant constraint* is a unary constraint μ_d^∞ for an arbitrary $d \in D$. We denote by $\Gamma_{\mathsf{const}} = \{\mu_d^\infty \mid d \in D\}$ the valued constraint language consisting of all constant constraints.

Clearly, (s,t)-SFM can be e-modelled by $\mathrm{VCSP}(\Gamma_{\mathsf{sub}} \cup \Gamma_{\mathsf{const}})$. Also, as constant constraints are submodular, we have $\Gamma_{\mathsf{const}} \subseteq \Gamma_{\mathsf{sub}}$, so (s,t)-SFM can be e-modelled by $\mathrm{VCSP}(\Gamma_{\mathsf{sub}})$ and we have already shown that Γ_{sub} is tractable.

Now we consider the Min-Cut problem. To e-model Min-Cut, we can use the valued constraint language Γ_{cut} defined above, and then forbid the empty and complete cuts by putting a crisp Not-All-Equal constraint over the variables $w_1, \ldots, w_n$ which represent V. In other words, Min-Cut can be e-modelled by $\mathrm{VCSP}(\Gamma_{\mathsf{cut}} \cup \Gamma_{\mathsf{nae}})$ where Γ_{nae} consists of Not-All-Equal constraints of all possible arities.

However, since Not-All-Equal Satisfiability is NP-complete, it follows that Γ_{nae} is intractable. We now show that for any valued constraint language Γ which e-models Min-Cut, $\mathrm{VCSP}(\Gamma)$ must be intractable.

Let Γ be a valued constraint language such that Min-Cut can be e-modelled by $\mathrm{VCSP}(\Gamma)$. Consider an instance of Min-Cut which is a triangle with all weights set to zero. The empty and complete cuts are forbidden, any other cut has cost 0, so Γ expresses a Not-All-Equal constraint, and hence is intractable.

Now, let Γ_{ssub} be the valued constraint language which consists of all symmetric submodular functions. The SSFM problem can be e-modelled by $\mathrm{VCSP}(\Gamma_{\mathsf{ssub}} \cup \Gamma_{\mathsf{nae}})$ in a similar way to Min-Cut. However, since Min-Cut is just a special case of SSFM, it follows from the argument just given that any suitable choice of Γ will again be intractable.

Finally, we consider the (s,t)-SSFM problem. Similarly to the arguments above, (s,t)-SSFM can be e-modelled by the language $\Gamma_{\mathsf{ssub}} \cup \Gamma_{\mathsf{const}}$ which is a subset of Γ_{sub}. □

We now examine more closely the language $\Gamma_{\mathsf{ssub}} \cup \Gamma_{\mathsf{const}}$ consisting of symmetric submodular constraints and constant constraints, which was introduced to model the (s,t)-SSFM problem. The following proposition shows that all submodular constraints can be expressed by this language.

Proposition 8. $\Gamma_{\mathsf{sub}} = \langle \Gamma_{\mathsf{ssub}} \cup \Gamma_{\mathsf{const}} \rangle$.

Proof. Since $\Gamma_{\mathsf{ssub}} \cup \Gamma_{\mathsf{const}} \subseteq \Gamma_{\mathsf{sub}}$, it only remains to prove that any submodular function can be expressed by symmetric submodular functions and unary constant constraints. Our proof is adapted from the construction given in [10].

Let ψ be a submodular function defined on subsets of a set V, with $|V| = n$. Let $M = \mathrm{Max}(\psi(U) | U \subseteq V, \psi(U) < \infty)$. Define $\overline{V} = V \cup \{s,t\}$ for $s,t \notin V$ and $\overline{\psi}$ as follows:

$$\overline{\psi}(U) \;=\; \begin{cases} \psi(U \setminus \{s\}) + M(n+2) & \text{if } s \in U \text{ and } t \notin U, \\ M \mid U \mid & \text{if } s,t \in U, \\ \overline{\psi}(V \setminus U) & \text{if } s \notin U. \end{cases}$$

[2] A collection of sets $\mathcal{C}$ is called a *ring family* if $\mathcal{C}$ is closed under union and intersection.

It can be shown that $\overline{\psi}$ is symmetric and submodular, and that $S \subseteq \overline{V}$ where $s \in S$ and $t \notin S$ minimises $\overline{\psi}$ if and only if $S \setminus \{s\}$ minimises ψ [10]. Hence ψ can be expressed (up to an additive constant) by $\overline{\psi}$ and two elements of Γ_{const}. □

Note that, in the CSP, it has been shown that adding constant constraints to a tractable language which is a core does not change the complexity (i.e., $\mathrm{CSP}(\Gamma)$ is linear-time equivalent to $\mathrm{CSP}(\Gamma \cup \Gamma_{\mathsf{const}})$, provided Γ is a core) [3]. However, in the valued constraint case Proposition 8 suggests that the situation may be rather different: the complexity of the valued language Γ_{ssub}, consisting of all symmetric submodular functions, is cubic, whereas adding constant constraints allows us to express all submodular functions. The best known algorithm for minimising arbitrary submodular functions is $\Omega(n^6)$.

The concept of e-modelling we have defined here is designed to avoid trivial models by requiring the constraints in the model to be capable of expressing the function being minimised. However, more relaxed notions of modelling can also yield non-trivial representations, and may provide more flexibility, as the following result indicates:

Theorem 9. *There is a tractable valued constraint language Γ over an infinite domain such that* Min-Cut *can be reduced to* $\mathrm{VCSP}(\Gamma)$ *in linear-time.*

Proof. Let $D = \{\langle i, d\rangle \mid i \in \mathbb{N}, d \in \{0,1\}\}$.

For any $w \in \mathbb{Q}_+$ and $k \in \mathbb{N}$, we define the binary cost function λ_k^w as follows:

$$\lambda_k^w(\langle i, d_1\rangle, \langle j, d_2\rangle) = \begin{cases} \infty & \text{if } \textsc{Max}(i,j) > k \text{ or } i \neq j, \\ 0 & \text{if } i = j \text{ and } d_1 = d_2, \\ w & \text{otherwise.} \end{cases}$$

Next, for any $d \in \{0,1\}$ and $k \in \mathbb{N}$, we define the unary cost function $\mu_{k \to d}$ as follows:

$$\mu_{k \to d}(\langle i, x\rangle) = \begin{cases} \infty & \text{if } i = k \text{ and } x \neq d, \\ 0 & \text{otherwise.} \end{cases}$$

Let Γ be the valued constraint language over D consisting of the cost functions λ_k^w and $\mu_{k \to d}$ for all $w \in \mathbb{Q}_+$, $k \in \mathbb{N}$ and $d \in \{0,1\}$. We first show how to reduce Min-Cut to $\mathrm{VCSP}(\Gamma)$.

Consider an instance of Min-Cut with graph $G = \langle V, E\rangle$ (without loss of generality, without isolated vertices), where $V = \{v_1, \ldots, v_n\}$, and weight $w(i,j)$ for every $\{v_i, v_j\} \in E$. We define a corresponding instance $\mathcal{I}$ of $\mathrm{VCSP}(\Gamma)$ as follows. Let $\mathcal{I} = \langle V, D, \mathcal{C}\rangle$, where $\mathcal{C} = \{\langle\langle v_i, v_j\rangle, \lambda_{n-1}^{w(i,j)}\rangle \mid i < j, \{v_i, v_j\} \in E\} \cup \{\langle v_1, \mu_{(i-1)\to 0}\rangle, \langle v_i, \mu_{(i-1)\to 1}\rangle \mid v_i \in V \setminus \{v_1\}\}$.

Clearly, if $\emptyset \neq U \subsetneq V$ is a minimum cut of G, then so is $V \setminus U$. Therefore, we can assume that $v_1 \in U$. Note that a necessary condition for an assignment of variables of $\mathcal{I}$ to have a finite cost is that there exists some $i \in \{1, \ldots, n-1\}$ such that every variable of $\mathcal{I}$ is assigned a value of the form $\langle i, .\rangle$. Such an i forces the variable v_1 to be assigned $\langle i, 0\rangle$ (i.e., v_1 belongs to the cut), and the variable v_{i+1} to be assigned $\langle i, 1\rangle$ (i.e., v_{i+1} does not belong to the cut). For the minimum cut U, there has to be an $i \in \{2, \ldots, n\}$ such that $v_i \notin U$ and the assignment of $\langle i-1, 0\rangle$ to variables in U, and $\langle i-1, 1\rangle$ to variables not in U, gives the weight of U. (This construction is similar to the one used in the proof of Theorem 7 which shows that (s,t)-Min-Cut can be e-modelled.)

It remains to show that Γ is tractable. Let Γ' be any finite subset of Γ, and let k be the biggest number which occurs in the subscript of any λ cost function in Γ'. The cost of an assignment to any instance of VCSP(Γ') which is of the form $\langle i, . \rangle$, for $i \in \{1, \ldots, k\}$, corresponds to the cost of an (s,t)-cut in an associated graph. Therefore, VCSP(Γ') can be solved by solving k instances of the (s,t)-Min-Cut problem, and hence its time complexity is $O(kn^3)$. For a fixed Γ' (and hence a fixed value of k) this is polynomial in the size of the instance. □

The language Γ used in Theorem 9 has a much higher complexity than Min-Cut. It is an interesting open question whether Min-Cut can be reduced to VCSP(Γ) for some Γ with the same complexity as Min-Cut (i.e., such that VCSP(Γ) is linear-time reducible to Min-Cut).

References

1. Bistarelli, S., Fargier, H., Montanari, U., Rossi, F., Schiex, T., Verfaillie, G.: Semiring-based CSPs and valued CSPs: Frameworks, properties, and comparison. Constraints **4** (1999) 199–240
2. Boros, E., Hammer, P.L.: Pseudo-boolean optimization. Discrete Applied Mathematics **123**(1-3) (2002) 155–225
3. Bulatov, A., Krokhin, A., Jeavons, P.: Classifying the complexity of constraints using finite algebras. SIAM Journal on Computing **34**(3) (2005) 720–742
4. Cohen, D., Cooper, M., Jeavons, P., Krokhin, A.: The complexity of soft constraint satisfaction. Artificial Intelligence **170** (2006) 983–1016
5. Creignou, N., Khanna, S., Sudan, M.: Complexity Classification of Boolean Constraint Satisfaction Problems. vol. 7 of SIAM Monographs on Discrete Mathematics and Applications, SIAM, 2001.
6. Goldberg, A., Tarjan, R.: A new approach to the maximum flow problem. Journal of the ACM **35** (1988) 921–940
7. Iwata, S., Orlin, J. B.: A Simple Combinatorial Algorithm for Submodular Function Minimization. In Proceedings of the 20th SODA, pages 1230–1237, 2009.
8. Narayanan, H.: A note on the minimization of symmetric and general submodular functions. Discrete Applied Mathematics **131**(2) (2003) 513–522
9. Orlin, J.B.: A faster strongly polynomial time algorithm for submodular function minimization. Mathematical Programming 118 (2009) 237–251.
10. Queyranne, M.: Minimising symmetric submodular functions. Mathematical Programming **82** (1998) 3–12
11. Rossi, F., van Beek, P., Walsh, T., eds.: The Handbook of Constraint Programming. Elsevier (2006)
12. Schiex, T., Fargier, H., Verfaillie, G.: Valued constraint satisfaction problems: hard and easy problems. In: Proceedings of the 14th IJCAI, Montreal, Canada (1995)
13. Schrijver, A.: A combinatorial algorithm minimizing submodular functions in strongly polynomial time. J. of Combinatorial Theory, Series B **80** (2000) 346–355
14. Schrijver, A.: Combinatorial Optimization: Polyhedra and Efficiency. Volume 24 of Algorithms and Combinatorics. Springer (2003)
15. Smith, B. Modelling. Chapter 11 of the Handbook of Constraint Programming. Elsevier (2006)
16. Stoer, M., Wagner, F.: A simple min-cut algorithm. Journal of the ACM **44**(4) (1997) 585–591