A Maximal Tractable Class of Soft Constraints

Content Areas: constraint satisfaction, computational complexity

Abstract

Many optimization problems can be expressed using some form of soft constraints, where different measures of desirability are associated with different combinations of domain values for specified subsets of variables. In this paper we identify a class of soft binary constraints for which the problem of finding the optimal solution is tractable. In other words, we show that for any given set of such constraints, there exists a polynomial time algorithm to determine the assignment having the best overall combined measure of desirability. This tractable class includes many commonly-occurring soft constraints, such as "as near as possible" or "as soon as possible after", as well as crisp constraints such as "greater than".

1 Introduction

The constraint satisfaction framework is widely acknowledged as a convenient and efficient way to model and solve a wide variety of problems arising in Artificial Intelligence, including planning and scheduling, image processing and natural language understanding.

In the standard framework a *constraint* is usually taken to be a predicate, or relation, specifying the allowed combinations of values for some fixed collection of variables: we will refer to such constraints here as *crisp* constraints. A number of authors have suggested that the usefulness of the constraint satisfaction framework could be greatly enhanced by extending the definition of a constraint to include also *soft* constraints, which allow different measures of desirability to be associated with different combinations of values [Bistarelli *et al.*, 1999]. In this extended framework a constraint can be seen as a *function*, mapping each possible combination of values to a measure of desirability or undesirability. Finding a solution to a set of constraints then means finding an assignment of values to all of the variables which has the best overall combined desirability measure.

Example 1.1 For example, consider an optimization problem where we have $2n$ variables, $v_1, v_2, \ldots, v_{2n}$, and we wish to assign each variable an integer value in the range $1, 2, \ldots, n$, subject to the following restrictions:

- Each variable v_i should be assigned a value that is as close as possible to $i/2$.
- Each pair of variables v_i, v_{2i} should be assigned a pair of values that are as similar as possible.

To model this situation we might impose the following soft constraints:

- A unary constraint on each v_i specified by a function ψ_i, where $\psi_i(x) = |x - i/2|^r$ for some $r \geq 1$.
- A binary constraint on each pair v_i, v_{2i} specified by a function δ_r, where $\delta_r(x, y) = |x - y|^r$ for some $r \geq 1$.

We would then seek an assignment to all of the variables which minimized the sum of these constraint functions,

$$\sum_{i=1}^{2n} \psi_i(v_i) + \sum_{i=1}^{n} \delta_r(v_i, v_{2i}).$$

□

The cost of allowing additional flexibility in the specification of constraints, in order to model requirements of this kind, is generally an increase in computational difficulty. In the case of crisp constraints there has been considerable progress in identifying classes of constraints which are *tractable*, in the sense that there exists a polynomial time algorithm to determine whether or not any collection of constraints from such a class can be simultaneously satisfied [Feder and Vardi, 1998; Jeavons *et al.*, 1997]. In the case of soft constraints there has not yet been any detailed investigation of the tractable cases, although there are many significant results in the literature on combinatorial optimization which are clearly relevant to this question [Nemhauser and Wolsey, 1988].

In this paper we make use of the idea of a *submodular function* [Nemhauser and Wolsey, 1988; Topkis, 1978] to identify a general class of soft constraints for which there exists a polynomial time solution algorithm. Submodular functions are usually defined as real-valued functions on Boolean tuples (≡sets) [Nemhauser and Wolsey, 1988], but we consider the more general case of functions on tuples over an arbitrary finite domain. We also allow our functions to take infinite values. By establishing a new decomposition result for this general class of binary submodular functions (Theorem 4.4), we obtain a cubic time algorithm to find the optimal assignment for any set of soft constraints defined by such functions.

We give a number of examples to illustrate the many different forms of soft constraint that can be defined using binary submodular functions, and we also show that this class is *maximal*, in the sense that no other form of binary constraint can be added without sacrificing tractability.

2 Definitions

To identify a tractable class of soft constraints we will need to restrict the set of functions that are used to specify constraints. Such a restricted set of possible functions will be called a soft constraint *language*.

Definition 2.1 Let D and E be fixed sets. A *soft constraint language over D with evaluations in E* is defined to be a set of functions, Γ, such that each $\phi \in \Gamma$ is a function from D^k to E, for some $k \in \mathbb{N}$, where k is called the arity of ϕ.

For any given choice of soft constraint language, Γ, we define an associated soft constraint satisfaction problem, which we will call $\mathrm{sCSP}(\Gamma)$, as follows.

Definition 2.2 Let Γ be a soft constraint language over D with evaluations in E. An instance $\mathcal{P}$ of $\mathrm{sCSP}(\Gamma)$ is a triple $\langle V, D, C\rangle$, where:

- V is a finite set of *variables*, which must be assigned values from the set D.
- C is a set of *soft constraints*. Each $c \in C$ is a pair $\langle \sigma, \phi\rangle$ where: σ is a list of variables, called the *scope* of c; and ϕ is an element of Γ of arity $|\sigma|$, called the *evaluation function* of c.

Note that, for any constraint $c = \langle \sigma, \phi\rangle$, the arity of the constraint is given by $|\sigma|$, the length of the constraint scope. The evaluation function ϕ will be used to specify some measure of desirability or undesirability associated with each possible tuple of values over σ.

To complete the definition of a soft constraint satisfaction problem we need to define how the evaluations obtained from each evaluation function are combined and compared, in order to define what constitutes an optimal overall solution. Several alternative mathematical approaches to this issue have been suggested in the literature:

- In the semiring based approach [Bistarelli *et al.*, 1999], the set of possible evaluations, E, is assumed to be an algebraic structure equipped with two binary operations, satisfying the axioms of a semiring.
- In the valued CSP approach [Bistarelli *et al.*, 1999], the set of possible evaluations E is assumed to be a totally ordered algebraic structure with a top and bottom element and a single monotonic binary operation known as *aggregation*.

For our purposes in this paper we require the same properties as the valued CSP approach, with the additional requirement that the aggregation operator has a partial inverse (so that any evaluation can be "subtracted" from any larger evaluation). For concreteness, we shall simply assume throughout this paper that the set of evaluations E is either the set of non-negative integers together with infinity, or else the set of non-negative real numbers together with infinity. Hence, for any two evaluations $\rho_1, \rho_2 \in E$, we have $\rho_1 + \rho_2 \in E$, and when $\rho_1 \geq \rho_2$ we also have $\rho_1 - \rho_2 \in E$. (Note that we set $\infty - \infty = \infty$).

The elements of the set E will be used to represent different measure of undesirability, or *penalties*, associated with different combinations of values. This allows us to complete the definition of a soft constraint satisfaction problem with the following simple definition of a solution to an instance.

Definition 2.3 For any soft constraint satisfaction problem instance $\mathcal{P} = \langle V, D, C\rangle$, an *assignment* for $\mathcal{P}$ is a mapping t from V to D. The *evaluation* of an assignment t, denoted $\Phi_{\mathcal{P}}(t)$, is given by the sum of the evaluations for the restrictions of t onto each constraint scope, that is,

$$\Phi_{\mathcal{P}}(t) = \sum_{\langle\langle v_1, v_2, \ldots, v_k\rangle, \phi\rangle \in C} \phi(t(v_1), t(v_2), \ldots, t(v_k)).$$

A *solution* to $\mathcal{P}$ is an assignment with the smallest possible evaluation, and the question is to find a solution.

Example 2.4 For any standard constraint satisfaction problem instance $\mathcal{P}$ with crisp constraints, we can define a corresponding soft constraint satisfaction problem instance $\mathcal{P}'$ in which the range of the evaluation functions of all the constraints is the set $\{0, \infty\}$. For each crisp constraint c of $\mathcal{P}$, we define a corresponding soft constraint c' of $\mathcal{P}'$ with the same scope; the evaluation function of c' maps each tuple allowed by c to 0, and each tuple disallowed by c to ∞.

In this case the evaluation of an assignment t for $\mathcal{P}'$ equals the minimal possible evaluation, 0, if and only if t satisfies all of the crisp constraints in $\mathcal{P}$. □

Example 2.5 For any standard constraint satisfaction problem instance $\mathcal{P}$ with crisp constraints, we can define a corresponding soft constraint satisfaction problem instance $\mathcal{P}^{\#}$ in which the range of the evaluation functions of all the constraints is the set $\{0, 1\}$. For each crisp constraint c of $\mathcal{P}$, we define a corresponding soft constraint $c^{\#}$ of $\mathcal{P}^{\#}$ with the same scope; the evaluation function of $c^{\#}$ maps each tuple allowed by c to 0, and each tuple disallowed by c to 1.

In this case the evaluation of an assignment t for $\mathcal{P}^{\#}$ equals the number of crisp constraints in $\mathcal{P}$ which are violated by t. Hence a solution to $\mathcal{P}^{\#}$ corresponds to an assignment which violates the minimal number of constraints of $\mathcal{P}$. □

Note that the problem of finding a solution to a soft constraint satisfaction problem is an NP optimization problem, that is, it lies in the complexity class NPO (see [Creignou *et al.*, 2001] for a formal definition of this class). If there exists a polynomial-time algorithm which finds a solution to all instances of $\mathrm{sCSP}(\Gamma)$, then we shall say that $\mathrm{sCSP}(\Gamma)$ is *tractable*. On the other hand, if there is a polynomial-time reduction from some NP-complete problem to $\mathrm{sCSP}(\Gamma)$, then we shall say that $\mathrm{sCSP}(\Gamma)$ is *NP-hard*.

Example 2.6 Let Γ be a soft constraint language over D, where $|D| = 2$. In this case $\mathrm{sCSP}(\Gamma)$ is a class of Boolean soft constraint satisfaction problems.

If we restrict Γ even further, to just those functions with range $\{0, \infty\}$, as in Example 2.4, then $\mathrm{sCSP}(\Gamma)$ corresponds precisely to a standard Boolean crisp constraint satisfaction problem. Such problems are sometimes known as GENERALIZED SATISFIABILITY problems [Schaefer, 1978]. The complexity of $\mathrm{sCSP}(\Gamma)$ for such restricted sets Γ has been completely characterised, and the six tractable cases have been identified [Schaefer, 1978; Creignou *et al.*, 2001].

Alternatively, if we restrict Γ to just those functions with range $\{0, 1\}$, as in Example 2.5, then $\mathrm{sCSP}(\Gamma)$ corresponds precisely to a standard Boolean maximum satisfiability problem, in which the aim is to satisfy the maximum number of crisp constraints. Such problems are sometimes known as MAX-SAT problems [Creignou *et al.*, 2001]. The complexity of $\mathrm{sCSP}(\Gamma)$ for such restricted sets Γ has been completely characterised, and the three tractable cases have been identified (see Theorem 7.6 of [Creignou *et al.*, 2001]).

We note, in particular, that when Γ contains just the single binary function ϕ_{XOR} defined by

$$\phi_{XOR}(x, y) = \begin{cases} 0 & \text{if } x \neq y \\ 1 & \text{otherwise} \end{cases}$$

then $\mathrm{sCSP}(\Gamma)$ corresponds to the MAX-SAT problem for the exclusive-or predicate, which is known to be NP-hard (see Lemma 7.4 of [Creignou *et al.*, 2001]). □

Example 2.7 Let Γ be a soft constraint language over D, where $|D| \geq 3$, and assume that Γ contains just the set of all unary functions, together with the single binary function ϕ_{EQ} defined by

$$\phi_{EQ}(x, y) = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{otherwise.} \end{cases}$$

Even in this very simple case it can be shown that $\mathrm{sCSP}(\Gamma)$ is NP-hard, by reduction from the MINIMUM MULTI-TERMINAL CUT problem [Dahlhaus *et al.*, 1994]. □

The examples above indicate that generalizing the constraint satisfaction framework to include soft constraints does indeed increase the computational complexity, in general. For example, the standard 2-SATISFIABILITY problem is tractable, but the soft constraint satisfaction problem involving only the single binary Boolean function, ϕ_{XOR}, defined at the end of Example 2.6, is NP-hard. Similarly, the standard constraint satisfaction problem involving only crisp unary constraints and equality constraints is clearly trivial, but the soft constraint satisfaction problem involving only soft unary constraints and a soft version of the equality constraint, specified by the function ϕ_{EQ} defined at the end of Example 2.7, is NP-hard.

However, in the next two sections we will show that it is possible to identify a large class of functions for which the corresponding soft constraint satisfaction problem is tractable.

3 Generalized interval functions

We begin with a rather restricted class of binary functions, with a very special structure.

Definition 3.1 Let D be a totally ordered set. A binary function, $\phi : D^2 \to E$ will be called a *generalized interval function* on D if it has the following form:

$$\phi(x, y) = \begin{cases} 0 & \text{if } x < a \vee y > b; \\ \rho & \text{otherwise} \end{cases}$$

for some $a, b \in D$ and some $\rho \in E$. Such a function will be denoted $\eta^{\rho}_{[a,b]}$.

We can explain the choice of name for these functions by considering the unary function $\eta^{\rho}_{[a,b]}(x, x)$. This function returns the value ρ if and only if its argument lies in the interval $[a, b]$; outside of this interval it returns the value 0.

We shall write Γ_{GI} to denote the set of all generalized interval functions on D, where $D = \{1, 2, \ldots, M\}$ with the usual ordering.

The main result of this section is Corollary 3.6, which states that $\mathrm{sCSP}(\Gamma_{GI})$ is tractable. To establish this result we first define a weighted directed graph[1] associated with each instance of $\mathrm{sCSP}(\Gamma_{GI})$.

Definition 3.2 Let $\mathcal{P} = \langle V, \{1, \ldots, M\}, C\rangle$ be an instance of $\mathrm{sCSP}(\Gamma_{GI})$. We define the weighted directed graph $G_{\mathcal{P}}$ as follows.

- The vertices of $G_{\mathcal{P}}$ are as follows:
$$\{S, T\} \cup \{v_d \mid v \in V,\ d \in \{0, 1, \ldots, M\}\}.$$
- The edges of $G_{\mathcal{P}}$ are defined as follows:
  - For each $v \in V$, there is an edge from S to v_M with weight ∞;
  - For each $v \in V$, there is an edge from v_0 to T with weight ∞;
  - For each $v \in V$ and each $d \in \{1, 2, \ldots, M-2\}$, there is an edge from v_d to v_{d+1} with weight ∞;
  - For each constraint $\langle\langle v, w\rangle, \eta^{\rho}_{[a,b]}\rangle \in C$, there is an edge from w_b to v_{a-1} with weight ρ. These edges are called "constraint edges".

Example 3.3 Let $\mathcal{P} = \langle\{x, y, z\}, \{1, 2, 3, 4\}, C\rangle$ be an instance of $\mathrm{sCSP}(\Gamma_{GI})$ with the following four constraints:

$$c_1 = \langle\langle y, x\rangle, \eta^{3}_{[3,4]}\rangle \qquad c_3 = \langle\langle z, y\rangle, \eta^{7}_{[1,3]}\rangle$$
$$c_2 = \langle\langle y, z\rangle, \eta^{2}_{[4,3]}\rangle \qquad c_4 = \langle\langle z, z\rangle, \eta^{\infty}_{[2,4]}\rangle$$

The corresponding directed weighted graph $G_{\mathcal{P}}$, is shown in Figure 1. □

Any set of edges C in the graph $G_{\mathcal{P}}$ whose removal leaves the vertices S and T disconnected will be called a *cut*. If every edge in C is a constraint edge, then C will be called a *proper cut*. The *weight* of a cut C is defined to be the sum of the weights of all the edges in C.

[1] This construction was inspired by a similar construction for certain Boolean constraints described in [Khanna *et al.*, 2000].

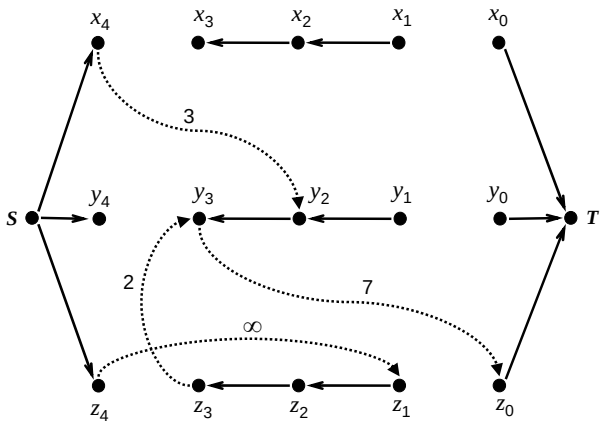


Figure 1: The graph $G_{\mathcal{P}}$ associated with the instance $\mathcal{P}$ defined in Example 3.3.

Example 3.4 Consider the graph $G_{\mathcal{P}}$ shown in Figure 1. The set $\{\langle y_3, z_0\rangle\}$ is a proper cut in $G_{\mathcal{P}}$ with weight 7, which is minimal with respect to inclusion. The set $\{\langle x_4, y_2\rangle, \langle z_3, y_3\rangle\}$ is a proper cut in $G_{\mathcal{P}}$ with weight 5, which is again minimal with respect to inclusion. □

Proposition 3.5 *Let $\mathcal{P}$ be any instance of $\mathrm{sCSP}(\Gamma_{GI})$, and let $G_{\mathcal{P}}$ be the corresponding weighted directed graph.*

1. *For each minimal proper cut in $G_{\mathcal{P}}$ with weight Φ, there is an assignment for $\mathcal{P}$ with evaluation Φ.*
2. *For each assignment t for $\mathcal{P}$ with evaluation Φ, there is a proper cut in $G_{\mathcal{P}}$ with weight Φ.*

Proof:

1. Let C be any minimal proper cut of the graph $G_{\mathcal{P}}$, and let C_S be the component of $G_{\mathcal{P}} \setminus C$ connected to S. Define the assignment t_C as follows:

   $$t_C(v) = \min\{d \mid v_d \in C_S\}$$

   (Note that t_C is well-defined because C_S always contains v_M, and never contains v_0, by construction.)

   By the construction of $G_{\mathcal{P}}$, it follows that:

   $$t_C(v) > d \Leftrightarrow v_d \notin C_S$$

   Now consider any constraint $c = \langle\langle v, w\rangle, \eta^{\rho}_{[a,b]}\rangle$ of $\mathcal{P}$, and its associated edge e in $G_{\mathcal{P}}$. By the definition of generalized interval constraint and the choice of t_C, $\eta^{\rho}_{[a,b]}(t_C(v), t_C(w)) = \rho$ if and only if e joins a vertex in C_S to a vertex not in C_S. Since C is minimal, this happens if and only if $e \in C$. Hence, the total weight of the cut C is equal to the evaluation of t_C.

2. Conversely, let t be an assignment to $\mathcal{P}$, and let K be the set of all constraints of $\mathcal{P}$ which give a non-zero evaluation on t.

   Now consider any path from S to T in $G_{\mathcal{P}}$. If we examine, in order, the constraint edges of this path, and assume that each of the corresponding constraints evaluates to 0, then we obtain a sequence of assertions of the following form:

   $$\begin{array}{rcll} (v_{i_0} > M) & \vee & (v_{i_1} < a_1) & \\ (v_{i_1} > b_2) & \vee & (v_{i_2} < a_2) & \text{for some } b_2 \geq a_1 \\ & \vdots & & \\ (v_{i_{k-1}} > b_k) & \vee & (v_{i_k} < a_k) & \text{for some } b_k \geq a_{k-1} \\ (v_{i_k} > b_{k+1}) & \vee & (v_{i_{k+1}} < 1) & \text{for some } b_{k+1} \geq a_k \end{array}$$

   Since the second disjunct of each assertion contradicts the first disjunct of the next, these assertions cannot all hold simultaneously, so one of the corresponding constraints must in fact give a non-zero evaluation on t. Hence, every path from S to T includes at least one edge corresponding to a constraint from K, and so the edges corresponding to the set K form a cut in $G_{\mathcal{P}}$. Furthermore, by the choice of K, the weight of this cut is equal to the evaluation of t.

□

Hence, by using a standard efficient algorithm for the MINIMUM WEIGHTED CUT problem [Goldberg and Tarjan, 1988], we can find an optimal assignment in cubic time.

Corollary 3.6 *The time complexity of $\mathrm{sCSP}(\Gamma_{GI})$ is $O(n^3|D|^3)$, where n is the number of variables.*

4 Submodular functions

In this section we will consider a rather more general class of functions, as described in [Topkis, 1978].

Definition 4.1 Let D be a totally ordered set. A function, $\phi : D^k \to E$ is a *submodular function* on D if, for all $\langle a_1, \ldots, a_k\rangle, \langle b_1, \ldots, b_k\rangle \in D^k$, we have

$$\phi(\min(a_1, b_1), \ldots, \min(a_k, b_k)) + \phi(\max(a_1, b_1), \ldots, \max(a_k, b_k)) \leq \phi(a_1, \ldots, a_k) + \phi(b_1, \ldots, b_k).$$

It is easy to check that all unary functions and all generalized interval functions are submodular. For binary functions, the definition of submodularity can be simplified, as follows.

Remark 4.2 Let D be a totally ordered set. A binary function, $\phi : D^2 \to E$ is submodular if and only if, for all $u < x, v < y \in D$, we have:

$$\phi(u, v) + \phi(x, y) \leq \phi(u, y) + \phi(x, v)$$

Example 4.3 Let D be the set $\{1, 2, \ldots, M\}$ with the usual ordering, and consider the binary function π_M, defined by $\pi_M(x, y) = M^2 - xy$.

Note that, for any $u < x, v < y \in D$, we have:

$$\begin{array}{rcl} \pi_M(u, v) + \pi_M(x, y) & = & 2M^2 - uv - xy \\ & = & 2M^2 - uy - xv - (x - u)(y - v) \\ & \leq & \pi_M(u, y) + \pi_M(x, v). \end{array}$$

Hence, by Remark 4.2, the function π_M is submodular. □

It follows from Definition 4.1 that the sum of any two submodular functions is submodular. This suggests that in some cases it may be possible to express a submodular function as a sum of simpler submodular functions. For example, for any unary function $\psi : D \to E$ we have

$$\psi(x) \equiv \sum_{d \in D} \eta^{\psi(d)}_{[d,d]}(x, x).$$

The main result of this section is Theorem 4.4, which states that any *binary* submodular function can also be expressed as a sum of generalized interval functions.

Theorem 4.4 *Let D be a totally ordered finite set. A binary function, $\phi : D^2 \to E$ is submodular if and only if it can be expressed as a sum of generalized interval functions on D.*

Proof: By the observations already made, any function ϕ which is equal to a sum of generalized interval functions is clearly submodular.

To establish the converse, we use induction on the *tightness* of ϕ, that is, the number of tuples for which the value of ϕ is non-zero. Details are given in the full version of this paper. □

Example 4.5 Consider the binary function π_M on $D = \{1, 2, \ldots, M\}$, defined in Example 4.3. When $M = 3$, the values of π_3 are given by the following table:

π_3	1	2	3
1	8	7	6
2	7	5	3
3	6	3	0

Note that:

$$\begin{pmatrix} 8 & 7 & 6 \\ 7 & 5 & 3 \\ 6 & 3 & 0 \end{pmatrix} = \begin{pmatrix} 6 & 6 & 6 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 3 & 3 & 3 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 2 & 0 & 0 \\ 2 & 0 & 0 \\ 2 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}.$$

Hence,

$$\begin{aligned} \pi_3(x, y) \quad = \quad & \eta^6_{[1,1]}(x, x) + \eta^3_{[2,2]}(x, x) + \\ & \eta^2_{[1,1]}(y, y) + \eta^1_{[2,2]}(y, y) + \eta^1_{[2,2]}(x, y) + \\ & \eta^1_{[2,1]}(x, y) + +\eta^1_{[3,2]}(x, y) + \eta^1_{[3,1]}(x, y). \end{aligned}$$

In general, for arbitrary values of M, we have $\pi_M(x, y) =$

$$\sum_{d=1}^{M-1} \left(\eta^{M(M-d)}_{[d,d]}(x, x) + \eta^{M-d}_{[d,d]}(y, y) + \sum_{e=1}^{M-1} \eta^1_{[d+1,e]}(x, y) \right)$$

We remark that this decomposition is not unique - other decompositions exist, including the symmetric decomposition $\pi_M(x, y) = \pi'_M(x, y) + \pi'_M(y, x)$, where $\pi'_M(x, y) =$

$$\sum_{d=1}^{M-1} \left(\eta^{\frac{(M^2-d^2)}{2}}_{[d,d]}(x, x) + \eta^{\frac{1}{2}}_{[d+1,d]}(x, y) + \sum_{e=1}^{d-1} \eta^1_{[d+1,e]}(x, y) \right)$$

□

Combining Theorem 4.4 with Corollary 3.6, gives:

Corollary 4.6 *For any finite soft constraint language Γ on a finite totally ordered set D, if Γ contains only unary or binary submodular functions, then the time complexity of $\mathrm{sCSP}(\Gamma)$ is $O(n^3|D|^3)$.*

The next result shows that the tractable class identified in Corollary 4.6 is maximal.

Theorem 4.7 *Let Γ be the set of all binary submodular functions on a totally ordered finite set D, with $|D| \geq 2$.*

For any binary function $\psi \notin \Gamma$, $\mathrm{sCSP}(\Gamma \cup \{\psi\})$ is NP-hard.

Proof: The proof is by reduction from $\mathrm{sCSP}(\{\phi_{XOR}\})$ to $\mathrm{sCSP}(\Gamma \cup \{\psi\})$, where ϕ_{XOR} is the binary function defined in Example 2.6. It was pointed out in Example 2.6 that $\mathrm{sCSP}(\{\phi_{XOR}\})$ corresponds to the MAX-SAT problem for the exclusive-or predicate, which is known to be NP-hard [Creignou *et al.*, 2001]. Hence $\mathrm{sCSP}(\Gamma \cup \{\psi\})$ is also NP-hard.

Details of the reduction are given in the full version of this paper. □

5 Applications

In this section we give a number of examples to illustrate the wide range of soft constraints which can be shown to be tractable using the results obtained in the previous sections.

Definition 5.1 For any k-ary relation R on a set D, we define an *associated function*, $\phi_R : D^k \to E$, as follows:

$$\phi_R(x_1, x_2, \ldots, x_k) = \begin{cases} 0 & \text{if } \langle x_1, x_2, \ldots, x_k \rangle \in R \\ \infty & \text{otherwise.} \end{cases}$$

By Corollary 4.6, any collection of crisp constraints, where each constraint is specified by a relation R for which ϕ_R is unary or binary submodular, can be solved in polynomial time, even when combined with other soft constraints that are also unary or binary submodular.

Example 5.2 The constraint programming language CHIP incorporates a number of constraint solving techniques for arithmetic and other constraints. In particular, it provides a constraint solver for a restricted class of crisp constraints over natural numbers, referred to as *basic constraints* [van Hentenryck *et al.*, 1992]. These basic constraints are of two kinds, which are referred to as "domain constraints" and "arithmetic constraints". The domain constraints described in [van Hentenryck *et al.*, 1992] are unary constraints which restrict the value of a variable to some specified finite subset of the natural numbers. The arithmetic constraints described in [van Hentenryck *et al.*, 1992] have one of the following forms:

$$\begin{array}{ll} aX \neq b & aX \leq bY + c \\ aX = bY + c & aX \geq bY + c \end{array}$$

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

For each of these crisp constraints the associated function given by Definition 5.1 is unary or binary submodular, hence, by Corollary 3.6, any problem involving constraints of this form can be solved in cubic time. Moreover, any other soft constraints with unary or binary submodular evaluation functions can be added to such problems without sacrificing tractability (including the examples below). □

Now assume, for simplicity, that $D = \{1, 2, \ldots, M\}$.

Example 5.3 Consider the binary linear function λ defined by $\lambda(x, y) = ax + by + c$, where $a, b \in \mathbb{R}^+$.

This function is submodular and hence, by Corollary 3.6, any collection of such binary linear soft constraints over the discrete set D can be solved in polynomial time. □

Example 5.4 The Euclidean length function $\sqrt{x^2 + y^2}$ is submodular, and can be used to express the constraint that a 2-dimensional point $\langle x, y\rangle$ is "as close to the origin as possible". □

Example 5.5 The following functions are all submodular:

- $\delta_r(x, y) = |x - y|^r$, where $r \in \mathbb{R}, r \geq 1$.
  The function δ_r can be used to express the constraint that: "The values assigned to the variables x and y should be as similar as possible".
- $\delta_r^+(x, y) = (\max(x - y, 0))^r$, where $r \in \mathbb{R}, r \geq 1$.
  The function δ_r^+ can be used to express the constraint that: "The value of x is either less than or as near as possible to y".
- $\delta_r^{\geq}(x, y) = \begin{cases} |x - y|^r & \text{if } x \geq y \\ \infty & \text{otherwise} \end{cases}$
  where $r \in \mathbb{R}, r \geq 1$.
  The function $\delta_r^{\geq}$ can be used to express the temporal constraint that: "x occurs as soon as possible after y".

□

Example 5.6 Reconsider the optimization problem defined in Example 1.1. Since ψ_i is unary, and δ_r is binary submodular (Example 5.5), this problem can be solved in cubic time, using the methods developed in this paper.

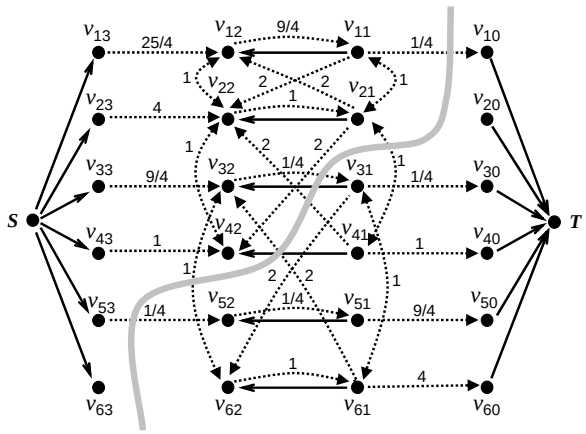
Let $\mathcal{P}$ be the instance with $n = 3$ and $r = 2$. The values of δ_2 are given by the following table:

δ_2	1	2	3
1	0	1	4
2	1	0	1
3	4	1	0

Hence,

$$\begin{aligned}\delta_2(x, y) \quad = \quad & \eta^1_{[2,1]}(x, y) + \eta^1_{[2,1]}(y, x) + \eta^1_{[3,2]}(x, y) + \\ & \eta^1_{[3,2]}(y, x) + \eta^2_{[3,1]}(x, y) + \eta^2_{[3,1]}(y, x)\end{aligned}$$

Using this decomposition for δ_2, we can construct the graph $G_{\mathcal{P}}$ corresponding to the instance $\mathcal{P}$, as follows.



The minimum weight of any cut in this graph is $\frac{11}{4}$, and hence the optimal evaluation of any assignment for $\mathcal{P}$ is $\frac{11}{4}$.

One of the several possible cuts with this weight is indicated by the gray line across the graph, which corresponds to the solution $v_1 = 1$, $v_2 = 1$, $v_3 = 2$, $v_4 = 2$, $v_5 = 3$, $v_6 = 3$. □

References

[Bistarelli *et al.*, 1999] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, and G. Verfaillie. Semiring-based CSPs and valued CSPs: Frameworks, properties, and comparison. *Constraints*, 4:199–240, 1999.

[Creignou *et al.*, 2001] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classification of Boolean Constraint Satisfaction Problems*, volume 7 of *SIAM Monographs on Discrete Mathematics and Applications*. 2001.

[Dahlhaus *et al.*, 1994] E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23(4):864–894, 1994.

[Feder and Vardi, 1998] T. Feder and M.Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM Journal of Computing*, 28:57–104, 1998.

[Goldberg and Tarjan, 1988] A. Goldberg and R.E. Tarjan. A new approach to the maximum flow problem. *Journal of the ACM*, 35:921–940, 1988.

[Jeavons *et al.*, 1997] P.G. Jeavons, D.A. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the ACM*, 44:527–548, 1997.

[Nemhauser and Wolsey, 1988] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, 1988.

[Schaefer, 1978] T.J. Schaefer. The complexity of satisfiability problems. In *Proceedings 10th ACM Symposium on Theory of Computing, STOC'78*, pages 216–226, 1978.

[Topkis, 1978] D.M. Topkis. Minimizing a submodular function on a lattice. *Operations Research*, pages 305–321, 1978.

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