Advanced Structured Prediction

Editors:

Sebastian Nowozin Sebastian.Nowozin@microsoft.com
Microsoft Research
Cambridge, CB1 2FB, United Kingdom

Peter V. Gehler pgehler@tuebingen.mpg.de
Max Planck Insitute for Intelligent Systems
72076 Tübingen, Germany

Jeremy Jancsary jermyj@microsoft.com
Microsoft Research
Cambridge, CB1 2FB, United Kingdom

Christoph Lampert chl@ist.ac.at
IST Austria
A-3400 Klosterneuburg, Austria

This is a draft version of the author chapter.

The MIT Press
Cambridge, Massachusetts
London, England

1 The Power of LP Relaxation for MAP Inference

Stanislav Živný standa@cs.ox.ac.uk
Department of Computer Science
University of Oxford
Oxford, UK

Tomáš Werner werner@cmp.felk.cvut.cz
Daniel Průša prusapa1@cmp.felk.cvut.cz
Center for Machine Perception
Faculty of Electrical Engineering
Czech Technical University
Prague, Czech Republic

Minimization of a partially separable function of many discrete variables is ubiquitous in machine learning and computer vision, in tasks like maximum a posteriori (MAP) inference in graphical models, or structured prediction. Among successful approaches to this problem is linear programming (LP) relaxation. We discuss this LP relaxation from two aspects. First, we review recent results which characterize languages (classes of functions permitted to form the objective function) for which the problem is solved by the relaxation exactly. Second, we show that solving the LP relaxation is not easier than solving any linear program, which makes a discovery of an efficient algorithm for the LP relaxation unlikely.

The topic of this chapter is the problem of minimizing a partially separable function of many discrete variables. That is, given a set of variables, we minimize the sum of functions each depending only on a subset of the variables. This NP-hard combinatorial optimization problem frequently arises in machine learning and computer vision, in tasks like MAP inference in graphical models (Lauritzen, 1996; Koller and Friedman, 2009; Wainwright and

Jordan, 2008) and structured prediction (Nowozin and Lampert, 2011). It is also known as discrete energy minimization or valued constraint satisfaction. The problem is formally defined in Section 1.1.

The problem has a natural linear programming (LP) relaxation, proposed independently by a number of authors (Shlezinger, 1976; Koster et al., 1998; Chekuri et al., 2005), that is defined in Section 1.2. Algorithms based on LP relaxation are among most successful ones for tackling the problem in practice (Szeliski et al., 2008).

In this chapter, we discuss the power of the relaxation from two aspects. In the first part of the chapter, Section 1.3, we focus on the question of what languages are exactly solved by the LP relaxation. This means, we consider subclasses of the problem in which the structure (hypergraph) is arbitrary but the functions belong to a given subset (language) of all possible functions. For instance, it is well-known that if all the functions are submodular then the problem is tractable, no matter what its structure is. In this case, the LP relaxation is tight. We review the recent results by Thapper and Živný (2013, 2012); Kolmogorov et al. (2013); Kolmogorov and Živný (2013), which characterize all languages solved by the LP relaxation. This is accompanied by a number of concrete examples of such languages.

Given the (widely accepted) usefulness of the LP relaxation, many authors have proposed algorithms to solve this linear program efficiently. In the second part of the chapter, Section 1.4, we review the result by Průša and Werner (2013) which states that solving the LP relaxation is not easier than solving any linear program. This result is negative, showing that finding a very efficient algorithm for the LP relaxation is as hard as improving the complexity of the best known algorithm for general LP.

In the sequel, we denote sets by $\{\cdots\}$ and ordered tuples by $\langle\cdots\rangle$. The set of all subsets of a set A is denoted by 2^A and the set of all k-element subsets of A by $\binom{A}{k}$. For a tuple $\mathbf{x}$, we denote by x_i its ith component.

1.1 Valued Constraint Satisfaction Problem

Let V be a finite set of *variables*. Each variable $i \in V$ can take states $x_i \in D$, where the *domain* D is the same for each variable. Let $\overline{\mathbb{Q}} = \mathbb{Q} \cup \{\infty\}$ denote the set of extended rational numbers. A function $\Phi\colon D^V \to \overline{\mathbb{Q}}$ is *partially separable* if it can be written as

$$\Phi(\mathbf{x}) = \sum_{S \in H} \phi_S(\mathbf{x}_S) \tag{1.1}$$

where $H \subseteq 2^V$ is a collection of subsets of V (so that $\langle V, H\rangle$ is a hypergraph) and each variable subset $S \in H$ is assigned a function $\phi_S\colon D^{|S|} \to \overline{\mathbb{Q}}$. Here, $\mathbf{x}_S = \langle\, x_i \mid i \in S \,\rangle \in D^S$ denotes the restriction of the assignment $\mathbf{x} = \langle\, x_i \mid i \in V \,\rangle \in D^V$ to variables S, where the order of elements of the tuple $\mathbf{x}_S$ is given by some fixed total order on V.

Example 1.1. For $V = \{1, 2, 3, 4\}$ and $H = \{\{2, 3, 4\}, \{1, 2\}, \{2, 3\}, \{1\}\}$, we have (where we abbreviated $\phi_{\{2,3,4\}}$ by ϕ_{234}, etc.)

$$\Phi(x_1, x_2, x_3, x_4) = \phi_{234}(x_2, x_3, x_4) + \phi_{12}(x_1, x_2) + \phi_{23}(x_2, x_3) + \phi_1(x_1).$$

Our aim is to minimize function (1.1) over all assignments $\mathbf{x} \in D^V$. In this chapter, we assume that the domain D has a finite size (that is, the variables are discrete). This problem is known under many names, such as MAP inference in graphical models (or Markov random fields), discrete energy minimization, or min-sum problem. In constraint programming (Rossi et al., 2006), it has been studied under the name *valued* (or *weighted*) *constraint satisfaction problem* (VCSP) (Schiex et al., 1995; Cohen et al., 2006b). We will follow this terminology. Here, each function ϕ_S is called a *constraint*[1] with *scope* S and *arity* $|S|$. The arity of the problem is $\max_{S\in H} |S|$. The values of the functions ϕ_S are called *costs*.

Problems involving only functions with costs from $\{0, \infty\}$ (so-called *hard* or *crisp* constraints) are known as *constraint satisfaction problems* (CSPs) (Cohen and Jeavons, 2006); these are decision problems asking for the existence of a zero-cost labelling. This type of problems has the longest history, started by the pioneering work of Montanari (1974). Problems involving functions with arbitrary costs from $\overline{\mathbb{Q}}$ are known as *valued* CSPs (VCSPs). Valued CSPs are sometimes called *general-valued*, to emphasize the fact that the costs can be both finite (from $\mathbb{Q}$) and infinite. The following two subclasses of valued CPSs have been studied intensively in the literature. Problems involving only functions with costs from $\{0, 1\}$ are known as *maximum constraint satisfaction problems* (Max-CSPs). Problems involving only functions with costs from $\mathbb{Q}$ (so-called *soft* constraints) are known as *finite-valued* CSPs.[2]

1. For historical reasons, costs are often required to be non-negative in the constraint community.
2. In the approximation community, Max-CSPs are referred to as CSPs and finite-valued CSPs are referred to as generalized CSPs.

1.2 Basic LP Relaxation

The LP relaxation of VCSP reads

$$\sum_{S\in H}\sum_{\mathbf{x}\in D^S}\phi_S(\mathbf{x})\,\mu_S(\mathbf{x}) \to \min \tag{1.2a}$$

$$\sum_{\mathbf{y}\in D^S \,|\, y_i=x} \mu_S(\mathbf{y}) = \mu_i(x), \qquad i\in S\in H,\ x\in D \tag{1.2b}$$

$$\sum_{x\in D}\mu_i(x) = 1, \qquad i\in V \tag{1.2c}$$

$$\mu_S(\mathbf{x}) \geq 0, \qquad S\in H,\ \mathbf{x}\in D^S \tag{1.2d}$$

$$\mu_i(x) \geq 0, \qquad i\in V,\ x\in D \tag{1.2e}$$

We minimize over functions $\mu_S\colon D^{|S|}\to\mathbb{R}$, $S\in H$, and $\mu_i\colon D\to\mathbb{R}$, $i\in V$. These functions can be seen as probability distributions on D^S and D, respectively. The marginalization constraint (1.2b) imposes that μ_i is the marginal of μ_S, for every $i\in S\in H$. In (1.2a) we define that $\infty\cdot 0 = 0$. Thus, if the LP is feasible then $\phi_S(\mathbf{x}_S)=\infty$ implies $\mu_S(\mathbf{x}_S)=0$.

An LP relaxation of VCSP, similar or closely related to (1.2), has been proposed independently by many authors (Shlezinger, 1976; Koster et al., 1998; Chekuri et al., 2005; Wainwright et al., 2005; Kingsford et al., 2005; Cooper, 2008; Cooper et al., 2010a; Kun et al., 2012). Equivalently, it can be understood as dual decomposition (or Lagrangian relaxation) of VCSP (Johnson et al., 2007; Komodakis et al., 2011; Sontag et al., 2011).

We refer to (1.2) as the *basic* LP relaxation (BLP) of VCSP. It is the first level in the hierarchy of Sherali and Adams (1990), which provides successively tighter LP relaxations of an integer LP. Several authors proposed finer-grained hierarchies of LP relaxations of VCSP (Wainwright and Jordan, 2008; Johnson et al., 2007; Werner, 2010; Franc et al., 2012).

1.3 Languages Solved by the Basic LP

In this section we will be interested in the question of which VCSPs are exactly (as opposed to, for instance, approximately) solved by BLP. Prior to this, we focus on a more general question of which classes of VCSPs can be solved in polynomial time. Such classes are called *tractable*.

Tractability of CSPs. Since CSPs are NP-hard in general, it is natural to study restrictions on the general framework that guarantee tractability.

The most studied are so-called *language* restrictions that impose restrictions on the types of constraints allowed in the instance. The computational complexity of language-restricted CSPs is known for problems over 2-element domains (Schaefer, 1978), 3-element domains (Bulatov, 2006), conservative CSPs (class of CSPs containing all unary functions) (Bulatov, 2011), and a few others (Barto et al., 2009). Most results rely heavily on algebraic methods (Jeavons et al., 1997; Bulatov et al., 2005).

Structural restrictions on CSPs do not impose any condition on the type of constraints (functions) but restrict how the constraints interact, that is, the hypergraph (Gottlob et al., 2000). Complete complexity classifications are known for structurally-restricted bounded-arity CSPs (Dalmau et al., 2002; Grohe, 2007) and unbounded-arity CSPs (Marx, 2010). Some results are also known for so-called *hybrid* CSPs, which combine structural and language restrictions; see, for instance, the work of Cooper et al. (2010b).

Tractability of Valued CSPs. The study of structural restrictions for valued CSPs has not led to essentially new results as hardness results for CSPs immediately apply to (more general) valued CSPs, and all known tractable (bounded-arity) structural classes for CSPs extend easily to valued CSPs, see (Dechter, 2003). There are not many results on hybrid restrictions for VCSPs (Cooper and Živný, 2011, 2012), including the permuted submodular VCSPs (Schlesinger, 2007) and planar max-cut (Hadlock, 1975).

The main topic of Section 1.3 is the tractability of language-restricted VCSPs. By a *language*, we mean a set Γ of functions $\phi\colon D^r \to \overline{\mathbb{Q}}$, possibly of different arities r. For a language Γ, we denote by VCSP(Γ) the set of all VCSP instances with constraints from Γ (that is, $\phi_S \in \Gamma$ for every $S \in H$) and an arbitrary hypergraph $\langle V, H\rangle$. We call a language Γ *tractable* if for every finite subset $\Gamma' \subseteq \Gamma$, any instance from VCSP(Γ') can be solved in polynomial time. A language Γ is called *intractable* if for some finite subset $\Gamma' \subseteq \Gamma$, the class VCSP(Γ') is NP-hard.

1.3.1 Examples of Languages

In this section, we give examples of languages and review tractability results for them that were obtained in the past.

As a motivation, we start with the well-known concept of submodularity (Schrijver, 2003; Fujishige, 2005). Let the set D be totally ordered. An r-ary function $\phi\colon D^r \to \overline{\mathbb{Q}}$ is *submodular* if and only if, for every $\mathbf{x}, \mathbf{y} \in D^r$,

$$\phi(\mathbf{x}) + \phi(\mathbf{y}) \;\geq\; \phi(\min(\mathbf{x}, \mathbf{y})) + \phi(\max(\mathbf{x}, \mathbf{y})). \tag{1.3}$$

Here, min and max returns the component-wise minimum and maximum,

respectively, of its two arguments, with respect to the total order on D.

The definition of submodularity can be straightforwardly generalized as follows. A binary *operation* is a mapping $f\colon D^2 \to D$. For r-tuples $\mathbf{x}, \mathbf{y} \in D^r$, we denote by $f(\mathbf{x}, \mathbf{y})$ the result of applying f on $\mathbf{x}$ and $\mathbf{y}$ component-wise, that is, $f(\mathbf{x}, \mathbf{y}) = (f(x_1, y_1), \ldots, f(x_r, y_r)) \in D^r$.

Definition 1.1 (Binary multimorphism (Cohen et al., 2006b)). *Let $f, g\colon D^2 \to D$ be binary operations. We say that an r-ary function $\phi\colon D^r \to \overline{\mathbb{Q}}$* admits *$\langle f, g\rangle$ as a* multimorphism *if for all $\mathbf{x}, \mathbf{y} \in D^r$ it holds that*

$$\phi(\mathbf{x}) + \phi(\mathbf{y}) \;\geq\; \phi(f(\mathbf{x}, \mathbf{y})) + \phi(g(\mathbf{x}, \mathbf{y})). \tag{1.4}$$

We say that a language Γ admits $\langle f, g\rangle$ as a multimorphism if every function $\phi \in \Gamma$ admits $\langle f, g\rangle$ as a multimorphism.

Example 1.2 (Submodularity). Let Γ be the set of functions $\phi\colon D^r \to \overline{\mathbb{Q}}$ (with D totally ordered and $r \geq 1$) that admit $\langle \min, \max\rangle$ as a multimorphism. Using a polynomial-time algorithm for minimizing submodular set functions (Schrijver, 2000; Iwata et al., 2001), Cohen et al. (2006b) have shown that the language Γ is tractable. For $\mathbb{Q}$-valued functions, this also immediately follows from the result by Schlesinger and Flach (2006).

Example 1.3 (Bisubmodularity). Let $D = \{0, 1, 2\}$. We define two binary operations $\min_0$ and $\max_0$ by

$$\min_0(x, y) \;=\; \begin{cases} 0 & \text{if } 0 \neq x \neq y \neq 0 \\ \min(x, y) & \text{otherwise} \end{cases},$$

$$\max_0(x, y) \;=\; \begin{cases} 0 & \text{if } 0 \neq x \neq y \neq 0 \\ \max(x, y) & \text{otherwise} \end{cases}.$$

Let Γ be the set of functions admitting $\langle \min_0, \max_0\rangle$ as a multimorphism. These functions are known as *bisubmodular* functions. The language Γ has been shown tractable for $\mathbb{Q}$-valued functions (even if given by oracles) by Fujishige and Iwata (2005).

Example 1.4 (k-submodularity). Let Γ be the set of functions, called *k-submodular*, with $D = \{0, 1, \ldots, d\}$ for some $d \geq 2$ and admitting $\langle \min_0, \max_0\rangle$, defined in Example 1.3, as a multimorphism. The tractability of this language for $d \geq 3$ was left open in the work of Huber and Kolmogorov (2012).

Example 1.5 ((Symmetric) tournament pair). A *tournament* operation is a binary operation $f\colon D^2 \to D$ such that (i) f is commutative (that is, $f(x, y) = f(y, x)$ for all $x, y \in D$) and (ii) f is conservative (that is,

$f(x, y) \in \{x, y\}$ for all $x, y \in D$). The *dual* of a tournament operation is the unique tournament operation g satisfying $x \neq y \Rightarrow f(x, y) \neq g(x, y)$. A *tournament pair* is a pair $\langle f, g \rangle$ where f and g are tournament operations. A tournament pair $\langle f, g \rangle$ is *symmetric* if g is the dual of f.

Let Γ be a $\overline{\mathbb{Q}}$-valued language that admits a symmetric tournament pair (STP) multimorphism. Cohen et al. (2008) have shown, by a reduction to the minimization problem for submodular functions (see Example 1.2), that any such Γ is tractable.

Let Γ be an arbitrary $\overline{\mathbb{Q}}$-valued language that admits any tournament pair multimorphism. Cohen et al. (2008) have shown, by a reduction to the symmetric tournament pair case, that any such Γ is also tractable.

Example 1.6 (Strong tree-submodularity)**.** Let the elements of D be arranged into a tree, T. Given $a, b \in T$, let P_{ab} denote the unique path in T between a and b of length (number of edges) $d(a, b)$, and let $P_{ab}[i]$ denote the ith vertex on P_{ab}, where $0 \leq i \leq d(a, b)$ and $P_{ab}[0] = a$. Define the binary operations $f(a, b) = P_{ab}[\lfloor d(a, b)/2 \rfloor]$ and $g(a, b) = P_{ab}[\lceil d(a, b)/2 \rceil]$.

A function (or language) admitting $\langle f, g \rangle$ as a multimorphism has been called *strongly tree-submodular*. The tractability of $\mathbb{Q}$-valued strongly tree-submodular languages on binary trees has been shown by Kolmogorov (2011) but the tractability of strongly tree-submodular languages on non-binary trees was left open.

Example 1.7 (Weak tree-submodularity)**.** Assume that the elements of D form a rooted tree T. For $a, b \in T$, let $f(a, b)$ be defined as the highest common ancestor of a and b in T, that is, the unique node on the path P_{ab} that is an ancestor of both a and b. Let $g(a, b)$ be the unique node on the path P_{ab} such that the distance between a and $g(a, b)$ is the same as the distance between b and $f(a, b)$.

A function (or language) admitting $\langle f, g \rangle$ as a multimorphism has been called *weakly tree-submodular*, since it can be shown that tree-submodularity implies weak tree-submodularity. The tractability of $\mathbb{Q}$-valued weakly tree-submodular languages on chains[3] and forks[4] has been shown by Kolmogorov (2011) and left open for all other trees.

Note that k-submodular functions are a special case of weakly tree-submodular functions, obtained for $D = \{0, 1, \ldots, d\}$ and T consisting of the root node 0 and d children.

---

3. A chain is a binary tree in which all nodes except leaves have exactly one child.
4. A fork is a binary tree in which all nodes except leaves and one special node have exactly one child. The special node has exactly two children.

Example 1.8 (1-defect)**.** Let b and c be two distinct elements of D and let $\preceq$ be a partial order on D which relates all pairs of elements except for b and c. We call $\langle f, g\rangle$, where $f, g\colon D^2 \to D$ are binary operations, a *1-defect* if f and g are both commutative and satisfy the following conditions:

- If $\{x, y\} \neq \{b, c\}$ then $f(x, y) = \min(x, y)$ and $g(x, y) = \max(x, y)$.
- If $\{x, y\} = \{b, c\}$ then $\{f(x, y), g(x, y)\} \cap \{x, y\} = \emptyset$ and $f(x, y) \preceq g(x, y)$.

The tractability of $\mathbb{Q}$-valued languages that admit a 1-defect multimorphism has been shown by Jonsson et al. (2011). This result generalizes the tractability result for weakly tree-submodular languages on chains and forks, but is incomparable with the tractability result for strongly tree-submodular languages on binary trees.

Example 1.9 (Submodularity on lattices)**.** Let the set D, endowed with a partial order, form a lattice, with the meet operation $\wedge$ and the join operation $\vee$. Let Γ be the language admitting $\langle \wedge, \vee\rangle$ as a multimorphism.

If the lattice is a chain (that is, the order on D is total), we obtain the language of submodular functions (Example 1.2). For distributive lattices, the tractability of Γ has been established by Schrijver (2000). Until recently, the tractability of Γ for non-distributive lattices was widely open and only partial results were known (Krokhin and Larose, 2008; Kuivinen, 2011), but the work of Thapper and Živný (2012), which we will discuss in Sections 1.3.2 and 1.3.3, settled this question.

Example 1.10 (Conservative languages)**.** A language that contains all unary functions (and possibly some other functions) is called *conservative*. Kolmogorov and Živný (2013) have shown that a $\mathbb{Q}$-valued conservative language can be only tractable if it admits an STP multimorphism (see Example 1.5). (Kolmogorov and Živný, 2013, Theorem 3.5) have given a precise condition under which a $\overline{\mathbb{Q}}$-valued conservative language is tractable. This condition is somewhat technical so we will not state it here but we mention that it involves a pair of complementary multimorphisms, one of which is an STP multimorphism and the other one is a ternary[5] multimorphism involving two majority and one minority operations. The algorithm involves a preprocessing step, after which the resulting instance admits an STP multimorphism.

Example 1.11 (Potts model)**.** Let Γ contain all unary functions and a

5. In order to state the property precisely one needs to generalize Definition 1.1 to a triple of ternary operations, see (Kolmogorov and Živný, 2013) for more details.

single binary function $\phi_{\mathsf{Potts}}\colon D^2 \to \mathbb{Q}$ defined by

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

This conservative language is known in statistical mechanics as the Potts model with external field (Mezard and Montanari, 2009) and is frequently used for image segmentation (Rother et al., 2004). For $|D| = 2$, ϕ_{Potts} is submodular and hence Γ is tractable. For $|D| > 2$, Γ is intractable.

Example 1.12 (Max-Cut)**.** Let Γ contain a single function $\phi_{\mathsf{mc}}\colon D^2 \to \mathbb{Q}$ defined by

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

This language models the well-known Max-Cut problem (Garey and Johnson, 1979) and thus Γ is intractable for any $|D| \geq 2$.

1.3.2 Power of BLP for Finite-Valued Languages

Given the long list of examples from Section 1.3.1, one might expect that perhaps multimorphism could define *all* tractable languages. It turns out that this is not the case and in order to capture more tractable languages one needs to consider a more general notion. We start with an example.

Example 1.13 (Skew bisubmodularity)**.** We extend the notion of bisubmodularity (Example 1.3) to *skew bisubmodularity* introduced by Huber et al. (2013). Let $D = \{0,1,2\}$. Recall the definition of operations $\min_0$ and $\max_0$ from Example 1.3. We define

$$\max_1(x,y) \;=\; \begin{cases} 1 & \text{if } 0 \neq x \neq y \neq 0 \\ \max(x,y) & \text{otherwise} \end{cases}.$$

A function $\phi\colon D^r \to \overline{\mathbb{Q}}$ is called α-bisubmodular, for some real $0 < \alpha \leq 1$, if for every $\mathbf{x}, \mathbf{y} \in D^r$,

$$\phi(\mathbf{x}) + \phi(\mathbf{y}) \;\geq\; \phi(\min_0(\mathbf{x},\mathbf{y})) + \alpha\phi(\max_0(\mathbf{x},\mathbf{y})) + (1-\alpha)f(\max_1(\mathbf{x},\mathbf{y})).$$

Note that 1-bisubmodular functions are (ordinary) bisubmodular functions.

The previous example suggests that it is not enough to consider only two operations with equal weight. In fact it is necessary to consider *probability distributions* over *all* binary operations. We denote by $\Omega_D^{(2)}$ the set of all binary operations $f\colon D^2 \to D$.

Definition 1.2 (Binary fractional polymorphism (Cohen et al., 2006a)). *Let ω be a probability distribution on $\Omega_D^{(2)}$. We say that ω is a* binary fractional polymorphism *of an r-ary function $\phi\colon D^r \to \mathbb{Q}$ if, for every $\mathbf{x}, \mathbf{y} \in D^r$,*

$$\frac{1}{2}(\phi(\mathbf{x}) + \phi(\mathbf{y})) \;\geq\; \sum_{f \in \Omega_D^{(2)}} \omega(f)\, \phi(f(\mathbf{x}, \mathbf{y})). \tag{1.5}$$

One can see the LHS of (1.5) as the average of $\phi(\mathbf{x})$ and $\phi(\mathbf{y})$ and the RHS as the expectation of $\phi(f(\mathbf{x}, \mathbf{y}))$ with respect to the probability distribution ω. We define the *support* of ω to be the set

$$\operatorname{supp}(\omega) = \{\, f \mid \omega(f) \neq 0 \,\} \tag{1.6}$$

of operations that get nonzero probability.

Note that a binary multimorphism $\langle f, g \rangle$ is a fractional polymorphism ω defined by $\omega(f) = \omega(g) = \frac{1}{2}$ and $\omega(h) = 0$ for all $h \notin \{f, g\}$. In this case, we have $\operatorname{supp}(\omega) = \{f, g\}$ and inequality (1.5) simplifies to (1.4).

A binary fractional polymorphism ω defined on D is called *symmetric* if every function from the support of ω is symmetric, that is, every $f \in \operatorname{supp}(\omega)$ satisfies $f(x, y) = f(y, x)$ for every $x, y \in D$. The following result is a consequence of the work of Thapper and Živný (2012) and Kolmogorov (2013), see also (Kolmogorov et al., 2013).

Theorem 1.1. *Let Γ be a $\mathbb{Q}$-valued language with a finite domain D. BLP solves all instance from* $\mathrm{VCSP}(\Gamma)$ *if and only if Γ admits a binary symmetric fractional polymorphism.*

Note that Theorem 1.1 proves tractability of all $\mathbb{Q}$-valued languages defined in Examples 1.2–1.10 as well as the skew bisubmodular languages defined in Example 1.13.

The following surprising result, due to Thapper and Živný (2013), shows that languages defined by binary symmetric fractional polymorphisms are the *only* tractable languages.

Theorem 1.2. *Let Γ be a $\mathbb{Q}$-valued language with a finite domain D. Either Γ admits a binary symmetric fractional polymorphism or* $\mathrm{VCSP}(\Gamma)$ *can be reduced to Max-Cut and thus is NP-hard.*

We remark that the reduction to Max-Cut mentioned in Theorem 1.2 is not just a polynomial-time reduction but a so-called *expressibility* reduction (Živný, 2012). Moreover, for a finite language Γ one can test for the existence of a binary symmetric fractional polymorphism of Γ via a linear program that has polynomial size in $|\Gamma|$ and double-exponential size in $|D|$. More details can be found in (Thapper and Živný, 2013).

1.3.3 Power of BLP for General-Valued Languages

In Section 1.3.2 we have given a complete characterization of tractable $\mathbb{Q}$-valued languages and have shown that BLP solves them all. In this section we will deal with $\overline{\mathbb{Q}}$-valued languages.

First, we will be interested in the question of which $\overline{\mathbb{Q}}$-valued languages are solvable by BLP. In order to do so, we need to extend the definition of binary fractional polymorphisms in two ways: firstly, to $\overline{\mathbb{Q}}$-valued functions and secondly, to fractional polymorphisms of arbitrary arities.

A k-ary *operation* is a mapping $f\colon D^k \to D$. We denote by $\Omega_D^{(k)}$ the set of all k-ary operations on D.

Definition 1.3 (Fractional polymorphism (Cohen et al., 2006a))**.** *Let ω be a probability distribution on $\Omega_D^{(k)}$. We say that ω is a* k-ary fractional polymorphism *of an r-ary function $\phi\colon D^r \to \overline{\mathbb{Q}}$ if, for every $\mathbf{x}^1, \dots, \mathbf{x}^k \in D^r$,*

$$\frac{1}{k}\sum_{i=1}^{k} \phi(\mathbf{x}^i) \;\geq\; \sum_{f \in \Omega_D^{(k)}} \omega(f)\, \phi(f(\mathbf{x}^1, \dots, \mathbf{x}^k)), \tag{1.7}$$

where we define $0 \cdot \infty = 0$ on the RHS of (1.7).

The support of ω is defined by (1.6). A k-ary fractional polymorphism ω is *symmetric* if every $f \in \operatorname{supp}(\omega)$ satisfies $f(x_1, \dots, x_k) = f(x_{\pi(1)}, \dots, x_{\pi(k)})$ for every $x_1, \dots, x_k \in D$ and every permutation π on $\{1, \dots, k\}$.

The following characterization of the power of BLP for general-valued languages is due to Thapper and Živný (2012), see also (Kolmogorov et al., 2013).

Theorem 1.3. *Let Γ be a $\overline{\mathbb{Q}}$-valued language with a finite domain D. BLP solves all instances from* $\mathrm{VCSP}(\Gamma)$ *if and only if Γ admits a k-ary symmetric fractional polymorphism of every arity $k \geq 2$.*

Note that unlike in the $\mathbb{Q}$-valued case (Theorem 1.1), it is not clear whether the characterization given in Theorem 1.3 is decidable. Nevertheless, Thapper and Živný (2012) have also given a sufficient condition on Γ for BLP to solve all instances from $\mathrm{VCSP}(\Gamma)$. We state this condition in Theorem 1.4.

A k-ary projection (on the ith coordinate) is the operation $e_i^{(k)}\colon D^k \to D$ defined by $e_i^{(k)}(x_1, \dots, x_k) = x_i$. A set $\mathcal{O}$ of operations defined on D *generates* an operation f if f can be obtained by composition from projections (of arbitrary arities) and operations from $\mathcal{O}$.

Theorem 1.4. *Let Γ be a $\overline{\mathbb{Q}}$-valued language with a finite domain D. Suppose that Γ admits a k-ary fractional polymorphism ω such that* $\operatorname{supp}(\omega)$

generates an m-ary symmetric operation. Then Γ admits an m-ary symmetric fractional polymorphism.

Corollary 1.5. *Let Γ be a $\overline{\mathbb{Q}}$-valued language with a finite domain D. Suppose that for every $k \geq 2$, Γ admits a (not necessarily k-ary) fractional polymorphisms ω so that* $\mathrm{supp}(\omega)$ *generates a k-ary symmetric operation. Then BLP solves any instance from* $\mathrm{VCSP}(\Gamma)$.

Note that the condition (of admitting symmetric fractional polymorphisms of all arities) from Theorem 1.3 trivially implies the condition from Corollary 1.5, thus showing that the condition from Corollary 1.5 is a characterization of the power of BLP.

A binary operation $f\colon D^2 \to D$ is called a *semi-lattice operation* if f is associative, commutative, and idempotent. Since any semi-lattice operation trivially generates symmetric operations of all arities, Corollary 1.5 shows that most $\overline{\mathbb{Q}}$-valued languages defined in Examples 1.2–1.10 as well as the skew bisubmodular languages from Example 1.13 are tractable. In the case of 1-defect languages from Example 1.8, a bit more work is needed to show the existence of symmetric operations of all arities, see (Thapper and Živný, 2012) for details. The $\overline{\mathbb{Q}}$-valued languages defined in Example 1.5 can be reduced, via a preprocessing described by Cohen et al. (2008), to an instance that is submodular and thus solvable by BLP as described in Example 1.2. The $\overline{\mathbb{Q}}$-valued languages defined in Example 1.10 can be reduced, via a preprocessing described by Kolmogorov and Živný (2013), to an instance that is submodular and thus solvable by BLP (see Example 1.2).

We finish this section with mentioning that obtaining a full complexity classification of all general-valued languages is extremely challenging. Indeed, even a classification of $\{0, \infty\}$-valued languages is not known. The so-called *Feder-Vardi Conjecture* (Feder and Vardi, 1998) states that every $\{0, \infty\}$-valued language is either tractable or intractable (note that assuming $P \neq NP$, Ladner (1975) showed that there are problems of intermediate complexity). However, there are some interesting results in this area. First, general-valued languages on 2-element domains have been classified by Cohen et al. (2006b). Second, an algebraic theory providing a powerful tool for analyzing the complexity of general-valued languages has been established by Cohen et al. (2011, 2013) and already used for simplifying the hardness part of the classification of general-valued languages on 2-element domains (Creed and Živný, 2011). Finally, conservative general-valued languages (see Example 1.10) have been completely classified by Kolmogorov and Živný (2013).

1.4 Universality of the Basic LP

We have seen that the basic LP relaxation solves many VCSP languages. Moreover, it has been empirically observed (Wainwright et al., 2005; Kolmogorov, 2006; Werner, 2007; Szeliski et al., 2008; Kappes et al., 2013) that it is tight for many VCSP instances that do not belong to any known tractable class. For other instances, it yields lower bounds which can be used, for instance, in exact search algorithms. For all these reasons, solving the BLP is of great practical interest.

The popular simplex and interior point methods are, due to their quadratic space complexity, applicable in practice only to small BLP instances. For larger instances, BLP can be solved efficiently for binary VCSPs with domain size $|D| = 2$, because in this case BLP can be reduced in linear time to the max-flow problem (Boros and Hammer, 2002; Rother et al., 2007). A lot of effort has been invested to develop efficient algorithms to exactly solve the BLP of more general VCSPs. Among the proposed algorithms are those based on subgradient methods (Schlesinger and Giginjak, 2007; Komodakis et al., 2011), smoothing methods (Weiss et al., 2007; Johnson et al., 2007; Ravikumar et al., 2008; Savchynskyy et al., 2011), and augmented Lagrangian methods (Martins et al., 2011; Schmidt et al., 2011; Meshi and Globerson, 2011).

In this section, we show that solving linear program (1.2) is not easier than solving an arbitrary linear program, in the following sense.

Theorem 1.6 (Průša and Werner (2013))**.** *Every linear program can be reduced in linear time to the basic LP relaxation (1.2) of a binary $\overline{\mathbb{Q}}$-valued VCSP with domain size $|D| = 3$.*

This result suggests that trying to find a very efficient algorithm to exactly solve the BLP may be futile because it might mean improving the complexity of the best known algorithm for general LP, which is unlikely.

In the rest of this section, we prove Theorem 1.6 by giving an algorithm that, for an arbitrary input LP, constructs a binary $\overline{\mathbb{Q}}$-valued VCSP with $|D| = 3$ whose basic LP relaxation solves the input LP.

1.4.1 The input linear program

The input linear program minimizes $\mathbf{c} \cdot \mathbf{x}$ over the polyhedron

$$P = \{\, \mathbf{x} = \langle x_1, \dots, x_n \rangle \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0} \,\}, \tag{1.8}$$

where $\mathbf{A} = [a_{ij}] \in \mathbb{Z}^{m\times n}$, $\mathbf{b} = \langle b_1, \dots, b_m\rangle \in \mathbb{Z}^m$, $\mathbf{c} = \langle c_1, \dots, c_n\rangle \in \mathbb{Z}^n$, and $m \le n$. Any LP representable by a finite number of bits can be described this way.

Before encoding, the system $\mathbf{Ax} = \mathbf{b}$ is rewritten as follows. Each equation

$$a_{i1}x_1 + \cdots + a_{in}x_n = b_i \tag{1.9}$$

is rewritten as

$$a^+_{i1}x_1 + \cdots + a^+_{in}x_n = a^-_{i1}x_1 + \cdots + a^-_{in}x_n + b_i \tag{1.10}$$

where $b_i \ge 0$, $a^+_{ij} \ge 0$, $a^-_{ij} \ge 0$, and $a_{ij} = a^+_{ij} - a^-_{ij}$. Moreover, it is assumed without loss of generality that neither side of (1.10) vanishes for any feasible $\mathbf{x}$.

The following lemmas are not surprising, their proofs can be found in (Průša and Werner, 2013).

Lemma 1.7. *Let $\mathbf{x} = \langle x_1, \dots, x_n\rangle$ be a vertex of the polyhedron P. Each component x_j of $\mathbf{x}$ satisfies either $x_j = 0$ or $M^{-1} \le x_j \le M$, where*

$$\begin{aligned} M &= m^{m/2}(B_1 \times \cdots \times B_{n+1}) \\ B_j &= \max(1, |a_{1j}|, \dots, |a_{mj}|), \quad j = 1, \dots, n \\ B_{n+1} &= \max(1, |b_1|, \dots |b_m|). \end{aligned}$$

Lemma 1.8. *Let P be bounded. Then for any $\mathbf{x} \in P$, each component of $\mathbf{A}^+\mathbf{x}$ and $\mathbf{A}^-\mathbf{x} + \mathbf{b}$ is not greater than $N = M(B_1 + \cdots + B_{n+1})$.*

The last lemma shows that we can restrict ourselves to input LPs with a bounded polyhedron P.

Lemma 1.9. *Every linear program can be reduced in linear time to a linear program over a bounded polyhedron.*

1.4.2 Elementary constructions

The output of the reduction will be a VCSP with domain size $|D| = 3$ and hypergraph $H = \binom{V}{1} \cup E$ where $E \subseteq \binom{V}{2}$ (that is, there is a unary constraint for each variable and binary constraints for a subset of variable pairs). We denote the binary constraints ϕ_S for $S = \{i, j\} \in E$ by ϕ_{ij}. Following Wainwright and Jordan (2008), we refer to the values of the functions μ_i and μ_{ij} as unary and binary *pseudomarginals*, respectively.

We will depict binary VCSPs by diagrams, commonly used in the constraint programming literature. Figure 1.1 illustrates the meaning of conditions (1.2b) and (1.2c) of the BLP in these diagrams.

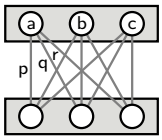
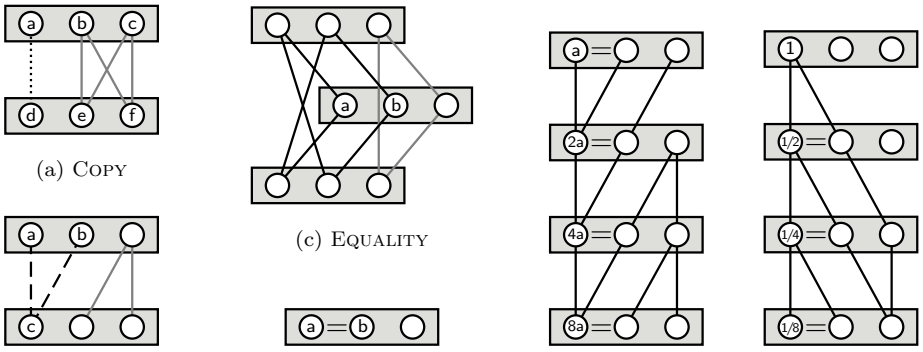


Figure 1.1: A pair of variables $\{i, j\} \in E$ with $|D| = 3$. Each variable is depicted as a box, its state $x \in D$ as a circle, and each state pair $\langle x, y \rangle \in D^2$ of two variables as an edge. Each circle is assigned a unary pseudomarginal $\mu_i(x)$ and each edge is assigned a binary pseudomarginal $\mu_{ij}(x, y)$. One normalization condition (1.2c) imposes for unary pseudomarginals a, b, c that $a + b + c = 1$. One marginalization condition (1.2b) imposes for pairwise pseudomarginals p, q, r that $a = p + q + r$.



(a) Copy (b) Addition (c) Equality (d) shorthand of Equality (e) Powers (f) NegPowers

Figure 1.2: Elementary constructions. The visible edges have costs $\phi_{ij}(x, y) = 0$ and the invisible edges have costs $\phi_{ij}(x, y) = \infty$. Different line styles of the visible edges distinguish different elementary constructions.

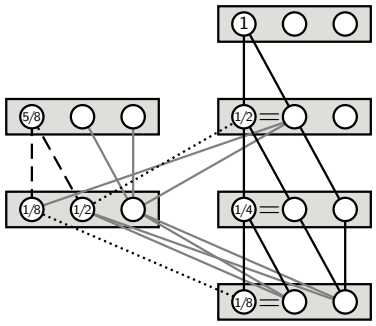


Figure 1.3: Construction of a unary pseudomarginal with value $\frac{5}{8}$. The example can be generalized in an obvious way to construct the value $2^{-d}k$ for any $d, k \in \mathbb{N}$ such that $2^{-d}k \leq 1$. If more than two values are added, intermediate results are stored in auxiliary variables using Copy.

The encoding algorithm uses several elementary constructions as its building blocks. Each construction is a standalone VCSP with crisp binary constraints, $\phi_{ij}\colon D^2 \to \{0,\infty\}$, that imposes a certain simple constraint on feasible unary pseudomarginals. Note that for any feasible pseudomarginals, $\phi_{ij}(x,y) = \infty$ implies $\mu_{ij}(x,y) = 0$. Each construction is defined by a diagram, in which visible edges have cost $\phi_{ij}(x,y) = 0$ and the invisible edges have cost $\phi_{ij}(x,y) = \infty$. The elementary constructions are as follows:

COPY, Figure 1.2(a), enforces equality of two unary pseudomarginals a, d in two variables $\{i,j\} \in E$ while imposing no other constraints on b, c, e, f. Precisely, if $a, b, c, d, e, f \geq 0$ and $a+b+c = 1 = d+e+f$, then there exist pairwise pseudomarginals feasible to (1.2) if and only if $a = d$.

ADDITION, Figure 1.2(b), adds two unary pseudomarginals a, b in one variable and represents the result as a unary pseudomarginal $c = a+b$ in another variable. No other constraints are imposed on the remaining unary pseudomarginals.

EQUALITY, Figure 1.2(c), enforces equality of two unary pseudomarginals a, b in a single variable, introducing two auxiliary variables. No other constraints are imposed on the remaining unary pseudomarginals. In the sequel, this construction will be abbreviated by omitting the two auxiliary variables and writing the equality sign between the two circles, as in Figure 1.2(d).

POWERS, Figure 1.2(e), creates the sequence of unary pseudomarginals with values $2^i a$ for $i = 0, \ldots, d$, each in a separate variable. We will call d the *depth* of the pyramid.

NEGPOWERS, Figure 1.2(f), is similar to POWERS but constructs values 2^{-i} for $i = 0, \ldots, d$.

Figure 1.3 shows an example of how the elementary constructions can be combined.

1.4.3 Encoding

Now we will formulate the encoding algorithm. The variables of the output VCSP and their states will be numbered by integers, $D = \{1,2,3\}$ and $V = \{1, \ldots, |V|\}$.

The algorithm is initialized as follows:

1.1. For each variable x_j in the input LP, introduce a new variable j into V and set $\phi_j(1) = c_j$. Pseudomarginal $\mu_j(1)$ will represent variable x_j. After this step, we have $V = \{1, \ldots, n\}$.

1.2. For each variable $j \in V$, build POWERS with the depth $d_j = \lfloor \log_2 B_j \rfloor$ based on state 1. This yields the sequence of numbers $2^i \mu_j(1)$, $i = 0, \ldots, d_j$.

1.3. Build NegPowers with the depth $d = \lceil \log_2 N \rceil$. By Lemma 1.8, the choice of d ensures that all values represented by pseudomarginals will be bounded by 1.

After initialization, the algorithm proceeds by encoding each equation (1.10) in turn. The ith equation (1.10) is encoded as follows:

2.1. Construct pseudomarginals with values $a_{ij}^{+}x_j$, $a_{ij}^{-}x_j$, $j = 1, \ldots, n$, by summing selected values from Powers built in Step 1.2, similarly as in Figure 1.3.

2.2. Construct a pseudomarginal with value $2^{-d}b_i$ by summing selected values from the NegPowers built in Step 1.3, similarly as in Figure 1.3. The value $2^{-d}b_i$ represents b_i, which sets the scale between the input and output polyhedron to 2^{-d}.

2.3. Represent each side of the equation by summing all its terms by repetitively applying Addition and Copy.

2.4. Apply Copy to enforce equality of the two sides of the equation.

Finally, set $\phi_i(x) = 0$ for all $i > n$ or $x \in \{2, 3\}$.

Figure 1.4 shows the output VCSP for an example input LP.

1.4.4 The length of the encoding

Here we finalize the proof of Theorem 1.6 by showing that the encoding time is linear. Since the encoding of vector $\mathbf{c}$ is clearly done in linear time, it suffices to show that the encoding time is linear in the length L of the binary representation of matrix $\mathbf{A}$ and vector $\mathbf{b}$. Since this time is obviously linear[6] in $|E|$, it suffices to show that $|E| = \mathcal{O}(L)$.

Variable pairs are created only when a variable is created and the number of variable pairs added with one variable is always bounded by a constant. Therefore $|E| = \mathcal{O}(|V|)$.

We clearly have the inequality $L \geq \max(mn, \log_2 B_1 + \cdots + \log_2 B_{n+1})$. The algorithm creates $\sum_{j=1}^{n}(d_j + 1)$ variables in Step 1.2 and $d+1$ variables in Step 1.3. By comparison with the above inequality, both of these numbers are $\mathcal{O}(L)$.

Finally, encoding one equality (1.10) adds at most as many variables as there are bits in the binary representation of all its coefficients. The

6. The only thing that may not be obvious is how to multiply large integers a, b in linear time. But this issue can be avoided by instead computing $p(a, b) = 2^{\lceil \log_2 a \rceil + \lceil \log_2 b \rceil}$, which can be done in linear time using bitwise operations. Since $ab \leq p(a, b) \leq (2a)(2b)$, the bounds like M become larger but this does not affect the overall complexity.

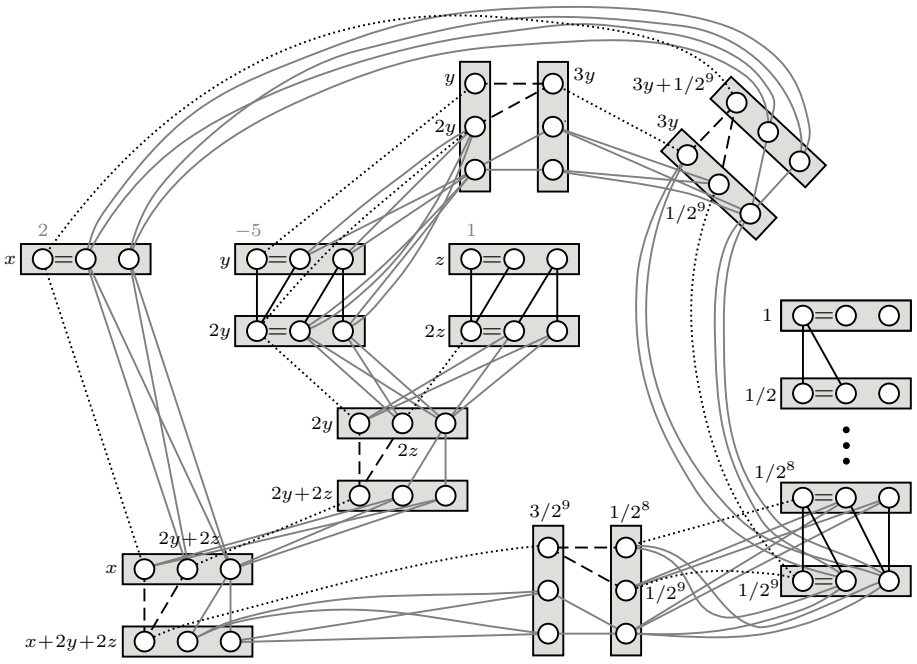


Figure 1.4: The VCSP whose basic LP relaxation solves the linear program $\min\{\, 2x - 5y + z \mid x + 2y + 2z = 3;\ x = 3y + 1;\ x, y, z \geq 0 \,\}$.

cumulative sum is thus $\mathcal{O}(L)$.

1.5 Conclusions

LP relaxation is a sucessfull approach to the problem of minimizing a partially separable function of many discrete variables, which is also known as the valued constraint satisfaction problem (VCSP). In this chapter, we have presented two types of theoretial results on the basic LP relaxation of VCSP: in Section 1.3, we characterized languages solves exactly by BLP and, in Section 1.4, we showed that solving BLP is as hard as solving an arbitrary LP.

These results suggest a number of questions. The first class of questions concerns the fact that rather than finding a global optimum of the LP relaxation, it is easier to find its local dual optimum with respect to block-coordinate moves. The latter in fact means reparameterizing the problem

such that the locally minimal tuples are arc consistent (Shlezinger, 1976; Werner, 2007), which has been called *virtual arc consistency* by Cooper et al. (2010a). Virtual arc consistency is enforced by the popular message-passing algorithms such as min-sum diffusion (Kovalevsky and Koval, approx. 1975; Werner, 2007, 2010), TRW-S (Kolmogorov, 2006) (see its generalization to VCSPs of any arity in Chapter **??**) and MPLP (Globerson and Jaakkola, 2008; Sontag et al., 2011), as well as by the algorithms (Koval and Schlesinger, 1976; Cooper et al., 2010a). Regarding Section 1.3, one can ask which languages are solved by enforcing virtual arc consistency. For instance, it is known that enforcing virtual arc consistency solves submodular languages of any arity (Werner, 2010; Cooper et al., 2010a) but for other languages the question is open. Regarding Section 1.4, one can ask whether enforcing virtual arc consistency is easier than solving the BLP exactly.

Recall that one can construct, in a number of ways, a hierarchy of increasingly tighter LP relaxations of VCSP (Sherali and Adams, 1990; Wainwright and Jordan, 2008; Johnson et al., 2007; Werner, 2010; Franc et al., 2012). BLP (1.2) is only one level of this hierarchy. As the second question, one can ask how much power these higher-order relaxations add to BLP. Theorems 1.1 and 1.2 imply the surprising fact that *all* tractable finite-valued languages are solved by BLP, hence higher-order relaxations do not allow us to solve any more languages. However, could BLP and more generally higher-order relaxations be useful for interesting, not necessarily language-restricted, classes of VCSPs?

Acknowledgment

D.Průša and T.Werner have been supported by the Grant Agency of the Czech Republic project P202/12/2071. Besides, D.Průša has been supported by the EC project FP7-ICT-247525 and T.Werner by the EC project FP7-ICT-270138. S.Živný has been supported by a Royal Society University Research Fellowship.

1.6 References

L. Barto, M. Kozik, and T. Niven. The CSP dichotomy holds for digraphs with no sources and no sinks (a positive answer to a conjecture of Bang-Jensen and Hell). *SIAM Journal on Computing*, 38(5):1782–1802, 2009.

E. Boros and P. L. Hammer. Pseudo-Boolean optimization. *Discrete Applied Mathematics*, 123(1-3):155–225, 2002.

A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-

element set. *Journal of the ACM*, 53(1):66–120, 2006.

A. Bulatov. Complexity of conservative constraint satisfaction problems. *ACM Transactions on Computational Logic*, 12(4), 2011. Article 24.

A. Bulatov, A. Krokhin, and P. Jeavons. Classifying the Complexity of Constraints using Finite Algebras. *SIAM Journal on Computing*, 34(3):720–742, 2005.

C. Chekuri, S. Khanna, J. Naor, and L. Zosin. A linear programming formulation and approximation algorithms for the metric labeling problem. *SIAM Journal on Discrete Mathematics*, 18(3):608–625, 2005.

D. Cohen and P. Jeavons. The complexity of constraint languages. In F. Rossi, P. van Beek, and T. Walsh, editors, *The Handbook of Constraint Programming*. Elsevier, 2006.

D. A. Cohen, M. C. Cooper, and P. G. Jeavons. An Algebraic Characterisation of Complexity for Valued Constraints. In *Intl. Conf. on Principles and Practice of Constraint Programming (CP)*, pages 107–121. Springer, 2006a.

D. A. Cohen, M. C. Cooper, P. G. Jeavons, and A. A. Krokhin. The Complexity of Soft Constraint Satisfaction. *Artificial Intelligence*, 170(11):983–1016, 2006b.

D. A. Cohen, M. C. Cooper, and P. G. Jeavons. Generalising submodularity and Horn clauses: Tractable optimization problems defined by tournament pair multimorphisms. *Theoretical Computer Science*, 401(1-3):36–51, 2008.

D. A. Cohen, P. Creed, P. G. Jeavons, and S. Živný. An algebraic theory of complexity for valued constraints: Establishing a Galois connection. In *Intl. Symp. on Mathematical Foundations of Computer Science (MFCS)*, pages 231–242. Springer, 2011.

D. A. Cohen, M. C. Cooper, P. Creed, P. Jeavons, and S. Živný. An algebraic theory of complexity for discrete optimisation. *SIAM Journal on Computing*, 2013. To appear.

M. C. Cooper. Minimization of Locally Defined Submodular Functions by Optimal Soft Arc Consistency. *Constraints*, 13(4):437–458, 2008.

M. C. Cooper, S. de Givry, M. Sánchez, T. Schiex, M. Zytnicki, and T. Werner. Soft arc consistency revisited. *Artificial Intelligence*, 174(7–8):449–478, 2010a.

M. C. Cooper, P. G. Jeavons, and A. Z. Salamon. Generalizing constraint satisfaction on trees: Hybrid tractability and variable elimination. *Artificial Intelligence*, 174(9–10):570–584, 2010b.

M. C. Cooper and S. Živný. Hybrid tractability of valued constraint problems. *Artificial Intelligence*, 175(9-10):1555–1569, 2011.

M. C. Cooper and S. Živný. Tractable triangles and cross-free convexity in discrete optimisation. *Journal of Artificial Intelligence Research*, 44:455–490, 2012.

P. Creed and S. Živný. On minimal weighted clones. In *Intl. Conf. on Principles and Practice of Constraint Programming (CP)*, pages 210–224. Springer, 2011.

V. Dalmau, P. G. Kolaitis, and M. Y. Vardi. Constraint Satisfaction, Bounded Treewidth, and Finite-Variable Logics. In *Intl. Conf. on Principles and Practice of Constraint Programming (CP)*, pages 310–326. Springer, 2002.

R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

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 on Computing*, 28(1):57–104, 1998.

V. Franc, S. Sonnenburg, and T. Werner. Cutting plane methods in machine

learning. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for Machine Learning*. MIT Press, 2012.

S. Fujishige. *Submodular Functions and Optimization*. North-Holland, 2005.

S. Fujishige and S. Iwata. Bisubmodular Function Minimization. *SIAM Journal on Discrete Mathematics*, 19(4):1065–1073, 2005.

M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.

A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *Conf. on Neural Information Processing Systems (NIPS)*, pages 553–560, 2008.

G. Gottlob, N. Leone, and F. Scarcello. A comparison of structural CSP decomposition methods. *Artificial Intelligence*, 124(2):243–282, 2000.

M. Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *Journal of the ACM*, 54(1):1–24, 2007.

F. Hadlock. Finding a maximum cut of a planar graph in polynomial time. *SIAM Journal on Computing*, 4(3):221–5, 1975.

A. Huber and V. Kolmogorov. Towards Minimizing k-Submodular Functions. In *Intl. Symp. on Combinatorial Optimization (ISCO)*, pages 451–462. Springer, 2012.

A. Huber, A. Krokhin, and R. Powell. Skew Bisubmodularity and Valued CSPs. In *ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 1296–1305. SIAM, 2013.

S. Iwata, L. Fleischer, and S. Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM*, 48(4): 761–777, 2001.

P. G. Jeavons, D. A. Cohen, and M. Gyssens. Closure Properties of Constraints. *Journal of the ACM*, 44(4):527–548, 1997.

J. K. Johnson, D. M. Malioutov, and A. S. Willsky. Lagrangian relaxation for MAP estimation in graphical models. In *Allerton Conf. on Communication, Control and Computing*, pages 64–73, 2007.

P. Jonsson, F. Kuivinen, and J. Thapper. Min CSP on Four Elements: Moving Beyond Submodularity. In *Intl. Conf. on Principles and Practice of Constraint Programming (CP)*, pages 438–453. Springer, 2011.

J. H. Kappes, B. Andres, F. A. Hamprecht, C. Schnörr, S. Nowozin, D. Batra, S. Kim, B. X. Kausler, J. Lellmann, N. Komodakis, and C. Rother. A comparative study of modern inference techniques for discrete energy minimization problem. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2013.

C. L. Kingsford, B. Chazelle, and M. Singh. Solving and analyzing side-chain positioning problems using linear and integer programming. *Bioinformatics*, 21 (7):1028–1039, 2005.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10): 1568–1583, 2006.

V. Kolmogorov. Submodularity on a tree: Unifying $L^\natural$-convex and bisubmodular functions. In *Intl. Symp. on Mathematical Foundations of Computer Science*

(MFCS), pages 400–411. Springer, 2011.

V. Kolmogorov. The power of linear programming for finite-valued CSPs: a constructive characterization. In *Intl. Coll. on Automata, Languages and Programming (ICALP)*, pages 625–636. Springer, 2013.

V. Kolmogorov, J. Thapper, and S. Živný. The power of linear programming for general-valued CSPs. 2013. Submitted for publication.

V. Kolmogorov and S. Živný. The complexity of conservative valued CSPs. *Journal of the ACM*, 60(2), 2013.

N. Komodakis, N. Paragios, and G. Tziritas. MRF energy minimization and beyond via dual decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):531–552, 2011.

A. Koster, S. van Hoesel, and A. Kolen. The partial constraint satisfaction problem: Facets and lifting theorems. *Operations Research Letters*, 23(3–5):89–97, 1998.

V. K. Koval and M. I. Schlesinger. Dvumernoe programmirovanie v zadachakh analiza izobrazheniy (Two-dimensional programming in image analysis problems). *Automatics and Telemechanics*, 8:149–168, 1976. In Russian.

V. A. Kovalevsky and V. K. Koval. A diffusion algorithm for decreasing the energy of the max-sum labeling problem. Glushkov Institute of Cybernetics, Kiev, USSR. Unpublished, approx. 1975.

A. Krokhin and B. Larose. Maximizing Supermodular Functions on Product Lattices, with Application to Maximum Constraint Satisfaction. *SIAM Journal on Discrete Mathematics*, 22(1):312–328, 2008.

F. Kuivinen. On the complexity of submodular function minimisation on diamonds. *Discrete Optimization*, 8(3):459–477, 2011.

G. Kun, R. O'Donnell, S. Tamaki, Y. Yoshida, and Y. Zhou. Linear programming, width-1 CSPs, and robust satisfaction. In *Innovations in Theoretical Computer Science (ITCS) Conf.*, pages 484–495. ACM, 2012.

R. Ladner. On the Structure of Polynomial Time Reducibility. *Journal of the ACM*, 22:155–171, 1975.

S. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

A. L. Martins, M. A. T. Figueiredo, P. M. Q. Aguiar, N. A. Smith, and E. P. Xing. An augmented Lagrangian approach to constrained MAP inference. In *Intl. Conf. on Machine Learning (ICML)*, pages 169–176. Omnipress, 2011.

D. Marx. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. In *ACM Symp. on Theory of Computing (STOC)*, pages 735–744. ACM, 2010.

O. Meshi and A. Globerson. An alternating direction method for dual MAP LP relaxation. In *Conf. on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, 2011.

M. Mezard and A. Montanari. *Information, Physics, and Computation*. Oxford University Press, 2009.

U. Montanari. Networks of Constraints: Fundamental properties and applications to picture processing. *Information Sciences*, 7:95–132, 1974.

S. Nowozin and C. H. Lampert. Structured learning and prediction in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 6(3-4):185–365, 2011.

D. Průša and T. Werner. Universality of the local marginal polytope. In *Conf. on*

Computer Vision and Pattern Recognition (CVPR). IEEE, 2013.

P. Ravikumar, A. Agarwal, and M. J. Wainwright. Message-passing for graph-structured linear programs: proximal projections, convergence and rounding schemes. In *Intl. Conf. on Machine Learning (ICML)*, pages 800–807. ACM, 2008.

F. Rossi, P. van Beek, and T. Walsh, editors. *Handbook of Constraint Programming*. Elsevier, 2006.

C. Rother, V. Kolmogorov, and A. Blake. "GrabCut": interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*, pages 309–314. ACM Press, 2004.

C. Rother, V. Kolmogorov, V. S. Lempitsky, and M. Szummer. Optimizing binary MRFs via extended roof duality. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2007.

B. Savchynskyy, J. Kappes, S. Schmidt, and C. Schnörr. A study of Nesterov's scheme for Lagrangian decomposition and MAP labeling. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1817–1823. IEEE, 2011.

T. J. Schaefer. The Complexity of Satisfiability Problems. In *ACM Symp. on Theory of Computing (STOC)*, pages 216–226. ACM, 1978.

T. Schiex, H. Fargier, and G. Verfaillie. Valued Constraint Satisfaction Problems: Hard and Easy Problems. In *Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 631–637, 1995.

D. Schlesinger. Exact solution of permuted submodular MinSum problems. In *Conf. on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, pages 28–38. Springer, 2007.

D. Schlesinger and B. Flach. Transforming an arbitrary MinSum problem into a binary one. Technical Report TUD-FI06-01, Dresden University of Technology, Germany, 2006.

M. I. Schlesinger and V. V. Giginjak. Solving (max,+) problems of structural pattern recognition using equivalent transformations. *Upravlyayushchie Sistemy i Mashiny (Control Systems and Machines), Kiev, Naukova Dumka*, 1 and 2, 2007. ISSN 0130-5395. In Russian, English translation available on www.

S. Schmidt, B. Savchynskyy, J. H. Kappes, and C. Schnörr. Evaluation of a first-order primal-dual algorithm for MRF energy minimization. In *Conf. on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 89–103. Springer, 2011.

A. Schrijver. A Combinatorial Algorithm Minimizing Submodular Functions in Strongly Polynomial Time. *Journal of Combinatorial Theory, Series B*, 80(2): 346–355, 2000.

A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer, 2003.

H. D. Sherali and W. P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal of Discrete Mathematics*, 3(3):411–430, 1990.

M. I. Shlezinger. Syntactic analysis of two-dimensional visual signals in noisy conditions. *Cybernetics and Systems Analysis*, 12(4):612–628, 1976. Translation from Russian.

D. Sontag, A. Globerson, and T. Jaakkola. Introduction to dual decomposition for inference. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for*

Machine Learning. MIT Press, 2011.

R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(6):1068–1080, 2008.

J. Thapper and S. Živný. The power of linear programming for valued CSPs. In *IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 669–678. IEEE, 2012.

J. Thapper and S. Živný. The complexity of finite-valued CSPs. In *ACM Symp. on the Theory of Computing (STOC)*, pages 695–704. ACM, 2013.

M. Wainwright, T. Jaakkola, and A. Willsky. MAP estimation via agreement on trees: message passing and linear programming. *IEEE Transactions on Information Theory*, 51(11):3697–3717, 2005.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

Y. Weiss, C. Yanover, and T. Meltzer. MAP estimation, linear programming and belief propagation with convex free energies. In *Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2007.

T. Werner. A linear programming approach to max-sum problem: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1165–1179, 2007.

T. Werner. Revisiting the linear programming relaxation approach to Gibbs energy minimization and weighted constraint satisfaction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1474–1488, 2010.

S. Živný. *The complexity of valued constraint satisfaction problems*. Cognitive Technologies. Springer, 2012.