Universality of the Local Marginal Polytope

Daniel Průša and Tomáš Werner

Abstract—We show that solving the LP relaxation of the min-sum labeling problem (also known as MAP inference problem in graphical models, discrete energy minimization, or valued constraint satisfaction) is not easier than solving any linear program. Precisely, every polytope is linear-time representable by a local marginal polytope and every LP can be reduced in linear time to a linear optimization (allowing infinite costs) over a local marginal polytope. The reduction can be done (though with a higher time complexity) even if the local marginal polytope is restricted to have a planar structure.

Index Terms—Graphical model, Markov random field, discrete energy minimization, valued constraint satisfaction, linear programming relaxation, local marginal polytope

1 Introduction

The *min-sum (labeling) problem* is defined as follows: given a set of discrete variables and a set of functions depending on one or two variables, minimize the sum of the functions over all variables. This problem arises in MAP inference in graphical models [22] and it is also known as discrete energy minimization [9] or valued constraint satisfaction [21].

This NP-complete problem has a natural linear programming (LP) relaxation, proposed by a number of authors [4], [13], [18], [22]. This relaxation is equivalent to the dual (Lagrangian) decomposition of the min-sum problem [8], [12], [19]. While the min-sum problem can be formulated as a linear optimization over the *marginal polytope*, the LP relaxation approximates this polytope by its outer bound, the *local marginal polytope* [22].

The relaxation is exact for a large class of min-sum instances and it is a basis for constructing good approximations for many other instances [9], [20], [23]. It is therefore of great practical interest to have efficient algorithms to solve the LP relaxation.

To solve the LP relaxation, the simplex and interior point methods are prohibitively inefficient for large-scale instances (which often occur, e.g., in computer vision). For min-sum problems with two labels, the LP relaxation can be solved efficiently because it reduces in linear time to max-flow [3], [17]. For more general problems, no really efficient algorithm is known to solve the LP.

In this paper we show that the quest for efficient algorithms to solve the LP relaxation of the general min-sum problem has a fundamental limitation, because this task is not easier than solving any linear program. Precisely, we prove the following theorems.

Theorem 1. *Every polytope is (up to scale) a coordinate-erasing projection of a face of a local marginal polytope with three labels, whose description can be computed from the input polytope in linear time.*

The input polytope is described by a set of linear inequalities with integer coefficients. By coordinate-erasing projection, we mean a projection that copies a subset of coordinates and erases the remaining ones.

Theorem 2. *Every linear program can be reduced in linear time to a linear optimization (allowing infinite costs) over a local marginal polytope with three labels.*

While Theorem 2 immediately follows from Theorem 1, the situation is more complex when infinite costs are not allowed. In this case, the reduction time and the output size are quadratic (see Theorem 9).

Given these negative results, one may ask whether the LP relaxation can be solved efficiently for some useful subclasses of the min-sum problem. One such subclass is the planar min-sum problem, which frequently occurs in computer vision. We show (in Theorem 11) that even in this case, the reduction can be done (with infinite costs allowed), in better than quadratic time.

Similar universality results are known also for other polytopes, e.g., the three-way transportation polytope [6] and the traveling salesman polytope [2].

2 The Local Marginal Polytope

Let (V, E) be an undirected graph, where V is a finite set of *objects* and $E \subseteq \binom{V}{2}$ is a set of object pairs. Let K be a finite set of *labels*. Let $g_u\colon K \to \overline{\mathbb{R}}$ and $g_{uv}\colon K \times K \to \overline{\mathbb{R}}$ be unary and binary *cost functions*, where $\overline{\mathbb{R}} = \mathbb{R} \cup \{\infty\}$ and we adopt that $g_{uv}(k,\ell) = g_{vu}(\ell,k)$. The *min-sum problem* is defined as

$$\min_{\mathbf{k}\in K^V} \left(\sum_{u\in V} g_u(k_u) + \sum_{\{u,v\}\in E} g_{uv}(k_u,k_v) \right). \tag{1}$$

All the costs $g_u(k), g_{uv}(k,\ell)$ form a vector $\mathbf{g} \in \overline{\mathbb{R}}^I$ where $I = (V \times K) \cup \{\{(u,k),(v,\ell)\} \,|\, \{u,v\} \in E;\ k,\ell \in K\}$. The problem instance is given by a tuple $(V, E, K, \mathbf{g})$.

The *local marginal polytope* [22] is the set Λ of vectors $\boldsymbol{\mu} \in \mathbb{R}^I_+$ satisfying

$$\sum_{\ell\in K} \mu_{uv}(k,\ell) = \mu_u(k), \quad u \in V,\ v \in N_u,\ k \in K, \tag{2a}$$

$$\sum_{k\in K} \mu_u(k) = 1, \quad u \in V, \tag{2b}$$

where $N_u = \{\, v \,|\, \{u,v\} \in E \,\}$ are the neighbors of u and we assume $\mu_{uv}(k,\ell) = \mu_{vu}(\ell,k)$. The numbers $\mu_u(k), \mu_{uv}(k,\ell)$ are known as *pseudomarginals* [22]. The local marginal polytope is given by a triplet (V, E, K).

The LP relaxation of the min-sum problem reads

$$\Lambda^*(\mathbf{g}) = \operatorname*{argmin}_{\boldsymbol{\mu}\in\Lambda} \langle \mathbf{g}, \boldsymbol{\mu} \rangle, \tag{3}$$

where in the scalar product $\langle \mathbf{g}, \boldsymbol{\mu} \rangle$ we define $0 \cdot \infty = 0$. The set (3) contains all vectors $\boldsymbol{\mu}$ for which $\langle \mathbf{g}, \boldsymbol{\mu} \rangle$ attains minimum over Λ. It is itself a polytope, a face of Λ.

We will depict min-sum problems by diagrams, as in Fig. 1. Objects $u \in V$ are depicted as boxes, labels $(u,k) \in I$ as nodes, label pairs $\{(u,k),(v,\ell)\} \in I$ as edges. Each node is assigned a unary pseudomarginal $\mu_u(k)$ and cost $g_u(k)$. Each edge is assigned a binary pseudomarginal $\mu_{uv}(k,\ell)$ and cost $g_{uv}(k,\ell)$.

Note the meaning of constraints (2) in Fig. 1. Constraint (2b) imposes for unary pseudomarginals a, b, c that $a + b + c = 1$. Constraint (2a) imposes for binary pseudomarginals p, q, r that $a = p + q + r$.

3 Input Polyhedron

We consider the input polyhedron in the form

$$P = \{\, \mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n \,|\, \mathbf{A}\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0} \,\}, \tag{4}$$

where[1] $\mathbf{A} = [a_{ij}] \in \mathbb{Z}^{m\times n}$, $\mathbf{b} = (b_1, \ldots, b_m) \in \mathbb{Z}^m$, $m \leq n$. We assume there is at least one non-zero entry in each row and column of $\mathbf{A}$.

• *The authors are with the Department of Cybernetics, Czech Technical University, Karlovo náměstí 13,12135 Praha, Czech Republic.*
E-mail: {prusapa1, werner}@cmp.felk.cvut.cz.

Manuscript received 16 Aug. 2013; revised 15 July 2014; accepted 28 July 2014. Date of publication 28 Aug. 2014; date of current version 3 Mar. 2015.
Recommended for acceptance by C. H. Lampert.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPAMI.2014.2353626

1. The assumption that $(\mathbf{A}, \mathbf{b})$ are integer-valued is common, see e.g., [10]. In the more general case of rational-valued $(\mathbf{A}, \mathbf{b})$, Lemma 4 would not hold. Linear complexity of the reduction could probably be maintained under some additional assumptions, such as prior bounds on the sizes of coordinates of the vertices of P.

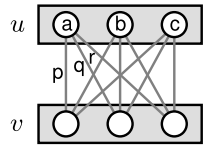

Fig. 1. A pair of objects $\{u, v\} \in E$ with $|K| = 3$ labels.

The instance of polyhedron (4) is given by $(\mathbf{A}, \mathbf{b})$ or, in short, by the extended matrix

$$\bar{\mathbf{A}} = [\bar{a}_{ij}] = [\mathbf{A} \,|\, \mathbf{b}] \in \mathbb{Z}^{m \times (n+1)}. \tag{5}$$

It will be convenient to rewrite the system $\mathbf{Ax} = \mathbf{b}$ as follows. In the ith equation

$$a_{i1}x_1 + \cdots + a_{in}x_n = b_i, \tag{6}$$

it is assumed that $b_i \geq 0$ (if not, multiply the equation by -1). Further, the terms with negative coefficients are moved to the right-hand side, such that both sides have only non-negative terms. Thus, (6) is rewritten as

$$a^+_{i1}x_1 + \cdots + a^+_{in}x_n = a^-_{i1}x_1 + \cdots + a^-_{in}x_n + b_i, \tag{7}$$

where $a^+_{ij} \geq 0$, $a^-_{ij} \geq 0$, $a_{ij} = a^+_{ij} - a^-_{ij}$. We assume w.l.o.g. that $a^+_{i1} + \cdots + a^+_{in} \neq 0$ and $a^-_{i1} + \cdots + a^-_{in} + b_i \neq 0$.

The following lemmas give some bounds that will be needed in the encoding algorithm.

Lemma 3. *For every matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ with columns $\mathbf{A}_j$,*

$$|\det \mathbf{A}| \leq \prod_{j=1}^{n} \|\mathbf{a}_j\|_2 \leq \prod_{j=1}^{n} \|\mathbf{a}_j\|_1.$$

Proof. The first inequality is well-known as Hadamard's inequality. The second inequality holds because $\|\mathbf{a}\|_2 \leq \|\mathbf{a}\|_1$ for every $\mathbf{a} \in \mathbb{R}^n$. □

Lemma 4. *Let $\mathbf{b} \neq \mathbf{0}$. Let $(x_1, \ldots, x_n)$ be a vertex of P. Then for each j we have $x_j = 0$ or $M^{-1} \leq x_j \leq M$ where*

$$M = \prod_{j=1}^{n+1} \sum_{i=1}^{m} |\bar{a}_{ij}|. \tag{8}$$

Proof. It is well-known from the theory of linear programming that every vertex $\mathbf{x}$ of P is a solution of a system $\mathbf{A}'\mathbf{x}' = \mathbf{b}'$, where $\mathbf{x}' = (x'_1, x'_2, \ldots)$ are the non-zero components of $\mathbf{x}$, $\mathbf{A}'$ is a non-singular submatrix of $\mathbf{A}$, and $\mathbf{b}'$ is a subvector of $\mathbf{b}$. By Cramer's rule,

$$x'_j = \frac{\det \mathbf{A}'_j}{\det \mathbf{A}'}, \tag{9}$$

where $\mathbf{A}'_j$ denotes $\mathbf{A}'$ with the jth column replaced by $\mathbf{b}'$. Lemma 3 implies $|\det \mathbf{A}'_j|, |\det \mathbf{A}'| \leq M$. □

Lemma 5. *Let P be bounded. Then for every $\mathbf{x} \in P$, each side of equation (7) is not greater than*

$$N = M \max_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|. \tag{10}$$

Proof. Since every point $(x_1, \ldots, x_n)$ of P is a convex combination of vertices of P, we have $x_j \leq M$ for each j. Hence, $a^+_{i1}x_1 + \cdots + a^+_{in}x_n \leq M(|a_{i1}| + \cdots + |a_{in}|) \leq N$ for each i. □

4 Encoding a Polytope

In this section, we prove Theorem 1 by constructing, in linear time, a min-sum problem $(V, E, K, \mathbf{g})$ with costs $\mathbf{g} \in \{0, 1\}^I$ such that the input polyhedron P is a scaled coordinate-erasing projection of $\Lambda^*(\mathbf{g})$. We assume that P is bounded, i.e., a polytope.[2]

4.1 Elementary Constructions

The output min-sum problem will be constructed from small building blocks, which implement certain simple operations on unary pseudomarginals. We call these blocks *elementary constructions*. An elementary construction is a min-sum problem with $|K| = 3$ labels, zero unary costs $g_u(k) = 0$, binary costs $g_{uv}(k, \ell) \in \{0, 1\}$, and optimal value $\min_{\mu \in \Lambda} \langle \mathbf{g}, \mu \rangle = 0$. It follows that $\mu \in \Lambda$ is optimal to the LP relaxation if and only if

$$g_{uv}(k, \ell)\, \mu_{uv}(k, \ell) = 0, \quad \{u, v\} \in E;\ k, \ell \in K. \tag{11}$$

We will define elementary constructions by diagrams such as in Fig. 1, in which we draw only edges with costs $g_{uv}(k, \ell) = 1$. Edges with costs $g_{uv}(k, \ell) = 0$ are not drawn. We will use the following elementary constructions (see Fig. 2):

Copy enforces equality of two unary pseudomarginals a, d in two objects while imposing no other constraints on b, c, e, f. Precisely, given any feasible unary pseudomarginals a, b, c, d, e, f, there exist feasible binary pseudomarginals satisfying (11) if and only if $a = d$.

Addition adds two unary pseudomarginals a, b in one object and represents the result as a unary pseudomarginal $c = a + b$ in another object. No other constraints are imposed on the remaining unary pseudomarginals.

Equality enforces equality of two unary pseudomarginals a, b in a single object, introducing two auxiliary objects. No other constraints are imposed on the remaining unary pseudomarginals. In the sequel, this construction will be abbreviated by omitting the two auxiliary objects and writing the equality sign between the two nodes, as shown in Fig. 2d.

Powers creates the sequence of unary pseudomarginals with values $2^i a$ for $i = 0, \ldots, d$, each in a separate object. We call d the *depth* of the pyramid.

NegPowers is similar to Powers but constructs values 2^{-i} for $i = 0, \ldots, d$.

Fig. 3 shows an example of how the elementary constructions can be combined. The edge colors distinguish different elementary constructions. By summing selected bits from NegPowers, the number $\frac{5}{8}$ is constructed. The example can be easily generalized to construct the value $2^{-d}k$ for any $d, k \in \mathbb{N}$ such that $2^{-d}k \leq 1$.

4.2 The Algorithm

Now we are ready to describe the encoding algorithm. The input of the algorithm is a set of equalities (7). Its output will be a min-sum problem $(V, E, K, \mathbf{g})$ with $|K| = 3$ labels and costs $g_u(k) = 0$, $g_{uv}(k, \ell) \in \{0, 1\}$. We will number labels and objects by integers, $K = \{1, 2, 3\}$ and $V = \{1, \ldots, |V|\}$.

The algorithm is initialized as follows:

1.1. For each variable x_j in (4), introduce a new object j into V. The variable x_j will be represented (up to scale) by pseudomarginal $\mu_j(1)$.

1.2. For each such object j, build Powers to the depth $d_j = \lfloor \log_2 \max_{i=1}^{m} |a_{ij}| \rfloor$ based on label 1. This yields the sequence of numbers $2^i \mu_j(1)$ for $i = 0, \ldots, d_j$.

1.3. Build NegPowers to the depth $d = \lceil \log_2 N \rceil$.

2. If the input polytope is in the general form $\{\, \mathbf{x} \in \mathbb{R}^n \,|\, \mathbf{Ax} \leq \mathbf{b} \,\}$, it can be transformed to the form (4) by adding slack variables and translating.

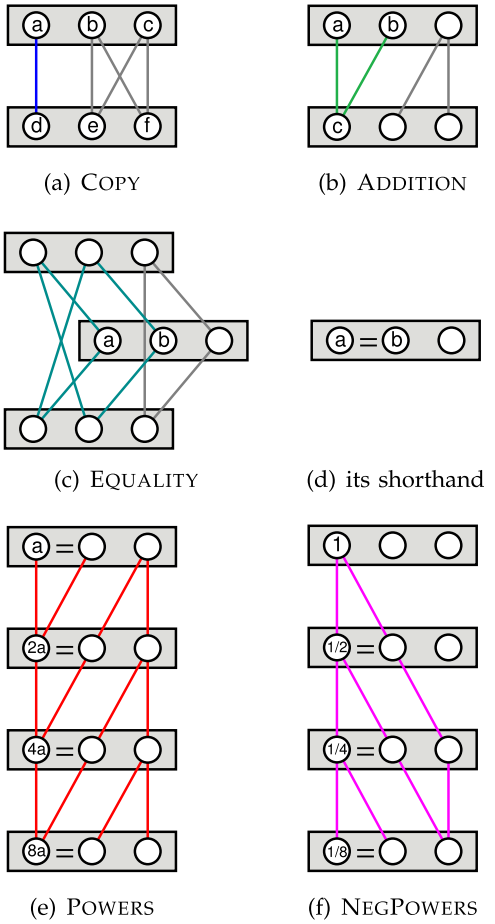

Fig. 2. Elementary constructions.

Then the algorithm proceeds by encoding each equation (7). The ith equation is encoded as follows:

2.1. Construct pseudomarginals with non-zero values $|a_{ij}|x_j$, $j = 1, \dots, n$, by summing selected values from POWERS built in Step 1.2, similarly as in Fig. 3. Note that the depths d_j are large enough to make this possible.

2.2. Construct a pseudomarginal with value $2^{-d}b_i$ by summing selected bits from NEGPOWERS built in Step 1.3, similarly as in Fig. 3. The value $2^{-d}b_i$ represents b_i, which sets the scale (mentioned in Theorem 1) between the input and output polytope to 2^{-d}. Note, the depth d is large enough to ensure that all pseudomarginals are bounded by 1.

2.3. Sum all the terms on each side of the equation by repetitively applying ADDITION and COPY.

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

Fig. 4 shows the output min-sum problem for an example polytope P. By construction, the resulting min-sum problem encodes the input polytope as follows:

- If $P = \emptyset$ then $\min_{\boldsymbol{\mu} \in \Lambda} \langle \mathbf{g}, \boldsymbol{\mu} \rangle > 0$.
- If $P \neq \emptyset$ then $\min_{\boldsymbol{\mu} \in \Lambda} \langle \mathbf{g}, \boldsymbol{\mu} \rangle = 0$ and

$$P = \pi(\Lambda^*(\mathbf{g})), \tag{12}$$

where $\pi\colon \mathbb{R}^I \to \mathbb{R}^n$ is the scaled coordinate-erasing projection given by

$$(x_1, \dots, x_n) = \pi(\boldsymbol{\mu}) = 2^d(\mu_1(1), \dots, \mu_n(1)). \tag{13}$$

Let us make some remarks on this construction. The output min-sum problem has costs $\mathbf{g} \in \{0, 1\}^I$ but we could also use $\mathbf{g} \in \{0, \infty\}^I$ without affecting the result. The min-sum problem

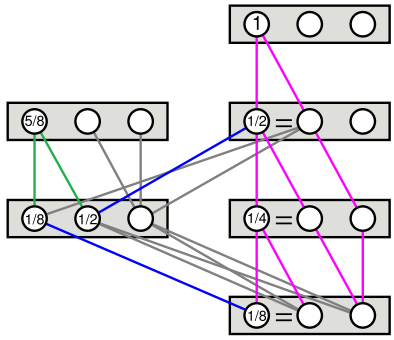

Fig. 3. Construction of the number $\frac{5}{8}$.

with costs in $\{0, \infty\}$ is well-known as the *constraint satisfaction problem* (CSP). An instance of CSP is *arc consistent* [1] if

$$\min_{\ell \in K} g_{uv}(k, \ell) = g_u(k), \qquad u \in V,\ v \in N_u,\ k \in K. \tag{14}$$

Our constructed min-sum problem is arc consistent.

Solving the LP relaxation of the problem $(V, E, K, \mathbf{g})$ decides whether $P \neq \emptyset$ and if so, it finds $\mathbf{x} \in P$. But this in fact means it solves the system $\{\, \mathbf{A}\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0} \,\}$. Thus, we have the following side-result.

Theorem 6. *Solving any system of linear inequalities reduces in linear time to the LP relaxation of an arc consistent min-sum problem with three labels and costs in* $\{0, \infty\}$.

4.3 The Complexity of Encoding

Let us show that the running time of the algorithm in Section 4.2 is linear in the size of P, i.e., in the size of the matrix (5). It is usual (see e.g. [10]) to define the description size of a matrix as the number of bits needed to encode all its entries in binary. Since an integer $a \in \mathbb{Z}$ needs at least $\log_2(|a| + 1)$ bits to encode, the number

$$L_1 = \sum_{j=1}^{n+1} \sum_{i=1}^{m} \log_2(|\overline{a}_{ij}| + 1) \tag{15}$$

is a lower bound on the size of $\overline{\mathbf{A}}$. Now it suffices to show that the running time is $\mathcal{O}(L_1)$ because then it will clearly be linear also in the true size of P.

Note that zero entries $\overline{a}_{ij} = 0$ do not contribute to L_1. Thus L_1 is a lower bound on a *sparse* representation of $\overline{\mathbf{A}}$, in which only non-zero entries are stored.

The running time of the algorithm is obviously[3] linear in $|E|$. Object pairs are created only when an object is created and the number of object pairs added with one object is bounded by a constant, hence $|E| = \mathcal{O}(|V|)$. So it suffices to show that $|V| = \mathcal{O}(L_1)$.

On initialization, the algorithm creates $\sum_{j=1}^{n}(d_j + 1)$ objects in Step 1.2 and $d + 1$ objects in Step 1.3. It is easy to verify that both these numbers are $\mathcal{O}(L_1)$. To show that $d + 1 = \mathcal{O}(L_1)$, one needs to show (referring to (10)) that $\log_2 M = \mathcal{O}(L_1)$ and $\log_2 \max_i \sum_j |a_{ij}| = \mathcal{O}(L_1)$.

For illustration, we only prove $\log_2 M = \mathcal{O}(L_1)$ and leave the rest up to the reader. For every j, we have

$$\sum_{i=1}^{m} |\overline{a}_{ij}| \leq \prod_{i=1}^{m} (|\overline{a}_{ij}| + 1)$$

because multiplying out the left-hand side yields the right-hand side plus additional non-negative terms. Taking logarithm and

3. 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.

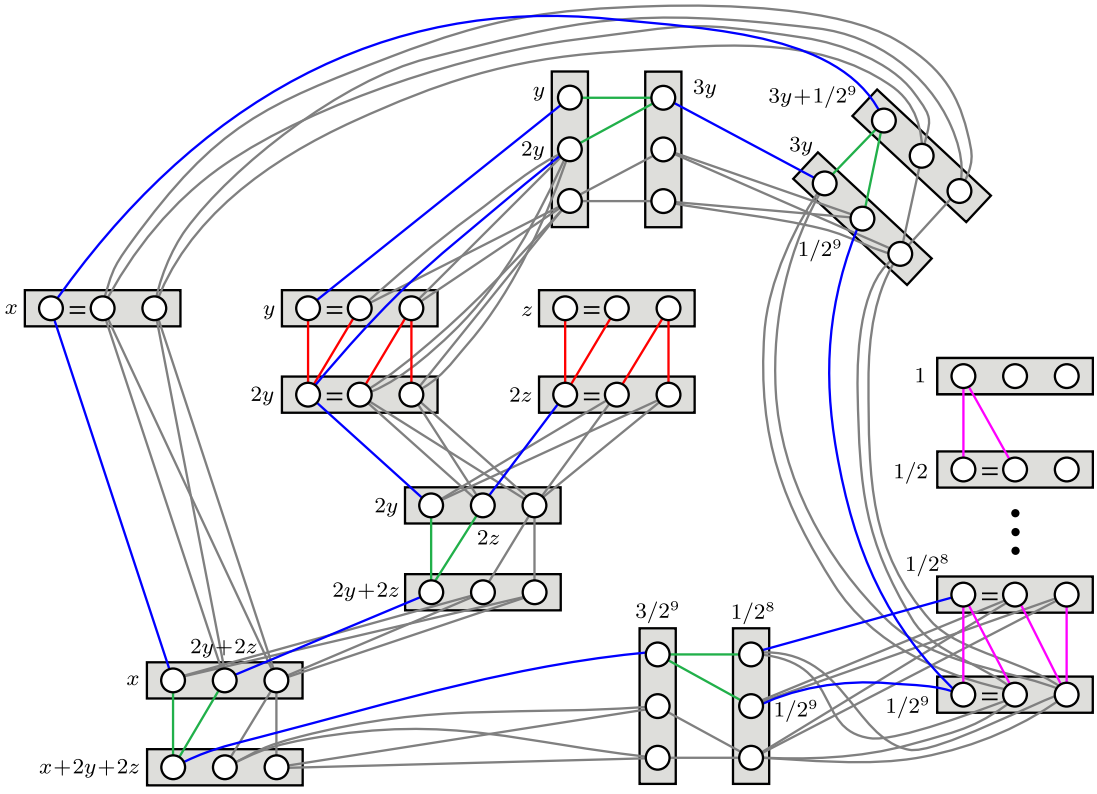

Fig. 4. The output min-sum problem for the polytope $P = \{\,(x, y, z) \,|\, x + 2y + 2z = 3;\ -x + 3y = -1;\ x, y, z \geq 0\,\}$.

summing over j yields

$$\log_2 M = \sum_{j=1}^{n+1} \log_2 \sum_{i=1}^{m} |\overline{a}_{ij}| \leq \sum_{j=1}^{n+1} \sum_{i=1}^{m} \log_2(|\overline{a}_{ij}| + 1) = L_1.$$

Finally, encoding one equality (7) adds at most as many objects as there are bits in the binary representation of all its coefficients. Thus, the number of objects added to encode all equalities (7) is $\mathcal{O}(L_1)$.

5 Encoding a Linear Program

Here we show how to reduce any linear program to linear optimization over a local marginal polytope. By saying that problem A reduces to problem B we mean there is an algorithm to solve problem A that can repeatedly[4] call an oracle for problem B (this is known as Turing reduction [15]). The complexity of the reduction is the complexity of this algorithm, assuming that the oracle for B takes constant time and space. If B is a linear program, we assume the oracle returns not only the optimal value but also an optimal argument.

We assume the input linear program in the form

$$P^*(\mathbf{c}) = \operatorname*{argmin}_{\mathbf{x} \in P} \langle \mathbf{c}, \mathbf{x} \rangle, \tag{16}$$

where $\mathbf{c} = (c_1, \ldots, c_n) \in \mathbb{Z}^n$. Since the encoding in Section 4 can be applied only to a bounded polyhedron but the LP (16) can be unbounded, we first need a lemma.

Lemma 7. *Every linear program can be reduced in linear time to a linear program over a bounded polyhedron.*

Proof. Denote $H(\alpha) = \{\,\mathbf{x} \in \mathbb{R}^n \,|\, \langle \mathbf{1}, \mathbf{x} \rangle \leq \alpha\,\}$. By Lemma 4, all vertices of P are contained in the halfspace $H(nM)$. Clearly,

$$\min_{\mathbf{x} \in P \cap H(nM)} \langle \mathbf{c}, \mathbf{x} \rangle \ \geq \ \min_{\mathbf{x} \in P \cap H(2nM)} \langle \mathbf{c}, \mathbf{x} \rangle. \tag{17}$$

Each side of (17) is a linear program over a bounded polyhedron. Inequality (17) is tight if and only if (16) is bounded, in which case (17) has the same optimum as (16). The linear programs (17) are infeasible if and only if (16) is infeasible.

The description size of numbers nM and $2nM$ is $\mathcal{O}(L_1)$, thus the reduction is done in linear time. □

By Lemma 7, we further assume that P is bounded. We also assume that $P \neq \emptyset$ because $P = \emptyset$ is indicated by $\min_{\boldsymbol{\mu} \in \Lambda} \langle \mathbf{g}', \boldsymbol{\mu} \rangle > 0$.

By Theorem 1, optimizing a linear function over P can be reduced in linear time to optimizing a linear function over a face of Λ. Given an oracle to optimize a linear function over Λ, it may seem unclear how to optimize a linear function over a *face* of Λ. This can be done by setting non-zero binary costs to a large constant.

Precisely, let $(V, E, K, \mathbf{g}')$ be the min-sum problem that encodes P, constructed in Section 4. Define $\mathbf{g} \in \overline{\mathbb{R}}^I$ by

$$g_i(k) = \begin{cases} c_i, & \text{if } k = 1 \text{ and } i \leq n, \\ 0, & \text{if } k > 1 \text{ or } i > n, \end{cases} \tag{18a}$$

$$g_{ij}(k, \ell) = \begin{cases} 0, & \text{if } g'_{ij}(k, \ell) = 0, \\ g_\infty, & \text{if } g'_{ij}(k, \ell) = 1, \end{cases} \tag{18b}$$

where the constant $g_\infty \geq 0$ is large enough to ensure that every $\boldsymbol{\mu} \in \Lambda^*(\mathbf{g})$ satisfies (11). It follows that

$$P^*(\mathbf{c}) = \pi(\Lambda^*(\mathbf{g})). \tag{19}$$

It remains to choose g_∞. The situation is different depending on whether or not we are allowed to use infinite costs. If infinite costs are allowed, we simply set $g_\infty = \infty$. This proves Theorem 2.

4. In our case, the oracle is called only twice, as given by Lemma 7.

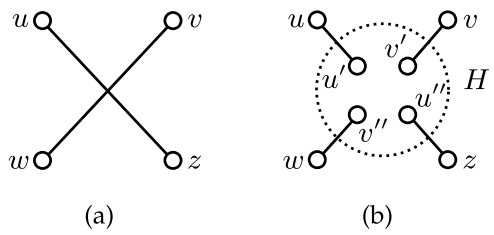

Fig. 5. Eliminating an edge crossing.

If infinite costs are not allowed, g_∞ must be large enough but finite. Unfortunately, manipulation with these large numbers increases the complexity of the reduction. This is given by Theorem 9. To prove it, we first need a lemma, which refines Lemma 4 for the special case of the local marginal polytope.

Lemma 8. *Let $\boldsymbol{\mu} \in \mathbb{R}^I$ be a vertex of the local marginal polytope defined by (V, E, K) with $|K| = 3$. Then each component μ of $\boldsymbol{\mu}$ satisfies $\mu = 0$ or $\mu \geq M_\Lambda^{-1}$ where*

$$M_\Lambda = 2^{|V|+6|E|}. \tag{20}$$

Proof. Write the local marginal polytope in the form (4), i.e., constraints (2) read $\mathbf{A}\mathbf{x} = \mathbf{b}$. Matrix $\mathbf{A}$ has $|V| + 6|E|$ rows and $3|V| + 9|E|$ columns. Each row of matrix $[\mathbf{A} \,|\, \mathbf{b}]$ has exactly 4 non-zeros, each of them in $\{-1, 1\}$. By Hadamard's inequality, in (9) we have $|\det \mathbf{A}'_j|, |\det \mathbf{A}'| \leq M_\Lambda$. □

Theorem 9. *Every linear program (16) can be reduced to a linear optimization (allowing only finite costs) over a local marginal polytope with three labels. The size of the output and the reduction time are $\mathcal{O}(L_1(L_1 + L_2))$ where L_2 is the description size of $\mathbf{c}$.*

Proof. Choose $g_\infty = 1 + M_\Lambda(C_2 - C_1)$ where

$$C_1 = \sum_{i=1}^{n} \min\{0, c_i\}, \qquad C_2 = \sum_{i=1}^{n} \max\{0, c_i\}.$$

We show that now every $\boldsymbol{\mu} \in \Lambda^*(\mathbf{g})$ satisfies (11). It suffices to show this only for vertices of $\Lambda^*(\mathbf{g})$ because taking convex combinations of vertices preserves (11).

Since $\boldsymbol{\mu} \in [0, 1]^I$, the contribution of the unary terms to $\langle \mathbf{g}, \boldsymbol{\mu} \rangle$ is in the interval $[C_1, C_2]$. Since $P \neq \emptyset$, we have $\min_{\boldsymbol{\mu} \in \Lambda} \langle \mathbf{g}', \boldsymbol{\mu} \rangle = 0$ and therefore $\min_{\boldsymbol{\mu} \in \Lambda} \langle \mathbf{g}, \boldsymbol{\mu} \rangle \leq C_2$.

Suppose there is a vertex $\boldsymbol{\mu}$ of $\Lambda^*(\mathbf{g})$ and a label pair $\{(u, k), (v, \ell)\}$ such that $g_{uv}(k, \ell) = g_\infty$ and $\mu_{uv}(k, \ell) > 0$. By Lemma 8, we have $\mu_{uv}(k, \ell) \geq M_\Lambda^{-1}$. Thus

$$\min_{\boldsymbol{\mu} \in \Lambda} \langle \mathbf{g}, \boldsymbol{\mu} \rangle \geq g_\infty M_\Lambda^{-1} + C_1 > C_2,$$

which is a contradiction.

Let us prove the claimed complexity. The binary length of g_∞ is $\mathcal{O}(L_1 + L_2)$. It occurs in $\mathbf{g}$ at $\mathcal{O}(L_1)$ positions, thus the binary length of $\mathbf{g}$ is $\mathcal{O}(L_1(L_1 + L_2))$. □

6 Reduction to Planar Min-Sum

In this section, we show that the reduction can be done even if we require the graph (V, E) of the output min-sum problem to be planar. For that, it suffices to modify the construction in Section 4.2 to ensure that (V, E) is planar.

Consider a drawing of the graph (V, E) in the plane, in which vertices are distinct points and edges are straight line segments connecting the vertices. We assume w.l.o.g. that no three edges intersect at a common point, except at graph vertices.

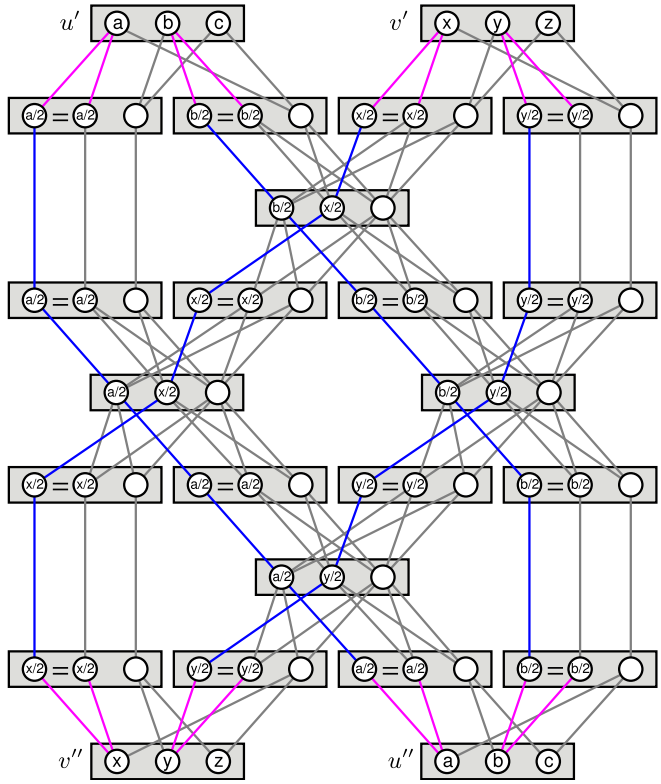

Fig. 6. Planar edge crossing using three labels.

The main idea is to replace every edge crossing with an equivalent planar min-sum problem. Consider a pair $\{u, z\}, \{v, w\} \in E$ of crossing edges, as shown in Fig. 5a. This pair is replaced by a construction in Fig. 5b. The cost functions $g_{uu'} = g_{vv'}$ copy unary pseudomarginals, i.e., they enforce $\mu_u = \mu_{u'}$ and $\mu_v = \mu_{v'}$. The other cost functions are set as $g_{u''z} = g_{uz}$ and $g_{v''w} = g_{vw}$. Problem H is a planar min-sum problem that enforces unary pseudomarginals in objects u', u'' and v', v'' to be equal, $\mu_{u'} = \mu_{u''}$ and $\mu_{v'} = \mu_{v''}$. This problem can be drawn arbitrarily small so that it is not intersected by any other edges.

Fig. 6 shows how the planar min-sum problem H can be designed. We work with halves of unary pseudomarginals, the first two from each object. The order of unary pseudomarginals is changed by swapping neighbors, imitating bubble sort on four elements.

Recall that the (non-planar) min-sum problem constructed in Section 4.2 has $E = \mathcal{O}(L_1)$ object pairs. Thus, there are $\mathcal{O}(L_1^2)$ edge crossings in this problem, which yields a reduction to a planar min-sum problem (allowing infinite costs) done in time $\mathcal{O}(L_1^2 + L_2)$.

It turns out that a more careful strategy of drawing the graph decreases the bound on edge crossings to $\mathcal{O}(mL_1)$. Before proving this in Theorem 11, we need a lemma.

Suppose we are given numbers $\alpha_1, \ldots, \alpha_p$ and sets $I_1, \ldots, I_q \subseteq \{1, \ldots, p\}$ and we want to compute numbers $\beta_j = \sum_{i \in I_j} \alpha_i$, $j = 1 \ldots, q$. The jth sum is constructed using a binary tree, T_j, in which every non-leaf vertex is the sum of its children (i.e., every non-leaf vertex with two children is Addition and every edge is Copy, as in Fig. 3). The leaves of T_j are α_i, $i \in I_j$, and its root is β_j. We refer to this construction as SumTrees.

Lemma 10. *Let SumTrees be drawn such that the leaves $\alpha_1, \ldots, \alpha_p$ lie on a common horizontal line and their positions on the line are given, and the roots $\beta_1, \ldots, \beta_q$ lie on a different horizontal line and their positions on the line can be arbitrary. Under this constraint, SumTrees can be drawn with $\mathcal{O}(q \sum_{j=1}^{q} |I_j|)$ edge crossings.*

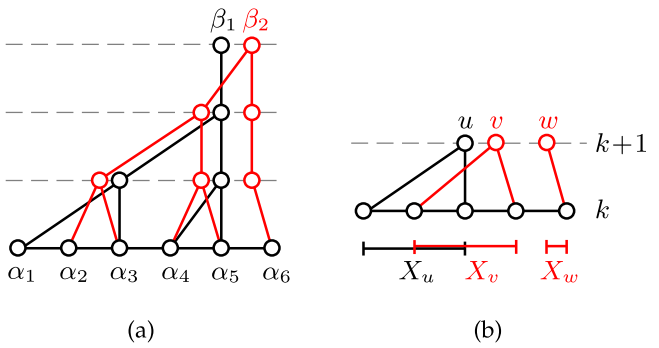

Fig. 7. (a) A drawing of SumTrees for $p=6$, $q=2$, $I_1=\{1,3,4,5\}$, $I_2=\{2,3,4,5,6\}$. (b) Crossing edges between two layers.

Proof. The construction is drawn as follows (see Fig. 7a). Each tree is drawn without edge crossings. In each tree T_j, all the leaves α_i, $i \in I_j$, have the same distance (i.e., the number of edges) to the root β_j. Let the *height* of a tree vertex be defined as its distance to the nearest leaf. The vertical coordinate of every non-root vertex is equal to its height. All the roots $\beta_1, \dots, \beta_q$ have the same vertical coordinate $h = \lceil \log_2 \max_{j=1}^{q} |I_j| \rceil$.

Let us focus on tree T_1. It is built in the bottom-up manner. All non-leaf vertices with the same height have two children except the right-most one, which can have only one child. The horizontal coordinate of a vertex equals the horizontal coordinate of its second child; if there is only one child, it equals the horizontal coordinate of this child. When a layer containing only one vertex has been drawn and its height is less than h, the vertex is linked by a single vertical edge with the layer of height h (thus, this edge can jump over several layers), where it forms the root β_1. Clearly, adding this vertical edge does not affect the overall complexity.

The trees $T_2, \dots, T_q$ are drawn similarly. The only difference is that all non-leaf vertices are shifted to the left by a small offset, to ensure that the non-leaf vertices of all the trees are distinct.

We will show that the number of edge crossings between two trees T_i and T_j is $\mathcal{O}(|I_i|+|I_j|)$. Consider all vertices with heights k and $k+1$ (see Fig. 7b). For a vertex u with height $k+1$, let $X_u \subset \mathbb{R}$ denote the smallest interval containing the horizontal coordinates of u and its children. Edges going down from u and v to height k can cross each other only if the intervals X_u and X_v intersect. Note that if u and v belong to the same tree, then X_u and X_v are disjoint.

Let $q_{i,k}$ and $q_{j,k}$ be the number of vertices with height k of T_i and T_j, respectively. The number of pairs of intersecting intervals is $\mathcal{O}(q_{i,k}+q_{j,k})$. To see this, observe that if an interval is included in another, then it appears only in one intersecting pair. If all such included intervals are discarded, each interval intersects at most two others. Thus the number of intersections is $\mathcal{O}(q_{i,k}+q_{j,k})$.

It follows that the number of edge crossings between T_i and T_j is $\mathcal{O}(|T_i|+|T_j|)$, where $|T|$ denotes the number of vertices of tree T. But we have $|T_j| = \mathcal{O}(|I_j|)$, because $q_{j,k+1} = \lceil q_{j,k}/2 \rceil$ for every j, k (recall, in every tree the highest non-root layer with a single node is linked with the root layer by a *single* edge).

The total number of crossings in the whole SumTrees graph is $\sum_{1 \le i \ne j \le q} \mathcal{O}(|I_i|+|I_j|) = \mathcal{O}(q \sum_{j=1}^{q} |I_j|)$. □

Theorem 11. *Every linear program can be reduced in $\mathcal{O}(mL_1+L_2)$ time to a linear optimization (allowing infinite costs) over a local marginal polytope with three labels over a planar graph.*

Proof. It suffices to show how to draw, in the algorithm from Section 4.2, the graph (V, E) with $\mathcal{O}(mL_1)$ edge crossings. We show this in the rest of the proof.

We start by drawing Powers for variable x_1 horizontally. Then we draw SumTrees over the objects of Powers, with roots being non-zero numbers $|a_{i1}|x_1$, $i=1,\dots,m$. The ith tree has $\mathcal{O}(\log_2(|a_{i1}|+1))$ leaves, therefore, by Lemma 10, this SumTrees construction has $\mathcal{O}(m \sum_{i=1}^{m} \log_2(|a_{i1}|+1))$ edge crossings.

This is repeated for the remaining variables $x_2, \dots x_n$, resulting in n independent SumTrees constructions. The numbers $2^{-d}b_i$, $i=1,\dots,m$, are constructed similarly, by drawing SumTrees over NegPowers. The total number of edge crossings is

$$\mathcal{O}\left(\sum_{j=1}^{n} m \sum_{i=1}^{m} \log_2(|a_{ij}|+1) + m \sum_{i=1}^{m} \log_2(|b_i|+1) \right) = \mathcal{O}(mL_1).$$

At this stage, we have objects representing all non-zero numbers $|a_{ij}|x_j$ and $2^{-d}b_i$. We assume that the vertical positions of all SumTrees were such that all these objects lie on a single horizontal line. Now we proceed to sum the terms of each side of each equality (7). This is done by drawing SumTrees over these objects, with $2m$ roots being the left-hand and right-hand sides of all equalities (7). The tree associated with any side of the ith equality (7) has $\mathcal{O}(n_i)$ leaves, where n_i is the number of non-zeros in the ith row of $\mathbf{A}$. Therefore, the number of edge crossings is $\mathcal{O}(m \sum_{i=1}^{m} n_i) = \mathcal{O}(mL_1)$.

At this stage, all objects representing both sides of all equalities (7) lie on a common horizontal line. It remains to join corresponding left- and right-hand sides using Copy. This creates $\mathcal{O}(m^2) \subseteq \mathcal{O}(mL_1)$ edge crossings. □

7 Consequences

Let us discuss some consequences of our results.

Most importantly, our results show that solving the LP relaxation of the min-sum problem is comparably hard as solving any LP. This is straightforward if infinite costs are allowed. Then, by Theorem 2, the reduction is done in time $\mathcal{O}(L)$ where $L = L_1 + L_2$, while the best known algorithm [10] for general LP has time complexity[5] $\mathcal{O}(n^{3.5}L^2 \log L \log\log L)$. Finding a very fast algorithm, such as $\mathcal{O}(L^2 \log L)$, to solve the LP relaxation would imply improving the best-known complexity of LP, which is unlikely.

The cases in which the reduction time is polynomial but higher than linear (Theorems 11 and 9) still impose a restriction on possible search for an efficient algorithm to solve the LP relaxation. There are not many principles how to solve the general LP in polynomial time (one is the ellipsoid algorithm), and finding a new such principle is expected to be difficult. Therefore, we should restrict our search to modifying these known principles rather than to discovering a new principle.

Our results make more precise the known observation that the LP relaxation of the min-sum problem is easier for two labels than for the general case. It is known that for two labels the LP relaxation reduces in linear time to max-flow [3], [17] and the local marginal polytope has half-integral vertices [11], [23]. For three labels, the coordinates of the vertices of local marginal polytopes can have much more general values, as shown in Section 4.1. Moreover, there is not much difference in complexity between the LP relaxation for three labels and for more than three labels (allowing infinite costs) because, by Theorem 2, the latter can be reduced to the former in linear time.

Rather than solving directly the LP relaxation (3), it is often more desirable to solve its dual. The dual seeks to maximize a lower bound on (1) by reparameterizations. One class of algorithms to tackle this dual LP converges only to its local minimum,

5. Note, Karmarkar [10] assumes full encoding of the LP matrix but we allow sparse encoding (see Section 4.3). To the best of our knowledge, the complexity of solving sparse LPs is largely open [16].

characterized by arc consistency. This class includes popular message passing algorithms [23, Section 6], [11], [7] and the algorithms [14], [23, Section 7], [5]. Theorem 6 has an interesting consequence. Suppose we are given a fixed point of say min-sum diffusion [23, Section 6] and want to decide whether it is (globally) optimal to the dual LP relaxation and if so, find a corresponding optimal solution to the primal LP (3). This problem is equivalent to the LP relaxation of an arc consistent min-sum problem with costs in $\{0, \infty\}$, therefore it is as hard as solving the general system of linear inequalities.

Acknowledgments

The authors were supported by the Czech Science Foundation grant P202/12/2071. Besides, Tomáš Werner was supported by the European Commission grant FP7-ICT-270138.

References

[1] C. Bessiere, "Constraint propagation," in *Handbook of Constraint Programming*. Amsterdam, The Netherlands: Elsevier, 2006, ch. 3.
[2] L. J. Billera and A. Sarangarajan, "All 0-1 polytopes are traveling salesman polytopes," *Combinatorica*, vol. 16, no. 2, pp. 175–188, 1996.
[3] E. Boros and P. L. Hammer, "Pseudo-Boolean optimization," *Discrete Appl. Math.*, vol. 123, nos. 1–3, pp. 155–225, 2002.
[4] C. Chekuri, S. Khanna, J. Naor, and L. Zosin, "Approximation algorithms for the metric labeling problem via a new linear programming formulation," in *Proc. 12th Annu. Symp. Discrete Algorithms*, 2001, pp. 109–118.
[5] M. C. Cooper, S. de Givry, M. Sanchez, T. Schiex, M. Zytnicki, and T. Werner, "Soft arc consistency revisited," *Artif. Intell.*, vol. 174, nos. 7/8, pp. 449–478, 2010.
[6] J. A. De Loera and S. Onn, "All linear and integer programs are slim 3-way transportation programs," *SIAM J. Optim.*, vol. 17, no. 3, pp. 806–821, 2006.
[7] A. Globerson and T. Jaakkola, "Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations," in *Proc. 21st Annu. Conf. Neural Inf. Process. Syst.*, 2008, pp. 553–560.
[8] J. K. Johnson, D. M. Malioutov, and A. S. Willsky, "Lagrangian relaxation for MAP estimation in graphical models," in *Proc. Allerton Conf. Commun., Control Comput.*, 2007.
[9] 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2013, pp. 1328–1335.
[10] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proc. 16th Anuu. ACM Symp. Theory Comput.*, 1984, pp. 302–311.
[11] V. Kolmogorov, "Convergent tree-reweighted message passing for energy minimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 10, pp. 1568–1583, Oct. 2006.
[12] N. Komodakis, N. Paragios, and G. Tziritas, "MRF energy minimization and beyond via dual decomposition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 3, pp. 531–552, Mar. 2011.
[13] A. Koster, S. P. van Hoesel, and A. W. Kolen, " The partial constraint satisfaction problem: Facets and lifting theorems," *Oper. Res. Lett.*, vol. 23, nos. 3–5, pp. 89–97, 1998.
[14] V. K. Koval and M. I. Schlesinger, "Dvumernoe programmirovanie v zadachakh analiza izobrazheniy (Two-dimensional programming in image analysis problems)," *USSR Acad. Sci., Autom. Telemech.*, vol. 8, pp. 149–168, 1976, in Russian.
[15] C. M. Papadimitriou, *Computational Complexity*. Reading, MA, USA: Addison-Wesley, 1994.
[16] P. M. Pardalos and S. A. Vavasis, "Open questions in complexity theory for numerical optimization," *Math. Program.*, vol. 57, pp. 337–339, 1992.
[17] C. Rother, V. Kolmogorov, V. S. Lempitsky, and M. Szummer, "Optimizing binary MRFs via extended roof duality," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2007, pp. 1–8.
[18] M. I. Shlezinger, "Syntactic analysis of two-dimensional visual signals in noisy conditions," *Cybern. Syst. Anal.*, vol. 12, no. 4, pp. 612–628, 1976.
[19] D. Sontag, A. Globerson, and T. Jaakkola, "Introduction to dual decomposition for inference," in *Optimization for Machine Learning*, Cambridge, MA, USA: MIT Press, 2011.
[20] J. Thapper and S. Živný, "The power of linear programming for valued CSPs," in *Proc. Symp. Found. Comput. Sci.*, 2012, pp. 669–678.
[21] S. Živný, *The Complexity of Valued Constraint Satisfaction Problems*. New York, NY, USA: Springer, 2012.
[22] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Found. Trends Mach. Learn.*, vol. 1, nos. 1/2, pp. 1–305, 2008.
[23] T. Werner, "A linear programming approach to max-sum problem: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 7, pp. 1165–1179, Jul. 2007.