Bounding Linear Programs by Constraint Propagation: Application to Max-SAT*

Tomáš Dlask[0000−0002−1944−6569] (✉) and Tomáš Werner[0000−0002−6161−7157]

Faculty of Electrical Engineering, Czech Technical University in Prague
{dlaskto2,werner}@fel.cvut.cz

Abstract. The Virtual Arc Consistency (VAC) algorithm by Cooper et al. is a soft local consistency technique that computes, in linear space, a bound on the basic LP relaxation of the Weighted CSP (WCSP). We generalize this technique by replacing arc consistency with a (problem-dependent) constraint propagation in a system of linear inequalities over the reals. When propagation detects infeasibility, the infeasibility certificate (a solution to the alternative system in Farkas' lemma) provides a dual improving direction. We illustrate this approach on the LP relaxation of Weighted Max-SAT. We show in experiments that the obtained bounds are often not far from global LP optima and we prove that they are exact for known tractable subclasses of Weighted Max-SAT.

Keywords: Linear programming relaxation · Constraint propagation · Weighted CSP · Virtual Arc Consistency · Weighted Max-SAT

1 Introduction

Although the linear programming (LP) problem is solvable in polynomial time, solving very large sparse linear programs can be challenging in practice. Such linear programs occur in many areas, a prominent example being the computation of bounds in branch-and-bound search by LP relaxation. To solve such LPs, the classical simplex and interior point methods may not always be suitable, if only for their worst-case space complexity which is super-linear in the number of non-zeros of the problem matrix[1]. First-order methods such as subgradient, smoothing or augmented Lagrangian methods have linear space complexity but tend to be slow (see experimental comparison [8] of methods for large-scale WCSP) and need a long time to re-converge when warm-started after a small change of the problem. This is a motivation to search for problem-specific (possibly approximate) solvers that would be more efficient than classical methods.

One such approach is known as the *primal-dual method*[2] [17], which is efficient for LP formulations of some tractable combinatorial optimization problems.

* This work has been supported by the Czech Science Foundation (grant 19-09967S), the OP VVV project CZ.02.1.01/0.0/0.0/16_019/0000765, and the Grant Agency of the Czech Technical University in Prague (grant SGS19/170/OHK3/3T/13).

[1] To the best of our knowledge, not much is known about worst-case complexity of solving sparse linear programs [18].

[2] As remarked in [17], this name is a misnomer as it is in fact a purely dual method.

Given a feasible dual solution, we consider the *restricted problem*, which minimizes infeasibility of the complementary slackness conditions. Optimal solutions of the *dual restricted problem* turn out to be dual-improving directions. The restricted problem is a linear program simpler than the original one, thus often amenable to combinatorial algorithms. Many classical algorithms for, e.g., flow and assignment problems can be seen as examples of the primal-dual method.

A similar idea has been employed in the VAC algorithm [3] (and the closely related Augmenting DAG algorithm [12, 25]), which computes an upper-bound on the basic LP relaxation of the WCSP [20, 25, 23, 19]. Strictly speaking, this is not a primal-dual method since the restricted problem is the LP relaxation of a CSP, which is a feasibility rather than optimization problem. Another difference is that the restricted problem is solved only approximately by arc consistency (AC), which not always detects infeasibility. Consequently, the method only obtains an upper bound on the LP relaxation of WCSP.

We propose a generalization of this technique. To detect infeasibility of the restricted problem, we propose to use a suitable (problem dependent) form of *constraint propagation in a system of linear inequalities*. If infeasibility is detected, a *certificate of infeasibility* (a solution to the alternative system in Farkas' lemma) is constructed, which provides a dual-improving direction. Since propagation may not always detect infeasibility, the approach yields only an upper bound on the global optimum of the LP. Note, while constraint propagation in CSP with infinite domains is well-known [2], the novelty of our approach is in using infeasibility certificates to iteratively improve the dual solution.

To illustrate the approach on a problem different than WCSP, we chose the LP relaxation of the Weighted Max-SAT problem [22]. We experimentally show that the obtained bounds are often not far from global LP optima and we prove that they are exact for known tractable subclasses of the Weighted Max-SAT.

2 Linear Optimization by Constraint Propagation

2.1 Constraint Propagation for Linear Inequalities

In the CSP, we are given a set of relations (constraints) $\phi_1, \dots, \phi_m \subseteq D^n$ and seek to find $x = (x_1, \dots, x_n) \in \phi_1 \cap \dots \cap \phi_m$ or prove that no such solution exists. A heuristic that can help achieve this is *constraint propagation*, where we iteratively generate new constraints that are implied by (i.e., inferred from) the constraint set and add them to the constraint set. By this, we make explicit some knowledge about the solution set, which before was only implicit in the constraints. As exhaustive enumeration of all implied constraints is usually impossible, only a small predefined set of simple inference (or propagation) rules is used. Since we are not doing complete inference, the procedure is refutation-incomplete: it need not infer a contradiction even if the CSP is infeasible.

Deciding feasibility, finding a solution and, more generally, deciding if the constraints imply a given relation, is usually intractable. The situation is much simpler if $D = \mathbb{R}$ (the reals) and ϕ_i's are linear inequalities. We write a linear inequality ϕ_i as $a_i^T x \leq b_i$ and the system $\phi_1, \dots, \phi_m$ as $Ax \leq b$ where $x \in \mathbb{R}^n$,

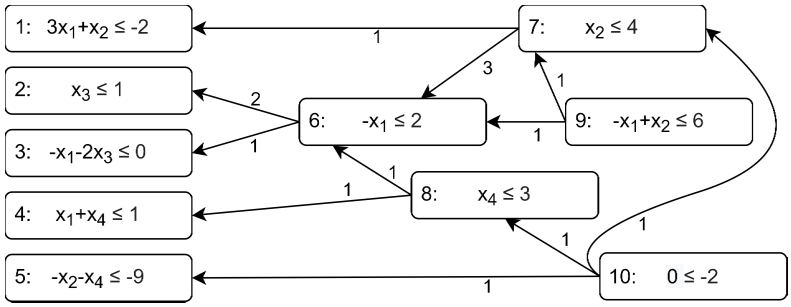

Fig. 1: Propagation in a simple system of linear inequalities. The inequalities are indexed by 1–10. Inequalities 1–5 are initial, inequalities 6–10 are inferred. Edge weights indicate the coefficients of non-negative combinations.

$A = \begin{bmatrix} a_1 \cdots a_m \end{bmatrix}^T \in \mathbb{R}^{m \times n}$ and $b = (b_1, \ldots, b_m) \in \mathbb{R}^m$. In this case, the above tasks can be solved by linear programming. In particular, the logic of linear inequalities over $\mathbb{R}$ is described by the affine form of Farkas' lemma [21, 7]:

Theorem 1. *A system $Ax \leq b$ implies an inequality $c^T x \leq d$ iff some non-negative combination of the inequalities $Ax \leq b$ implies $c^T x \leq d$, i.e., there is $y \geq 0$ such that $A^T y = c$ and $b^T y \leq d$.*

In particular (Farkas' lemma), the system $Ax \leq b$ is infeasible iff some non-negative combination of the inequalities equals $0^T x \leq d$ where $d < 0$, i.e., there is a vector $y \geq 0$ such that $A^T y = 0$ and $b^T y < 0$. The vector y can be seen as a proof (certificate, cause) for the inequality $c^T x \leq d$ resp. infeasibility.

Thus, constraint propagation for linear inequalities works as follows. Using a fixed set of inference rules (which depends on the problem solved), we generate new linear inequalities until either no new inequality can be generated or a contradiction is found. Each time a new inequality is generated, its 'cause' vector is stored, encoding how the inequality was created from the existing inequalities. When a contradiction is found, a certificate of infeasibility can be computed by tracking the newly generated inequalities back to the original system and composing the cause vectors.

2.2 Computing Certificate of Infeasibility

Let us focus on obtaining the certificate of infeasibility. As an example, consider the system of $m = 5$ initial inequalities on the left in Figure 1. From inequalities ϕ_2 and ϕ_3, we infer inequality $\phi_6 = 2\phi_2 + \phi_3$. Next, we gradually infer inequalities $\phi_7 = \phi_1 + 3\phi_6$, $\phi_8 = \phi_4 + \phi_6$, $\phi_9 = \phi_6 + \phi_7$, and finally $\phi_{10} = \phi_5 + \phi_7 + \phi_8$. Since ϕ_{10} reads $0 \leq -2$, the initial system $\phi_1, \ldots, \phi_5$ is infeasible.

The history of propagation is represented by a directed acyclic graph (DAG) $E \subseteq V \times V$ with edge weights α: $E \to \mathbb{R}_+$, where V is the set of all (initial and inferred) inequalities and each inferred inequality is given by $\phi_i = \sum_{j \in N_i} \alpha_{ij} \phi_j$ where $N_i = \{\, j \in V \mid (i, j) \in E \,\}$. By composing the inferences, each inequality

ϕ_i can be expressed in terms of the initial inequalities as $\phi_i = \sum_{j=1}^{m} y_j^i \phi_j$, where we call $y^i = (y_1^i, \ldots, y_m^i) \in \mathbb{R}^m$ the *cause vector* of ϕ_i. For $i \leq m$, we have $y^i = e^i$ where e^i is the ith standard-basis vector of $\mathbb{R}^m$. For $i > m$, we have

$$y^i = \sum_{j \in N_i \colon N_j = \emptyset} \alpha_{ij} e^j + \sum_{j \in N_i \colon N_j \neq \emptyset} \alpha_{ij} y^j. \tag{1}$$

In the example, $V = \{1, \ldots, 10\}$, $y^6 = 2e^2 + e^3 = (0, 2, 1, 0, 0)$, $y^7 = e^1 + 3y^6$, $y^8 = e^4 + y^6$, $y^9 = y^6 + y^7$, and $y^{10} = e^5 + y^7 + y^8 = (1, 8, 4, 1, 1)$. Since $b^T y^{10} = -2$ and $A^T y^{10} = 0$, vector y^{10} is a certificate of infeasibility by Theorem 1.

As we need the cause vector only for the final (contradictory) inequality (ϕ_{10} in the example), storing all cause vectors explicitly in the memory is wasteful. In addition, some inferred inequalities may not be needed for the proof of infeasibility (ϕ_9 in the example). We show that any single cause vector can be computed more efficiently by dynamic programming.

For any initial inequality ϕ_i and any derived inequality ϕ_k, y_i^k is the sum of weight-products[3] of all directed paths from node k to node i in the DAG. Suppose we want to compute y^k for some single k. We can consider only the subgraph of the DAG reachable from node k along directed paths. We introduce auxiliary variables z_j, which are to equal the sum of weight-products of all directed paths from node k to node j. Initially, we set $y^k = 0$, $z_k = 1$, and $z_j = 0$ for all $j \neq k$. Then we process the nodes i of the subgraph in a topological order as follows: if $N_i = \emptyset$ then set $y_i^k := z_i$, otherwise update $z_j := z_j + \alpha_{ij} z_i$ for all $j \in N_i$. Eventually, we have $y_i^k = z_i$ for all $i = 1, \ldots, m$. The time and space complexity of this algorithm is linear in the size of the graph.

2.3 Application to Linear Programming

Now we show how constraint propagation can be used to possibly improve a feasible dual solution of a linear program. Consider a pair of mutually dual linear programs (the primal on the left, the dual on the right)

$$c^T x \to \max \qquad b^T y \to \min \tag{2a}$$
$$Ax \leq b \qquad y \geq 0 \tag{2b}$$
$$x \lesseqgtr 0 \qquad A^T y = c \tag{2c}$$

where[4] $x \lesseqgtr 0$ denotes that the components of x can have arbitrary signs (as in [17]). By the complementary slackness theorem, a primal feasible solution x and a dual feasible solution y are simultaneously optimal iff for every i we have $a_i^T x = b_i$ or $y_i = 0$ (or both). Denoting by $I = \{\, i \mid y_i = 0 \,\}$ the set of dual constraints (2b) active at y, this condition can be written as the left-hand system

[3] The weight-product of a path is the product of all edge weights along the path.
[4] A, b in (2) denote different matrices than A, b in the previous sections.

of the pair

$$b^T\bar{y} < 0 \tag{3a}$$

$$a_i^T x \le b_i \qquad \bar{y}_i \ge 0 \qquad \forall i \in I \tag{3b}$$

$$a_i^T x = b_i \qquad \bar{y}_i \lesseqgtr 0 \qquad \forall i \notin I \tag{3c}$$

$$x \lesseqgtr 0 \qquad A^T\bar{y} = 0. \tag{3d}$$

On the right of (3), we wrote the Farkas alternative system to these conditions[5]. Thus, a point y feasible for the dual in (2) is optimal iff the left-hand system in (3) is feasible, which holds iff the right-hand system in (3) is infeasible. Moreover, any solution $\bar{y}$ to the right-hand system is an improving direction for the dual in (2), i.e., there is $\epsilon > 0$ such that $b^T(y + \epsilon\bar{y}) < b^T y$, $y + \epsilon\bar{y} \ge 0$ and $A^T(y + \epsilon\bar{y}) = c$.

The method thus proceeds as follows. Having a feasible solution y for the dual in (2), try to prove infeasibility of the left-hand system in (3) by constraint propagation and find a certificate of infeasibility $\bar{y}$, i.e., a solution to the right-hand system in (3). Then choose (by exact or approximate line search) a step size ϵ and update $y := y + \epsilon\bar{y}$. By repeating this iteration, a better and better upper bound on linear program (2) is obtained. Terminate when the propagation fails to detect infeasibility of the left-hand system in (3).

In the rest of the paper, we apply this approach to LP relaxations of WCSP and Max-SAT. These LPs will involve equality constraints and non-negative variables. Though they could be transformed to the general form (2) by well-known tricks (such as replacing an equality with two inequalities or adding slack variables), it will be more convenient to adapt the basic approach described in §2 to these cases, resulting in somewhat different and more complex algorithms.

3 LP Relaxation of Weighted CSP

In the (binary) WCSP, we are given a graph $E \subseteq \binom{V}{2}$, a finite domain D, and weights $c_\emptyset \in \mathbb{R}$, $c_{uk} \le 0$ ($u \in V$, $k \in D$) and $c_{uk,vl} \le 0$ ($uv \in E$, $k, l \in D$). We maximize

$$f_c(\lambda) = c_\emptyset + \sum_{u \in V} c_{u\lambda(u)} + \sum_{uv \in E} c_{u\lambda(u),v\lambda(v)} \tag{4}$$

over all assignments $\lambda\colon V \to D$. We abbreviated $\{u, v\}$ by uv and adopted that $c_{uk,vl} = c_{vl,uk}$. The basic LP relaxation of WCSP can be written as[6]

$$c^T x \to \max \qquad h \to \min \tag{5a}$$

$$Ax = 0 \qquad y \lesseqgtr 0 \tag{5b}$$

$$x_\emptyset = 1 \qquad h \lesseqgtr 0 \tag{5c}$$

$$x \ge 0 \qquad A^T y + e^\emptyset h \ge c \tag{5d}$$

[5] The two systems (3) correspond to a more general form of Farkas' lemma than Theorem 1, allowing for equality constraints and non-negative variables [14, §6.4].

[6] The basic LP relaxation of WCSP can be written in several different ways, see e.g. [25, 23, 19]). We chose the one that is closest to the VAC paper [3].

where the system $Ax = 0$ reads

$$\sum_{l\in D} x_{uk,vl} - x_{uk} = 0 \qquad \forall u \in V,\ v \in N_u,\ k \in D \tag{6a}$$
$$\sum_{k\in D} x_{uk} - x_{\emptyset} = 0 \qquad \forall u \in V. \tag{6b}$$

The system $A^T y + e^{\emptyset} h \geq c$ reads

$$y_u - \sum_{v\in N_u} y_{uk,v} \geq c_{uk} \qquad \forall u \in V,\ k \in D \tag{7a}$$
$$y_{uk,v} + y_{vj,u} \geq c_{uk,vl} \qquad \forall uv \in E,\ k,l \in D \tag{7b}$$
$$\sum_{u\in V} y_u + h \geq c_{\emptyset} \tag{7c}$$

where $e^{\emptyset}$ is the standard-basis vector such that $x^T e^{\emptyset} = x_{\emptyset}$.

For any y, replacing the weight vector c with the vector $c' = c - A^T y$ is an *equivalent transformation* [3] (a.k.a. a *reparameterization* [24, 19]) of the WCSP objective[7]. Indeed, $c'^T x = (c^T - y^T A)x = c^T x$ for all feasible x, hence also $f_{c'}(\lambda) = f_c(\lambda)$ for all assignments λ. A reparameterization is feasible (satisfying (7a) and (7b)) if $c'_{uk} \leq 0$ and $c'_{uk,vl} \leq 0$. After eliminating variable h, the dual thus minimizes $c'_{\emptyset}$ over feasible reparameterizations. Note that for feasible reparameterizations, $c'_{\emptyset}$ is an upper bound on the WCSP optimal value.

Given a feasible dual solution (y, h), let J denote the set of indices of dual inequalities (5d) that are active at (y, h). Then, the complementary slackness conditions read as the left-hand system of

$$\bar{h} < 0 \tag{8a}$$
$$Ax = 0 \qquad \bar{y} \lesseqgtr 0 \tag{8b}$$
$$x_{\emptyset} = 1 \qquad \bar{h} \lesseqgtr 0 \tag{8c}$$
$$x_j \geq 0 \qquad A_j^T \bar{y} + \bar{h} e_j^{\emptyset} \geq 0 \qquad \forall j \in J \tag{8d}$$
$$x_j = 0 \qquad \forall j \notin J \tag{8e}$$

where A_j denotes the jth column of A and $e_j^{\emptyset} = [\![j = \emptyset]\!]$ is[8] the jth component of $e^{\emptyset}$. On the right, we wrote the Farkas alternative system.

The left-hand system in (8) is the LP relaxation of the CSP instance formed by the active tuples of the reparameterized WCSP instance. The WCSP is *virtually arc-consistent (VAC)* if this CSP has a non-empty AC closure [3]. Indeed, propagating zero values of the primal variables x_j in (8) using the marginalization constraint (6a) is equivalent to the AC algorithm in this CSP.

When propagation detects a contradiction, we construct a certificate $(\bar{y}, \bar{h})$ satisfying the right-hand system of (8). If a variable x_j is inferred to be zero, we store a cause vector y^j of the coefficients of linear combination $x_j + t_j = 0$ of the primal constraints (6), where t_j is a non-negative combination of x_i, $i \in J$, and may contain x_i, $i \notin J$, with arbitrary sign. Under constraints (8e)+(8d), t_j is non-negative, therefore $x_j + t_j = 0$ implies $x_j = 0$. For $j \notin J$, we initialize

[7] Note, c' is 'almost' (up to variable h) the reduced cost vector of the primal of (5).
[8] The symbol $[\![\cdot]\!]$ denotes the Iverson bracket.

$y^j = 0$. All vectors y^j have the same dimension, equal to the number of primal constraints (6). We denote the standard-basis vector of this space corresponding to (6a) (resp. (6b)) by $e^{uk,v}$ (resp. e^u).

We now describe the propagation rules in detail, including the cause vectors y^j. If $j \in J$, each y^j will represent an equality $x_j + t_j = 0$. If $j \notin J$, y^j is initialized to zero and thus represent equality $0 = 0$; in this case the excess variable $-x_j$ on the left-hand side will be included into t_i on the right-hand side, which is allowed by the definition of t_i. The propagation rules are as follows:

- If $x_{uk} = 0$ for some $u \in V$ and $k \in D$, constraints (6a) imply $x_{uk,vl} = 0$ for all $v \in N_u$ and $l \in D$. Inference in terms of equalities:

$$\big(x_{uk} + t_{uk} = 0\big) + \big(\sum_{l \in D} x_{uk,vl} - x_{uk} = 0\big) = \big(\sum_{l \in D} x_{uk,vl} + t_{uk} = 0\big)$$

  The cause vectors are given by $y^{uk,vl} = y^{uk} + e^{uk,v}$, for $v \in N_u$ and $l \in D$.
- If for some $uv \in E$ and $k \in D$ we have $x_{uk,vl} = 0$ for all $l \in D$, constraint (6a) implies $x_{uk} = 0$. Inference in terms of equalities:

$$\sum_{l \in D} \big(x_{uk,vl} + t_{uk,vl} = 0\big) - \big(\sum_{l \in D} x_{uk,vl} - x_{uk} = 0\big) = \big(x_{uk} + \sum_{l \in D} t_{uk,vl} = 0\big)$$

  The cause vector is given by $y^{uk} = \sum_{l \in D} y^{uk,vl} - e^{uk,v}$.
- If for some $u \in V$ we have $x_{uk} = 0$ for all $k \in D$, constraint (6b) implies a contradiction (domain wipe-out). Inference in terms of equalities:

$$\sum_{k \in D} \big(x_{uk} + t_{uk} = 0\big) - \big(\sum_{l \in D} x_{ul} - x_\emptyset = 0\big) - \big(x_\emptyset = 1\big) = \big(\sum_{k \in D} t_{uk} = -1\big)$$

  The certificate of infeasibility is given by $\bar{y} = \sum_{k \in D} y^{uk} - e^u$, $\bar{h} = -1$.

By properties of t_j and the fact that coefficients $(\bar{y}, \bar{h})$ encode an equality in the form $\sum_j t_j = -1$, it is not hard to show that $(\bar{y}, \bar{h})$ are feasible for the right-hand side (8) and therefore constitute an improving direction for the dual in (5) from the current point (y, h).

The described algorithm is 'almost' equivalent to the VAC / Augmenting DAG algorithm [3, 12, 25]. The fixed points of both algorithms are characterized by the same property, namely VAC. However, our improving directions $\bar{y}$ are in general different from the ones in [3, 12, 25], sometimes having larger absolute values of their components (and thus allowing smaller step size ϵ). It is subject to further research to clarify the relation between them.

4 LP Relaxation of Weighted Max-SAT

In the Weighted Max-SAT problem, we are given a set V of variables and a set C of clauses with positive weights w: $C \to \mathbb{R}_{++}$ and we seek to maximize the weighted sum of satisfied clauses. Let V_c^+ (resp. V_c^-) denote the set of variables that occur in clause $c \in C$ non-negated (resp. negated). Let

$C_i^{\pm} = \{\, c \in C \mid i \in V_i^{\pm} \,\}$ denote the set of clauses where variable $i \in V$ occurs non-negated/negated. We denote $n_c = |V_c^-|$. For any $S \subseteq V$, we will use the shortcut $x(S) = \sum_{i \in S} x_i$, similarly for y. The classical LP relaxation of Weighted Max-SAT [22] reads

$$
\begin{aligned}
&w^T z \to \max && && && \text{(9a)}\\
&z_c \le x(V_c^+) + n_c - x(V_c^-) && y_c \ge 0 && \forall c \in C && \text{(9b)}\\
&x_i \ge 0 && p_i - y(C_i^+) + y(C_i^-) \ge 0 && \forall i \in V && \text{(9c)}\\
&x_i \le 1 && p_i \ge 0 && \forall i \in V && \text{(9d)}\\
&z_c \ge 0 && q_c + y_c \ge w_c && \forall c \in C && \text{(9e)}\\
&z_c \le 1 && q_c \ge 0 && \forall c \in C && \text{(9f)}
\end{aligned}
$$

where we wrote also the dual LP on the right. The primal variables x_i represent the (relaxed) original Boolean variables. Clearly, at dual optimum we have

$$
\begin{aligned}
p_i &= \max\{y(C_i^+) - y(C_i^-), 0\} && \forall i \in V && \text{(10a)}\\
q_c &= \max\{w_c - y_c, 0\} && \forall c \in C. && \text{(10b)}
\end{aligned}
$$

Substituting (10) into the dual objective together with $n^T y = \sum_{i \in V} y(C_i^-)$ results in a simpler form of the dual,

$$
\min_{y \ge 0} \sum_{c \in C} \max\{w_c - y_c, 0\} + \sum_{i \in V} \max\{y(C_i^+), y(C_i^-)\} \tag{11}
$$

which minimizes a convex piecewise-affine function of non-negative variables.

Theorem 2. *Point $y \in \mathbb{R}_+^C$ is optimal for* (11) *iff there exists $x \in \mathbb{R}^V$ satisfying the left-hand system of*

$$
\begin{aligned}
x(V_c^+) + n_c - x(V_c^-) &\ge 1 && \bar{y}_c \ge 0 && \forall c \in C^{\ge 1} && \text{(12a)}\\
x(V_c^+) + n_c - x(V_c^-) &= 1 && \bar{y}_c \lesseqgtr 0 && \forall c \in C^{=1} && \text{(12b)}\\
x(V_c^+) + n_c - x(V_c^-) &\le 1 && \bar{y}_c \le 0 && \forall c \in C^{\le 1} && \text{(12c)}\\
x(V_c^+) + n_c - x(V_c^-) &= 0 && \bar{y}_c \lesseqgtr 0 && \forall c \in C^{=0} && \text{(12d)}\\
x_i &= 1 && \bar{p}_i \lesseqgtr 0 && \forall i \in X^1 && \text{(12e)}\\
x_i &= 0 && \bar{p}_i \lesseqgtr 0 && \forall i \in X^0 && \text{(12f)}\\
x_i &\lesseqgtr 0 && \bar{p}_i + \bar{y}(C_i^+) - \bar{y}(C_i^-) = 0 && \forall i \in X^0 \cup X^1 && \text{(12g)}\\
x_i &\ge 0 && \bar{p}_i + \bar{y}(C_i^+) - \bar{y}(C_i^-) \le 0 && \forall i \in X^U && \text{(12h)}\\
x_i &\le 1 && \bar{p}_i \le 0 && \forall i \in X^U && \text{(12i)}
\end{aligned}
$$

where

$$
\begin{aligned}
X^0 &= \{\, i \in V \mid y(C_i^+) < y(C_i^-) \,\} && C^{\ge 1} = \{\, c \in C \mid y_c = 0 \,\} && \text{(13a)}\\
X^1 &= \{\, i \in V \mid y(C_i^+) > y(C_i^-) \,\} && C^{=1} = \{\, c \in C \mid 0 < y_c < w_c \,\} && \text{(13b)}\\
X^U &= \{\, i \in V \mid y(C_i^+) = y(C_i^-) \,\} && C^{\le 1} = \{\, c \in C \mid y_c = w_c \,\} && \text{(13c)}\\
& && C^{=0} = \{\, c \in C \mid y_c > w_c \,\} && \text{(13d)}
\end{aligned}
$$

are partitions of V and C. If the left-hand system is infeasible, then there exist values $(\bar{p}, \bar{y})$ for the right-hand system[9] (12) *such that*

$$\sum_{c \in C} (\llbracket c \notin C^{=0} \rrbracket - n_c)\bar{y}_c + \bar{p}(X^1 \cup X^U) > 0, \tag{14}$$

where $\bar{y}$ is an improving direction for (11) *from y.*

Proof. The left-hand system in (12) is the complementary slackness condition for the primal-dual pair (9), where we used (10) and eliminated variables z_c.

The right-hand system of (12) and (14) form the alternative system to the left-hand system of (12) in Farkas' lemma. To show that $\bar{y}$ is improving for (11), realize that $\bar{p}$ can w.l.o.g. satisfy $\bar{p}_i = \bar{y}(C_i^-) - \bar{y}(C_i^+)$ for all $i \in X^0 \cup X^1$ and $\bar{p}_i = \min\{\bar{y}(C_i^-) - \bar{y}(C_i^+), 0\}$ for all $i \in X^U$. Then, (14) can be reformulated (after multiplying by -1 and substituting $-\bar{p}_i$ terms) as

$$-\bar{y}(C - C^{=0}) + \sum_{i \in X^U} \max\{\bar{y}(C_i^+), \bar{y}(C_i^-)\} + \sum_{i \in X^1} \bar{y}(C_i^+) + \sum_{i \in X^0} \bar{y}(C_i^-) < 0,$$

which states that (11) decreases in terms of the affine functions that are active[10] in the current point y, as defined by the sets (13). □

We now define propagation rules for the left-hand system (12). These rules set the values of some of the undecided variables x_i, $i \in X^U$, to 0 or 1. Precisely, we iteratively visit each constraint (12a)-(12d) and look whether with the already decided variables it permits only a single value of some so-far undecided variable. If so, we fix the value of this variable (i.e., make it decided). If some constraint (12a)-(12d) cannot be satisfied by any assignment subject to the already decided variables, the left-hand system in (12) is infeasible. During propagation, we update the dual variables of (12), so that if infeasibility is detected, we are able to construct an improving direction $\bar{y}$ for (11).

We now need a technical definition. For $j \in V$, we call a cause vector (p^j, y^j)

- *1-j-defining* if it satisfies the right-hand system in (12), except for $i = j$ when it satisfies $p_j^j + y^j(C_j^+) - y^j(C_j^-) = 1$ and left-hand side of (14) equals 1. This cause vector defines an inequality $x_j + t_j \geq 1$ derived from the left-hand system in (12), where t_j is a non-positive weighted sum of x_i, $i \in X^U$. Clearly, this inequality implies $x_j = 1$.
- *0-j-defining* if it satisfies the right-hand system in (12) and $p_j^j + y^j(C_j^+) - y^j(C_j^-) = -1$, and the left-hand side of (14) equals 0. This cause vector defines an inequality $-x_j + t_j \geq 0$, which implies $x_j = 0$.

Note that the sign constraints on the right-hand side (12) ensure that the inequalities on left-hand side (12) are combined in correct directions.

[9] Note, the right-hand system (12) has opposite direction of inequalities. This is due to writing left-hand inequalities (12a)-(12d) with opposite directions than in (9b).

[10] For $f(x) = \max\{a^T x, b^T x\}$ and a fixed x, we say that the function $a^T x$ is active and $b^T x$ is inactive at x if $a^T x > b^T x$. If $a^T x = b^T x$, both functions are active at x.

Constraint		Rule
$C^{\geq 1} \cup C^{=1}$	A1	If there is only one undecided variable x_k, $k \in V_c$, and $x_i^c = 0$ for all other $i \in V_c - \{k\}$, we set $x_k = [\![k \in V_c^+]\!]$ and $y^k = e^c + \sum_{i \in V_c - \{k\}} y^i$.
	A2	If all the variables x_i for $i \in V_c$ are decided and satisfy $x_i^c = 0$, then we obtain a contradiction and set $\bar{y} = e^c + \sum_{i \in V_c} y^i$.
$C^{=1} \cup C^{\leq 1}$	B1	If there is exactly one decided variable $x_i, i \in V_c$, with $x_i^c = 1$, then we set $x_k = [\![k \in V_c^-]\!]$ and $y^k = -e^c + y^i$ for each undecided $x_k, k \in V_c$.
	B2	If there are two (or more) decided variables x_i, x_j for $i, j \in V_c$ with $x_i^c = x_j^c = 1$, then we obtain a contradiction and set $\bar{y} = -e^c + y^i + y^j$.
$C^{=0}$	C1	If there is no decided variable $x_i, i \in V_c$, with $x_i^c = 1$, then set all undecided variables $x_k, k \in V_c$, as $x_k = [\![k \in V_c^-]\!]$ and $y^k = -e^c$.
	C2	If there is a decided variable $x_i, i \in V_c$, with $x_i^c = 1$, then we obtain a contradiction and set $\bar{y} = -e^c + y^i$.

Table 1: Propagation rules for system (12). The first column determines the type of constraints to which the rule applies.

As mentioned in Theorem 2, it is sufficient to store vectors $y^j \in \mathbb{R}^C$ because these will be used to construct the improving direction $\bar{y}$ if a contradiction is detected. Thus, for each decided variable x_j that is set to a value $v \in \{0, 1\}$, we can store only the y^j component of a v-j-defining vector (p^j, y^j).

The propagation rules are listed in Table 1, divided into groups according to which set (13) clause c belongs. For each rule, we also specify how to construct the cause vector y^i for each inferred variable x_i. For $i \in X^1 \cup X^0$, we define $y^i = 0$ so that it can be referred to in the equations for creation of other y^j or $\bar{y}$. To simplify the explanation of the rules, for any $i \in V$ and $c \in C$ we denote

$$x_i^c = \begin{cases} x_i, & \text{if } i \in V_c^+, \\ 1 - x_i, & \text{if } i \in V_c^-. \end{cases} \tag{15}$$

We denote $e^c \in \mathbb{R}^C$ to be the standard-basis vector with 1 in the place corresponding to clause c. We also define $V_c = V_c^+ \cup V_c^-$.

The derivation of the updates for cause vectors y^i is technical and must be done for each rule separately. The proof relies on the fact that for each initially decided variable x_j, $j \in X^1$, (resp. $j \in X^0$), we can initialize a 1-j-defining (resp. 0-j-defining) cause vector (p^j, y^j) as $(e^j, 0)$ (resp. $(-e^j, 0)$) where $e^j \in \mathbb{R}^V$ is a standard-basis vector. This corresponds to setting $y^j = 0$ for the initially decided variables. Then it is possible to show how to derive a v-k-defining vector for a newly decided variable x_k from the previous ones. We are going to show this in detail for rule B1, which we believe is most complicated.

Theorem 3. *Let $c \in C^{=1} \cup C^{\leq 1}$ such that there is exactly one decided variable x_i, $i \in V_c$, with $x_i^c = 1$ and $x_k, k \in V_c$, is an undecided variable. Let x_i be decided to a value $v \in \{0, 1\}$, and let (p^i, y^i) be v-i-defining. Then there exists a vector p^k such that $(p^k, -e^c + y^i)$ is $[\![k \in V_c^-]\!]$-k-defining.*

Proof. First of all, see that $x(V_c^+) + n_c - x(V_c^-) = \sum_{j \in V_c} x_j^c$ by definition (15). The inequality encoded by the v-i-defining cause vector (p^i, y^i) can be compactly rewritten as $x_j^c + t_j \geq 1$ and analogously, each defining equality for $j \in V_c \cap (X^0 \cup X^1)$ with $x_j^c = 0$ can be expressed as $-x_j^c = 0$, hence $x_j^c \geq 0$. Then, the derivation of the defining inequality for undecided variable x_k is given as follows:

$$
\begin{array}{rll}
-\sum\limits_{j \in V_c} x_j^c \geq -1 & & (0, -e^c) \quad (16a) \\
x_i^c + t_i \geq 1 & & (p^i, y^i) \quad (16b) \\
x_j^c \geq 0 & \forall j \in (V_c \cap X^0) - \{i\} & (e^j, 0) \quad (16c) \\
x_j^c \geq 0 & \forall j \in (V_c \cap X^1) - \{i\} & (-e^j, 0) \quad (16d) \\
x_j^c \geq 0 & \forall j \in (V_c^- \cap X^U) - \{i, k\} & (-e^j, 0) \quad (16e) \\
\hline
-x_k^c + t_k \geq 0 & & (p^k, y^k) \quad (16f)
\end{array}
$$

where

$$t_k = - \sum_{j \in (X^U \cap V_c^+) - \{i,k\}} x_j + t_i. \quad (17)$$

Inequality (16a) is (12b) or (12c) multiplied by -1, (16b), (16c), and (16d) are inequalities determining the values of already decided variables, and (16e) is (12i). Inequality (16f) is given as the sum of the inequalities above it. Each row in (16) is marked on the right by the coefficients (p, y) with which it was derived from the original system (12). It is easy to check that the sign constraints in the right-hand system (12) are satisfied for each pair (p, y) in (16).

The coefficients for the last row are determined by summing the above coefficients, i.e., $y^k = -e^c + y^i$ (which is the same equation as in Table 1) and $p^k = p^i + \sum_{j \in J'} e^j - \sum_{j \in J''} e^j$ where J' (resp. J'') is set used on line (16c) (resp. union of the sets on lines (16d) and (16e)). To show that the vector (p^k, y^k) is $[\![k \in V_c^-]\!]$-k-defining, substitute the definition (15) into (16f) and see that t_k is again a non-positive combination of other variables from X^U as this held for t_i by the assumption of the theorem. □

Each propagation rule can be formulated in a general form similar to (16) that defines vectors (y^k, p^k) as described in Table 1. Using these vectors, a contradiction defined by $(\bar{p}, \bar{y})$ encodes an inequality $\bar{t} \geq 1$ where $\bar{t}$ is a non-positive sum of $x_i, i \in X^U$, and it is possible to show that the pair $(\bar{p}, \bar{y})$ satisfies conditions of Theorem 2.

Remark 1. One can ask whether it is possible to infer other values of undecided variables than 0 or 1 (such as $\frac{1}{2}$). Assuming that inference is done only from a single constraint from (12a)-(12d), this is impossible because the polyhedron defined by a single (in)equality from (12a)-(12d) subject to $0 \leq x_i \leq 1$, where some of the variables may be already set to 0 or 1, has integral vertices.

Remark 2. For general Max-SAT, the propagation rules in Table 1 do not always prove infeasibility of the left-hand system in (12). However, for Weighted

Max-2SAT they do: if no more propagation is possible and no contradiction is detected, setting all undecided variables x_i to $\frac{1}{2}$ satisfies all constraints of (12).

Remark 3. The restricted system (12) is the LP relaxation of a CSP with Boolean variables. The propagation corresponds to enforcing arc consistency of this CSP. The whole algorithm seeks to find a feasible dual solution of the LP relaxation of Max-SAT that enforces this CSP to have a non-empty AC closure. Compare this with the WCSP case, where the restricted system (8) is the LP relaxation of the CSP formed by the active tuples and the VAC algorithm seeks to find an equivalent transformation (a linear tranformation of the weight vector that preserves the objective function) that makes this CSP arc-consistent. Note that, in contrast to WCSP, there is no obvious analogy of equivalent transformations for Weighted Max-SAT.

4.1 Finding Step Size by Approximate Line Search

If a contradiction is detected in (12) and improving direction $\bar{y}$ at the point y is constructed, we need to find a feasible step size $\epsilon > 0$, as mentioned in §2.3. The optimal way (exact line search) would be to minimize $f(y+\epsilon\bar{y})$ over $\epsilon > 0$ subject to $y + \epsilon\bar{y} \geq 0$, where f is the objective of (11). Since this is too costly for large instances, we do only approximate line search: we find the first breakpoint of the univariate convex piece-wise affine function $\epsilon \mapsto f(y + \epsilon\bar{y})$, i.e., the smallest ϵ at which at least one previously inactive affine function becomes active. This ensures a non-zero improvement of f. Such ϵ is the maximum number satisfying the following constraints:

- To stay within the feasible set, we need $y_c + \epsilon\bar{y} \geq 0$, therefore $\epsilon \leq -y_c/\bar{y}_c$ for all $c \in C - C^{\geq 1}$ with $\bar{y}_c < 0$.
- For terms $\max\{w_c - y_c, 0\}$, if $w_c - y_c > 0$ (resp. $w_c - y_c < 0$) and $w_c - y_c - \epsilon\bar{y}_c$ decreases (resp. increases), then we need $\epsilon \leq (w_c - y_c)/\bar{y}_c$ where the bound is the point where the terms equalize. This is for all $c \in C$ such that $(w_c - y_c)\bar{y}_c > 0$.
- For terms $\max\{y(C_i^+), y(C_i^-)\}$, if $y(C_i^+) > y(C_i^-)$ and $\bar{y}(C_i^+) < \bar{y}(C_i^-)$ (resp. with inverted inequalities), we need $\epsilon \leq (y(C_i^+) - y(C_i^-))/(\bar{y}(C_i^-) - \bar{y}(C_i^+))$ where the bound is the point where the terms equal. This is for all $i \in V$ with $(y(C_i^+) - y(C_i^-))(\bar{y}(C_i^-) - \bar{y}(C_i^+)) > 0$.

Using the conditions on $\bar{y}$ determined by Theorem 2, it can be shown that there always exists $\epsilon > 0$ satisfying these bounds.

4.2 Algorithm Overview

Let us summarize the algorithm. We start with $y = 0$ (which is dual-feasible) and repeat the following iteration: From the current y, construct system (12). Apply rules in Table 1 to fix values of undecided variables. During that, construct the DAG defining each y^i until no rule is applicable or contradiction is detected. If no

contradiction is detected, stop. If contradiction is detected, compute $\bar{y}$ from the DAG, similarly as in §2.2. Calculate step size ϵ as in §4.1 and update $y := y + \epsilon\bar{y}$.

To speed up the algorithm and facilitate convergence[11], we redefine sets (13) up to a tollerance $\delta > 0$, by replacing $y_c > 0$ with $y_c > \delta$, $y(C_i^+) < y(C_i^-)$ with $y(C_i^+) + \delta < y(C_i^-)$, etc. We start with some large value of δ. When the algorithm achieves a fixed point, we keep the current y and set $\delta := \delta/10$. We continue until δ is not very small (10^{-6}).

All data structures used by the algorithm need space that is linear in the input size, i.e., in the number $\sum_{c\in C} |V_c|$ of non-zeros in linear program (9). In particular, it can be shown that the DAG (used to calculate $\bar{y}$) can be conveniently stored as a subgraph of the bipartite clause-variable incidence graph.

4.3 Results

We compared the upper bound on the optimal value of (9) obtained by our algorithm with the exact optimal value of (9) obtained by an off-the-shelf LP solver (we used Gurobi with default parameters) on the Max-SAT Evaluations 2018 benchmark [1]. This benchmark contains 2591 instances of Weighted Max-SAT. Gurobi was able to optimize (without memory overflow) the smallest 2100 instances, the largest of which had up to 600 thousand clauses, 300 thousand variables and 1.6 million non-zeros. The largest instances in the benchmark have up to 27 million clauses, 19 million variables and 77 million non-zeros and was still manageable by our algorithm.

From the smallest 2100 instances, 154 instances were Max-2SAT and 91 instances did not contain any unit clause. As discussed in Remark 2, the algorithm attained the exact optimum of the LP on instances of Max-2SAT. Similarly, if an instance does not contain any unit clause, then setting $x_i = \frac{1}{2}$ for all $i \in V$ yields an optimal solution of (9) with objective value $w(C)$. The algorithm also attains optimality on these instances because $y = 0$ is already optimal for the dual. These instances are excluded from the evaluation.

Each of the remaining 1855 instances contains a clause of length at least 3 and also contain a unit clause, thus the bound is not guaranteed to be optimal. We measure the quality of the bound by the criterion $R = (U - U^*)/(w(C) - U^*)$ where U^* is the globally optimal value of (9) and U is the upper bound computed by our algorithm. This criterion is invariant to scaling the weights and shows how tight the bound is relative to the trivial bound $w(C)$.

The sorted numbers R for the selected 1855 instances are plotted in Figure 2. For 802 instances the bound was tight ($U = U^*$). Due to this, the vertical (logarithmic) axis in the right-hand plot is trimmed, starting from 10^{-20}. The left-hand plot shows that the obtained upper bound is informative in at least 1000-1100 cases. In fact, R was higher than 0.6 only in 35 instances.

[11] Though we do not present any convergence analysis of our method, it is known that the VAC / Augmenting DAG algorithm with $\delta = 0$ can converge to a point that does not satisfy virtual arc consistency [12, 26, 3].

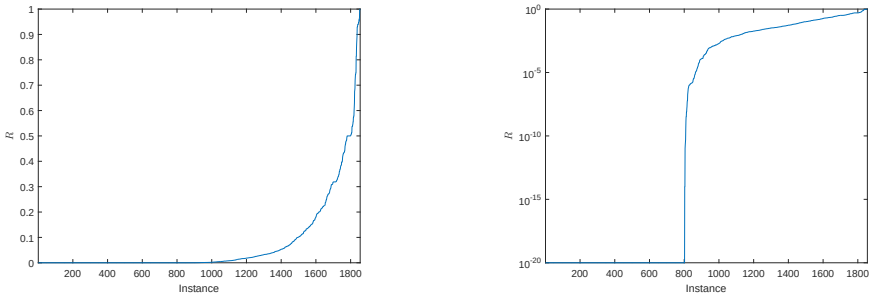

Fig. 2: Sorted values of R with linear (left) and logarithmic (right) scale.

We computed also another criterion $Q = (U - U^*)/U^*$, which was lower than 10^{-6} (resp. 10^{-8}) on 1644 (resp. 1308) from the 1855 instances. Overall, Q was always lower than 0.029.

For 152 out of the 2100 considered instances, the integrality gap of the LP relaxation is known to be tight. In 133 of them, our algorithm attained this optimum. Only 2 of these were Max-2SAT and each contained a unit clause, so optimality was not guaranteed trivially.

An unoptimized implementation of our algorithm was on average 3.3 times faster than Gurobi. We believe a significant speed-up could be achieved by warm-starting. The part of the DAG needed to explain the found contradiction (see §2.2) is usually very small. If the DAG is built in every iteration from scratch, most of it is therefore thrown away. Since the system (12) changes only slightly between consecutive updates, it makes sense to re-use a part of the DAG in the next iteration. Such warm-starting was presented for the VAC algorithm in [16] and for the Augmenting DAG algorithm in [26] with significant speed-ups.

4.4 Tightness of the Bound on Tractable Classes

We show that the constraint propagation in system (12) is refutation-complete for tractable subclasses of Weighted Max-SAT that either use tractable clause types (language) or have acyclic structure (clause-variable incidence graph). For these instances, integrality gap of the LP relaxation (9) is zero and all fixed points of our algorithm are the optima of the unrelaxed problem.

It was shown in [9, Theorem 1] that a subclass of generalized Max-SAT (i.e., Max-CSP with Boolean variables) defined by restricting constraint types (language) is tractable if and only if one of the following holds:

- All constraints are 0-valid (resp. all are 1-valid). In this case, if the constraints are given as clauses (i.e., we restrict ourselves to the ordinary Weighted Max-SAT), the optimal value is $w(C)$, which coincides with the optimum of the LP and our algorithm attains this optimum already at $y = 0$.
- All constraints are 2-monotone. Again, restricting these constraints to clauses results in clauses with at most two literals where at most one of them is

positive (resp. negative). In this case, Max-SAT can be reduced to minimum st-cut problem [9, Lemma 3] and the optimum of its LP formulation equals (up to a trivial recalculation) the optimum of the LP relaxation of Max-SAT which is thus tight. Since this is an instance of Weighted Max-2SAT, by Remark 2 all fixed points of our algorithm are the optima of the LP.

If we view (12) as the LP relaxation of a CSP with Boolean variables, then the propagation rules in Table 1 enforce arc consistency of this CSP. If the factor graph of this CSP is acyclic, arc consistency solves this CSP exactly [5, Theorem 1]. Hence, if the clause-variable incidence graph is acyclic, our constraint propagation rules are refutation-complete and the fixed points of our algorithm are optimal. Additionally, if no contradiction is detected, an integral solution to the left-hand system (12) can be constructed, so the integrality gap is zero.

5 Conclusion

We have proposed a technique to compute, with small space complexity, bounds on certain large sparse linear programs with suitable structure. Having a feasible dual solution, infeasibility of the complementary slackness conditions (a system of linear inequalities) is detected by constraint propagation and the infeasibility certificate is recovered, providing a dual improving direction. This technique can be seen as a generalization of the VAC algorithm [3] for WCSP. We have newly applied it to the LP relaxation of the Weighted Max-SAT.

The main purpose of soft local consistencies in WCSP, such as FDAC, EDAC, VAC and OSAC [3], is to bound the optimal value of WCSP during search. Each local consistency has a different trade-off point between bound tightness and computational complexity. In this view, our approach can be seen as a soft local consistency technique for other problems than WCSP. It is open whether the trade-off point of our method for Max-SAT will allow designing better algorithms to compute exact or approximate solutions of the unrelaxed Max-SAT problem.

Though in principle our approach can also be applied to other LPs (if an initial dual-feasible solution is available), the existence of good propagation rules and the quality of obtained bounds critically depends on the problem structure in a so-far unknown way. It is yet to be seen if there are other such 'propagation friendly' classes of LPs beyond the LP relaxation of WCSP and Max-SAT.

In comparison with the first-order optimization methods (such as subgradient methods or ADMM, see [8]), our approach may have the advantage that it reconverges faster after a small change of the problem instance. Though this claim would need more experimental support, evidence can be found in [15] for the VAC algorithm and in [11] for the Augmenting DAG algorithm.

VAC in WCSP is closely related to convergent message-passing methods developed within computer vision, such as [13, 25, 10, 6]. Their fixed points are also characterized by a local consistency (usually AC in disguise) and they can be seen as versions of block-coordinate descent applied to the dual LP relaxation. This suggests there is a close connection between our approach and block-coordinate descent methods. We clarify this connection in [4].

References

1. Bacchus, F., Järvisalo, M., Martins, R.: MaxSAT Evaluation 2018: New developments and detailed results. Journal on Satisfiability, Boolean Modeling and Computation **11**(1), 99–131 (2019), instances available at https://maxsat-evaluations.github.io/.
2. Benhamou, F., Granvilliers, L.: Continuous and interval constraints. In: Handbook of Constraint Programming, chap. 16. Elsevier (2006)
3. Cooper, M.C., de Givry, S., Sanchez, M., Schiex, T., Zytnicki, M., Werner, T.: Soft arc consistency revisited. Artificial Intelligence **174**(7-8), 449–478 (2010)
4. Dlask, T., Werner, T.: On relation between constraint propagation and block-coordinate descent in linear programs. In: International Conference on Principles and Practice of Constraint Programming. Springer (2020)
5. Freuder, E.C.: A sufficient condition for backtrack-free search. Journal of the ACM (JACM) **29**(1), 24–32 (1982)
6. Globerson, A., Jaakkola, T.S.: Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In: Advances in Neural Information Processing Systems. pp. 553–560 (2008)
7. Hooker, J.: Logic-based methods for optimization: combining optimization and constraint satisfaction. Wiley series in discrete mathematics and optimization, Wiley (2000)
8. Kappes, J.H., Andres, B., Hamprecht, F.A., Schnörr, C., Nowozin, S., Batra, D., Kim, S., Kausler, B.X., Kröger, T., Lellmann, J., Komodakis, N., Savchynskyy, B., Rother, C.: A comparative study of modern inference techniques for structured discrete energy minimization problems. Intl. J. of Computer Vision **115**(2), 155–184 (2015)
9. Khanna, S., Sudan, M.: The optimization complexity of constraint satisfaction problems. In: Electonic Colloquium on Computational Complexity. Citeseer. Citeseer (1996)
10. Kolmogorov, V.: Convergent tree-reweighted message passing for energy minimization. IEEE Transactions on Pattern Analysis and Machine Intelligence **28**(10), 1568–1583 (2006)
11. Komodakis, N., Paragios, N.: Beyond loose LP-relaxations: Optimizing MRFs by repairing cycles. In: European Conference on Computer Vision (2008)
12. Koval, V.K., Schlesinger, M.I.: Dvumernoe programmirovanie v zadachakh analiza izobrazheniy (Two-dimensional programming in image analysis problems). Automatics and Telemechanics **8**, 149–168 (1976), in Russian.
13. Kovalevsky, V., Koval, V.: A diffusion algorithm for decreasing energy of max-sum labeling problem. Glushkov Institute of Cybernetics, Kiev, USSR (1975), unpublished.
14. Matoušek, J., Gärtner, B.: Understanding and using linear programming (universitext). Springer-Verlag (2006)
15. Nguyen, H., de Givry, S., Schiex, T., Bessiere, C.: Maintaining virtual arc consistency dynamically during search. In: International Conference on Tools with Artificial Intelligence (ICTAI). pp. 8–15. IEEE Computer Society (2014)
16. Nguyen, H., Schiex, T., Bessiere, C.: Dynamic virtual arc consistency. In: The 28th Annual ACM Symposium on Applied Computing. pp. 98–103 (2013)
17. Papadimitriou, C.H., Steiglitz, K.: Combinatorial optimization: algorithms and complexity. Courier Corporation (1998)

18. Pardalos, P.M., Vavasis, S.A.: Open questions in complexity theory for numerical optimization. Mathematical programming **57**, 337–339 (1992)
19. Savchynskyy, B.: Discrete graphical models – an optimization perspective. Foundations and Trends in Computer Graphics and Vision **11**(3-4), 160–429 (2019)
20. Schlesinger, M.: Sintaksicheskiy analiz dvumernykh zritelnikh signalov v usloviyakh pomekh (syntactic analysis of two-dimensional visual signals in noisy conditions). Kibernetika **4**(113-130), 2 (1976)
21. Schrijver, A.: Theory of Linear and Integer Programming. John Wiley & Sons, Inc. (1986)
22. Vazirani, V.V.: Approximation Algorithms. Springer-Verlag New York (2001)
23. Živný, S.: The Complexity of Valued Constraint Satisfaction Problems. Cognitive Technologies, Springer (2012)
24. Wainwright, M.J., Jordan, M.I.: Graphical models, exponential families, and variational inference. Foundations and Trends in Machine Learning **1**(1-2), 1–305 (2008)
25. Werner, T.: A linear programming approach to max-sum problem: A review. IEEE Transactions on Pattern Analysis and Machine Intelligence **29**(7), 1165–1179 (July 2007)
26. Werner, T.: On coordinate minimization of piecewise-affine functions. Tech. Rep. CTU-CMP-2017-05, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague (September 2017)