Journal of Artificial Intelligence Research 80 (2024) 665-718 Submitted 10/2023; published 06/2024

Using Constraint Propagation to Bound Linear Programs

Tomáš Dlask[https://orcid.org/0000-0002-1944-6569] DLASKTO2@FEL.CVUT.CZ
Tomáš Werner[https://orcid.org/0000-0002-6161-7157] WERNER@FEL.CVUT.CZ
Machine Learning Group, Department of Cybernetics,
Faculty of Electrical Engineering, Czech Technical University in Prague

Abstract

We present an approach to compute bounds on the optimal value of linear programs based on constraint propagation. Given a feasible dual solution, we apply constraint propagation to the complementary slackness conditions and, if propagation succeeds to prove these conditions infeasible, the infeasibility certificate (in the sense of Farkas' lemma) is reconstructed from the propagation history. This certificate is a dual-improving direction, which allows us to improve the bound. As constraint propagation need not always detect infeasibility of a linear inequality system, the method is not guaranteed to converge to a global solution of the linear program but only to an upper bound, whose tightness depends on the structure of the program and the used propagation method. The approach is suited for large sparse linear programs (such as LP relaxations of combinatorial optimization problems), for which the classical LP algorithms may be infeasible, if only for their super-linear space complexity. The approach can be seen as a generalization of the Virtual Arc Consistency (VAC) algorithm to bound the LP relaxation of the Weighted CSP (WCSP). We newly apply it to the LP relaxation of the Weighted Max-SAT problem, experimentally showing that the obtained bounds are often not far from optima of the relaxation and proving that they are exact for known tractable subclasses of Weighted Max-SAT.

1. Introduction

Although linear programs are solvable in polynomial time, solving very large (though possibly sparse) instances can be challenging in practice. Such linear programs occur in many areas, a prominent example being linear programming (LP) relaxations of hard combinatorial optimization problems, used to compute bounds in branch-and-bound search. For example, even verifying feasibility of LP relaxations of some hard combinatorial problems may require significant amount of time, maybe even exceed the overall time limit for solving the original (non-relaxed) combinatorial problem (Devriendt, Gleixner, & Nordström, 2021, Section 3.2). As another example, LP instances originating in computer vision may have millions of constraints and variables (Yanover, Meltzer, Weiss, Bennett, & Parrado-Hernández, 2006), (Savchynskyy, 2019, Example 4.2). Solving such large instances by traditional LP algorithms such as simplex and interior point methods is sometimes inefficient or even impossible (due to super-linear space-complexity of these methods) (Yanover et al., 2006; Swoboda, Kuske, & Savchynskyy, 2017; Swoboda & Andres, 2017; Haller, Prakash, Hutschenreiter, Pietzsch, Rother, Jug, Swoboda, & Savchynskyy, 2020; Průša & Werner, 2019). Subgradient and first-order iterative methods (such as ADMM) tend to converge slowly (Szeliski, Zabih, Scharstein, Veksler, Kolmogorov, Agarwala, Tappen, & Rother, 2008; Kappes, Andres, Hamprecht, Schnörr, Nowozin, Batra, Kim, Kausler, Kröger, Lellmann, Komodakis, Savchynskyy, & Rother, 2015) and re-converge slowly when warm-started after a small

change of the problem (a significant drawback in branch-and-bound search). Since solving LP relaxations of many NP-hard problems is not easier than solving the general linear programming problem (Průša & Werner, 2019), it is unlikely that any specialized methods will ever beat the general LP algorithms to exactly solve these LP relaxations. This calls for the development of approximate but more efficient methods.

In this article, we propose one such method. Given a pair of mutually dual linear programs, by weak duality the value of the dual objective at any feasible dual solution bounds the common optimal value. Our method iteratively improves a given feasible dual solution as follows. A feasible dual solution is optimal if and only if the complementary slackness conditions, which is a system of linear inequalities in the primal variables, are feasible. This system is infeasible if and only if its alternative system (in the sense of Farkas' lemma) is feasible, so any solution of this alternative system is a certificate of infeasibility of the complementary slackness system. This certificate is also an improving direction for the dual, which can be used to improve the current dual solution and hence the bound. Since proving infeasibility of the complementary slackness system (and possibly finding its infeasibility certificate) can be still too hard for large instances, we propose to do it more efficiently by constraint propagation. The cost for this is that constraint propagation need not always detect infeasibility (i.e., it is refutation incomplete), so the method can converge only to a suboptimal solution. In the examples that we consider, the space complexity of this method is linear in the number of non-zeros of the instance.

This approach can be seen as a generalization of the Virtual Arc Consistency (VAC) algorithm (Cooper, de Givry, Sanchez, Schiex, Zytnicki, & Werner, 2010) and the closely related Augmenting DAG algorithm (Koval & Schlesinger, 1976; Werner, 2007, 2005) to compute a bound on the LP relaxation of the Weighted CSP (WCSP) (Schlesinger, 1976; Werner, 2007; Živný, 2012; Savchynskyy, 2019). In this particular case, the complementary slackness conditions describe the LP relaxation of a CSP (defined by the current feasible dual solution) and the constraint propagation is arc consistency.

Another method closely related to ours is the primal-dual algorithm (Papadimitriou & Steiglitz, 1998, Section 5), proposed to efficiently solve LP formulations of some tractable combinatorial optimization problems. Given a feasible dual solution, it considers the so-called restricted problem, which minimizes infeasibility of the complementary slackness conditions. This is a linear program simpler than the original one, thus often amenable to combinatorial algorithms. Optimal solutions to the dual restricted problem are then dual-improving directions. Many classical algorithms, e.g., for solving shortest paths, maximum flow, or assignment problem, can be seen as examples of the primal-dual algorithm (Papadimitriou & Steiglitz, 1998). The difference to our approach is that the restricted problem is an optimization one (a linear program) rather than a feasibility one (a linear inequality system) and that it is solved exactly (using, e.g., the simplex method) rather than approximately (using constraint propagation). To our knowledge, the VAC/AugDAG algorithms have never been related to the primal-dual algorithm.

Our next source of inspiration was the logic-theoretic view on Farkas' lemma and LP duality (Matoušek & Gärtner, 2006, Section 6.4), (Hooker, 2000). However, our approach is not directly related to inference duality introduced by Hooker (2000, Chapter 17) because we use inference (constraint propagation) in an iterative scheme.

Since constraint propagation need not always detect infeasibility of a system of linear inequalities, our method in general converges only to a suboptimal dual solution. The tightness of the corresponding bound depends on how 'well' the complementary slackness constraints propagate, which in turn depends on the structure of the linear program and the used propagation method. One can expect that practically useful bounds can be obtained for sparse and highly structured linear programs, such as LP relaxations of some combinatorial optimization problems. To illustrate this on a problem different from the WCSP, we apply our method to the LP relaxation of the Weighted Max-SAT problem (Vazirani, 2001). We experimentally show that the obtained bounds are often not far from global optima of the LP relaxation and we prove that they are exact for known tractable subclasses of Weighted Max-SAT.

Compared to the conference version of this paper (Dlask & Werner, 2020), we extended the current paper in the following ways:

- We recall the well-known bounds propagation (which turns out to be equivalent to enforcing arc consistency) and show how to compute infeasibility certificates whenever propagation of bounds detects infeasibility (Section 3.1). We also mention the not-so-known activity propagation and state sufficient conditions for it to be equivalent to bounds propagation (Section 3.3 and Appendix A).
- We comment on the fixed points of the proposed method, state sufficient conditions that guarantee its finiteness, and discuss convergence (Sections 4.1 and 4.2).
- We re-run the experiments to gather more data and provide a detailed analysis that shows the achievable trade-offs between runtime and quality of the obtained bound when applied to LP relaxation of Weighted Max-SAT (Section 6.4).
- In many places throughout our exposition, we provide more detailed comments and insights and also illustrate the theory by additional examples to improve readability.

2. Inference in Systems of Linear Inequalities

A linear inequality in n real variables $x = (x_1, \dots, x_n) \in \mathbb{R}^n$ is the predicate $a^T x \leq b$ where $a = (a_1, \dots, a_n) \in \mathbb{R}^n$ and $b \in \mathbb{R}$. Here, a^T denotes the transposition of vector a, so that $a^T x = a_1 x_1 + \cdots + a_n x_n$. A system of linear inequalities[1] $a_i^T x \leq b_i$, $i \in \{1, \dots, m\} = [m]$ can be written in matrix form as $Ax \leq b$ where $A = [a_{ij}] \in \mathbb{R}^{m \times n}$ is the matrix with rows $a_1^T, \dots, a_m^T$ and $b = (b_1, \dots, b_m) \in \mathbb{R}^m$.

Inference in a system of constraints is a process to derive new (not present in the system) constraints that are implied by the system. We say that an inequality $c^T x \leq d$ is *implied* by (or inferred from, or valid for) an inequality system $Ax \leq b$ if $c^T x \leq d$ holds for all x satisfying $Ax \leq b$. There are two obvious inference rules for linear inequalities:

1. Make a non-negative linear combination of existing inequalities. Precisely, for any vector $y \geq 0$, a system $Ax \leq b$ implies the inequality $y^T Ax \leq y^T b$. We call y the *cause vector* of the new inequality (w.r.t. the system $Ax \leq b$).

1. We will speak about a system of linear inequalities even if it contains also linear equalities, as a linear equality $a^T x = b$ can be represented by two linear inequalities $a^T x \leq b$ and $-a^T x \leq -b$.

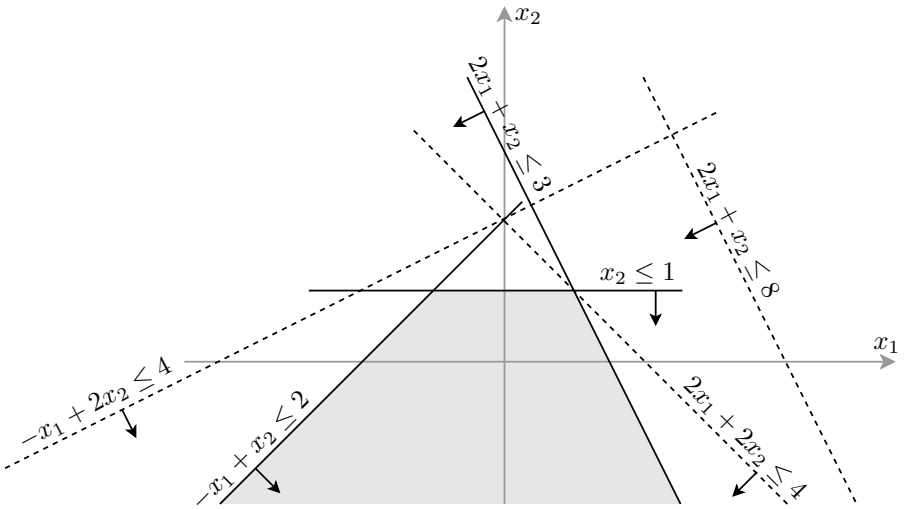


Figure 1: Illustration of Example 1. Half-planes defined by the initial inequalities (1) (solid lines, their intersection is shaded) and by the three inferred inequalities (dashed lines).

2. Loosen the right-hand side of an existing inequality. Precisely, an inequality $a^T x \leq b$ implies $a^T x \leq c$ if $b \leq c$.

Example 1. *Let $m = 3$, $n = 2$ and the system $Ax \leq b$ read*

$$2x_1 + x_2 \leq 3, \qquad x_2 \leq 1, \qquad -x_1 + x_2 \leq 2. \tag{1}$$

Using the first rule, we can, e.g., sum the first and second inequality to infer the inequality $2x_1 + 2x_2 \leq 4$, whose cause vector w.r.t. the system (1) is thus $y = (1, 1, 0)$. Or we can infer the inequality $-x_1 + 2x_2 \leq 4$ with cause vector $y = (\frac{1}{4}, \frac{1}{4}, \frac{3}{2})$.

Using the second rule, from the first inequality in (1) we can infer, e.g., $2x_1 + x_2 \leq 8$.

These three inferred inequalities, as well as the polyhedron defined by (1), are depicted in Figure 1.

A system of linear inequalities is *feasible* if it has at least one solution, and *infeasible* if it has no solution. It follows from the first rule that if the inequality $y^T Ax \leq y^T b$ is infeasible (i.e., $y^T Ax > y^T b$ for all x) for some $y \geq 0$, then the system $Ax \leq b$ is infeasible. In that case, we call the cause vector y a *certificate of infeasibility* of the system $Ax \leq b$.

Example 2. *Let the system $Ax \leq b$ read*

$$x_1 - 2x_2 \leq -1, \qquad -x_1 \leq 0, \qquad x_2 \leq 0. \tag{2}$$

Using the cause vector $y = (1, 1, 2)$, we infer the inequality $0 \leq -1$ (which we can also write as $0x_1 + 0x_2 \leq -1$). As this inequality is infeasible, system (2) is infeasible and y is its certificate of infeasibility.

The celebrated lemma by Farkas says that the condition above is also necessary:

Theorem 1. *A system $Ax \leq b$ is feasible if and only if the inequality $y^TAx \leq y^Tb$ is feasible for every $y \geq 0$.*

Farkas' lemma is usually stated in a slightly different form (Schrijver, 1998, Section 7.3) (Matoušek & Gärtner, 2006, Section 6.4): a system $Ax \leq b$ is infeasible if and only if there exists $y \geq 0$ such that $y^TA = 0$ and $y^Tb < 0$. To see equivalence with Theorem 1, just notice that an inequality $y^TAx \leq y^Tb$ is infeasible if and only if $y^TA = 0$ and $y^Tb < 0$.

As our interest in this paper is to prove infeasibility of linear inequality systems rather than general inference, we will actually never use the second inference rule. Nevertheless, for completeness we state also the so-called affine form of Farkas' lemma, which says that any inequality valid for a linear inequality system can be inferred from this system by combining the two inference rules:

Theorem 2. *The following are equivalent:*

(a) $Ax \leq b$ implies $c^Tx \leq d$.

(b) There exists $y \geq 0$ such that $y^TAx \leq y^Tb$ implies $c^Tx \leq d$.

Again, this is usually stated in a different form (Schrijver, 1998, Section 7.6), which we recover by noting that $y^TAx \leq y^Tb$ implies $c^Tx \leq d$ if $A^Ty = c$ and $y^Tb \leq d$ (the second inference rule).[2]

Proof. Direction (b)⇒(a) is obvious. To prove (a)⇒(b), suppose (a) holds, i.e., the system

$$Ax \leq b, \qquad -c^Tx \leq -d - \delta$$

is infeasible for some $\delta > 0$. By Farkas' lemma, this implies that there are a vector $y \geq 0$ and a scalar $z \geq 0$ such that the inequality $(y^TA - zc)x \leq y^Tb - z(d + \delta)$ is infeasible, i.e., $y^TA - zc = 0$ and $y^Tb - z(d + \delta) < 0$.

If $z = 0$, this reads $y^TA = 0$ and $y^Tb < 0$. In this case, the inequality $y^TAx \leq y^Tb$ is infeasible (i.e., the system $Ax \leq b$ is infeasible), hence it implies anything.

If $z > 0$, we substitute $y := y/z$ to obtain $y^TA = c$ and $y^Tb < d + \delta$, hence (b) holds. □

2.1 Composing Multiple Inference Steps

Inference can be split to multiple steps: from a subset of the initial inequalities we infer (by taking their non-negative combination) a new inequality and add it to the initial set of inequalities, then we infer another inequality from this enlarged set of inequalities, etc. It is easy to see that the composition of such steps is equivalent to taking a single non-negative combination of the initial inequalities. If we obtain an infeasible inequality in this way, we have proved that the initial system was infeasible and the certificate of infeasibility w.r.t. the initial system can be reconstructed from the cause vectors of the intermediate inequalities. We illustrate this with an example.

2. Theorem 2 is trivially equivalent to strong duality for the LP pair (12). Assuming that the primal and dual problem (12) are both feasible and bounded, indeed (a) says that the optimal value of the primal is at most d and (b) means that the optimal value of the dual is also at most d. Since this holds for every d, Theorem 2 is equivalent to equality of the optimal values of the primal and dual problem in (12).

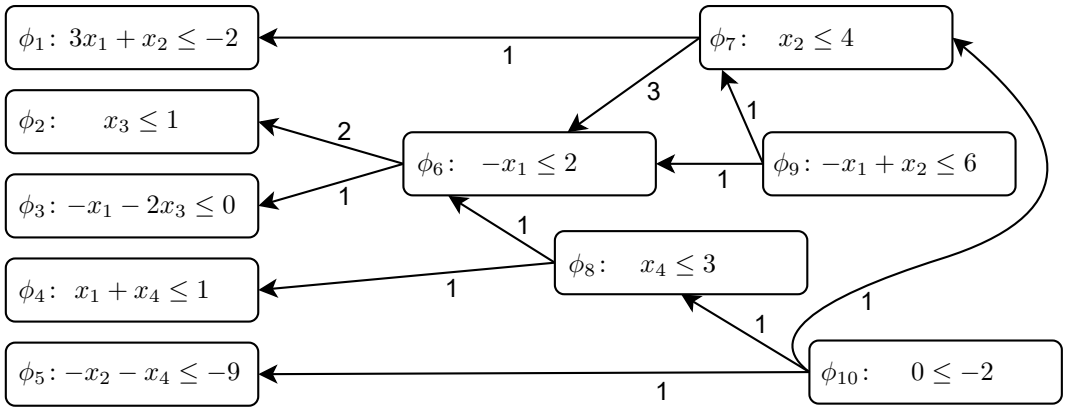


Figure 2: Propagation in a system of linear inequalities. The inequalities are indexed by 1–10. Inequalities 1–5 are initial, inequalities 6–10 are inferred. Edge weights indicate the coefficients of non-negative combinations.

Example 3. *Consider $m = 5$ initial inequalities on the left in Figure 2, where we denote the i-th inequality by ϕ_i. From inequalities ϕ_2 and ϕ_3, we infer inequality $\phi_6 = 2\phi_2 + \phi_3$ (i.e., ϕ_6 is obtained by multiplying ϕ_2 by 2 and summing it with ϕ_3). This derivation is depicted in the figure by arrows with numbers which are the coefficients of the non-negative combination. 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 $\phi_{10} = \phi_5 + \phi_7 + \phi_8$. Inequality ϕ_{10} reads $0 \leq -2$, which is infeasible, therefore the initial system $\phi_1, \ldots, \phi_5$ was infeasible.*

Let us now reconstruct the cause vector y^i of each inequality ϕ_i w.r.t. the initial inequalities $\phi_1, \ldots, \phi_5$. For any $i \leq 5$, we trivially have $y^i = e^i$, the i-th standard-basis vector of $\mathbb{R}^5$. The cause vector of inequality ϕ_6 is $y^6 = 2e^2 + e^3 = (0, 2, 1, 0, 0)$. Further we have $y^7 = e^1 + 3y^6$, $y^8 = e^4 + y^6$, $y^9 = y^6 + y^7$, and finally $y^{10} = e^5 + y^7 + y^8 = (1, 8, 4, 1, 1)$ is a certificate of infeasibility of the initial system $\phi_1, \ldots, \phi_5$.

In later sections, we will often write a single inference step in a table. We illustrate this here with inferring $\phi_7 = \phi_1 + 3\phi_6$:

$$\begin{aligned} 3x_1 + x_2 &\leq -2 & y^1 = e^1 &= (1,0,0,0,0) && \text{(3a)}\\ -3x_1 &\leq 6 & 3y^6 &= (0,6,3,0,0) && \text{(3b)}\\ \hline x_2 &\leq 4 & y^7 &= (1,6,3,0,0). && \text{(3c)} \end{aligned}$$

In the first and second column, we write the inequalities and the cause vectors, respectively. The input entities are above the line and the output below it.

The history of inference is represented by an edge-weighted directed acyclic graph (DAG) (V, E, w) where V is the index set of all (initial and inferred) inequalities, $U \subseteq V$ is the index set of initial inequalities (so in Example 3 we have $V = \{1, \ldots, 10\}$ and $U = \{1, \ldots, 5\}$), and $E \subseteq V \times V$ is a set of directed edges with non-negative weights $w\colon E \to \mathbb{R}_+$ so that each inferred inequality ϕ_i (i.e., $i \in V \setminus U$) is given by $\phi_i = \sum_{j \in N_i^+} w_{ij}\phi_j$ where $N_i^+ = \{\, j \in V \mid (i,j) \in E \,\}$. By composing inference steps, each inequality ϕ_i can be expressed in

terms of the initial inequalities as $\phi_i = \sum_{j \in U} y^i_j \phi_j$ where $y^i \in \mathbb{R}^U_+$ is the cause vector of ϕ_i w.r.t. the initial inequalities. For any initial inequality ϕ_i, we have $y^i = e^i$ where e^i is the i-th standard-basis vector of $\mathbb{R}^U$. For any inferred inequality ϕ_i, we have $y^i = \sum_{j \in N^+_i} w_{ij} y^j$.

As we need the cause vector only for the final (infeasible) inequality (ϕ_{10} in the example), storing all cause vectors explicitly in the memory can be 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.

Let ϕ_i, for $i \in U$, be an initial inequality and ϕ_k, for $k \in V \setminus U$, be an inferred inequality. Then y^k_i is the sum of weight-products of all directed paths from node k to node i in the DAG. By the weight-product of a path, we mean the product of all edge weights along the path. If 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 to store 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 $y^k_i := z_i$, otherwise update $z_j := z_j + w_{ij} z_i$ for all $j \in N^+_i$. Eventually we have $y^k_i = z_i$ for all $i \in U$. The time and space complexity of this algorithm is linear in the size of the graph.

2.2 Other Versions of Farkas' Lemma

So far, we have assumed linear inequality systems in the general form $Ax \leq b$. Farkas' lemma can be formulated for inequality systems in other forms and these versions can be easily proved from one another (Matoušek & Gärtner, 2006, Proposition 6.4.3). The composition algorithm from the previous section can then be easily adapted to these versions. Since it is often convenient to work with a different version, we state some of them:

Theorem 3. *A system $Ax = b$, $x \geq 0$ is feasible if and only if the system $y^T Ax = y^T b$, $x \geq 0$ is feasible for every $y \in \mathbb{R}^m$.*

Theorem 4. *A system $Ax \leq b$, $x \geq 0$ is feasible if and only if the system $y^T Ax \leq y^T b$, $x \geq 0$ is feasible for every $y \geq 0$.*

Sometimes it is convenient to 'dualize' only some of the constraints, considering the convex polyhedron defined by the remaining constraints as our 'universe'. Note that the following theorem subsumes Theorems 3 and 4:

Theorem 5. *A system $Ax \leq b$, $Cx \leq d$ is feasible if and only if the system $y^T Ax \leq y^T b$, $Cx \leq d$ is feasible for every $y \geq 0$.*

Proof. The 'only if' direction is obvious. To prove the 'if' direction, we proceed by contrapositive and suppose the system $Ax \leq b$, $Cx \leq d$ is infeasible. By Theorem 1, this means there exist vectors $y, z \geq 0$ such that the inequality $y^T Ax + z^T Cx \leq y^T b + z^T d$ is infeasible. For such y, the system $y^T Ax \leq y^T b$, $Cx \leq d$ is thus also infeasible. □

Yet other versions can be obtained by combining the versions above. Formulating and proving these is just an exercise on linear programming duality, see, e.g., Matoušek and Gärtner (2006).

3. Constraint Propagation in Systems of Linear Inequalities

In the *constraint satisfaction problem (CSP)*, we are given domains $D_1, \ldots, D_n$ and a set of relations (constraints) over subsets of domains and we want to find a solution $(x_1, \ldots, x_n) \in D_1 \times \cdots \times D_n$ satisfying all the relations. A popular heuristic to tackle the CSP is *constraint propagation*, where we iteratively delete some tuples from some relations (or, as a special case, delete some elements from some domains, i.e., unary relations) without changing the solution set. If some domain or relation becomes empty, the initial CSP was infeasible. Constraint propagation is a form of inference because deleting a tuple is equivalent to inferring a new constraint from the current constraint set and adding it to the constraint set. Instead of inferring all constraints implied by the constraint set, only a small set of simple inference (or propagation) rules is used, able to generate only a subset of all implied constraints. In that case, the procedure may be refutation incomplete: it need not infer an empty relation if the CSP is infeasible.

A system of linear inequalities can be seen as a CSP with the domains $D_1 = \cdots = D_n = \mathbb{R}$ and the constraints being the linear inequalities. Though such a system can be proved infeasible by linear programming algorithms, for large systems this may be too time consuming or memory demanding. In such a case, constraint propagation can be a more efficient option, despite the fact that it need not always detect infeasibility.

Constraint propagation in CSPs with numeric variables is well-studied (Benhamou & Granvilliers, 2006) but typically assumes finite integral domains and/or non-linear constraints (Belotti, Cafieri, Lee, Liberti, et al., 2010; Bordeaux, Katsirelos, Narodytska, & Vardi, 2011; Schulte & Stuckey, 2005; Harvey & Stuckey, 2003; Puranik & Sahinidis, 2017). Perhaps the closest to our setting are CSPs with linear equality/inequality constraints over integer variables, see, e.g., Bordeaux et al. (2011), Harvey and Stuckey (2003), Yuanlin and Yap (2000). Although linear inequality systems over real variables are occasionally mentioned in the CSP literature (Bordeaux et al., 2011; Schulte & Stuckey, 2005; Sofranac, Gleixner, & Pokutta, 2022; Yuanlin & Yap, 2000), they were given limited attention because of their relative simplicity.

Next, we discuss examples of constraint propagation methods applicable to systems of linear inequalities over real variables. We do not aim to give a comprehensive survey of such methods, instead we focus on construction of cause vectors and infeasibility certificates, which have not been studied before in connection with constraint propagation.

3.1 Bounds Propagation

The propagation of lower and upper bounds for numeric variables in CSPs is well-known (Bordeaux et al., 2011; Schulte & Stuckey, 2005; Benhamou & Granvilliers, 2006; Harvey & Stuckey, 2003; Sofranac et al., 2022; Belotti et al., 2010) and, among other applications, finds its use in practical presolve methods that are used to simplify linear programs before a general solver is invoked (Brearley, Mitra, & Williams, 1975; Achterberg, Bixby, Gu, Rothberg, & Weninger, 2014, 2020).

Suppose each variable x_j has a lower bound $l_j \in \mathbb{R} \cup \{-\infty\}$ and upper bound $u_j \in \mathbb{R} \cup \{+\infty\}$ with $l_j \leq u_j$, so that its domain $D_j \subseteq \mathbb{R}$ is the interval $\{x_j \in \mathbb{R} \mid l_j \leq x_j \leq u_j\}$.[3]

3. E.g., for $l_j, u_j \in \mathbb{R}$, the domain D_j is the finite interval $[l_j, u_j]$. For $l_j \in \mathbb{R}$ and $u_j = +\infty$, D_j is the left-bounded interval $D_j = [l_j, +\infty)$.

Before propagation, the bounds are initialized to $l_j = -\infty$ and $u_j = +\infty$ for all j. In every iteration, the algorithm tightens the bounds on a single variable x_j based on a single linear inequality[4] $a_i^T x \le b_i$ with $a_{ij} \neq 0$. Namely, we have the bound (Puranik & Sahinidis, 2017, Section 5.1)

$$a_{ij}x_j \le b_i - \sum_{j' \in [n]\setminus\{j\}} a_{ij'}x_{j'} \le \underbrace{b_i - \sum_{j' \in [n]\setminus\{j\}:\ a_{ij'}>0} a_{ij'}l_{j'} - \sum_{j' \in [n]\setminus\{j\}:\ a_{ij'}<0} a_{ij'}u_{j'}}_{r} \tag{4}$$

where[5] the first inequality is $a_i^T x \le b_i$ and the second inequality follows from the bounds. If $a_{ij} > 0$, then (4) implies $x_j \le r/a_{ij}$, so we have derived an upper bound $u'_j = r/a_{ij}$. If $a_{ij} < 0$, then (4) implies $x_j \ge r/a_{ij}$, so we have a lower bound $l'_j = r/a_{ij}$.

For each inferred bound, i.e., inequality of the form $x_j \le u_j$ or $-x_j \le -l_j$, we will denote the corresponding cause vector of this inequality by $\overline{y}^j$ and $\underline{y}^j$, respectively. Next, we describe one iteration of bounds propagation. We assume that if some upper or lower bound is finite, it has been derived in some previous iteration, including its cause vector.

To derive the new bound inequality along with its cause vector, we need to express this inequality as a non-negative combination of the used inequalities. We write this in a table together with the cause vectors of the involved inequalities (like in Example 3). We assume $a_{ij} > 0$ (for $a_{ij} < 0$ we would analogically obtain a lower bound):

$a_i^T x/\lvert a_{ij}\rvert \le b_i/\lvert a_{ij}\rvert$	$e^i/\lvert a_{ij}\rvert$		(5a)
$-a_{ij'}x_{j'}/\lvert a_{ij}\rvert \le -a_{ij'}l_{j'}/\lvert a_{ij}\rvert$	$a_{ij'}\underline{y}^{j'}/\lvert a_{ij}\rvert$	$\forall j' \in [n]\setminus\{j\}\colon a_{ij'} > 0$	(5b)
$-a_{ij'}x_{j'}/\lvert a_{ij}\rvert \le -a_{ij'}u_{j'}/\lvert a_{ij}\rvert$	$-a_{ij'}\overline{y}^{j'}/\lvert a_{ij}\rvert$	$\forall j' \in [n]\setminus\{j\}\colon a_{ij'} < 0$	(5c)
$x_j \le u'_j$	$\overline{y}'^j$.		(5d)

Here, (5a) is the inequality $a_i^T x \le b_i$ divided by $|a_{ij}|$, (5b) are lower-bound inequalities $-x_{j'} \le -l_{j'}$ multiplied by $a_{ij'}/|a_{ij}| > 0$, and (5c) are upper-bound inequalities $x_{j'} \le u_{j'}$ multiplied by $-a_{ij'}/|a_{ij}| > 0$. The resulting inequality and its cause vector in (5d) are the sum of the entities above them. We see that the cause vector $\overline{y}'^j$ of the derived inequality $x_j \le u'_j$ is

$$\overline{y}'^j = \big(e^i + \sum_{j' \in [n]\setminus\{j\}:\ a_{ij'}>0} a_{ij'}\underline{y}^{j'} - \sum_{j' \in [n]\setminus\{j\}:\ a_{ij'}<0} a_{ij'}\overline{y}^{j'}\big)/|a_{ij}|. \tag{6}$$

Since it suffices to keep only the so-far best bound for each variable, we update the bound (by setting $u_j := u'_j$ and $\overline{y}^j := \overline{y}'^j$) only if it has improved (i.e., $u'_j < u_j$).

Note, a necessary condition for the bounds to get updated from the initially infinite values is that the constraint system contains at least one unary constraint, i.e., a constraint involving only a single variable.

4. If our constraint is a linear equality, $a_i^T x = b_i$, the propagation update is the same as the updates for two inequalities $a_i^T x \le b_i$ and $-a_i^T x \le -b_i$ (Bordeaux et al., 2011, Section 3.3).
5. In more complex settings (such as non-linear constraints), interval arithmetic is often used to compute the updates (Yuanlin & Yap, 2000; Benhamou & Granvilliers, 2006; Schulte & Stuckey, 2005; Belotti et al., 2010). However, in case of linear inequalities, both approaches yield identical bounds.

Example 4. *Consider the inequality* $-2x_1 + x_2 - 3x_3 \leq -5$ *and the domains* $D_1 = [1, 3]$, $D_2 = [2, 6]$, *and* $D_3 = [0, 1]$ *(obtained, e.g., by applying the iteration to individual inequalities* $-x_1 \leq -1$, $x_1 \leq 3$, $-x_2 \leq -2$, $x_2 \leq 6$, $-x_3 \leq 0$, *and* $x_3 \leq 1$*).*

For $j = 1$, (4) *reads* $-2x_1 \leq -5 - x_2 + 3x_3 \leq -5 - l_2 + 3u_3 = -4 = r$, *which yields* $x_1 \geq 2$ *and we increase the lower bound of* x_1 *to* $l_1 = \max\{1, 2\} = 2$. *Following* (5), *one can derive this formally as*

$$\begin{aligned} -x_1 + \tfrac{1}{2}x_2 - \tfrac{3}{2}x_3 &\leq -\tfrac{5}{2} & \tfrac{1}{2}e^i & \quad (7a)\\ -\tfrac{1}{2}x_2 &\leq -1 & \tfrac{1}{2}\underline{y}^2 & \quad (7b)\\ \tfrac{3}{2}x_3 &\leq \tfrac{3}{2} & \tfrac{3}{2}\overline{y}^3 & \quad (7c)\\ \hline -x_1 &\leq -2 & \underline{y}'^1 & \quad (7d) \end{aligned}$$

where $\underline{y}'^1 = \frac{1}{2}e^i + \frac{1}{2}\underline{y}^2 + \frac{3}{2}\overline{y}^3$.

Next, for $j = 2$, *we obtain* $x_2 \leq -5 + 2x_1 + 3x_3 \leq 4$, *which decreases the upper bound of* x_2 *to* $u_2 = \min\{6, 4\} = 4$. *Finally, for* $j = 3$, *the inequalities* $-3x_3 \leq -5 + 2x_1 - x_2 \leq -1$ *imply* $x_3 \geq \frac{1}{3}$, *which improves the previous lower bound to* $l_3 = \max\{0, \frac{1}{3}\} = \frac{1}{3}$. *The new domains thus are* $D_1 = [2, 3]$, $D_2 = [2, 4]$, $D_3 = [\frac{1}{3}, 1]$.

If during propagation we get $l_j > u_j$ (i.e., $D_j = \emptyset$, 'domain wipe-out') for some $j \in [n]$, then the initial CSP is infeasible. This contradiction is inferred as follows:

$$\begin{aligned} -x_j &\leq -l_j & \underline{y}^j & \quad (8a)\\ x_j &\leq u_j & \overline{y}^j & \quad (8b)\\ \hline 0 &\leq u_j - l_j & y = \underline{y}^j + \overline{y}^j. & \quad (8c) \end{aligned}$$

Due to $u_j - l_j < 0$ the inferred inequality (8c) is infeasible, therefore its cause vector $\underline{y}^j + \overline{y}^j$ is a certificate of infeasibility of the system $Ax \leq b$.

An established form of local consistency in this setting is bound($\mathbb{R}$) consistency (Bordeaux et al., 2011; Schulte & Stuckey, 2005; Harvey & Stuckey, 2003; Lhomme, 1993). As we do not consider other forms of bounds consistency (such as bound($\mathbb{Z}$) consistency from Bordeaux et al., 2011; Schulte & Stuckey, 2005), we call it simply 'bounds consistency'. We state its definition only for the case when all bounds are finite[6] (i.e., for all $j \in [n]$ we have $-\infty < l_j \leq u_j < +\infty$):

Definition 1 (Bounds consistency)**.** *Domains* $D_j = [l_j, u_j]$, $j \in [n]$ *are bounds consistent w.r.t. a given constraint if for each* $j \in [n]$ *and* $x_j \in \{l_j, u_j\}$ *there exist values* $x_{j'} \in D_{j'}$, $j' \in [n] \setminus \{j\}$ *satisfying the constraint. The domains* D_j, $j \in [n]$ *are bounds consistent w.r.t. a set of constraints if they are bounds consistent w.r.t. each constraint in the set.*

After executing the propagation update for each variable of a single inequality, the resulting domains become bounds consistent w.r.t. that inequality (Yuanlin & Yap, 2000, Section 3) (see Example 4). By iterating the updates for different inequalities of a system $Ax \leq b$, as in Algorithm 1, the domains gradually shrink. This process in general need

6. We did not find in the literature a definition of bounds consistency for the case when some bounds can be infinite. We did not want to discuss this question here because it is not essential for the paper.

inputs: matrix $A \in \mathbb{R}^{m \times n}$, vector $b \in \mathbb{R}^m$
For each $j \in [n]$, set $l_j := -\infty$ and $u_j := +\infty$.
while bounds change
  Choose $j \in [n]$ and $i \in [m]$ with $a_{ij} \neq 0$.
  Set $d := r/a_{ij}$ where r is defined in (4).
  if $a_{ij} > 0$ and $d < u_j$ **then**
    Set $u_j := d$. Set $\overline{y}^j$ as in (6).
  else if $a_{ij} < 0$ and $d > l_j$ **then**
    Set $l_j := d$. Set $\underline{y}^j$ analogically to (6).
  if $l_j > u_j$ **then**
    return $y = \underline{y}^j + \overline{y}^j$. (System $Ax \leq b$ is infeasible, with certificate y.)

Algorithm 1: Bounds propagation in system $Ax \leq b$.

not reach a fixed point in finite time[7] (Davis, 1987, Section 9), (Puranik & Sahinidis, 2017, Section 5.2), which may not matter in practice because the propagation is refutation incomplete anyway. To achieve termination in finite time, one option is to discretize the domains (e.g., by using floating-point arithmetic) (Benhamou & Granvilliers, 2006; Bordeaux et al., 2011; Lhomme, 1993), in which case the propagation takes pseudo-polynomial time (Bordeaux et al., 2011). Another option is to terminate the propagation before reaching a fixed point (e.g., when the bounds do not change 'too much'), which is related to the notion of arc $B(w)$-consistency (Lhomme, 1993).

For this particular kind of propagation, it may be practical to store all cause vectors explicitly, as opposed to Section 2.

3.2 Arc Consistency

Still assuming that the domains $D_1, \ldots, D_n$ are real closed intervals, bounds consistency is equivalent to (generalized) arc consistency (a.k.a. domain consistency, see Schulte & Stuckey, 2005, Section 2.2 and Harvey & Stuckey, 2003). To show this formally, let us recall the definition of arc consistency in this setting.

Definition 2 (Arc consistency, Benhamou & Granvilliers, 2006)**.** *Domains D_j, $j \in [n]$ are arc consistent w.r.t. a given constraint if for each $j \in [n]$ and $x_j \in D_j$ there exist values $x_{j'} \in D_{j'}$, $j' \in [n] \setminus \{j\}$ satisfying the constraint. The domains D_j, $j \in [n]$ are arc consistent w.r.t. a set of constraints if they are arc consistent w.r.t. each constraint in the set.*

The only difference between Definitions 1 and 2 lies in whether we require $x_j \in \{l_j, u_j\}$ or $x_j \in D_j$, which is why bounds consistency was called arc B-consistency ('bounds arc consistency') by Lhomme (1993, Section 3.1). It is easy to see that bounds consistency and arc consistency are equivalent for constraints defined by linear inequalities because the

7. The bounds consistency closure can be in this case computed in polynomial time by solving a linear program (Belotti et al., 2010, Section 4), (Bordeaux et al., 2011, Section 3.4.1). Unfortunately, this linear program is even larger than the original system $Ax \leq b$. It is an interesting question whether the structure of this linear program would allow us to solve it efficiently (at least in some cases).

feasible set is convex. We formalize this fact in Proposition 1. Analogous results were shown for a slightly different setting by, e.g., Lhomme (1993, Corollary 1) or Yuanlin and Yap (2000, Section 4) (also see Schulte & Stuckey, 2005).

Proposition 1. *Interval domains $D_1, \ldots, D_n$ are arc consistent w.r.t. a linear constraint ϕ_i (given by $a_i^T x \leq b_i$), $i \in [m]$ if and only if they are bounds consistent w.r.t. ϕ_i.*

Proof. The 'only if' direction is obvious, so we proceed with the 'if' direction. Let $j \in [n]$ and let $\underline{x}$ and $\overline{x}$ be such that $\underline{x}_j = l_j$, $\overline{x}_j = u_j$, $a_i^T \underline{x} \leq b_i$, and $a_i^T \overline{x} \leq b_i$. For any value $x_j^* \in D_j$, there exists $\alpha \in [0, 1]$ such that $x_j^* = \alpha l_j + (1 - \alpha) u_j$. Consequently, $x^* = \alpha \underline{x} + (1 - \alpha)\overline{x}$ satisfies $a_i^T x^* \leq b_i$. □

Remark 1. *Even if the initial domains are intervals, they need not be intervals after propagating arc consistency for general (possibly non-convex) constraints. On the other hand, the projections of any convex sets (in particular, polyhedral sets) onto individual coordinates are intervals, which is why it is natural to restrict Definition 2 to intervals in this setting.*

3.3 Activity Propagation

In Dlask and Werner (2023), we proposed a natural kind of constraint propagation in a system $Ax \leq b$ of linear inequalities and called it *activity propagation*. Recall that an inequality $a^T x \leq b$ is *active* at point $x \in \mathbb{R}^n$ if it holds with equality at that point, i.e., $a^T x = b$. An inequality $a_i^T x \leq b_i$ in a system $Ax \leq b$ is *always active* if $Ax \leq b$ implies $a_i^T x = b_i$, i.e., the inequality $a_i^T x \leq b_i$ is active at all points $x \in \mathbb{R}^n$ satisfying $Ax \leq b$ (Freund, Roundy, & Todd, 1985). Clearly, if some inequality is always active in a subset of the set of inequalities $Ax \leq b$, then it is always active also in the whole set $Ax \leq b$. In one iteration of activity propagation, we choose a subset of the inequalities, infer which (if any) of them are always active in this subset, and turn these inequalities to equalities.

To describe the activity propagator formally, suppose we already know that some of the inequalities of the system $Ax \leq b$ (where $A \in \mathbb{R}^{m \times n}$), given by indices $[m] \setminus I$ where $I \subseteq [m]$, are always active. Then the system $Ax \leq b$ is equivalent to

$$a_i^T x \leq b_i \qquad \forall i \in I \tag{9a}$$

$$a_i^T x = b_i \qquad \forall i \in [m] \setminus I. \tag{9b}$$

The activity propagator chooses a subset ('block')

$$a_i^T x \leq b_i \qquad \forall i \in B \cap I \tag{10a}$$

$$a_i^T x = b_i \qquad \forall i \in B \setminus I. \tag{10b}$$

of constraints (9) (determined by a set $B \subseteq [m]$), infers which of the inequalities (10a) are always active in system (10), and removes the indices of these inequalities from I.

The algorithm starts with $I = [m]$ (so that (9) reads $Ax \leq b$) and then iterates the activity propagator for different blocks $B \subseteq [m]$, which progressively shrinks the set I. If (10) becomes infeasible for some B, we know that the original system $Ax \leq b$ was infeasible.

If the blocks B are chosen from some collection $\mathcal{B} \subseteq 2^{[m]}$ fixed in advance, activity propagation leads to the following local consistency:

Definition 3 ($\mathcal{B}$-consistency, Dlask & Werner, 2023)**.** *For $B \subseteq [m]$, a set $I \subseteq [m]$ is B-consistent w.r.t. a system $Ax \leq b$ if system (10) is feasible and does not contain any always-active inequality. For $\mathcal{B} \subseteq 2^{[m]}$, a set $I \subseteq [m]$ is $\mathcal{B}$-consistent w.r.t. the system $Ax \leq b$ if it is B-consistent for all $B \in \mathcal{B}$.*

We proved in Dlask and Werner (2023) that, for any subset $\mathcal{B} \subseteq 2^{[m]}$, activity propagation either detects infeasibility or attains $\mathcal{B}$-consistency after a finite number of iterations, assuming that each block from $\mathcal{B}$ is visited again after a finite number of iterations. Note that rather than an equality $a_i^T x = b_i$, it suffices that the propagator derives the inequality $-a_i^T x \leq -b_i$ because the inequality $a_i^T x \leq b_i$ is already present in the system. The existence of the cause vector for the newly derived inequality is guaranteed by Theorem 2. Whenever infeasibility is detected, it is possible to compute a certificate of infeasibility from the history of the propagation, according to Theorem 1. In general, the certificate can be also computed by the algorithm outlined in Section 2.1 using the intermediate cause vectors.

We described a procedure which computes the certificate in anti-chronological order of the propagations (as opposed to the chronological approach from Section 2.1) in Dlask and Werner (2023, Appendix A), but it considers a system in a different form, namely $Ax = b$, $x \geq 0$. Investigating the relation between the chronological method from Section 2.1 and the anti-chronological method from Dlask and Werner (2023) is subject to further research. We will also briefly comment on this later in Section 5.2.

In Appendix A, we show that, under precise additional conditions on the system $Ax \leq b$ and the set of blocks $\mathcal{B}$, bounds consistency (hence also arc consistency) coincides with $\mathcal{B}$-consistency.

3.4 Tightening Right-Hand Sides

Not only due to the relation between bounds/arc consistency and $\mathcal{B}$-consistency, it may be natural to ask whether there exists a common generalization of these consistencies. Here, we propose one such local consistency. It is based on tightening the right-hand side (RHS) of individual linear inequalities.

Definition 4 ($\mathcal{B}$-RHS-consistency)**.** *Let $A \in \mathbb{R}^{m \times n}$. For $B \subseteq [m]$, a vector $b \in \mathbb{R}^m$ is B-RHS-consistent if, for every $i \in B$,*

$$b_i = \max\{a_i^T x \mid x \in \mathbb{R}^n,\ a_{i'}^T x \leq b_{i'}\ \forall i' \in B\}. \tag{11}$$

For $\mathcal{B} \subseteq 2^{[m]}$, a vector $b \in \mathbb{R}^m$ is $\mathcal{B}$-RHS-consistent if it is B-RHS-consistent for all $B \in \mathcal{B}$.

A propagator for B-RHS-consistency is simply the following one: for each $i \in B$, compute the value (11), denote it by b'_i, and update $b_i := b'_i$. It can be shown that after processing all constraints from a set B, the vector b becomes B-RHS-consistent.

The proposed propagation generalizes bounds propagation because one can take $a_i = e^j$ or $a_i = -e^j$ so that the right-hand side is tightened in $x_j \leq b_i$ or $-x_j \leq b_i$, respectively. It also generalizes activity propagation since we may have $a_i = -a_{i'}$ for $i \neq i'$ and inferring that $a_i^T x \leq d$ and $a_{i'}^T x \leq -d$ for some $d \in \mathbb{R}$ is then equivalent to deriving $a_i^T x = d$. One might think that it would be more general to explicitly keep both upper and lower bound on $a_i^T x$ resulting in $l_i \leq a_i^T x \leq u_i$ as in (Sofranac et al., 2022; Belotti et al., 2010; Achterberg, 2007) – but this can be represented by two inequalities $-a_i^T x \leq -l_i$ and $a_i^T x \leq u_i$.

The existence of cause vectors of the newly inferred (tightened) inequalities is guaranteed by Theorem 2. To provide more details, if the system $a_{i'}^T x \leq b_{i'}$, $i' \in B$ implies $a_i^T x \leq b'_i$, then we may formally add the constraint $a_i^T x \leq b'_i$ into the system and replace the original constraint $a_i^T x \leq b_i$ by the new one in all sets B rather than update $b_i := b'_i$. However, the difference is only semantic. If the system (11) is infeasible for some B, one can compute the certificate of infeasibility w.r.t. the original system by the approach described in Section 2.1. For this particular kind of propagation, each node in the DAG has at most $\max_{B \in \mathcal{B}} |B| - 1$ successors, so the graph is sparse if the sets in $\mathcal{B}$ are small.

Remark 2. *We did not suggest how to choose the set of blocks $\mathcal{B}$ in activity propagation and $\mathcal{B}$-RHS-consistency. In general, it is desirable to choose the blocks such that the propagation can be done efficiently, preferably without the need to invoke a general LP solver. In relation to Remark 7 (stated later), a similar question can be asked for the related block-coordinate descent method, see Remark 2 in Dlask and Werner (2023).*

Remark 3. *A similar method for tightening the RHSs of linear constraints is mentioned by Puranik and Sahinidis (2017, Equation (14)) (also see Savelsbergh, 1994). However, this method considers all the other inequalities, i.e., we have $B = [m]$ in* (11)*. This method is not useful for our purposes because it requires to directly check feasibility of the whole system $Ax \leq b$, contradicting our focus on (simpler) local consistency techniques. Although such a method can be useful for pre-processing constraints in (mixed) integer linear programs (or LP relaxations of non-linear programs), it is also admitted by Puranik and Sahinidis (2017) that solving such problems may be too expensive. Solving only a subproblem (as in* (11) *for a smaller set B) may thus provide a certain trade-off, which was noted by Achterberg et al. (2014, Section 6).*

Another method related to bounds propagation with more constraints was proposed and analyzed by Belotti (2013) where the new bounds for a variable are not inferred only from a single linear inequality, but instead from a pair of linear inequalities, which can provide tighter bounds at higher computational cost.

Remark 4. *It is known that most local consistencies can be unified using the theory of partially ordered sets (Apt, 1999; Bessiere, 2006; Dlask, 2022). Locally consistent CSP instances (with a fixed structure) form a subset L of a complete lattice S, where L is closed under the join operation. Enforcing the local consistency is a closure operator on S (namely, the closure of $s \in S$ is the join of the elements of L that are less than or equal to s). Finally, the enforcing algorithm (constraint propagation) can be seen as chaotic applications of propagators on S.*

The described local consistencies fit into this framework. For bounds consistency (and arc consistency), S is the set of boxes $D_1 \times \cdots \times D_n$ where $D_1, \ldots, D_n$ are real closed intervals, partially ordered by inclusion (hence the join operation is the convex hull of union). For activity propagation, S is the set of subsets of $[m]$, partially ordered by inclusion (hence the join is union) (Dlask & Werner, 2023). For $\mathcal{B}$-RHS-consistency, S is $\mathbb{R}^m$, partially ordered by element-wise arithmetic ordering $\leq$ (hence the join is the element-wise maximum). For all of these consistencies, it is easy to check that the set L of consistent elements of S is closed under join.

4. Linear Optimization by Constraint Propagation

In Sections 2 and 3, we showed how a certificate of infeasibility of a linear inequality system can be obtained by constraint propagation. Here, we show how this certificate can be used to improve a non-optimal feasible solution to a linear program. We will show this on the pair of mutually dual linear programs

$$\begin{aligned} \max\ & c^T x && \min\ b^T y && \text{(12a)}\\ Ax &\le b && y \ge 0 && \text{(12b)}\\ x &\in \mathbb{R}^n && A^T y = c && \text{(12c)} \end{aligned}$$

where we will refer to the problem on the left as primal and to the problem on the right as dual. We assume that both problems are feasible and bounded.

By complementary slackness, a primal-feasible solution x and a dual-feasible solution y are simultaneously optimal if and only if $y^T(Ax - b) = 0$, i.e., for every $i \in [m]$ we have $a_i^T x = b_i$ or $y_i = 0$ (or both), i.e., at least one of the two corresponding constraints on line (12b) is active. Denoting by

$$I^*(y) = \{i \in [m] \mid y_i = 0\} \tag{13}$$

the set of dual constraints (12b) active at point y, this condition can be rewritten as follows: a dual-feasible solution y is optimal for the dual if and only if the left-hand system in

$$\begin{aligned} & && b^T \bar{y} < 0 && && \text{(14a)}\\ a_i^T x &\le b_i && \bar{y}_i \ge 0 && \forall i \in I && \text{(14b)}\\ a_i^T x &= b_i && \bar{y}_i \in \mathbb{R} && \forall i \in [m] \setminus I && \text{(14c)}\\ x &\in \mathbb{R}^n && A^T \bar{y} = 0 && && \text{(14d)} \end{aligned}$$

is feasible for $I = I^*(y)$. On the right in (14), we wrote Farkas' alternative system to the left-hand system.[8] The left-hand system is infeasible if and only if the right-hand system is feasible. In that case, any $\bar{y}$ feasible for the right-hand system is not only a certificate of infeasibility of the left-hand system but also a *dual-improving direction* from the current point y, as given by the following proposition.

Proposition 2. *Let $y \in \mathbb{R}^m$ be feasible for the dual* (12) *and $\bar{y}$ be a solution to the right-hand system* (14) *for $I = I^*(y)$. Let*

$$\alpha^* = \min_{i \in [m]:\ \bar{y}_i < 0} -y_i/\bar{y}_i. \tag{15}$$

Then for every $0 < \alpha \le \alpha^*$, the point $y' = y + \alpha\bar{y}$ is feasible for the dual (12) *and $b^T y' < b^T y$.*

Proof. First, see that $\alpha^* > 0$ because $\bar{y}_i < 0$ implies $i \in [m] \setminus I^*(y)$ by (14b), i.e., $y_i > 0$ by definition of $I^*(y)$. Additionally, since we assume that both primal and dual are bounded, the value α^* is well-defined: if there was no $i \in [m]$ with $\bar{y}_i < 0$, then the dual (12) would

8. The pair (14) corresponds to a more general version of Farkas' lemma than the one considered in Theorem 1 because the left-hand system contains also equality constraints (recall Section 2.2).

be unbounded and α could be chosen arbitrarily large, making the dual objective $b^T y'$ arbitrarily small.

To show feasibility, if $i \in [m]$ is such that $\bar{y}_i \geq 0$, then $y_i + \alpha \bar{y}_i \geq 0$ by $y_i, \alpha \geq 0$. On the other hand, if $y_i < 0$, then $\alpha \leq \alpha^* \leq -y_i/\bar{y}_i$, i.e., $y_i + \alpha \bar{y}_i \geq 0$. This implies $y' \geq 0$ and $A^T y' = A^T y = c$ due to $A^T \bar{y} = 0$. By $b^T \bar{y} < 0$ and $\alpha > 0$, the dual objective strictly improves: $b^T y' = b^T y + \alpha b^T \bar{y} < b^T y$. □

Note that the step size $\alpha = \alpha^*$ is optimal in the sense that, for the fixed improving direction $\bar{y}$, it provides the largest possible improvement of the dual objective.

These properties can be used to iteratively optimize the dual linear program if an initial dual-feasible solution y is provided. We construct the left-hand system (14) and, if it is infeasible for $I = I^*(y)$, find an improving direction $\bar{y}$ satisfying the right-hand system, update $y := y + \alpha^* \bar{y}$, and improve the current objective while maintaining feasibility. By repeating this iteration, one obtains a better and better bound on the common optimal value of (12). Eventually, if the left-hand system becomes feasible for current y, y is optimal for the dual. Note, even though the dual objective improves after each iteration, this general scheme need not terminate in a finite number of steps or even converge to an optimal solution.

Remark 5. *This optimization scheme is related to the primal-dual algorithm (Papadimitriou & Steiglitz, 1998, Section 5), where complementary slackness conditions are not enforced strictly, but their violation is minimized instead. This change ensures convergence and finiteness of the algorithm. Note, in modern terminology, calling such a method 'primal-dual' can be seen as a misnomer because only a dual-feasible solution is maintained (Papadimitriou & Steiglitz, 1998, Section 5.2).*

Since deciding feasibility of the left-hand system (14) is in general as hard as solving the general linear programming problem, it may be inefficient for large instances. We propose that in such cases it can be desirable to do it by constraint propagation, despite the fact that constraint propagation need not always detect infeasibility (i.e., is refutation incomplete[9]). In one iteration of this scheme, we apply a suitable (problem-dependent) set of constraint propagation rules for the left-hand system (14) and, whenever infeasibility is detected, construct the certificate of infeasibility (the dual-improving direction) and use it to improve the dual-feasible solution y. This is outlined in Algorithm 2. For constraint propagation and construction of the improving direction, one can use, e.g., the approaches reviewed in Section 3.

As constraint propagation need not always find a solution to the right-hand system (14) even if it exists, points y returned by Algorithm 2 need not be optimal. However, even non-optimal solutions can be useful in practice. As an example, if the primal (12) is an LP relaxation of a hard combinatorial optimization problem, then any solution feasible for the dual (12) provides an upper bound on the optimal value of the original combinatorial problem. Such bounds can be used in branch-and-bound search where it may not be necessary

9. Recall from Section 2 that we call a propagation method *refutation complete* (for a given class of problems) if it detects a contradiction in any infeasible/unsatisfiable problem from the class and *refutation incomplete* otherwise. Synonymous terms to refutation-completeness are that the method is a *decision procedure* (Cooper et al., 2010; Thapper & Živný, 2012; Cooper & Živný, 2016) or a *complete refutation method* (Hooker, 2000, Section 3.2.1).

inputs: instance of problem (12), dual-feasible solution y, constraint propagation rules

repeat
  Set $I := I^*(y)$.
  Try to detect infeasibility of left-hand system (14) by constraint propagation.
  if left-hand system (14) is proved infeasible **then**
    Find an improving direction $\bar{y}$ satisfying right-hand system (14).
    Compute (possibly non-optimal) step size $\alpha > 0$ so that $y + \alpha\bar{y}$ is feasible.
    Update $y := y + \alpha\bar{y}$.
  else
    return y (At this point, we are unable to prove infeasibility.)

Algorithm 2: Iterative scheme for approximate optimization of the dual (12).

or efficient to solve the LP relaxation to global optimality, as there is a trade-off between the time spent in computing the bound and the time spent in search. Note, we do not use the computed bounds in branch-and-bound search in this paper, our focus is only on computing the bounds.

4.1 Fixed Points and Faces

Let us now comment on the fixed points of Algorithm 2 (i.e., the points which it is unable to improve) and show that these points cluster to the (relative interiors of) faces of the feasible set of the dual (12).

Let

$$Y = \{y \in \mathbb{R}^m \mid y \geq 0,\ A^T y = c\} \tag{16}$$

denote the feasible set of the dual (12), which is a convex polyhedron. Recall that, for any $I \subseteq [m]$, the set

$$F_I = \{y \in Y \mid y_i = 0\ \forall i \in I\} \tag{17}$$

is a *face* of the polyhedron Y (Schrijver, 2004, Section 5.6). Moreover, if[10] $\{y \in Y \mid I^*(y) = I\} \neq \emptyset$, then

$$\operatorname{ri} F_I = \{y \in Y \mid I^*(y) = I\} \subseteq F_I \tag{18}$$

is the *relative interior* (Rockafellar, 1972, Section 6) of the face F_I.

Further in this section we assume a fixed set of constraint propagation rules is given for the left-hand system (14). Let $\mathcal{I} \subseteq 2^{[m]}$ be the set of all sets $I \subseteq [m]$ for which this system is feasible or the constraint propagation rules do *not* detect its infeasibility.[11] Consequently, the set of fixed points of Algorithm 2 with such constraint propagation rules is

$$\{y \in Y \mid I^*(y) \in \mathcal{I}\} = \bigcup_{I \in \mathcal{I}} \{y \in Y \mid I^*(y) = I\} = \bigcup_{I \in \mathcal{I}'} \operatorname{ri} F_I \tag{19}$$

10. For completeness, if this assumption is not satisfied, then we have $\operatorname{ri} F_I = \{y \in Y \mid I^*(y) = I'\}$ where $I' = \{i \in [m] \mid y \in F_I \implies y_i = 0\}$ (Dlask, 2022, Section 1.1.3.1), (Greenberg, 1996, Theorem 8).
11. The existence of such a set $\mathcal{I}$ implicitly implies that the constraint propagation rules are deterministic, i.e., whether they detect infeasibility or not depends only on $I \subseteq [m]$.

where $\mathcal{I}' = \{I \in \mathcal{I} \mid \{y \in Y \mid I^*(y) = I\} \neq \emptyset\}$. Therefore, the set of fixed points (19) is the union of relative interiors of some faces of Y. Note that $\{\operatorname{ri} F_I \mid I \subseteq [m],\ F_I \neq \emptyset\}$ is a partition of Y (Rockafellar, 1972, Theorem 18.2) and the set of fixed points (19) is thus formed as a union of some elements of this partition, which depends only on the chosen propagation rules.

These results can be further extended if a natural assumption on the constraint propagation rules is adopted, namely that $I \in \mathcal{I}$ implies $I' \in \mathcal{I}$ for all $I' \supseteq I$. This means, if the constraint propagation rules do not detect infeasibility of the left-hand system (14), then they will not detect infeasibility even if the set I is enlarged, i.e., some of the equalities (14c) are relaxed to inequalities. With this assumption, the set of fixed points (19) can be also expressed as

$$\{y \in Y \mid I^*(y) \in \mathcal{I}\} = \bigcup_{I \in \mathcal{I}} \bigcup_{I' \supseteq I} \{y \in Y \mid I^*(y) = I'\} = \bigcup_{I \in \mathcal{I}} \{y \in Y \mid I^*(y) \supseteq I\} = \bigcup_{I \in \mathcal{I}} F_I, \tag{20}$$

i.e., as the union of some faces of Y.

Remark 6. *The result stated by Dlask and Werner (2023, Appendix B), which connects the B-consistent sets (in the sense of Definition 3) and faces of a certain polyhedron is not related to the discussion in this section. First, in Dlask and Werner (2023), these are faces of a polyhedron which is a superset of the feasible set of the primal whereas we consider faces of the feasible set of the dual. Second, the results stated here apply to any constraint propagation method, not only activity propagation.*

Remark 7. *A method related to our iterative scheme from Section 4 is* block-coordinate descent *(BCD). It is also initialized at a dual-feasible point y and iteratively improves it. In each iteration, BCD chooses a block $B \in \mathcal{B}$ of dual variables (where $\mathcal{B} \subseteq 2^{[m]}$ is a pre-defined set of blocks) and optimizes the dual over the variables y_i, $i \in B$ while keeping the other dual variables y_i, $i \notin B$ constant. By choosing different blocks, the dual objective gradually improves until a fixed point is reached. Except for special cases (Zadeh, 1970; Bertsekas, 1997; Beck, 2014; Tseng, 2001; Dlask & Werner, 2022), the fixed points of BCD need not be global optima of the dual. We proved in Dlask and Werner (2023) that if activity propagation (see Section 3.3) is applied to the left-hand system* (14) *such that the set $\mathcal{B}$ in activity propagation coincides with the set of coordinate blocks in BCD, then BCD and Algorithm 2 have identical fixed points.*

4.2 Finiteness and Capacity Scaling

As already mentioned, Algorithm 2 need not in general terminate in a finite number of iterations. In this section, we state sufficient conditions for its finiteness.

First, notice that the improving direction $\bar{y}$ used on line 5 of Algorithm 2 can be chosen as any vector satisfying the right-hand system (14). To guarantee finiteness, we require a technical assumption, namely that there exists a finite set $\bar{Y}$ such that any improving direction used by Algorithm 2 is from this set. This condition is not satisfied trivially because the same set $I^*(y)$ may be encountered repeatedly during the run of Algorithm 2 and, in each such an iteration, the improving direction $\bar{y}$ may be chosen as a different vector satisfying the right-hand system (14).

The following theorem shows that if there also exists a positive lower bound on the step sizes used in Algorithm 2, then the algorithm terminates after a finite number of iterations.

Theorem 6. *Let the dual* (12) *be feasible and bounded. Let there exist a constant* $\alpha_{\min} > 0$ *and a finite set* $\bar{Y} \subseteq \mathbb{R}^m$ *such that in every iteration of Algorithm 2:*

(a) improving direction $\bar{y}$ *found on line 5 belongs to* $\bar{Y}$*,*

(b) step size α *computed on line 6 satisfies* $\alpha \geq \alpha_{\min}$*.*

Then the algorithm terminates after a finite number of iterations (i.e., updates of y*).*

Proof. We follow a proof technique analogous to the one in Dlask (2018, Section 3.2.4): we show that there is a value $\Delta > 0$ that depends only on the instance (i.e., on A, b, c) such that the dual objective improves at least by Δ in each iteration. Thus, if the dual has an optimal solution y^* and the algorithm is initialized at y, it terminates after at most $\lfloor (b^T y - b^T y^*)/\Delta \rfloor$ iterations.

Without loss of generality, we assume that $b^T \bar{y} < 0$ for each $\bar{y} \in \bar{Y}$. Indeed, if $b^T \bar{y} \geq 0$ for some $\bar{y} \in \bar{Y}$, then $\bar{y}$ can be removed from $\bar{Y}$ as it is never used by the algorithm because it does not satisfy the right-hand system (14) for any $I \subseteq [m]$. With this assumption, the objective improves in each iteration at least by $\Delta = \min_{\bar{y} \in \bar{Y}} (-\alpha_{\min} b^T \bar{y})$ because after any update from y to $y + \alpha \bar{y}$ we have $b^T y - b^T (y + \alpha \bar{y}) = -\alpha b^T \bar{y} \geq -\alpha_{\min} b^T \bar{y} \geq \Delta$.

For completeness, Δ is not well-defined if $\bar{Y} = \emptyset$. But, in such case, Algorithm 2 always terminates already with the initial point as it cannot use any improving direction. □

We will now discuss how the assumptions of Theorem 6 can be satisfied in practice. We begin with condition (a).

Because the set $[m]$ has only a finite number of subsets, the existence of a finite set $\bar{Y}$ satisfying condition (a) is equivalent to the following: for each $I \subseteq [m]$ there exists a finite set $\bar{Y}_I \subseteq \{\, \bar{y} \in \mathbb{R}^m \mid \bar{y} \text{ satisfies the right-hand system (14)} \,\}$ such that, in any iteration, the improving direction chosen on line 5 in Algorithm 2 is from the set $\bar{Y}_{I^*(y)}$ where y is the current dual-feasible solution. See that $\bar{Y}_I = \emptyset$ if the left-hand system (14) is feasible because then right-hand system is infeasible. Moreover, if we are unable to prove infeasibility of the left-hand system for some $I = I^*(y)$, then one can set $\bar{Y}_I = \emptyset$.

In particular, if the method for constructing the improving direction on line 5 in Algorithm 2 is deterministic and depends only on I, then each set $\bar{Y}_I$ is either empty or a singleton, so $\bar{Y} = \bigcup_{I \subseteq [m]} \bar{Y}_I$ is finite and satisfies condition (a) in Theorem 6.

To satisfy condition (b), we introduce a heuristic analogous to *capacity scaling* in maximum flow algorithms (Magnanti, Ahuja, & Orlin, 1993, Section 7.3). This heuristic has been used to ensure finiteness of the Augmenting DAG algorithm (Koval & Schlesinger, 1976; Werner, 2007, 2005), VAC algorithm (Cooper et al., 2010, Section 11.1),[12] and later by Werner (2017, Equation (21)) and Dlask (2018, Section 3.1.7) in a more general setting of minimizing unconstrained convex piecewise-affine functions. Let

$$I^*_\theta(y) = \{ i \in [m] \mid y_i \leq \theta \} \tag{21}$$

12. To be more precise, Cooper et al., 2010 ensured finiteness of VAC by stopping if the bound does not sufficiently improve in several consecutive iterations. Capacity scaling was used only for improving the speed of the algorithm, but the authors presented an example where the VAC algorithm can enter an infinite loop if neither capacity scaling nor the previously mentioned heuristic is used. However, we discuss later in Remark 8 that finiteness of the VAC algorithm can be ensured by capacity scaling only.

denote the set of dual constraints (12b) that are 'almost' active, i.e., active up to a threshold $\theta \geq 0$. Clearly, $I^*_\theta(y) = I^*(y)$ for $\theta = 0$ (see (13)). If $I^*(y)$ is replaced by $I^*_\theta(y)$ on line 2 in Algorithm 2, the scheme remains valid. In detail, if the left-hand system (14) is infeasible for $I = I^*_\theta(y)$, then it is also infeasible for $I = I^*(y)$ due to $I^*(y) \subseteq I^*_\theta(y)$ for any $\theta \geq 0$. This is equivalent to the fact that any $\bar{y}$ feasible for the right-hand system (14) with $I = I^*_\theta(y)$ is also feasible for this system with $I = I^*(y)$.

Next, we prove that, if the computed step size on line 6 of Algorithm 2 is optimal and $I^*(y)$ on line 2 is replaced by $I^*_\theta(y)$ for some constant $\theta > 0$, then there exists a positive lower bound on the computed step sizes. Thus, condition (b) in Theorem 6 is implied by the aforementioned conditions which yields the next theorem.

Theorem 7. *Let the dual* (12) *be feasible and bounded. Suppose that*

(a) there exists a finite set $\bar{Y} \subseteq \mathbb{R}^m$ such that in every iteration, improving direction $\bar{y}$ found on line 5 belongs to $\bar{Y}$,

(b) $I^*(y)$ on line 2 is replaced by $I^*_\theta(y)$ where $\theta > 0$ is a constant,

(c) the step size computed on line 6 is optimal, i.e., $\alpha = \alpha^*$ where α^* is (15)*.*

Then Algorithm 2 terminates after a finite number of iterations (i.e., updates of y).

Proof. We prove that there exists $\alpha_{\min} > 0$ such that for any α computed by the algorithm (with the modifications assumed in this theorem), it holds that $\alpha \geq \alpha_{\min}$. The claim will then follow from Theorem 6. Analogously to the proof of Theorem 6, we assume (without loss of generality) that for any $\bar{y} \in \bar{Y}$, there exists dual-feasible y such that $\bar{y}$ satisfies the right-hand system (14) for $I = I^*_\theta(y)$.

Let us define

$$\delta = \max_{\bar{y} \in \bar{Y}} \max_{i \in [m] \colon \bar{y}_i < 0} -\bar{y}_i \tag{22}$$

so that $\delta \geq -\bar{y}_i > 0$ for any $\bar{y}$ used by Algorithm 2 and any i considered in definition of step size (15). Clearly, δ is positive and well-defined because the maxima are always taken over finite sets. The case with $\bar{Y} = \emptyset$ is treated as in the proof of Theorem 6. On the other hand, if $\bar{Y} \neq \emptyset$ but all $\bar{y} \in \bar{Y}$ satisfy $\bar{y} \geq 0$, then the dual is unbounded, which violates our assumption on its boundedness.

Now, we claim that $\alpha_{\min} = \theta/\delta$ is positive (because $\theta > 0$ and $\delta > 0$) and constitutes a lower bound on the step sizes computed by the algorithm. To show this, let y be any feasible solution for the dual and $\bar{y} \in \bar{Y}$ satisfy the right-hand system (14) for $I = I^*_\theta(y)$. We claim that $\alpha^* \geq \alpha_{\min}$, i.e., for all $i \in [m]$ with $\bar{y}_i < 0$, $-y_i/\bar{y}_i \geq \alpha_{\min}$. Indeed, as discussed in the proof of Proposition 2, if $\bar{y}_i < 0$, then $i \notin I = I^*_\theta(y)$ due to the left-hand constraints (14b), so $y_i > \theta$ by definition of $I^*_\theta(y)$. Combined with $\delta \geq -\bar{y}_i$, the claim follows. □

Note, instead of using a single fixed $\theta > 0$, one can run the iterative algorithm multiple times, each time with a lower value of θ, starting from the previously attained fixed point and thus gradually improving the current feasible solution y even further, as it was done by Cooper et al. (2010, Section 11.1) and later by Dlask (2018, Algorithm 14) or Dlask, Werner, and de Givry (2023). In this way, we obtain an anytime algorithmic scheme outlined in Algorithm 3. It is experimentally observed that this modification results in larger step sizes and faster decrease of the objective (Cooper et al., 2010; Dlask, 2018).

inputs: instance of problem (12), dual-feasible solution y, initial threshold $\theta > 0$, constraint propagation rules for left-hand system (14).
while θ is not small enough or time limit is not reached **do**
  Improve y by Algorithm 2 with $I^*(y)$ replaced by $I^*_\theta(y)$ on line 2.
  Decrease θ while keeping $\theta > 0$ (e.g., $\theta := \theta/10$).
return y

Algorithm 3: Approximate optimization of the dual (12) with gradually decreasing threshold θ.

4.2.1 Convergence

As in Section 4.1, let us for simplicity fix an arbitrary method for detecting infeasibility of the left-hand system (14). Again, we will denote by $\mathcal{I} \subseteq 2^{[m]}$ the set of all $I \subseteq [m]$ such that this system is feasible or the propagation algorithm does *not* detect its infeasibility.

Consider the function $\Theta\colon Y \to \mathbb{R}_+$ (where Y is the feasible set (16) of the dual) defined by[13]

$$\Theta(y) = \min\{\theta \in \mathbb{R}_+ \mid I^*_\theta(y) \in \mathcal{I}\}, \tag{23}$$

which can be interpreted as a distance of the point y to the set of fixed points (19) because y is a fixed point of Algorithm 2 if and only if $\Theta(y) = 0$.

To discuss convergence, let us assume that the main loop in Algorithm 3 is repeated infinitely, producing an infinite sequence $(y^k)_{k\in\mathbb{N}}$ of dual-feasible solutions y. Similarly, let $(\theta_k)_{k\in\mathbb{N}}$ be the decreasing sequence of positive values θ computed by Algorithm 3. It is easy to see that if $\lim_{k\to\infty} \theta_k = 0$, then

$$\lim_{k\to\infty} \Theta(y^k) = 0. \tag{24}$$

Indeed, for all $k \in \mathbb{N}$ we have $I^*_{\theta_k}(y^k) \in \mathcal{I}$, so $0 \le \Theta(y^k) \le \theta_k$. Note, the assumption on convergence of θ_k towards zero is naturally satisfied, e.g., by using the geometric decrease schedule $\theta_{k+1} := \theta_k/10$, as on line 3 of Algorithm 3.

The interpretation of (24) is that the distance between y^k and the set of fixed points (19) (measured by Θ) converges to zero, i.e., y^k converges to the set of the fixed points. In particular, if the constraint propagation rules are refutation complete (i.e., the chosen method is able to detect infeasibility of the left-hand system (14) whenever it is infeasible), then y^k converges to the set of optima of the dual (12).

In the following two sections, we exemplify the proposed approach on LP relaxations of two combinatorial problems. Namely, in Section 5, we show that the VAC algorithm (Cooper et al., 2010) is its special case and, in Section 6, we apply the approach to the LP relaxation of Weighted Max-SAT. These LP relaxations involve equality constraints and/or non-negative variables and even though they could be transformed to the general form (12)

13. The function Θ is well-defined because, for any $y \in Y$, there exists a sufficiently large θ such that $I^*_\theta(y) \in \mathcal{I}$, namely for $\theta = \max_{i\in[m]} y_i$ we have $I^*_\theta(y) = [m] \in \mathcal{I}$ (because feasibility of the left-hand system (14) for $I = [m]$ is equivalent to feasibility of the primal). A similar function has been used in convergence analysis of BCD algorithms for the LP relaxation of WCSP (Schlesinger & Antoniuk, 2011), (Tourani, Shekhovtsov, Rother, & Savchynskyy, 2018, Appendix B), (Savchynskyy, 2019, Section 6.2.4).

by well-known tricks, such as replacing an equality with two inequalities or adding slack variables (Matoušek & Gärtner, 2006; Leiserson, Rivest, Cormen, & Stein, 1994), it will be more convenient to adapt the basic approach described here to these cases. Indeed, our approach is applicable to a primal-dual pair in any form and all of our theoretical results from this section also generalize in this way. E.g., the analogous results for the case where the roles of the primal and dual are interchanged were presented in the dissertation by Dlask (2022, Section 2.2).

5. LP Relaxation of Weighted CSP

In this section, we define the Weighted Constraint Satisfaction Problem (WCSP) and show that the well-known Virtual Arc Consistency (VAC) algorithm by Cooper et al. (2010) is subsumed by our iterative scheme.

To keep the presentation concise, we consider only binary WCSPs (i.e., WCSPs with unary and binary constraints) and assume that no constraint is hard (i.e., with weight $-\infty$) – generalizing the analysis to WCSPs of any arity and/or with hard constraints would be straightforward.[14] An instance of such WCSP is defined by (V, D, E, c) where V is a finite set of variables, D is a finite domain, $E \subseteq \binom{V}{2}$ is a set of variable pairs (so that (V, E) is an undirected graph) and $c \in \mathbb{R}^P$ is a cost vector where

$$P \;=\; \{(u,k) \mid u \in V,\; k \in D\} \;\cup\; \{\{(u,k),(v,l)\} \mid \{u,v\} \in E,\; k,l \in D\} \tag{25}$$

denotes the set of *tuples*. The component of c with index $j \in P$ will be abbreviated by $c_j = c_{uk}$ for a unary tuple $j = (u,k)$ and by $c_j = c_{uk,vl}$ for a binary tuple $j = \{(u,k),(v,l)\}$, understanding that $c_{uk,vl} = c_{vl,uk}$. The goal is to find an assignment $\lambda\colon V \to D$ maximizing[15] the objective

$$\sum_{u \in V} c_{u\lambda(u)} + \sum_{\{u,v\} \in E} c_{u\lambda(u),v\lambda(v)}. \tag{26}$$

The LP relaxation of the binary WCSP can be written[16] as the left-hand problem of the primal-dual pair

$$\begin{aligned}
\max\; & c^T x && \min\; \sum_{u \in V} y_u && && \text{(27a)}\\
\sum_{l \in D} x_{uk,vl} - x_{uk} &= 0 && y_{uk,v} \in \mathbb{R} && \forall u \in V,\; v \in N_u,\; k \in D && \text{(27b)}\\
\sum_{k \in D} x_{uk} &= 1 && y_u \in \mathbb{R} && \forall u \in V && \text{(27c)}\\
x_{uk} &\geq 0 && y_u - \sum_{v \in N_u} y_{uk,v} \geq c_{uk} && \forall u \in V,\; k \in D && \text{(27d)}\\
x_{uk,vl} &\geq 0 && y_{uk,v} + y_{vl,u} \geq c_{uk,vl} && \forall \{u,v\} \in E,\; k,l \in D && \text{(27e)}
\end{aligned}$$

14. Hard constraints can be naturally modelled by removing the corresponding primal variables and dual constraints or setting the corresponding costs to a negative number large in absolute value.
15. To fit our framework, we assume that the WCSP is a maximization problem so that we could minimize in the dual (27), as in Section 4. The only difference lies in inverting the sign of the weights c.
16. While the basic LP relaxation of WCSP can be written in several different ways (Werner, 2007; Živný, 2012; Savchynskyy, 2019), we chose the form whose dual coincides with the one in the VAC paper (Cooper et al., 2010, Figure 2). The dual linear program in (27) can be interpreted as minimizing an upper bound on the WCSP optimal value over *equivalent transformations* (a.k.a. *reparametrizations*) of the cost vector c (Cooper et al., 2010; Wainwright & Jordan, 2008; Savchynskyy, 2019).

where $c^T x = \sum_{j \in P} c_j x_j = \sum_{u \in V} \sum_{k \in D} c_{uk} x_{uk} + \sum_{\{u,v\} \in E} \sum_{k,l \in D} c_{uk,vl} x_{uk,vl}$ and $N_u = \{v \in V \mid \{u,v\} \in E\}$ is the set of neighbors of variable $u \in V$. In matrix form, (27) reads

$$\max c^T x \qquad \min b^T y \tag{28a}$$
$$Ax = b \qquad y \in \mathbb{R}^Q \tag{28b}$$
$$x \geq 0 \qquad A^T y \geq c \tag{28c}$$

where the matrix A and vector b are determined by (27) and $Q = \{(u,k,v) \mid u \in V,\ v \in N_u,\ k \in D\} \cup V$ is the index set of the dual variables y.

Given a dual-feasible solution y, the complementary slackness conditions can be written as the left-hand system of

$$b^T \bar{y} < 0 \tag{29a}$$
$$Ax = b \qquad \bar{y} \in \mathbb{R}^P \tag{29b}$$
$$x_j \geq 0 \qquad A_j^T \bar{y} \geq 0 \qquad \forall j \in J \tag{29c}$$
$$x_j = 0 \qquad \forall j \in P \setminus J \tag{29d}$$

for $J = J^*(y)$ where A_j denotes the column of A corresponding to tuple j and $J^*(y) \subseteq P$ the set of the dual constraints (28c) (i.e., (27d)-(27e)) active at y. On the right, we again wrote Farkas' alternative system, so that any $\bar{y}$ satisfying this system is a dual-improving direction from the current point y. The left-hand system (29) is the LP relaxation of the CSP with allowed tuples J (Cooper et al., 2010; Nguyen, Bessiere, de Givry, & Schiex, 2017; Trösser, de Givry, & Katsirelos, 2020; Werner, 2007), which we denote by (V, D, E, J).[17]

5.1 Constraint Propagation

Given a dual-feasible solution $y \in \mathbb{R}^Q$, we want to apply constraint propagation to the left-hand system (29). In our particular case, we will propagate zero values of variables x (i.e., inferring equalities $x_j = 0$, i.e., inferring that some of inequalities (29c) are always active) based on a small subset of marginalization constraints (27b). E.g., if we know that $x_{uk} = 0$ for some $(u,k) \in P$, then equality (27b) implies (assuming $x \geq 0$) that $x_{uk,vl} = 0$ for all $v \in N_u$ and $l \in D$. If for some $u \in V$ we get $x_{uk} = 0$ for all $k \in D$ ('domain wipe-out'), then the normalization constraint (27c) is violated, which proves that the initial system was infeasible. The propagation algorithm starts with $J = J^*(y)$ and then iteratively sets some variables to zero by removing tuples from J according to (29d).

Let us now focus on deriving the cause vectors. While above we were setting variables to zero by removing elements from J, here we will do it by adding a suitable (in)equality to the system (as in Section 2). As our system is in the form of linear equalities over non-negative variables, we will follow Theorem 3.[18] We will not directly infer an equality $x_j = 0$ (or

17. This notation for CSP is non-standard: the set J of allowed tuples is usually represented by a set of unary relations (equivalently, domains of the variables) and binary relations.

18. An alternative approach would be to treat the left-hand system (29) as a system of linear inequalities over unconstrained variables (by replacing each equality with two inequalities with opposite directions) and apply Farkas' lemma in the general form of Theorem 1. However, this would lead to more complex derivation of the cause vectors.

inequality $x_j \leq 0$) as a linear combination of all existing (in)equalities, but instead infer a more complex equality as a linear combination of equalities $Ax = b$ which nevertheless implies $x_j = 0$. Namely, we infer an equality

$$x_j + t_j = 0 \tag{30}$$

where t_j is a linear combination of the variables x such that $t_j \geq 0$ holds for all vectors $x \geq 0$ satisfying $x_{j'} = 0$ for all $j' \notin J^*(y)$. Under these conditions and assuming $x \geq 0$, (30) clearly implies $x_j = 0$. The cause vector (the coefficients of a linear combination of the equalities $Ax = b$) of equality (30) is denoted $y^j \in \mathbb{R}^P$. Since a single equality (30) can imply multiple x-variables to be zero, it can happen that $y^j = y^{j'}$ for some $j \neq j'$. Equality (30) trivially (by (29d)) implies $x_j = 0$ also for $j \notin J^*(y)$, in which case we simply set $t_j = -x_j$. Then (30) reduces to the equality $0 = 0$, which is the trivial linear combination of equalities $Ax = b$ with cause vector $y^j = 0$.

Below, we describe the propagation rules in detail. Although we are applying constraint propagation to the left-hand system (29) (which is a CSP with continuous variables), the propagation corresponds to enforcing certain well-known local consistencies for the CSP (V, D, E, J) (which has finite domains), namely pairwise consistency (PWC) and arc consistency (AC) (Janssen, Jégou, Nouguier, & Vilarem, 1989; Bessiere, 2006) (more in Remark 10). We first describe the rules for PWC and then for AC. Although PWC and AC have equal power for binary CSPs, we describe them both to better illustrate the derivations of cause vectors. Let us remark that it would be straightforward to extend the PWC and AC propagation rules to CSPs with constraints of any arity, but we do not present this extension as it would require more complex notation.

We denote the standard-basis vector of the space $\mathbb{R}^P$ corresponding to (27b) and (27c) by $e^{uk,v}$ and e^u, respectively. While elements are removed from J only to keep track which variables have been set to zero, the inference tables (such as (31)) are understood as adding equalities to the system (29) with $J = J^*(y)$.

5.1.1 Pairwise Consistency

Definition 5. *A binary CSP (V, D, E, J) is* pairwise consistent (PWC) *if*[19] *$(u, k) \in J$ if and only if for each $v \in N_u$ there is $l \in D$ such that $\{(u, k), (v, l)\} \in J$.*

PWC is enforced by iterating the following propagation rules. We always first state a rule and then derive it from a suitable linear combination of existing equalities, which provides the cause vector:

1. If there is $\{(u, k), (v, l)\} \in J$ such that $(u, k) \notin J$, then we remove the tuple $\{(u, k), (v, l)\}$ from J. This rule can be derived as follows:

$$x_{uk} + t_{uk} = 0 \qquad\qquad y^{uk} \tag{31a}$$

$$\sum_{l' \in D} x_{uk,vl'} - x_{uk} = 0 \qquad\qquad e^{uk,v} \tag{31b}$$

19. While PWC in general requires consistency of all possible constraint pairs (with overlapping scopes), it is known that for binary CSPs with all unary constraints (our case), this is equivalent to consistency only of all constraint pairs consisting of a unary and a binary constraint.

$$x_{uk,vl} + \underbrace{\sum_{l' \in D \setminus \{l\}} x_{uk,vl'} + t_{uk}}_{t_{uk,vl}} = 0 \qquad y^{uk,vl} = y^{uk} + e^{uk,v} \tag{31c}$$

Equality (31a) represents the equality $x_{uk} = 0$, which holds by the assumption $(u,k) \notin J$. Equality (31b) is the marginalization constraint (27b). Finally, (31c) is the new inferred equality. We wrote the corresponding cause vectors on the right.

2. If there are $(u,k) \in J$ and $v \in N_u$ such that $\{(u,k),(v,l)\} \notin J$ for all $l \in D$, then we remove the tuple (u,k) from J. This is inferred as follows:

$$\begin{aligned} x_{uk,vl} + t_{uk,vl} &= 0 && y^{uk,vl} && \forall l \in D && (32a)\\ -\sum_{l \in D} x_{uk,vl} + x_{uk} &= 0 && -e^{uk,v} && && (32b)\\ \hline x_{uk} + \underbrace{\sum_{l \in D} t_{uk,vl}}_{t_{uk}} &= 0 && y^{uk} = \sum_{l \in D} y^{uk,vl} - e^{uk,v} && && (32c) \end{aligned}$$

where (32a) represents the equalities $x_{uk,vl} = 0$ (which hold because $\{(u,k),(v,l)\} \notin J$) for all $l \in D$ and (32b) is the marginalization constraint (27b) multiplied by -1. In the inferred equality, we have $t_{uk} \geq 0$, which implies $x_{uk} = 0$.

3. If for some $u \in V$ it happens that $(u,k) \notin J$ for all $k \in D$, then the CSP is infeasible due to domain wipe-out. This is inferred as follows:

$$\begin{aligned} x_{uk} + t_{uk} &= 0 && y^{uk} && \forall k \in D && (33a)\\ -\sum_{k \in D} x_{uk} &= -1 && -e^{u} && && (33b)\\ \hline \sum_{k \in D} t_{uk} &= -1 && \bar{y} = \sum_{k \in D} y^{uk} - e^{u} && && (33c) \end{aligned}$$

where (33b) is the normalization constraint (27c) multiplied by -1. The inferred equality (33c) is infeasible because $t_{uk} \geq 0$ for all k. The derived vector $\bar{y}$ in (33c) is a solution to the right-hand system (29).

The whole process of propagation using the three PWC rules above is summarized in Algorithm 4. If the rules detect infeasibility of the left-hand system (29), the algorithm returns the computed infeasibility certificate satisfying the right-hand system (29).

5.1.2 Arc Consistency

Definition 6. *A binary CSP (V,D,E,J) is* arc consistent (AC) *if for each $(u,k) \in J$ and $v \in N_u$ there is $l \in D$ such that $\{(u,k),(v,l)\} \in J$ and $(v,l) \in J$.*

AC is enforced by iterating the following propagation rule (besides the last PWC rule):

inputs: V, D, E and $J \subseteq P$
For each $j \in P \setminus J$, set $y^j := 0$.
while the set J changes
  if there is $\{(u,k),(v,l)\} \in J$ with $(u,k) \notin J$ **then**
    Remove $\{(u,k),(v,l)\}$ from J. Set $y^{uk,vl} := y^{uk} + e^{uk,v}$.
  if there are $(u,k) \in J$ and $v \in N_u$ with $\{(u,k),(v,l)\} \notin J$ for all $l \in D$ **then**
    Remove (u,k) from J. Set $y^{uk} := \sum_{l \in D} y^{uk,vl} - e^{uk,v}$.
  if there is $u \in V$ with $(u,k) \notin J$ for all $k \in D$ **then**
    return $\bar{y} = \sum_{k \in D} y^{uk} - e^u$. (Domain wipe-out, with certificate $\bar{y}$.)

Algorithm 4: PWC propagation in LP relaxation of CSP (V, D, E, J).

1. If there are $(u,k) \in J$ and $v \in N_u$ such that $\{(u,k),(v,l)\} \notin J$ or $(v,l) \notin J$ for all $l \in D$, then we remove the tuple (u,k) from J. This rule is derived as follows:

$$x_{uk,vl} + t_{uk,vl} = 0 \qquad y^{uk,vl} \qquad \forall l \in D\colon \{(u,k),(v,l)\} \notin J \tag{34a}$$

$$x_{vl} + t_{vl} = 0 \qquad y^{vl} \qquad \forall l \in D\colon \{(u,k),(v,l)\} \in J \tag{34b}$$

$$\sum_{k' \in D} x_{uk',vl} - x_{vl} = 0 \qquad e^{vl,u} \qquad \forall l \in D\colon \{(u,k),(v,l)\} \in J \tag{34c}$$

$$-\sum_{l \in D} x_{uk,vl} + x_{uk} = 0 \qquad -e^{uk,v} \tag{34d}$$

$$x_{uk} + t_{uk} = 0 \qquad y^{uk} \tag{34e}$$

where

$$y^{uk} = \sum_{\substack{l \in D \\ \{(u,k),(v,l)\} \notin J}} y^{uk,vl} + \sum_{\substack{l \in D \\ \{(u,k),(v,l)\} \in J}} (y^{vl} + e^{vl,u}) - e^{uk,v} \tag{35}$$

and

$$t_{uk} = \sum_{\substack{l \in D \\ \{(u,k),(v,l)\} \notin J}} (x_{uk,vl} + t_{uk,vl}) + \sum_{\substack{l \in D \\ \{(u,k),(v,l)\} \in J}} \Big(x_{vl} + t_{vl} + \sum_{k' \in D} x_{uk',vl} - x_{vl} \Big) - \sum_{l \in D} x_{uk,vl} \tag{36a}$$

$$= \sum_{\substack{l \in D \\ \{(u,k),(v,l)\} \notin J}} t_{uk,vl} + \sum_{\substack{l \in D \\ \{(u,k),(v,l)\} \in J}} \Big(t_{vl} + \sum_{k' \in D \setminus \{k\}} x_{uk',vl} \Big). \tag{36b}$$

We have $t_{uj} \geq 0$ as required, because all terms in (36b) are non-negative. Note, we used that for each $l \in D$ we have $(v,l) \notin J$ or $\{(u,k),(v,l)\} \notin J$ by assumption.

5.2 Final Algorithm

When plugging the constraint propagation described in Section 5.1 into Algorithm 2, we obtain an algorithm that is 'almost' equivalent to the VAC / Augmenting DAG algorithm (Cooper et al., 2010; Koval & Schlesinger, 1976; Werner, 2007). The stopping points of both algorithms are characterized by the same property, namely non-empty AC/PWC

closure of the CSP $(V, D, E, J^*(y))$. However, improving directions $\bar{y}$ constructed in Section 5.1 are in general different from the ones constructed by Cooper et al. (2010), Koval and Schlesinger (1976), Werner (2007), sometimes having larger absolute values of their components and thus possibly leading to smaller step sizes α. We illustrate such a case in detail in Appendix B. The reason is that our algorithm does not take into account the values of the individual coefficients in the cause vectors y^j. In contrast, the algorithms in (Cooper et al., 2010; Koval & Schlesinger, 1976; Werner, 2007) compute the cause vectors in an anti-chronological order after the contradiction has been detected, rather than fixing them already when inferring individual equalities. This is the same difference which we discussed previously in Section 3.3 considering the chronological and anti-chronological approaches for computing certificates of infeasibility. The latter approach (Dlask & Werner, 2023, Appendix A) is a generalization of the algorithms in (Cooper et al., 2010; Koval & Schlesinger, 1976; Werner, 2007) to a general system in the form $Ax = b$, $x \geq 0$. Another method using the anti-chronological computation of cause vectors was proposed by Dlask et al. (2023, Section 4.2.2), which is however applicable only to a certain class of linear programs. It may be an important direction for future research to compare and study the quality of the improving directions produced by the chronological and anti-chronological approach or propose even other methods.

Remark 8. *It is known (Cooper et al., 2010, Appendix A) that the VAC algorithm without capacity scaling can enter an infinite loop. In order to avoid this, one can replace the set of active tuples $J^*(y)$ by 'almost' active tuples $J^*_\theta(y)$, analogously to* (21) *(see Cooper et al., 2010, Section 11.1).*

Despite finiteness of the VAC algorithm was already ensured by Cooper et al. (2010), we would like to point out that Theorem 7 is easily applicable here. First, both the primal and the dual (28) *are clearly feasible and bounded. Second, despite there might be many orders in which the PWC or AC propagation rules can be applied to the CSP $J^*_\theta(y)$, the set of tuples P is finite, so there is only finitely many ways in which the rules can be applied to each CSP. For each such an order, the improving direction $\bar{y}$ is constructed deterministically (both by the VAC algorithm or by our propagation rules above). Finally, using $\theta > 0$ and optimal step size, both of these approaches will terminate in finite time.*

Remark 9. *Since non-empty AC/PWC closure of the CSP $(V, D, E, J^*(y))$ is in general not sufficient for optimality of y for the dual* (28) *(Werner, 2007, Section 5), our propagation algorithm is refutation incomplete, i.e., it may not detect infeasibility of the left-hand system* (29) *in some cases. Consequently, the VAC algorithm and the algorithm outlined above may terminate in a non-optimal point y.*

Remark 10. *The words 'arc consistency' in VAC refer to enforcing arc consistency in the CSP $(V, D, E, J^*(y))$ where a tuple $j \in P$ is forbidden in the CSP implicitly by setting the variable x_j to zero. However, one can see this also as enforcing arc consistency w.r.t. the continuous constraints* (27b)-(27c)*. By* (29c)-(29d)*, the initial domains of the variables x_j are either $D_j = \{0\}$ (if the variable is set to zero, i.e., $j \notin J^*(y)$) or $D_j = \mathbb{R}_+$ (otherwise). Enforcing arc consistency w.r.t. a marginalization constraint* (27b) *then shrinks some of the domains from $\mathbb{R}_+$ to $\{0\}$. In this case (see Proposition 1, cf. Appendix A), enforcing arc consistency is not only equivalent to propagation of bounds but also to activity propagation*

since inferring an equality $x_j = 0$ can be seen as inferring that the inequality $x_j \geq 0$ is always active.

Remark 11. *The fact that there exists y such that the CSP $(V, D, E, J^*(y))$ is PWC/AC (or has a non-empty PWC/AC closure) is given by the properties of the LP relaxation (27) (Werner, 2007, 2010). Not every local consistency of this CSP can be enforced in this way unless a different linear program is used (Dlask et al., 2023) or additional constraints are introduced (Batra, Nowozin, & Kohli, 2011; Sontag, Choe, & Li, 2012; Sontag, Meltzer, Globerson, Jaakkola, & Weiss, 2008; Werner, 2014, 2010).*

6. LP Relaxation of Weighted Max-SAT

In this section, we show how the iterative constraint-propagation-based scheme from Section 4 can be applied to the LP relaxation of the Weighted Max-SAT problem. In this problem, we are given a finite set V of Boolean variables and a finite set C of clauses with positive weights w_c, $c \in C$. The task is to find an assignment to the variables such that the sum of weights of satisfied clauses is maximized. Let V_c^+ and V_c^- denote the set of variables that occur in clause $c \in C$ non-negated and negated, respectively. Let $C_i^\pm = \{\, c \in C \mid i \in V_c^\pm \,\}$ denote the set of clauses where variable $i \in V$ occurs non-negated/negated.

We consider the classical LP relaxation of Weighted Max-SAT (Vazirani, 2001, Section 16), (Biere, Heule, & van Maaren, 2009, Section 19.2) in a slightly different but equivalent[20] form (cf. Min-UNSAT in Biere et al., 2009). To adhere to the form of the primal-dual pair from Section 4, i.e., the dual being a minimization problem, we make the auxiliary primal variables z negative. Moreover, we represent each Boolean variable by two variables, x_i^+ for a non-negated and x_i^- for a negated Boolean variable. The relaxation is the left-hand problem of the primal-dual pair[21]

$$\begin{aligned}
\max\ & w^T z && \min\ p(V) - y(C) && && (37a)\\
z_c - x^+(V_c^+) - x^-(V_c^-) &\leq -1 && y_c \geq 0 && \forall c \in C && (37b)\\
z_c &\leq 0 && y_c \leq w_c && \forall c \in C && (37c)\\
x_i^+ + x_i^- &= 1 && p_i \lesseqgtr 0 && \forall i \in V && (37d)\\
x_i^+ &\geq 0 && p_i - y(C_i^+) \geq 0 && \forall i \in V && (37e)\\
x_i^- &\geq 0 && p_i - y(C_i^-) \geq 0 && \forall i \in V && (37f)
\end{aligned}$$

where we use the usual shortcuts $x^+(V_c^+) = \sum_{i \in V_c^+} x_i^+$ and $y(C_i^+) = \sum_{c \in C_i^+} y_c$, similarly for $x^-(V_c^-)$, $y(C_i^-)$, $y(C)$, and $p(V)$.

Note, for any upper bound u on the common optimal value of (37), $w(C)+u$ is an upper bound on the optimal value of the original (non-relaxed) Max-SAT problem. In particular,

20. In our conference paper (Dlask & Werner, 2020), we used the classical LP relaxation, called Max-SAT by Biere et al. (2009). Both relaxations have the same global optima and behave the same way w.r.t. the propagation algorithm. Precisely, using our result in (Dlask & Werner, 2023), it is easy to see that every pre-interior local minimum of problem (39) is a pre-interior local minimum of problem (11) in (Dlask & Werner, 2020).

21. The meaning of the primal constraint (37b) becomes clearer if it is multiplied by -1. However, we keep it in its current form so that the pair (37) complies to the standard primal-dual correspondence in linear programming (Matoušek & Gärtner, 2006, Section 6.2).

if u^* is the common optimal value of (37), then $w(C) + u^*$ is equal to the optimal value of the LP relaxation in the form considered by Vazirani (2001, Section 16).

Remark 12. *Despite we do not consider this explicitly, our approach can be also applied to instances of Weighted Partial Max-SAT which contain hard clauses (i.e., clauses that need to be satisfied) by removing z_c from the corresponding primal constraints* (37b)*, which leads to removing the dual constraint* (37c)*. Another, less elegant, alternative is to assign large weights to the hard clauses which does not require any changes in the algorithm. Comparing these approaches is subject to further research.*

At any primal and dual optimum, we clearly have

$$z_c = \max\{x^+(V_c^+) + x^-(V_c^-) - 1,\, 0\} \geq -1 \qquad \forall c \in C, \tag{38a}$$
$$p_i = \max\{y(C_i^+),\, y(C_i^-)\} \qquad \forall i \in V, \tag{38b}$$

respectively. Using (38b), the dual thus can be simplified to the problem

$$\min_{0 \leq y \leq w} \sum_{i \in V} \max\{y(C_i^+), y(C_i^-)\} - y(C), \tag{39}$$

in which we minimize a convex piecewise-affine function on a box. In the following, we will work with the dual in the form (39) because it simplifies the algorithm.

By eliminating variables z and p, the complementary slackness conditions for the pair (37) can be simplified to

$$-x^+(V_c^+) - x^-(V_c^-) \leq -1 \qquad \forall c \in C\colon y_c = 0 \tag{40a}$$
$$-x^+(V_c^+) - x^-(V_c^-) = -1 \qquad \forall c \in C\colon 0 < y_c < w_c \tag{40b}$$
$$-x^+(V_c^+) - x^-(V_c^-) \geq -1 \qquad \forall c \in C\colon y_c = w_c \tag{40c}$$
$$x_i^+ + x_i^- = 1 \qquad \forall i \in V \tag{40d}$$
$$x_i^+ \geq 0 \qquad \forall i \in I_+ \tag{40e}$$
$$x_i^+ = 0 \qquad \forall i \in V \setminus I_+ \tag{40f}$$
$$x_i^- \geq 0 \qquad \forall i \in I_- \tag{40g}$$
$$x_i^- = 0 \qquad \forall i \in V \setminus I_- \tag{40h}$$

for $I_+ = I_+^*(y)$ and $I_- = I_-^*(y)$ where

$$I_+^*(y) = \{i \in V \mid y(C_i^+) \geq y(C_i^-)\}, \tag{41a}$$
$$I_-^*(y) = \{i \in V \mid y(C_i^+) \leq y(C_i^-)\}. \tag{41b}$$

Thus, system (40) is feasible if and only if y is optimal for problem (39). If system (40) is infeasible, the following theorem shows how to conveniently obtain an improving direction for (39). Following Theorem 5, we dualize only some of the constraints, namely (40a)-(40c).

Theorem 8. *Let*

$$\sum_{c \in C} \bar{y}_c (1 - x^+(V_c^+) - x^-(V_c^-)) \leq 0 \tag{42}$$

be a linear combination of (in)equalities (40a)-(40c) *with coefficients $\bar{y} \in \mathbb{R}^C$ (where the signs of the components of $\bar{y}$ comply to the (in)equality signs in* (40a)-(40c)*: $\bar{y}_c \geq 0$*

for (40a), $\bar{y}_c \in \mathbb{R}$ *for* (40b), *and* $\bar{y}_c \leq 0$ *for* (40c)*). Inequality* (42) *is infeasible on conditions* (40d)-(40h) *if and only if* $\bar{y}$ *is an improving direction for problem* (39) *from the point* y*.*

Proof. The complementary slackness conditions for the pair (37) before eliminating z read

$$z_c - x^+(V_c^+) - x^-(V_c^-) \leq -1 \qquad \forall c \in C\colon y_c = 0 \tag{43a}$$
$$z_c - x^+(V_c^+) - x^-(V_c^-) = -1 \qquad \forall c \in C\colon y_c > 0 \tag{43b}$$
$$z_c \leq 0 \qquad \forall c \in C\colon y_c = w_c \tag{43c}$$
$$z_c = 0 \qquad \forall c \in C\colon y_c < w_c \tag{43d}$$

plus constraints (40d)-(40h). Consider the linear combination of (in)equalities (43a)-(43b) with the coefficients $\bar{y}$:

$$\sum_{c \in C} \bar{y}_c (1 + z_c - x^+(V_c^+) + x^-(V_c^-)) \leq 0. \tag{44}$$

It is easy to see that inequality (44) is infeasible on conditions (43c)-(43d) and (40d)-(40h) if and only if $\bar{y}$ is an improving direction for (39). So it remains to prove that (44) is feasible on conditions (43c)-(43d) and (40d)-(40h) if and only if (42) is feasible on conditions (40d)-(40h).

Suppose there is x satisfying (40d)-(40h) and (42). If we simply set $z = 0$, then (x, z) clearly satisfies (43c)-(43d) and (44).

Suppose there are (x, z) satisfying (43c)-(43d), (40d)-(40h) and (44). Then x satisfies (42), which can be shown by comparing corresponding terms in (42) and (44) as follows. For $c \in C$ with $y_c < w_c$, we have $z_c = 0$ by (43d), hence the c-th terms in (42) and (44) are the same. For $c \in C$ with $y_c = w_c$, we have $\bar{y}_c \leq 0$ by (40c) and $z_c \leq 0$, hence the c-th term in (42) is not greater than the c-th term in (44). □

6.1 Constraint Propagation

Given a current solution y feasible for problem (39), we want to apply constraint propagation to system (40). Similarly to Section 5.1, in every iteration we propagate zero values of the variables x^+ and x^- based on a single constraint (40a)-(40c), assuming (40d)-(40h). That is, we choose one of the constraints (40a)-(40c), infer from this constraint equalities $x_i^+ = 0$ for some $i \in I_+ \cap V_c^+$ and/or $x_i^- = 0$ for some $i \in I_- \cap V_c^-$, and remove these indices from the sets I_+ and/or I_-.[22] The propagation algorithm starts with $I_+ = I_+^*(y)$ and $I_- = I_-^*(y)$ and then iteratively shrinks the sets I_+ and I_-. If at some point any of the constraints (40a)-(40c) becomes infeasible on conditions (40d)-(40h), then we know that the initial (i.e., for $I_\pm = I_\pm^*(y)$) system (40) was infeasible, hence there exists an improving direction $\bar{y}$ for (39) from y, which can be obtained using Theorem 8.

To derive the cause vectors, we need to formulate each propagation rule as a linear combination of existing equalities and inequalities, according to Theorem 8. Similarly to Section 5.1, rather than directly inferring an equality $x_i^\pm = 0$ for some $i \in I_\pm^*(y)$, we instead infer an inequality

$$x_i^\pm + t_i^\pm \leq 0 \tag{45}$$

22. To satisfy constraint (40d), any index i can be removed from at most one of the sets I_+ and I_-, so that we always maintain $I_+ \cup I_- = V$.

where $t_i^\pm$ is a linear combination of the variables x^+ and x^- such that $t_i^\pm \geq 0$ on conditions (40d)-(40h) with $I_\pm = I_\pm^*(y)$. This implies $x_i^\pm = 0$ due to $x^\pm \geq 0$. Inequality (45) is a linear combination of constraints (40a)-(40c) with the cause vector (the coefficients of the linear combination) denoted by $y^i \in \mathbb{R}^C$. Note that (45) trivially (by (40e)-(40h)) implies $x_i^\pm = 0$ also for $i \notin I_\pm^*(y)$, in which case we set $t_i^\pm = -x_i^\pm$, so that (45) reads $0 \leq 0$ with the cause vector $y^i = 0$.

To simplify the formal descriptions of the rules, we will use the following abbreviations. For any $c \in C$, we denote $V_c = V_c^+ \cup V_c^-$. For any $c \in C$ and $i \in V_c$, we denote

$$x_i^c = \begin{cases} x_i^+ & \text{if } i \in V_c^+, \\ x_i^- & \text{if } i \in V_c^- \end{cases} \tag{46}$$

and define the following phrases:

- By saying that *variable x_i^c is fixed to 0*, we mean that $i \in (V_c^+ \setminus I_+) \cup (V_c^- \setminus I_-)$.
- By saying that *variable x_i^c is fixed to 1*, we mean that $i \in (V_c^+ \setminus I_-) \cup (V_c^- \setminus I_+)$.
- By saying that *variable x_i^c is not fixed*, we mean that $i \in I_+ \cap I_-$.
- By *fixing x_i^c to 0*, we mean removing i from I_+ or I_- if $i \in V_c^+$ or $i \in V_c^-$, respectively.
- By *fixing x_i^c to 1*, we mean removing i from I_- or I_+ if $i \in V_c^+$ or $i \in V_c^-$, respectively.

Similarly to previous sections, $e^c \in \mathbb{R}^C$ denotes the standard-basis vector of $\mathbb{R}^C$.

Here are the used propagation rules in detail and the derivations of the cause vectors:

1. In a constraint (40a) or (40b), if one variable is not fixed and all other variables are fixed to 0, then the non-fixed variable can be fixed to 1. Formally, if there is $i \in V_c$ such that x_i^c is not fixed and x_j^c is fixed to 0 for all $j \in V_c \setminus \{i\}$, then x_i^c is fixed to 1.

   We show how this rule is inferred for the case $i \in V_c^+$:

$$x_j^+ + t_j^+ \leq 0 \qquad y^j \qquad \forall j \in V_c^+ \setminus \{i\} \tag{47a}$$
$$x_j^- + t_j^- \leq 0 \qquad y^j \qquad \forall j \in V_c^- \tag{47b}$$
$$1 - x^+(V_c^+) - x^-(V_c^-) \leq 0 \qquad e^c \tag{47c}$$
$$\underbrace{1 - x_i^+}_{x_i^-} + \underbrace{\sum_{j \in V_c^+ \setminus \{i\}} t_j^+ + \sum_{j \in V_c^-} t_j^-}_{t_i^-} \leq 0 \qquad y^i = \sum_{j \in V_c \setminus \{i\}} y^j + e^c \tag{47d}$$

   Since all $t_j^\pm$ are non-negative, the inferred inequality implies $x_i^- = 0$. Note that, due to the implicitly assumed constraint (40d), we replaced $1 - x_i^+$ with x_i^- in the inferred inequality.

   The inference for the case $i \in V_c^-$ is similar, resulting in the same cause vector.

2. In a constraint (40a) or (40b), if all variables are fixed to 0 (formally, if x_i^c is fixed to 0 for all $i \in V_c$), then there is a contradiction. This is inferred as follows:

$$x_j^+ + t_j^+ \leq 0 \qquad y^j \qquad \forall j \in V_c^+ \tag{48a}$$
$$x_j^- + t_j^- \leq 0 \qquad y^j \qquad \forall j \in V_c^- \tag{48b}$$
$$1 - x^+(V_c^+) - x^-(V_c^-) \leq 0 \qquad e^c \tag{48c}$$

$$1 + \sum_{j \in V_c^+} t_j^+ + \sum_{j \in V_c^-} t_j^- \leq 0 \qquad \bar{y} = \sum_{j \in V_c} y^j + e^c \tag{48d}$$

Since all $t_j^\pm$ are non-negative, the inferred inequality is infeasible.

3. In a constraint (40b) or (40c), if exactly one variable is fixed to 1 (i.e., there is exactly one $i \in V_c$ such that x_i^c is fixed to 1), then all non-fixed variables have to be fixed to 0. For $i \in V_c^+$ (i.e., $1 - x_i^+ = x_i^- = 0$), this is inferred as follows:

$$\begin{aligned} x_i^- + t_i^- &\leq 0 && y^i && (49a)\\ x^+(V_c^+) + x^-(V_c^-) - 1 &\leq 0 && -e^c && (49b)\\ \hline x_i^- + t_i^- + x^+(V_c^+) + x^-(V_c^-) - 1 &\leq 0 && y^j = y^i - e^c && (49c) \end{aligned}$$

Using (40d) we have $x_i^- + x^+(V_c^+) - 1 = x_i^- + x^+(V_c^+ \setminus \{i\}) + x_i^+ - 1 = x^+(V_c^+ \setminus \{i\})$, therefore the inferred inequality implies $x_j^c = 0$ for all $j \in V_c \setminus \{i\}$. Note that the cause vectors for all these variables are identical, $y^j = y^i - e^c$.

The inference for the case $i \in V_c^-$ is similar, resulting in the same cause vector.

4. In a constraint (40b) or (40c), if two (or more) variables are fixed to 1 (i.e., there are distinct $i, j \in V_c$ such that x_i^c and x_j^c are fixed to 1), then there is a contradiction. For $i, j \in V_c^+$, this is inferred as follows:

$$\begin{aligned} x_i^- + t_i^- &\leq 0 && y^i && (50a)\\ x_j^- + t_j^- &\leq 0 && y^j && (50b)\\ x^+(V_c^+) + x^-(V_c^-) - 1 &\leq 0 && -e^c && (50c)\\ \hline x_i^- + t_i^- + x_j^- + t_j^- + x^+(V_c^+) + x^-(V_c^-) - 1 &\leq 0 && \bar{y} = y^i + y^j - e^c && (50d) \end{aligned}$$

Using (40d) we have $x_i^- + x_j^- + x^+(V_c^+) = x^+(V_c^+ \setminus \{i, j\}) + 2$, therefore the inferred inequality is infeasible.

For all other cases than $i, j \in V_c^+$, the inference is similar, resulting in the same infeasibility certificate $\bar{y}$.

Example 5. *Let $V = C = \{1, 2, 3\}$ and let the system* (40a)-(40c) *read*[23]

$$\begin{aligned} x_1^+ + x_2^- + x_3^- &= 1 && (51a)\\ x_1^+ + x_2^+ &\geq 1 && (51b)\\ x_2^+ + x_3^+ &= 1. && (51c) \end{aligned}$$

I.e., $V_1^+ = \{1\}$, $V_1^- = \{2, 3\}$, $V_2^+ = \{1, 2\}$, $V_2^- = \emptyset$, $V_3^+ = \{2, 3\}$, and $V_3^- = \emptyset$. Let us also initially have $I_+ = \{1, 2, 3\}$ and $I_- = \{2, 3\}$. Note that $x_1^- = 0$ due to $1 \notin I_-$, so we initialize $y^1 = (0, 0, 0)$.

The propagation proceeds as follows:

1. *Since $x_1^- = 1 - x_1^+ = 0$,* (51a) *implies $x_2^- = x_3^- = 0$, by propagation rule 3. Following* (49)*, $y^2 = y^3 = y^1 - e^1 = (-1, 0, 0)$.*

23. We write the constraints (40a)-(40c) multiplied by -1 here.

2. *All variables in* (51b) *are fixed and the constraint is satisfied, so no rule is applicable to this constraint.*
3. *Finally, since* $x_2^+ + x_3^+ = 1 - x_2^- + 1 - x_3^- = 2$, *constraint* (51c) *is infeasible by rule 4. The certificate of infeasibility is* $\bar{y} = y^2 + y^3 - e^3 = (-2, 0, -1)$ *by* (50).

Remark 13. *For general Weighted Max-SAT, the propagation rules need not always detect infeasibility of system* (40), *i.e., they are refutation incomplete. As an example, let* $V = I_+ = I_- = \{1, 2, 3\}$ *and* (40a)-(40c) *read*

$$x_1^+ + x_2^+ + x_3^+ = 1 \tag{52a}$$
$$x_1^+ + x_2^+ = 1 \tag{52b}$$
$$x_1^+ + x_3^+ = 1 \tag{52c}$$
$$x_2^+ + x_3^+ = 1 \tag{52d}$$

This system is infeasible but no propagation rule is applicable.

However, for Weighted Max-2SAT (i.e., $|V_c| \leq 2$ *for all* $c \in C$*), the rules are refutation complete. In detail, if no more propagation is possible and no contradiction was detected, setting all undecided variables* $x_i^\pm$ *to* $\frac{1}{2}$ *satisfies* (40). *This is easily verified by case analysis.*

Remark 14. *If a propagation rule 1 or 3 is applied to a single constraint of* (40a)-(40c), *then all variables in this constraint become fixed and the constraint becomes satisfied. Thus, none of the propagation rules is applicable to this constraint anymore. Consequently, propagation stops after at most* $|C|$ *inference rules are applied.*

Remark 15. *One can ask whether it is possible to infer other values of non-fixed variables than 0 or 1, such as* $\frac{1}{2}$. *If inference is done only from a single constraint of* (40a)-(40c), *this is impossible because the polyhedron defined by a single (in)equality from* (40a)-(40c) *subject to* (40d)-(40h) *has integral vertices (see Proposition 4 in Appendix A).*

Remark 16. *System* (40) *can be interpreted as an LP relaxation of a Boolean CSP with domains* $D = \{+, -\}$ *and constraints* (40a)-(40c). *It is easy to see that the propagation rules correspond to enforcing arc consistency in this CSP, i.e., reducing the initial domains to* $D_i = \{+\}$ *or* $D_i = \{-\}$ *which is represented by* $x_i^- = 0$ *or* $x_i^+ = 0$, *respectively.*

Seeing (40) *as a CSP with continuous variables, described propagation can be also interpreted as activity propagation, making some of the inequalities* (40e) *and* (40g) *active in every iteration. Note that we need not explicitly infer activity of constraints* (40a) *and* (40c) *because this does not make the propagation algorithm stronger. The propagation rules can be also equivalently seen as enforcing bounds consistency (or arc consistency) w.r.t. the constraints* (40a)-(40c) *while implicitly assuming the constraints* (40d).

6.2 Finding Step Size by Approximate Line Search

If a contradiction is detected in (40) and an improving direction $\bar{y}$ for problem (39) from the current point y is constructed, we need to find a step size $\alpha > 0$ to update $y := y + \alpha\bar{y}$ as in Section 4. The optimal way (exact line search) would be to minimize the univariate convex piecewise-affine function $g(\alpha) = f(y + \alpha\bar{y})$ over $\alpha > 0$ subject to $0 \leq y + \alpha\bar{y} \leq w$ where f is the objective function of (39). As this is too costly for large instances, we do

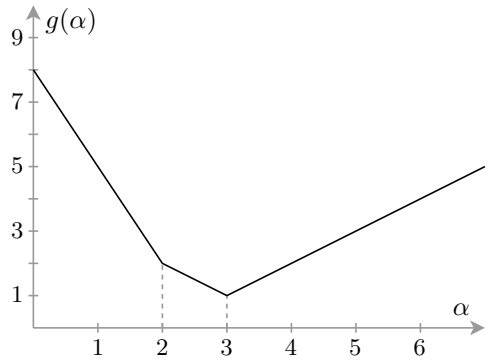


Figure 3: Graph of function g from Example 6.

only approximate line search: we find the first breakpoint (i.e., non-differentiable point) $\alpha > 0$ of the function g, i.e., the smallest $\alpha > 0$ at which at least one of the affine functions that is inactive at $\alpha = 0$ becomes active.[24] This value of α may be further decreased to ensure feasibility. Such α is the greatest number satisfying the following conditions:

1. To maintain feasibility, for all $c \in C$ we need $0 \leq y_c + \alpha \bar{y}_c \leq w_c$. Since $0 \leq y_c \leq w_c$, this means $\alpha \leq -y_c/\bar{y}_c$ for all c with $\bar{y}_c < 0$ and $\alpha \leq (w_c - y_c)/\bar{y}_c$ for all c with $\bar{y}_c > 0$.
2. For the terms $\max\{y(C_i^+), y(C_i^-)\}$, if $y(C_i^+) > y(C_i^-)$ and $\bar{y}(C_i^+) < \bar{y}(C_i^-)$ (or with both inequalities inverted), we need $\alpha \leq (y(C_i^+) - y(C_i^-))/(\bar{y}(C_i^-) - \bar{y}(C_i^+))$ where the bound is the point where the terms equal, i.e., $y(C_i^+) + \alpha\bar{y}(C_i^+) = y(C_i^-) + \alpha\bar{y}(C_i^-)$. So, this is for all $i \in V$ with $(y(C_i^+) - y(C_i^-))(\bar{y}(C_i^-) - \bar{y}(C_i^+)) > 0$.

By writing Farkas' alternative system to the complementary slackness conditions (40), it can be shown that there always exists $\alpha > 0$ satisfying these bounds.

Example 6. *Let the sets V, C, $V_1^{\pm}$, $V_2^{\pm}$, and $V_3^{\pm}$ be as in Example 5. In that case, the objective of* (39) *reads*

$$f(y) = \max\{y_1 + y_2,\, 0\} + \max\{y_2 + y_3,\, y_1\} + \max\{y_3,\, y_1\} - y_1 - y_2 - y_3. \tag{53}$$

Furthermore, let the current point be $y = (6, 0, 4)$, the improving direction be $\bar{y} = (-2, 0, -1)$, and $w = (7, 8, 9)$. Thus, we have (see Figure 3)

$$g(\alpha) = f(y+\alpha\bar{y}) = \max\{6-2\alpha,\, 0\} + \max\{4-\alpha,\, 6-2\alpha\} + \max\{4-\alpha,\, 6-2\alpha\} - 10 + 3\alpha. \tag{54}$$

By condition 1, we need $y_c + \alpha\bar{y}_c \geq 0$ for each c with $\bar{y}_c < 0$, which yields $\alpha \leq -y_1/\bar{y}_1 = 3$ and $\alpha \leq -y_3/\bar{y}_3 = 4$. By condition 2, we need $\alpha \leq (y(C_i^+) - y(C_i^-))/(\bar{y}(C_i^-) - \bar{y}(C_i^+))$ for all $i \in V$, which yields $\alpha \leq 3$, $\alpha \leq 2$, and $\alpha \leq 2$. The maximum number α satisfying these bounds is $\alpha = 2$ and y is therefore updated to $y + 2\bar{y} = (2, 0, 2)$.

Notice that the upper-bounds obtained from condition 2, i.e., $\{2, 3\}$, are precisely the breakpoints α of function g with $\alpha > 0$. The chosen step size $\alpha = 2$ is then the smallest breakpoint of g with $\alpha > 0$ and also the point where the affine functions $4 - \alpha$ from (54) *become active (while they are inactive for $\alpha < 2$).*

24. For a convex piecewise-affine function $\sum_{k \in K} \max_{l \in L_k} \left(c_{kl}^T y + d_{kl}\right)$, an affine function $c_{kl^*}^T y + d_{kl^*}$ is *active* (for some y) if $c_{kl^*}^T y + d_{kl^*} = \max_{l \in L_k} \left(c_{kl}^T y + d_{kl}\right)$.

It is clear from Figure 3 that the unique optimal step size is $\operatorname{argmin}_{0<\alpha\leq 3} g(\alpha) = \{3\}$*, so the computed step size* $\alpha = 2$ *is not optimal in this case.*

6.3 Algorithm Overview and Implementation Details

Let us summarize the algorithm that follows the general iterative scheme previously shown in Algorithm 2. We start with $y = 0$ (which is feasible for (39)) and repeat the following iteration. For the current y, apply appropriate propagation rules. During that, construct the DAG defining each y^i (recall Section 2.1) 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 as in Section 2.1. Calculate step size α as in Section 6.2 and update $y := y + \alpha\bar{y}$.

Remark 17. *Following on Remark 16, this algorithm seeks to find a feasible dual solution to the LP relaxation of Weighted Max-SAT that enforces the CSP defined by* (40) *to have a non-empty AC closure. Compare this with the WCSP case, where* (29) *is an LP relaxation of a CSP and the VAC algorithm seeks to find a reparametrization that makes this CSP have a non-empty AC closure. In contrast to WCSP, there is no obvious analogy of reparametrizations (or equivalent transformations) for Weighted Max-SAT.*

To speed up the algorithm and ensure finiteness, we use the trick similar to capacity scaling as described in Section 4.2. Namely, we redefine the conditions in (40a)-(40c) and (41) up to a tolerance $\theta > 0$: we replace $y_c > 0$ with $y_c > \theta$, $y(C_i^+) \leq y(C_i^-)$ with $y(C_i^+) \leq y(C_i^-) + \theta$, $y_c = w_c$ with $w_c - \theta \leq y_c \leq w_c + \theta$, etc. We follow the general scheme outlined in Algorithm 3 where we initialize $\theta = w(C) = \sum_{c\in C} w_c$ and whenever the algorithm cannot detect infeasibility with the current θ, we keep the current y and update $\theta := \theta/10$. We continue until θ is not small enough (10^{-6}). To improve performance, we also decrease θ whenever $(D - D')/(w(C) + D') < 10^{-12}$ where D and D' is the dual objective before and after the iteration, respectively.

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 (37). In particular, it can be shown that the DAG (used to calculate $\bar{y}$) can be conveniently stored as a directed subgraph of the clause-variable incidence graph.[25] By Remark 14, we only need to store for each clause $c \in C$ which rule was applied to this clause and which variables were fixed by the rule.

Remark 18. *We argue that this algorithm terminates after a finite number of iterations (recall Theorem 6). First, the primal* (37) *is always feasible and bounded by* $-w(C)$*, so the dual* (37) *is also feasible and bounded. Second, by Remark 14 there are only finitely many options in which the propagation rules can be applied to system* (40) *and, for each order in which the rules were applied, the improving direction* $\bar{y}$ *is defined deterministically. Consequently, there exists a finite set* $\bar{Y}$ *of improving directions used by the algorithm (for each instance). Finally, although the computed step size need not be optimal (for the formulation in terms of the convex piecewise-affine function* (39)*), it can be shown that there exists a positive lower bound on the step size by analyzing the bounds listed in Section 6.2 and noting that* $\theta > 0$ *(this is similar to the proof of Theorem 7).*

25. The clause-variable incidence graph is the bipartite graph whose nodes correspond to variables V and clauses C. The graph contains an edge between nodes $i \in V$ and $c \in C$ if $i \in V_c$.

	Solved by LP solver? Yes	No	Total
Instances with no unit clauses	91	11	102
Max-2SAT instances with at least one unit clause	154	0	154
Other instances	1855	480	2335
Total	2100	491	2591

Table 1: Survey of Max-SAT instances in experiments.

Alternatively, one can apply Theorem 7 directly to the dual linear program in (37)*. It is possible to show that the step size* α *is in fact optimal for the dual* (37) *via the update* $(y, p) := (y, p) + \alpha(\bar{y}, \bar{p})$ *if the improving direction* $(\bar{y}, \bar{p})$ *for the dual* (37) *is obtained from* $\bar{y}$ *as*

$$\bar{p}_i = \begin{cases} \max\{\bar{y}(C_i^+), \bar{y}(C_i^-)\} & \text{if } y(C_i^+) = y(C_i^-) \\ \bar{y}(C_i^+) & \text{if } y(C_i^+) > y(C_i^-) \\ \bar{y}(C_i^-) & \text{if } y(C_i^+) < y(C_i^-) \end{cases} \qquad \forall i \in V. \tag{55}$$

Remark 19. *Li, Xu, Coll, Manyà, Habet, & He, 2021 recently proposed another method for computing bounds on the optimal value of Max-SAT. It is combinatorial in nature: based on the current assignment, it finds conflicts (cores) among the clauses, which provides a bound on the number of falsified soft clauses. The method is applicable only to Unweighted Partial Max-SAT. As our interest is only in the bounds obtained from the LP relaxation, we do not compare our results with this method.*

6.4 Experimental Results

We compared the upper bound on the optimal value of (37) obtained by our algorithm with the exact optimal value of (37) computed by an off-the-shelf LP solver. We used Gurobi (Gurobi Optimization, LLC, 2020) version 7.5.2 with default parameters, which uses a concurrent solver. Note that the concurrent solver runs several algorithms in parallel, stopping as soon as any of them signals optimality. This option can thus use more memory than any of the algorithms individually. The experiments were performed on a laptop with i7-4710MQ CPU at 2.5 GHz, providing 8 threads to the concurrent solver, and 16GB RAM without any time limit. We implemented our algorithm to use only a single thread.

We used the Max-SAT Evaluations 2018 benchmark (Bacchus, Järvisalo, & Martins, 2019), which contains 2591 instances of Weighted Max-SAT. Gurobi was able to solve (without memory overflow) the LP relaxation for 2100 instances, the largest of which had up to 640 thousand clauses, 400 thousand variables, and 3.5 million non-zeros (i.e., the number $\sum_{c \in C} |V_c|$). The largest instances in the benchmark have up to 27 million clauses, 19 million variables, and 95 million non-zeros and were still managed by our algorithm.[26]

Note that if an instance does not contain any unit clause, then setting $x_i^+ = x_i^- = \frac{1}{2}$, $i \in V$ and $z_c = 0$, $c \in C$ yields an optimal solution to the primal (37) with objective value 0

26. The instance sizes reported here correspond to maximal values of these properties and, e.g., the instance with the maximal number of clauses need not have the maximal number of variables. In contrast, Dlask and Werner (2020) reported these values for the largest instances in terms of file size. However, in both publications, the dataset and results are the same.

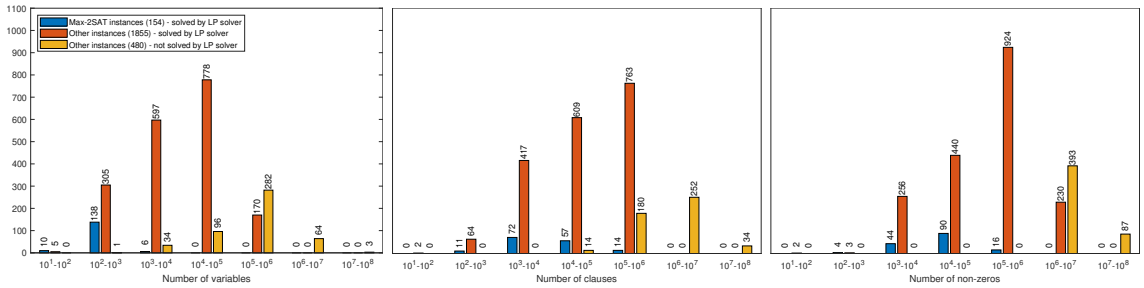


Figure 4: Histograms of the number of clauses, number of variables, and number of non-zeros of the instances, grouped as in Table 1. Instances with no unit clauses are omitted. Lower bounds of the intervals are inclusive and upper bounds are exclusive.

(cf. Hooker, 2000, Section 13.1.1). Our algorithm attains optimality on these instances because $y = 0$ is already optimal for the dual. Therefore, we exclude these instances from further evaluation.

In Table 1, we divide instances according to lengths of clauses and whether the Gurobi LP solver was able to solve them. Further on, we compare our method with the LP solver on the 1855 instances with at least one clause of length at least 3 and on the 154 Max-2SAT instances, respectively, and discuss the 480 instances that are too large for the LP solver in Section 6.4.3. For these three groups of instances, we show the histograms of the instance sizes in Figure 4.

6.4.1 Comparison on Instances with a Clause of Length at Least 3

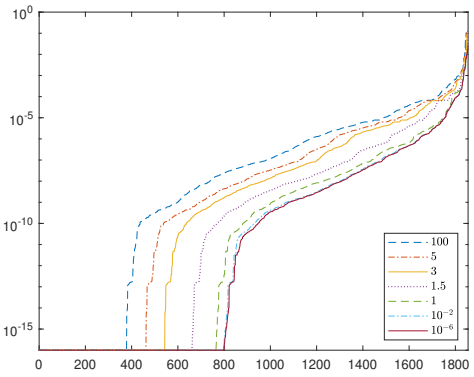
If an instance has at least one clause of length at least 3 and at least one unit clause, the bound provided by our algorithm is not guaranteed to coincide with the optimum of the LP relaxation. We measure the quality of the bound on each such instance i by two criteria

$$Q_i = \frac{U_i^{\mathrm{CP}} - U_i^{\mathrm{LP}}}{U_i^{\mathrm{LP}}} \qquad \text{and} \qquad R_i = \frac{U_i^{\mathrm{CP}} - U_i^{\mathrm{LP}}}{w(C) - U_i^{\mathrm{LP}}} \tag{56}$$

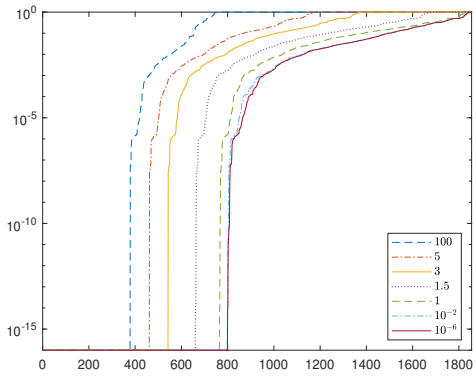
where U_i^{LP} is the upper bound computed using an LP solver and U_i^{CP} is the upper bound computed by our algorithm based on constraint propagation on the instance. In detail, U_i^{LP} equals $w(C)$ plus the optimal value of (37) and U_i^{CP} equals $w(C)$ plus the bound on the optimal value of (37) computed by our algorithm. Note that U_i^{LP} is equal to the optimal value of the classical LP relaxation (Vazirani, 2001, Section 16). Both criteria (56) are invariant to scaling the weights. Criterion Q_i is the relative difference between the optimal value of the classical LP relaxation and the upper bound computed by our algorithm whereas criterion R_i shows how tight the bound is relative to the trivial bound $w(C)$.

To analyze the achievable trade-offs between runtime and quality of the obtained bound, we also report the results obtained for the setting where the algorithm is stopped already when $\theta \leq \theta^*$ for different (larger) values of θ^*, namely 100, 5, 3, 1.5, 1, 10^{-2}, and 10^{-6}. Clearly, a higher value of θ^* results in stopping earlier and a worse bound. For each value of θ^*, the schedule of decrease of θ is the same, as described in Section 6.3.

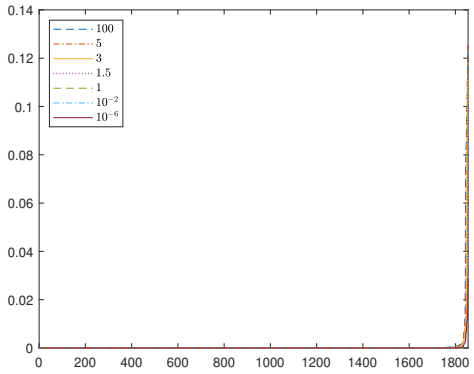
The sorted numbers Q_i and R_i for the 1855 instances are plotted in Figure 5 for each considered value of θ^*. The values θ^* are chosen so that the curves in the plots are evenly



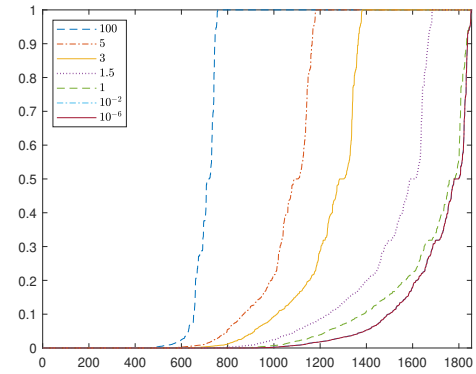
(a) Sorted values Q_i (log scale).



(b) Sorted values R_i (log scale).



(c) Sorted values Q_i (linear scale).



(d) Sorted values R_i (linear scale).

Figure 5: Sorted values of Q_i (left) and R_i (right) with linear (bottom) and logarithmic (top) scale for different values of θ^*.

distributed (except that there is not much difference between the results obtained with $\theta^* = 10^{-2}$ and $\theta^* = 10^{-6}$). Since our algorithm sometimes attains the optimal value, the vertical logarithmic axes are trimmed, starting from 10^{-16}. Note, the least attained positive value of the values (56) is $1.2 \cdot 10^{-16}$. Based on the linear plot in Figure 5d, the obtained upper bound U_i^{CP} is close to U_i^{LP} in around 1100 cases (if θ^* is sufficiently small). To be precise, criterion R_i was lower than 10^{-2} in 1036, 1120, and 1127 cases for $\theta^* = 1$, 10^{-2}, and 10^{-6}, respectively.

To support the claim that our algorithm is useful even for larger instances, it is important to analyze whether the bounds do not get worse with increasing instance size. To confirm this, we group instances into quartiles based on their properties and plot the overlaid sorted values of Q_i and R_i for each quartile in Figure 6. Regarding the properties determining the size of the instance, we consider number of clauses, number of variables, and number of non-zeros, i.e., the sum of lengths of all clauses. For all the plots, it seems that the numbers Q_i and R_i are either getting lower or staying the same with increasing the instance size. For the criterion Q_i, this can be explained by the fact that, with increasing size of the instance,

θ^*	100	5	3	1.5	1	10^{-2}	10^{-6}
Instances with at least one clause of length at least 3 and at least one unit clause (Sec. 6.4.1)							
$\sum_i T_i^{\mathrm{LP}} / \sum_i T_i^{\mathrm{CP}}$	15.26	7.38	6.45	3.91	3.48	3.29	3.23
arithmetic mean of $T_i^{\mathrm{LP}}/T_i^{\mathrm{CP}}$	200.40	159.84	85.77	53.25	34.32	26.50	20.37
geometric mean of $T_i^{\mathrm{LP}}/T_i^{\mathrm{CP}}$	42.16	22.04	16.86	10.39	7.48	5.43	4.52
instances with $R_i \leq 10^{-16}$	378	461	542	660	764	798	799
instances with $U_i^{\mathrm{CP}} \leq U_i^{\mathrm{LP}} + 10^{-6}$	378	462	543	665	769	807	814
Max-2SAT instances with at least one unit clause (Section 6.4.2)							
$\sum_i T_i^{\mathrm{LP}} / \sum_i T_i^{\mathrm{CP}}$	16.59	2.01	0.86	0.65	0.60	0.59	0.59
arithmetic mean of $T_i^{\mathrm{LP}}/T_i^{\mathrm{CP}}$	36.57	23.71	11.40	7.08	3.74	3.44	3.03
geometric mean of $T_i^{\mathrm{LP}}/T_i^{\mathrm{CP}}$	15.62	7.26	3.29	2.16	1.56	1.53	1.48
instances with $R_i \leq 10^{-16}$	0	53	99	134	153	154	154
instances with $U_i^{\mathrm{CP}} \leq U_i^{\mathrm{LP}} + 10^{-6}$	0	53	99	134	153	154	154

Table 2: Comparison of runtimes of the LP solver (T_i^{LP}) and our method (T_i^{CP}) and the number of instances solved exactly.

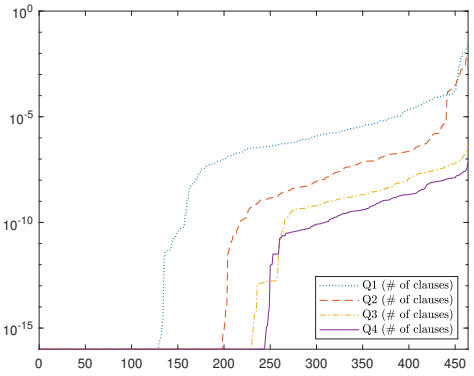
U_i^{LP} is increasing too. However, the number of instances solved to optimality is increasing with the size of the instance. Overall, we conclude that the bounds do not get worse with increasing instance size.

Next, we report the number of instances solved exactly and the speed-up obtained for different values of θ^* in Table 2 (upper part), denoting by T_i^{LP} and T_i^{CP} the time required by the LP solver and by our method on instance i, respectively. For each value of θ^*, the table shows the overall runtime of the LP solver divided by the overall runtime of our algorithm and the arithmetic and geometric mean of speed-ups for the individual instances. Figure 7a shows the cactus plot, which indicates that there is indeed a systematic speed-up with our method when compared to the LP solver. The longest runtime of our algorithm with $\theta^* = 10^{-6}$ and the LP solver is 111 seconds and 679 seconds, respectively.

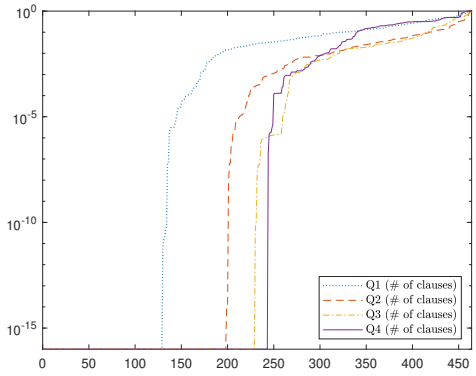
We believe that an additional speed-up could be achieved by warm-starting. The part of the DAG needed to explain the found contradiction (see Section 2.1) is usually very small. If the DAG is built in every iteration from scratch, most of it is therefore thrown away. Since the system (40) changes only slightly between consecutive updates, it makes sense to reuse a part of the DAG in the next iteration and thus avoid repeatedly applying many rules in the same way. Such a warm-starting was presented for the VAC algorithm by Nguyen, Schiex, and Bessiere (2013) and for the Augmenting DAG algorithm by Werner (2005) with significant speed-ups (also see Komodakis & Paragios, 2008).

6.4.2 Comparison on Max-2SAT Instances

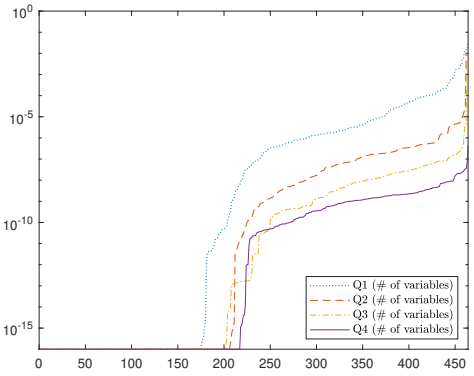
Regarding the 154 Max-2SAT instances, Table 2 (lower part) shows the achievable trade-offs in terms of runtime and number of instances solved to optimality. In relation to Remark 13, with low-enough θ^*, all Max-2SAT instances are solved to optimality. Concerning runtime, some speed-up is often achieved, but the overall time needed by our algorithm for the 154 instances is higher than the overall runtime of the LP solver, as seen in Table 2. This is explained by the cactus plot in Figure 7b which shows that our approach is able to solve many instances quickly, but then spends a lot of time converging on a few ones. In detail, more than 80% and 90% of time is spent on 12% and 23% of instances by our algorithm



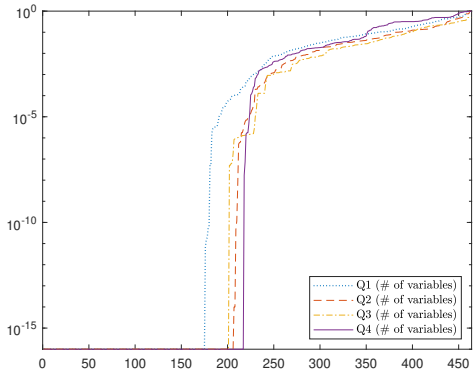
(a) Division based on number of clauses in the instance, values of Q_i.



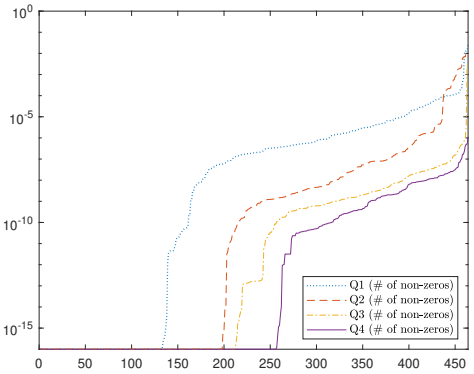
(b) Division based on number of clauses in the instance, values of R_i.



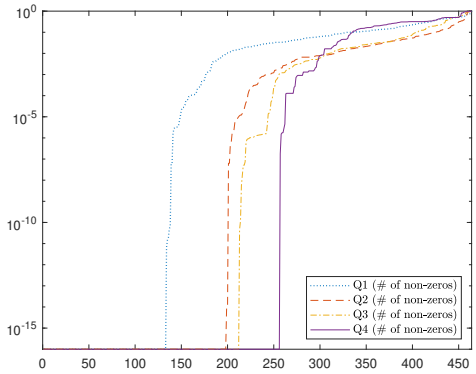
(c) Division based on number of variables in the instance, values of Q_i.



(d) Division based on number of variables in the instance, values of R_i.

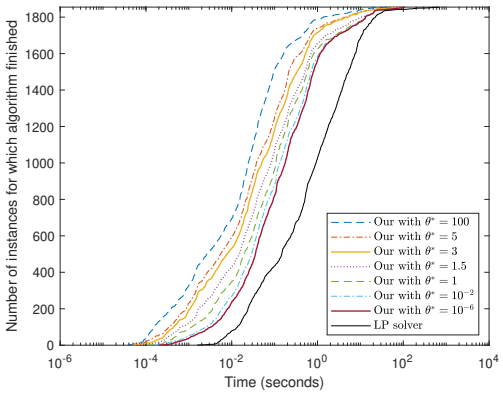


(e) Division based on number of non-zeros, values of Q_i.

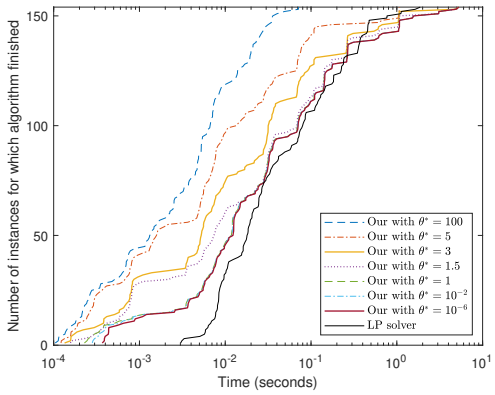


(f) Division based on number of non-zeros, values of R_i.

Figure 6: Dependence of bound quality on the instance size (specified by quartiles Q1–Q4). For each quartile, sorted values of Q_i or R_i are shown with logarithmic scale of the vertical axis. The division into quartiles is done according to the number of clauses, number of variables, and number of non-zeros.



(a) Cactus plot for the 1855 instances with at least one clause of length at least 3 and at least one unit clause.



(b) Cactus plot for the 154 Max-2SAT instances with at least one unit clause.

Figure 7: Cactus plots comparing the speed of the LP solver and our method for different values of θ^*. The plots show for how many instances the algorithm finishes (vertical axis) within the specified time (horizontal axis).

with $\theta^* = 10^{-6}$, respectively, and the longest runtime is 5.3 seconds. On the other hand, the LP solver solves each Max-2SAT instance in less than 1.9 seconds.

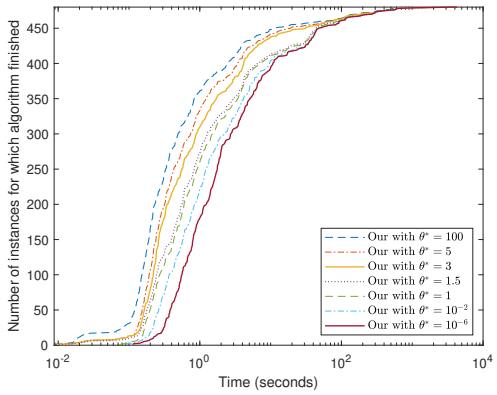
Importantly, the LP relaxation of Weighted Max-2SAT can be solved by a reduction to the maximum-flow problem (Rother, Kolmogorov, Lempitsky, & Szummer, 2007; Boros & Hammer, 2002), which makes it easier than the general case (Průša & Werner, 2019). It is possible that Gurobi takes advantage of this.

6.4.3 Instances Too Large for the LP Solver

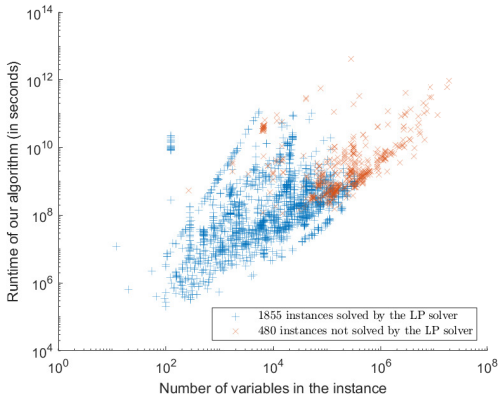
Let us now analyze the runtime of our algorithm on the 480 instances that are too large for the LP solver. Figure 8a shows the cactus plot which is, in comparison to those in Figure 7, more inclined towards longer runtimes. In particular, 90% and 99% of these instances are solved (with $\theta^* = 10^{-6}$) within 36 seconds and 556 seconds, respectively. The instance with second-longest and longest runtime required 16 and 72 minutes for our algorithm (with $\theta^* = 10^{-6}$) to converge, respectively.

To demonstrate that there is no abrupt change of behavior of our algorithm on these instances, we show the dependence of runtime on the number of variables, clauses, and non-zeros in Figures 8b, 8c, and 8d, respectively. All of these figures show that there is a clear trend between the runtime and the size of the instance and that the 480 instances adhere to it.

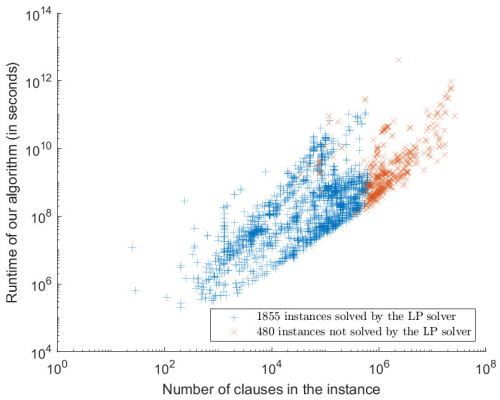
Putting this together with the results from Section 6.4.1 (recall Figure 6), we believe that our algorithm scales well even to large instances and that neither the runtime (relative to the instance size) nor the bound quality will systematically deteriorate.



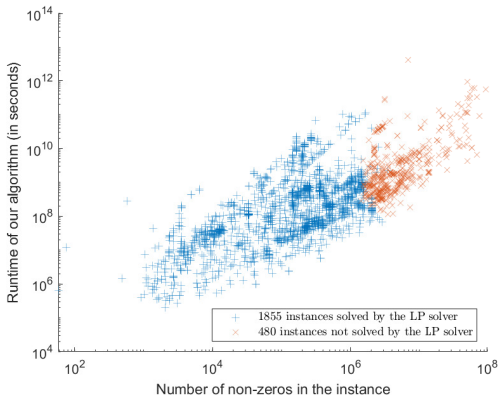
(a) Cactus plot for the 480 instances with at least one clause of length at least 3 and at least one unit clause that could not be solved by the LP solver.



(b) Dependence of runtime of our algorithm (with $\theta^* = 10^{-6}$) on the number of variables in the instance.



(c) Dependence of runtime of our algorithm (with $\theta^* = 10^{-6}$) on the number of clauses in the instance.



(d) Dependence of runtime of our algorithm (with $\theta^* = 10^{-6}$) on the number of non-zeros in the instance.

Figure 8: Information on the runtime of our algorithm and its dependence on instance size. Note the logarithmic scales of some of the axes.

6.5 Tightness of the Bound on Tractable Max-SAT Classes

We show that our constraint propagation rules for system (40) are refutation complete for tractable subclasses of Max-SAT that either use tractable clause types (language) or have acyclic structure (clause-variable incidence graph). For these instances, the LP relaxation is tight and any point returned by our algorithm is an optimum of the dual LP relaxation.

It was proved by Khanna and Sudan (1996) and Creignou (1995) 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 or all are 1-valid. In this case, the optimal value is $w(C)$, which coincides with the bound provided by the linear program (optimal value shifted by $w(C)$, as explained previously) and our algorithm attains this optimum already at $y = 0$.

- All constraints are 2-monotone. Restricting these constraints to clauses results in clauses with at most two literals where at most one is positive and at most one is negative. In this case, Max-SAT can be reduced to minimum *st*-cut problem (Khanna & Sudan, 1996, Lemma 3), (Creignou, 1995) and the optimal value of its LP formulation equals (up to a trivial recalculation) the optimal value of the LP relaxation of Max-SAT which is thus tight. Since this is an instance of Max-2SAT, any point returned by our algorithm is optimal by Remark 13.

If we view (40) as the LP relaxation of a CSP with Boolean variables, then the propagation rules enforce AC of this CSP (Remark 16). If the factor graph of this CSP is acyclic, AC solves this CSP exactly (Freuder, 1982, 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 (40) can be constructed, so the bound computed by the LP relaxation is tight.

7. Conclusion

We have proposed and analyzed a technique to compute, with small space complexity, bounds on linear programs. Given a dual-feasible solution, we apply constraint propagation to the complementary slackness conditions, which is a system of linear inequalities in the primal variables. If propagation succeeds to detect infeasibility of this system, we reconstruct a certificate of infeasibility (a solution of Farkas' alternative system) from the propagation history. This certificate is at the same time a dual-improving direction, which is used, after line search, to improve the current dual solution and thus the bound on the optimal value of the linear program.

While constraint propagation for systems of constraints (here, linear inequalities) with continuous variables has been studied before (Benhamou & Granvilliers, 2006), the main novelty of our approach is in reconstructing infeasibility certificates from constraint propagation and using them to iteratively improve the dual solution.

The proposed method can be seen as a generalization of the VAC / Augmenting DAG algorithm (Cooper et al., 2010; Koval & Schlesinger, 1976; Werner, 2007) for WCSP. Recall that the main purpose of (soft) local consistencies in WCSP, such as VAC, EDAC (de Givry, Heras, Zytnicki, & Larrosa, 2005), FDAC, DAC (Cooper, 2003; Larrosa & Schiex, 2003) or OSAC (Cooper, de Givry, & Schiex, 2007), is to bound the optimal value of WCSP during search. Each (soft) local consistency has a different trade-off point between bound quality and computational complexity (those that provide tighter bounds take longer to compute and vice versa). In this view, our approach can be seen as a (soft) local consistency technique for other problems than WCSP.

In particular, we applied it to the LP relaxation of Weighted Max-SAT where it provided different trade-offs between quality of the bound and runtime. Since the original announcement of our method in (Dlask & Werner, 2020), we applied it to approximately optimize an LP formulation of WCSP (Dlask et al., 2023), which is an exponentially large linear program. There, a necessary condition for feasibility of the complementary slackness is satisfiability of an underlying CSP (a generalization of the CSP $(V, D, E, J^*(y))$ from Section 5), so one can use any classical CSP local consistency to improve the bound.

In principle, our approach can be applied to any linear program (assuming an initial feasible dual solution is available), with any propagation method from Section 3. Alternatively, new propagation methods can be designed for the problem at hand. However, the quality of the obtained bounds will heavily depend on the linear program and the used constraint propagation method: for some classes of linear programs, the complementary slackness constraints will 'propagate well', for some others not. Like in the (W)CSP, no general enough theoretical results currently exist here, so this question is largely empirical.

Acknowledgments

This work has been supported by the Grant Agency of the Czech Technical University in Prague (grants SGS19/170/OHK3/3T/13 and SGS22/061/OHK3/1T/13), the OP VVV project CZ.02.1.01/0.0/0.0/16_019/0000765, the CTU institutional support (future fund), and the Czech Science Foundation (grant 19-09967S).

Appendix A. Relation Between Bounds Propagation and Activity Propagation

Here, we state specific conditions under which $\mathcal{B}$-consistency is related to bounds consistency (and thus also to arc consistency, by Proposition 1). As before, we assume that the constraint system is $Ax \leq b$ where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, but we additionally require the variables to be bounded to the interval $[0, 1]$, i.e., $Ax \leq b$ reads

$$A'x \leq b' \tag{57a}$$
$$-x \leq 0 \tag{57b}$$
$$x \leq 1 \tag{57c}$$

where $A' \in \mathbb{R}^{m' \times n}$ consists of the first $m' = m - 2n$ rows of A and $b' \in \mathbb{R}^{m'}$ contains the first m' components of b. For clarity, we assume that the inequalities in (57) are numbered from top to bottom so that $-x_j \leq 0$ is the $(m' + j)$-th inequality and $x_j \leq 1$ is the $(m' + n + j)$-th inequality.

Next, we assume that the set of blocks $\mathcal{B}$ is defined by

$$\mathcal{B} = \{\{i\} \cup \{m' + 1, \ldots, m\} \mid i \in [m']\}, \tag{58}$$

so that each subset of inequalities given by $B \in \mathcal{B}$ contains a single inequality from (57a) and all inequalities (57b)-(57c).

Following on this setting, we state our result that connects bounds consistency and $\mathcal{B}$-consistency in Proposition 3.

Proposition 3. *Let $\{X_1, X_0, X_U\}$ be a partition of $[n]$. Let us define*

$$D_j = \begin{cases} \{1\} & \text{if } j \in X_1 \\ \{0\} & \text{if } j \in X_0 \\ [0,1] & \text{if } j \in X_U \end{cases} \quad \forall j \in [n] \tag{59a}$$

$$I = [m'] \cup \{m' + j \mid j \in X_1\} \cup \{m' + n + j \mid j \in X_0\} \tag{59b}$$

$$I' = I \setminus \Big\{ i \in [m'] \;\Big|\; \sum_{j \in X_1} a_{ij} + \sum_{j \in X_U} \min\{0, a_{ij}\} = b_i \Big\}. \tag{59c}$$

Let $i \in [m']$ and $B = \{i\} \cup \{m'+1, \ldots, m\}$. If the polyhedron

$$\{x \in \mathbb{R}^n \mid a_i^T x \le b_i,\ 0 \le x \le 1\} \tag{60}$$

is integral, then the domains (59a) *are bounds consistent w.r.t. the constraint $a_i^T x \le b_i$ if and only if I' is B-consistent w.r.t.* (57)*.*

Consequently, if the polyhedron (60) *is integral for each $i \in [m']$, then the domains* (59a) *are bounds consistent w.r.t. the system* (57) *if and only if I' is $\mathcal{B}$-consistent w.r.t.* (57) *(where $\mathcal{B}$ is* (58)*). Lastly, if these statements hold, then there exists a $\mathcal{B}$-consistent subset of I, namely I'.*

Proof. Let $i \in [m']$ and B be as in the statement of the theorem. We prove both implications by contradiction. The system (10) for such B reads

$$\begin{cases} a_i^T x \le b_i & \text{if } i \in I' \\ a_i^T x = b_i & \text{if } i \notin I' \end{cases} \tag{61a}$$

$$x_j = 1 \qquad \forall j \in X_1 \tag{61b}$$

$$x_j = 0 \qquad \forall j \in X_0 \tag{61c}$$

$$0 \le x_j \le 1 \qquad \forall j \in X_U. \tag{61d}$$

Clearly, integrality of (60) implies integrality of $\{x \in \mathbb{R}^n \mid a_i^T x = b_i,\ 0 \le x \le 1\}$ and consequently also integrality of the polyhedron defined by (61).

Let I' be B-consistent, i.e., (61) is feasible and implies neither $x_j = 0$ nor $x_j = 1$ for any $j \in X_U$. Also, if $i \in I'$, it does not imply $a_i^T x = b_i$. Let (59a) not be bounds consistent, i.e., for some $j \in [n]$, there is no x feasible for (61) with $x_j = d$ where $d \in \{\min D_j, \max D_j\}$. Necessarily, $j \in X_U$ because otherwise D_j is a singleton set and this in turn implies infeasibility of (61), which would be contradictory. Assuming feasibility of (61) and $j \in X_U$, (61) implies $x_j = 1 - d$ because the polyhedron defined by (61) is integral. This is contradictory with B-consistency of I'.

On the other hand, let (59a) be bounds consistent w.r.t. $a_i^T x \le b_i$. The system

$$a_i^T x \le b_i \tag{62a}$$

$$x_j = 1 \qquad \forall j \in X_1 \tag{62b}$$

$$x_j = 0 \qquad \forall j \in X_0 \tag{62c}$$

$$0 \le x_j \le 1 \qquad \forall j \in X_U. \tag{62d}$$

is feasible because none of the domains is empty. Moreover, for each $j \in X_U$, (62) implies neither $x_j = 0$ nor $x_j = 1$ (otherwise, the domains could be reduced). Next, we argue that $i \in I'$ if and only if (62) implies $a_i^T x = b_i$: indeed, the minimal value of $a_i^T x$ among x satisfying (62b)-(62d) is

$$\sum_{j \in X_1} a_{ij} + \sum_{j \in X_U} \min\{0, a_{ij}\}.$$

By feasibility, this value is at most b_i and it equals b_i if and only if (62) implies $a_i^T x = b_i$, i.e., if and only if $i \notin I'$ by definition (59c). Consequently, I' is B-consistent, i.e., (61) does not contain any always-active inequality.

The remaining statements follow directly from the definition of $\mathcal{B}$-consistency and bounds consistency. □

Even though the setting outlined above might seem very restrictive, LP relaxations of combinatorial problems often satisfy these assumptions. In such cases, the bounded variables $x_j \in [0, 1]$, $j \in [n]$ can originate as the continuous relaxation of binary variables and the requirement on integrality of the polyhedra (60) can be assured, e.g., by the following proposition.

Proposition 4 (Dlask, 2022, Lemma 5.5, cf. Hooker, 2000, Theorem 45)**.** *For each $a \in \{-1, 0, 1\}^n$ and $b \in \mathbb{Z}$, the polyhedra $\{x \in [0, 1]^n \mid a^T x \leq b\}$ and $\{x \in [0, 1]^n \mid a^T x = b\}$ are integral.*

It is straightforward to extend the previous result to the more general case where the system (57a) contains also equalities, but we chose not to formalize this since it would complicate the explanation to some extent.[27]

In practice, it is indeed typically the case that the relaxed variables are in the interval $[0, 1]$ and the coefficients in the constraints satisfy $a_{ij} \in \{-1, 0, 1\}$ and $b_i \in \mathbb{Z}$, which turns out sufficient for integrality by Proposition 4. In particular, all of the aforementioned assumptions are satisfied by the constraints in the LP relaxations of the following combinatorial problems: set multicover (Vazirani, 2001) (including its special cases, set cover and vertex cover), quadratic assignment problem (Du & Pardalos, 1999) (also for the form in Zhang, Shi, McAuley, Wei, Zhang, & Van Den Hengel, 2016), max-cut (de la Vega & Kenyon-Mathieu, 2007), correlation clustering (Demaine & Immorlica, 2003) (a.k.a. multiway cut), Steiner forest (Vazirani, 2001), maximum independent set (Matoušek & Gärtner, 2006), facility location (Vazirani, 2001), travelling salesperson problem (Jünger, Reinelt, & Rinaldi, 1995; Dantzig, Fulkerson, & Johnson, 1954), linear ordering problem (Pardalos, Du, & Graham, 2013), planar and axial three-dimensional problem (Burkard, 2002; Průša & Werner, 2019). To be precise, some of these linear programs do not contain the upper bounds $x \leq 1$ (at least explicitly), but their optimal value is not changed by including them.

27. Additionally, we are aware of the fact that this is not the only case where these notions of consistency coincide – e.g., this is also the case for the basic LP relaxation of WCSP (Remark 10). Even though all coefficients in the constraints (27) are from the set $\{-1, 0, 1\}$, the variables are only constrained to be non-negative and are not (explicitly) bounded to be at most 1 (even though this is done implicitly by the simplex constraint (27c)). We decided to omit a more general analysis that would cover also such cases because it would be technical and lengthy.

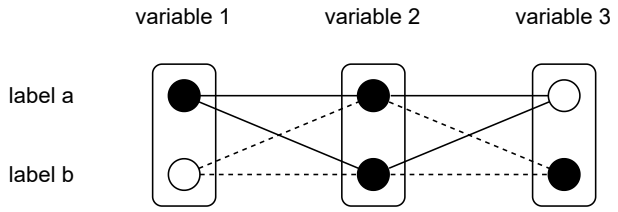


Figure 9: Illustration of the CSP considered in Appendix B. Allowed domain elements and binary tuples are shown by black circles and full lines, respectively, and forbidden domain elements and binary tuples are depicted by white circles and dashed lines, respectively.

Remark 20. *It is of interest in the constraint programming community to identify subsets of problems where enforcing a precise form of local consistency is a refutation-complete method, i.e., where attaining the local consistency implies satisfiability of the original system. This was studied, e.g., for arc consistency by Cohen and Jeavons (2017) and Cooper and Živný (2016). Dlask and Werner (2022) identified two classes of problems where activity propagation is refutation complete (more precisely, where BCD applied to the dual linear program is optimal, which is equivalent to refutation-completeness of activity propagation by Remark 7). In both of them, the variables are bounded by* $0 \leq x \leq 1$*, the constraint matrix contains only* $\{-1, 0, 1\}$ *coefficients, and vector* b *is integral, so the constraints are in the form* (57) *and satisfy all the assumptions stated in this section. Other classes of problems could be inherited by the connection between activity propagation and bounds/arc consistency stated here.*

Appendix B. Example of Propagation: Difference Between VAC and Our Algorithm

In this part, we present an example in which the propagation rules from Section 5.1 yield a different infeasibility certificate (dual-improving direction) than the VAC algorithm (Cooper et al., 2010; Koval & Schlesinger, 1976; Werner, 2007) or its generalization by Dlask et al. (2023), even if the sequence of propagation steps is the same.

Let $V = \{1, 2, 3\}$, $E = \{\{1, 2\}, \{2, 3\}\}$, and $D = \{\mathsf{a}, \mathsf{b}\}$. We thus have

$$\begin{aligned} P = \{&(1, \mathsf{a}), (1, \mathsf{b}), (2, \mathsf{a}), (2, \mathsf{b}), (3, \mathsf{a}), (3, \mathsf{b}), \\ &\{(1, \mathsf{a}), (2, \mathsf{a})\}, \{(1, \mathsf{a}), (2, \mathsf{b})\}, \{(1, \mathsf{b}), (2, \mathsf{a})\}, \{(1, \mathsf{b}), (2, \mathsf{b})\}, \\ &\{(2, \mathsf{a}), (3, \mathsf{a})\}, \{(2, \mathsf{a}), (3, \mathsf{b})\}, \{(2, \mathsf{b}), (3, \mathsf{a})\}, \{(2, \mathsf{b}), (3, \mathsf{b})\}\}. \end{aligned} \tag{63}$$

Next, let $c = (2, 0, 2, 2, 0, 2, 2, 2, 0, 0, 2, 0, 2, 0)$ where the order of the components is determined by (63) and $y = 0$ is the zero vector of $\mathbb{R}^Q$. Consequently,

$$\begin{aligned} J^*(y) = \{&(1, \mathsf{a}), (2, \mathsf{a}), (2, \mathsf{b}), (3, \mathsf{b}), \\ &\{(1, \mathsf{a}), (2, \mathsf{a})\}, \{(1, \mathsf{a}), (2, \mathsf{b})\}, \\ &\{(2, \mathsf{a}), (3, \mathsf{a})\}, \{(2, \mathsf{b}), (3, \mathsf{a})\}\} \end{aligned} \tag{64}$$

and the left-hand system (29) thus reads

$$\begin{aligned}
\phi_{1\mathsf{a},2}\colon&\quad x_{1\mathsf{a},2\mathsf{a}} + x_{1\mathsf{a},2\mathsf{b}} - x_{1\mathsf{a}} = 0\\
\phi_{1\mathsf{b},2}\colon&\quad x_{1\mathsf{b},2\mathsf{a}} + x_{1\mathsf{b},2\mathsf{b}} - x_{1\mathsf{b}} = 0\\
\phi_{2\mathsf{a},1}\colon&\quad x_{1\mathsf{a},2\mathsf{a}} + x_{1\mathsf{b},2\mathsf{a}} - x_{2\mathsf{a}} = 0\\
\phi_{2\mathsf{b},1}\colon&\quad x_{1\mathsf{a},2\mathsf{b}} + x_{1\mathsf{b},2\mathsf{b}} - x_{2\mathsf{b}} = 0\\
\phi_{2\mathsf{a},3}\colon&\quad x_{2\mathsf{a},3\mathsf{a}} + x_{2\mathsf{a},3\mathsf{b}} - x_{2\mathsf{a}} = 0\\
\phi_{2\mathsf{b},3}\colon&\quad x_{2\mathsf{b},3\mathsf{a}} + x_{2\mathsf{b},3\mathsf{b}} - x_{2\mathsf{b}} = 0\\
\phi_{3\mathsf{a},2}\colon&\quad x_{2\mathsf{a},3\mathsf{a}} + x_{2\mathsf{b},3\mathsf{a}} - x_{3\mathsf{a}} = 0\\
\phi_{3\mathsf{b},2}\colon&\quad x_{2\mathsf{a},3\mathsf{b}} + x_{2\mathsf{b},3\mathsf{b}} - x_{3\mathsf{b}} = 0\\
\phi_{1}\colon&\quad x_{1\mathsf{a}} + x_{1\mathsf{b}} = 1\\
\phi_{2}\colon&\quad x_{2\mathsf{a}} + x_{2\mathsf{b}} = 1\\
\phi_{3}\colon&\quad x_{3\mathsf{a}} + x_{3\mathsf{b}} = 1\\
&\quad x_j \geq 0 \quad \forall j \in J^*(y)\\
&\quad x_j = 0 \quad \forall j \in P \setminus J^*(y)
\end{aligned} \tag{65}$$

where we named the individual equalities in accordance to the corresponding dual variables $y_{1\mathsf{a},2}, \ldots, y_3$. The CSP with the set of allowed tuples (64) is depicted in Figure 9.

To initialize the propagation algorithm, we set $y^j = 0$ (the zero vector of $\mathbb{R}^P$) for all $j \in P \setminus J^*(y)$. The PWC propagation rules from Section 5.1.1 iteratively infer the following. Note that, to improve readability, we underline variables x_j with $j \in P \setminus J^*(y)$.

1. Using rule 1, set $x_{2\mathsf{a},3\mathsf{a}} = x_{2\mathsf{b},3\mathsf{a}} = 0$ via the inequality $x_{2\mathsf{a},3\mathsf{a}} + x_{2\mathsf{b},3\mathsf{a}} - \underline{x_{3\mathsf{a}}} = 0$ with cause vectors $y^{2\mathsf{a},3\mathsf{a}} = y^{2\mathsf{b},3\mathsf{a}} = e^{3\mathsf{a},2} + y^{3\mathsf{a}}$.
2. Using rule 2, set $x_{2\mathsf{a}} = 0$ via the equality $x_{2\mathsf{b},3\mathsf{a}} - \underline{x_{2\mathsf{a},3\mathsf{b}}} + x_{2\mathsf{a}} - \underline{x_{3\mathsf{a}}} = 0$ with cause vector $y^{2\mathsf{a}} = y^{2\mathsf{a},3\mathsf{a}} + y^{2\mathsf{a},3\mathsf{b}} - e^{2\mathsf{a},3}$.
3. Using rule 2, set $x_{2\mathsf{b}} = 0$ via the equality $x_{2\mathsf{a},3\mathsf{a}} - \underline{x_{2\mathsf{b},3\mathsf{b}}} + x_{2\mathsf{b}} - \underline{x_{3\mathsf{a}}} = 0$ with cause vector $y^{2\mathsf{b}} = y^{2\mathsf{b},3\mathsf{a}} + y^{2\mathsf{b},3\mathsf{b}} - e^{2\mathsf{b},3}$.
4. Using rule 1, set $x_{1\mathsf{a},2\mathsf{a}} = 0$ via the equality $x_{1\mathsf{a},2\mathsf{a}} + \underline{x_{1\mathsf{b},2\mathsf{a}}} + x_{2\mathsf{b},3\mathsf{a}} - \underline{x_{2\mathsf{a},3\mathsf{b}}} - \underline{x_{3\mathsf{a}}} = 0$ with cause vector $y^{1\mathsf{a},2\mathsf{a}} = y^{2\mathsf{a}} + e^{2\mathsf{a},1}$.
5. Using rule 1, set $x_{1\mathsf{a},2\mathsf{b}} = 0$ via the equality $x_{1\mathsf{a},2\mathsf{b}} + \underline{x_{1\mathsf{b},2\mathsf{b}}} + x_{2\mathsf{a},3\mathsf{a}} - \underline{x_{2\mathsf{b},3\mathsf{b}}} - \underline{x_{3\mathsf{a}}} = 0$ with cause vector $y^{1\mathsf{a},2\mathsf{b}} = y^{2\mathsf{b}} + e^{2\mathsf{b},1}$.
6. Using rule 2, set $x_{1\mathsf{a}} = 0$ via the equality $\underline{x_{1\mathsf{b},2\mathsf{a}}} + x_{2\mathsf{b},3\mathsf{a}} - \underline{x_{2\mathsf{a},3\mathsf{b}}} + \underline{x_{1\mathsf{b},2\mathsf{b}}} + x_{2\mathsf{a},3\mathsf{a}} - \underline{x_{2\mathsf{b},3\mathsf{b}}} - 2\underline{x_{3\mathsf{a}}} + x_{1\mathsf{a}} = 0$ with cause vector $y^{1\mathsf{a}} = y^{1\mathsf{a},2\mathsf{a}} + y^{1\mathsf{a},2\mathsf{b}} - e^{1\mathsf{a},2}$.
7. Using rule 3, detect infeasibility via the equality $\underline{x_{1\mathsf{b},2\mathsf{a}}} + x_{2\mathsf{b},3\mathsf{a}} - \underline{x_{2\mathsf{a},3\mathsf{b}}} + \underline{x_{1\mathsf{b},2\mathsf{b}}} + x_{2\mathsf{a},3\mathsf{a}} - \underline{x_{2\mathsf{b},3\mathsf{b}}} - 2\underline{x_{3\mathsf{a}}} - \underline{x_{1\mathsf{b}}} = -1$ with cause vector $\bar{y} = y^{1\mathsf{a}} + y^{1\mathsf{b}} - e^1$.

Stated explicitly, the found infeasibility certificate is $\bar{y} = (-1, 0, 1, 1, -1, -1, 2, 0, -1, 0, 0)$ where the order of the components corresponds to the order of the constraints in (65). The optimal step size with this improving direction $\bar{y}$ from the current point y is $\alpha^* = 1$ and, by updating from y to $y + \alpha^*\bar{y}$, the objective improves by 1.

However, the VAC algorithm with the same order of propagations finds the improving direction $\bar{y}' = (-1, 0, 1, 1, -1, -1, 1, 0, -1, 0, 0)$ which differs in the component $y_{3\mathsf{a},2}$. Due to its lower magnitude, the optimal step size is larger, namely $\alpha^{*\prime} = 2$. By updating from y

to $y + \alpha^{*\prime}\bar{y}'$, the objective improves by 2, which is a greater improvement when compared to the improving direction $\bar{y}$ above.

References

Achterberg, T. (2007). *Constraint integer programming.* Ph.D. thesis.

Achterberg, T., Bixby, R. E., Gu, Z., Rothberg, E., & Weninger, D. (2014). Multi-row presolve reductions in mixed integer programming. In *Proceedings of the Twenty-Sixth RAMP Symposium*, pp. 181–196.

Achterberg, T., Bixby, R. E., Gu, Z., Rothberg, E., & Weninger, D. (2020). Presolve reductions in mixed integer programming. *INFORMS Journal on Computing*, *32*(2), 473–506.

Apt, K. R. (1999). The rough guide to constraint propagation. In *Conference on Principles and Practice of Constraint Programming*, pp. 1–23. Springer.

Bacchus, F., Järvisalo, M., & Martins, R. (2019). MaxSAT Evaluation 2018: New developments and detailed results. *Journal on Satisfiability, Boolean Modeling and Computation*, *11*(1), 99–131. Instances available at `https://maxsat-evaluations.github.io/`.

Batra, D., Nowozin, S., & Kohli, P. (2011). Tighter relaxations for MAP-MRF inference: A local primal-dual gap based separation algorithm. In *Proceedings of the Fourteenth international conference on artificial intelligence and statistics*, pp. 146–154. JMLR Workshop and Conference Proceedings.

Beck, A. (2014). The 2-coordinate descent method for solving double-sided simplex constrained minimization problems. *Journal of Optimization Theory and Applications*, *162*, 892–919.

Belotti, P. (2013). Bound reduction using pairs of linear inequalities. *Journal of Global Optimization*, *56*, 787–819.

Belotti, P., Cafieri, S., Lee, J., Liberti, L., et al. (2010). Feasibility-based bounds tightening via fixed points.. In *COCOA (1)*, pp. 65–76.

Benhamou, F., & Granvilliers, L. (2006). Continuous and interval constraints. In *Handbook of Constraint Programming*, chap. 16. Elsevier.

Bertsekas, D. P. (1997). Nonlinear programming. *Journal of the Operational Research Society*, *48*(3), 334–334.

Bessiere, C. (2006). Constraint propagation. In *Handbook of Constraint Programming*, chap. 3. Elsevier.

Biere, A., Heule, M., & van Maaren, H. (2009). *Handbook of satisfiability*, Vol. 185. IOS press.

Bordeaux, L., Katsirelos, G., Narodytska, N., & Vardi, M. Y. (2011). The complexity of integer bound propagation. *Journal of Artificial Intelligence Research*, *40*, 657–676.

Boros, E., & Hammer, P. L. (2002). Pseudo-boolean optimization. *Discrete applied mathematics*, *123*(1-3), 155–225.

Brearley, A., Mitra, G., & Williams, H. P. (1975). Analysis of mathematical programming problems prior to applying the simplex algorithm. *Mathematical programming*, *8*, 54–83.

Burkard, R. E. (2002). Selected topics on assignment problems. *Discrete applied mathematics*, *123*(1-3), 257–302.

Cohen, D. A., & Jeavons, P. G. (2017). The power of propagation: when GAC is enough. *Constraints*, *22*(1), 3–23.

Cooper, M. C. (2003). Reduction operations in fuzzy or valued constraint satisfaction. *Fuzzy Sets and Systems*, *134*(3), 311–342.

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

Cooper, M. C., de Givry, S., & Schiex, T. (2007). Optimal soft arc consistency.. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, Vol. 7, pp. 68–73.

Cooper, M. C., & Živný, S. (2016). The power of arc consistency for CSPs defined by partially-ordered forbidden patterns. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, pp. 652–661.

Creignou, N. (1995). A dichotomy theorem for maximum generalized satisfiability problems. *Journal of Computer and System Sciences*, *51*(3), 511–522.

Dantzig, G., Fulkerson, R., & Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, *2*(4), 393–410.

Davis, E. (1987). Constraint propagation with interval labels. *Artificial intelligence*, *32*(3), 281–331.

de Givry, S., Heras, F., Zytnicki, M., & Larrosa, J. (2005). Existential arc consistency: Getting closer to full arc consistency in weighted CSPs. In *International Joint Conference on Artificial Intelligence*, Vol. 5, pp. 84–89.

de la Vega, W. F., & Kenyon-Mathieu, C. (2007). Linear programming relaxations of Maxcut. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 53–61. Citeseer.

Demaine, E. D., & Immorlica, N. (2003). Correlation clustering with partial information. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques: 6th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2003 and 7th International Workshop on Randomization and Approximation Techniques in Computer Science, RANDOM 2003, Princeton, NJ, USA, August 24-26, 2003. Proceedings*, pp. 1–13. Springer.

Devriendt, J., Gleixner, A., & Nordström, J. (2021). Learn to relax: Integrating 0-1 integer linear programming with pseudo-boolean conflict-driven search. *Constraints*, *26*(1), 26–55.

Dlask, T. (2018). *Minimizing Convex Piecewise-Affine Functions by Local Consistency Techniques*. Master's thesis, Czech Technical University in Prague, Faculty of Electrical Engineering.

Dlask, T. (2022). *Block-Coordinate Descent and Local Consistencies in Linear Programming*. Dissertation, available online: `https://dspace.cvut.cz/handle/10467/102874?locale-attribute=en`, Czech Technical University in Prague, Faculty of Electrical Engineering.

Dlask, T., & Werner, T. (2020). Bounding linear programs by constraint propagation: application to Max-SAT. In *International Conference on Principles and Practice of Constraint Programming*, pp. 177–193. Springer.

Dlask, T., & Werner, T. (2022). Classes of linear programs solvable by coordinate-wise minimization. *Annals of Mathematics and Artificial Intelligence*, *90*(7), 777–807.

Dlask, T., & Werner, T. (2023). Activity propagation in systems of linear inequalities and its relation to block-coordinate descent in linear programs. *Constraints*, *28*, 244—276.

Dlask, T., Werner, T., & de Givry, S. (2023). Super-reparametrizations of weighted CSPs: properties and optimization perspective. *Constraints*, *28*, 277–319.

Du, D., & Pardalos, P. M. (1999). *Handbook of combinatorial optimization*, Vol. 4. Springer Science & Business Media.

Freuder, E. C. (1982). A sufficient condition for backtrack-free search. *Journal of the ACM (JACM)*, *29*(1), 24–32.

Freund, R. M., Roundy, R., & Todd, M. J. (1985). Identifying the set of always-active constraints in a system of linear inequalities by a single linear program. Massachusetts Institute of Technology, Alfred P. Sloan School of Management.

Greenberg, H. J. (1996). Consistency, redundancy, and implied equalities in linear systems. *Annals of Mathematics and Artificial Intelligence*, *17*(1), 37–83.

Gurobi Optimization, LLC (2020). Gurobi Optimizer Reference Manual..

Haller, S., Prakash, M., Hutschenreiter, L., Pietzsch, T., Rother, C., Jug, F., Swoboda, P., & Savchynskyy, B. (2020). A primal-dual solver for large-scale tracking-by-assignment. In *International Conference on Artificial Intelligence and Statistics*, pp. 2539–2549. PMLR.

Harvey, W., & Stuckey, P. J. (2003). Improving linear constraint propagation by changing constraint representation. *Constraints*, *8*(2), 173–207.

Hooker, J. (2000). *Logic-based methods for optimization: combining optimization and constraint satisfaction*. Wiley series in discrete mathematics and optimization. Wiley.

Janssen, P., Jégou, P., Nouguier, B., & Vilarem, M.-C. (1989). A filtering process for general constraint-satisfaction problems: achieving pairwise-consistency using an associated binary representation. In *IEEE International Workshop on Tools for Artificial Intelligence*, pp. 420–421. IEEE Computer Society.

Jünger, M., Reinelt, G., & Rinaldi, G. (1995). The traveling salesman problem. *Handbooks in operations research and management science*, *7*, 225–330.

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. (2015). A comparative study of modern inference techniques for structured discrete

energy minimization problems. *International Journal of Computer Vision*, *115*(2), 155–184.

Khanna, S., & Sudan, M. (1996). The optimization complexity of constraint satisfaction problems. In *Electonic Colloquium on Computational Complexity*. Citeseer.

Komodakis, N., & Paragios, N. (2008). Beyond loose LP-relaxations: Optimizing MRFs by repairing cycles. In *European Conference on Computer Vision*.

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

Larrosa, J., & Schiex, T. (2003). In the quest of the best form of local consistency for weighted CSP. In *International Joint Conference on Artificial Intelligence*, Vol. 3, pp. 239–244.

Leiserson, C. E., Rivest, R. L., Cormen, T. H., & Stein, C. (1994). *Introduction to algorithms*, Vol. 3. MIT press Cambridge, MA, USA.

Lhomme, O. (1993). Consistency techniques for numeric CSPs. In *IJCAI*, Vol. 93, pp. 232–238.

Li, C.-M., Xu, Z., Coll, J., Manyà, F., Habet, D., & He, K. (2021). Combining Clause Learning and Branch and Bound for MaxSAT. In Michel, L. D. (Ed.), *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*, Vol. 210 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 38:1–38:18, Dagstuhl, Germany. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Magnanti, T., Ahuja, R., & Orlin, J. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Upper Saddle River, NJ.

Matoušek, J., & Gärtner, B. (2006). *Understanding and using linear programming (universitext)*. Springer-Verlag.

Nguyen, H., Bessiere, C., de Givry, S., & Schiex, T. (2017). Triangle-based consistencies for cost function networks. *Constraints*, *22*(2), 230–264.

Nguyen, H., Schiex, T., & Bessiere, C. (2013). Dynamic virtual arc consistency. In *The 28th Annual ACM Symposium on Applied Computing*, pp. 98–103.

Papadimitriou, C. H., & Steiglitz, K. (1998). *Combinatorial optimization: algorithms and complexity*. Courier Corporation.

Pardalos, P. M., Du, D.-Z., & Graham, R. L. (2013). *Handbook of combinatorial optimization*. Springer.

Průša, D., & Werner, T. (2019). Solving LP relaxations of some NP-hard problems is as hard as solving any linear program. *SIAM Journal on Optimization*, *29*(3), 1745–1771.

Puranik, Y., & Sahinidis, N. V. (2017). Domain reduction techniques for global NLP and MINLP optimization. *Constraints*, *22*(3), 338–376.

Rockafellar, R. T. (1972). *Convex Analysis*. Princeton University Press.

Rother, C., Kolmogorov, V., Lempitsky, V., & Szummer, M. (2007). Optimizing binary MRFs via extended roof duality. In *2007 IEEE conference on computer vision and pattern recognition*, pp. 1–8. IEEE.

Savchynskyy, B. (2019). Discrete graphical models – an optimization perspective. *Foundations and Trends in Computer Graphics and Vision*, *11*(3-4), 160–429.

Savelsbergh, M. W. (1994). Preprocessing and probing techniques for mixed integer programming problems. *ORSA Journal on Computing*, *6*(4), 445–454.

Schlesinger, M. (1976). Sintaksicheskiy analiz dvumernykh zritelnikh signalov v usloviyakh pomekh (Syntactic analysis of two-dimensional visual signals in noisy conditions). *Kibernetika*, *4*(113-130), 2.

Schlesinger, M. I., & Antoniuk, K. (2011). Diffusion algorithms and structural recognition optimization problems. *Cybernetics and Systems Analysis*, *47*, 175–192.

Schrijver, A. (1998). *Theory of linear and integer programming*. John Wiley & Sons.

Schrijver, A. (2004). *Combinatorial optimization: polyhedra and efficiency*. Springer Science & Business Media.

Schulte, C., & Stuckey, P. J. (2005). When do bounds and domain propagation lead to the same search space?. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, *27*(3), 388–425.

Sofranac, B., Gleixner, A., & Pokutta, S. (2022). An algorithm-independent measure of progress for linear constraint propagation. *Constraints*, *27*(4), 432–455.

Sontag, D., Choe, D. K., & Li, Y. (2012). Efficiently searching for frustrated cycles in MAP inference. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, UAI'12, p. 795–804, Arlington, Virginia, USA. AUAI Press.

Sontag, D., Meltzer, T., Globerson, A., Jaakkola, T., & Weiss, Y. (2008). Tightening LP relaxations for MAP using message passing. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, UAI'08, p. 503–510, Arlington, Virginia, USA. AUAI Press.

Swoboda, P., & Andres, B. (2017). A message passing algorithm for the minimum cost multicut problem. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1617–1626.

Swoboda, P., Kuske, J., & Savchynskyy, B. (2017). A dual ascent framework for Lagrangean decomposition of combinatorial problems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1596–1606.

Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., & Rother, C. (2008). 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.

Thapper, J., & Živný, S. (2012). The power of linear programming for valued CSPs. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pp. 669–678. IEEE.

Tourani, S., Shekhovtsov, A., Rother, C., & Savchynskyy, B. (2018). MPLP++: Fast, parallel dual block-coordinate ascent for dense graphical models. In *Proceedings of the European Conference on Computer Vision*, pp. 251–267.

Trösser, F., de Givry, S., & Katsirelos, G. (2020). Relaxation-aware heuristics for exact optimization in graphical models. In Hebrard, E., & Musliu, N. (Eds.), *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pp. 475–491, Cham. Springer International Publishing.

Tseng, P. (2001). Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, *109*(3), 475.

Vazirani, V. V. (2001). *Approximation Algorithms*. Springer-Verlag New York.

Živný, S. (2012). *The Complexity of Valued Constraint Satisfaction Problems*. Cognitive Technologies. Springer.

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

Werner, T. (2005). A linear programming approach to max-sum problem: A review. Tech. rep. CTU-CMP-2005-25, Center for Machine Perception, Czech Technical University.

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

Werner, T. (2010). 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.

Werner, T. (2014). Marginal consistency: Upper-bounding partition functions over commutative semirings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(7), 1455–1468.

Werner, T. (2017). 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.

Yanover, C., Meltzer, T., Weiss, Y., Bennett, K. P., & Parrado-Hernández, E. (2006). Linear programming relaxations and belief propagation–an empirical study.. *Journal of Machine Learning Research*, *7*(9).

Yuanlin, Z., & Yap, R. H. (2000). Arc consistency on n-ary monotonic and linear constraints. In *International Conference on Principles and Practice of Constraint Programming*, pp. 470–483. Springer.

Zadeh, N. (1970). A note on the cyclic coordinate ascent method. *Management Science*, *16*(9), 642–644.

Zhang, Z., Shi, Q., McAuley, J., Wei, W., Zhang, Y., & Van Den Hengel, A. (2016). Pairwise matching through max-weight bipartite belief propagation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1202–1210.