Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics

Block-Coordinate Descent and Local Consistencies in Linear Programming

Dissertation Thesis

Ing. Tomáš Dlask

Supervisor: doc. Ing. Tomáš Werner, PhD.

Study Programme: Electrical Engineering and Information Technology
Branch of Study: Artificial Intelligence and Biocybernetics

Prague, May 2022

Abstract

Even though linear programming (LP) problems can be solved in polynomial time, solving large-scale LP instances using off-the-shelf solvers may be difficult in practice, which creates demand for specialized scalable methods. One such method for large-scale problems is block-coordinate descent (BCD). However, the fixed points of this method need not be global optima even for convex optimization problems. Despite this limitation, various BCD algorithms (also called 'convergent message-passing algorithms') are successfully used for approximately solving the dual LP relaxation of the weighted constraint satisfaction problem (WCSP, also known as MAP inference in graphical models) and their fixed points can be characterized using local consistencies, typically variants of arc consistency.

In this work, we focus on optimizing linear programs by BCD or constraint propagation and theoretically relating these approaches. To this end, we propose a general constraint-propagation-based framework for approximate optimization of large-scale linear programs whose applicability is evaluated on publicly available benchmarks. In detail, we employ this approach to approximately optimize the dual LP relaxation of weighted Max-SAT and an LP formulation of WCSP. In the latter case, we show that one can use any classical CSP constraint propagation method in order to obtain an upper bound on the optimal value. This is in contrast to existing methods that needed to be tailored to a specific chosen kind of local consistency. However, the cost for this is that our approach may not preserve the properties of the input WCSP instance, such as the set of optimal assignments, and only provides an upper bound on its optimal value, which is nevertheless important for pruning the search space during branch-and-bound search.

Although one can use our general framework with any constraint propagation method in a system of linear inequalities, we identify the precise form of constraint propagation such that the stopping points of the resulting algorithm coincide with the fixed points of BCD. In other words, we identify the kind of local consistency that is enforced by BCD in any linear program. Depending on the problem being solved, this condition may be interpreted, e.g., as arc consistency or positive consistency. Thanks to these results, we characterize linear programs that are optimally solvable by BCD by refutation-completeness of the associated propagator (i.e., whether it can always detect infeasibility of a certain class of systems of linear inequalities and equalities). This allows us to identify new classes of linear programs exactly solvable by BCD, including, e.g., an LP formulation of the maximum flow problem or LP relaxations of some combinatorial problems.

We believe that this work may initiate further research on large-scale non-smooth constrained convex optimization problems.

Keywords: Linear Programming, Block-Coordinate Descent, Local Consistency, Constraint Propagation, Weighted Constraint Satisfaction Problem

Abstrakt

Přestože problémy lineárního programování (LP) lze řešit v polynomiálním čase, běžné metody nemusí být v praxi dostatečné pro problémy velkého rozsahu, což vytváří poptávku po specializovaných škálovatelných metodách. Takovou metodou pro rozsáhlé problémy je sestup po blocích souřadnic (BCD), jejíž fixní body ovšem nemusí být globální optima ani pro konvexní optimalizační problémy. Navzdory tomuto omezení jsou různé BCD algoritmy úspěšně používány pro přibližné řešení duální LP relaxace problému s váženými omezeními (WCSP, také známého jako MAP inference v grafových modelech) a jejich fixní body mohou být charakterizovány pomocí lokálních konzistencí, typicky variant hranové konzistence.

V této práci se zaměřujeme na optimalizaci lineárních programů pomocí BCD nebo propagace podmínek a teoreticky oba tyto přístupy propojujeme. Abychom dosáhli tohoto cíle, navrhneme obecné schéma založené na propagaci podmínek pro přibližnou optimalizaci lineárních programů velkého rozsahu, jehož použitelnost vyhodnocujeme na veřejně dostupných instancích. Konkrétně tento přístup aplikujeme k přibližné optimalizaci duální LP relaxace váženého Max-SAT problému a LP formulace WCSP. V případě LP formulace WCSP ukážeme, že k získání horní meze na optimální hodnotu je možné použít jakoukoli klasickou metodu propagace podmínek v CSP, což se liší od stávajících metod, které bylo nutno přizpůsobit konkrétnímu zvolenému druhu lokální konzistence. Nevýhodou našeho přístupu je, že nemusí zachovávat vlastnosti vstupní WCSP instance, jako například množinu optimálních řešení, ale poskytuje pouze horní mez na optimální hodnotu, která je nicméně důležitá pro prořezávání prohledávaného prostoru během metody větví a mezí.

Ačkoli je možné použít naše obecné schéma s jakoukoli metodou propagace podmínek v soustavě lineárních nerovností, našli jsme takový konkrétní způsob propagace, že fixní body výsledného algoritmu se shodují s fixními body BCD. Jinými slovy, našli jsme druh lokální konzistence, který BCD vynucuje v jakémkoli lineárním programu. V závislosti na řešeném problému se tato lokální konzistence může interpretovat například jako hranová konzistence nebo pozitivní konzistence. Díky těmto výsledkům charakterizujeme lineární programy, které jsou optimálně řešitelné pomocí BCD, pomocí úplnosti odpovídající metody propagace podmínek (tj. jestli daná metoda vždy dokáže detekovat nesplnitelnost jisté třídy lineárních rovnic a nerovnic). Tyto výsledky umožňují identifikovat nové třídy lineárních programů, které lze optimálně řešit pomocí BCD. Takové lineární programy zahrnují například LP formulaci problému hledání maximálního toku v síti nebo LP relaxace určitých kombinatorických problémů.

Věříme, že tato práce může podnítit další výzkum v oblasti konvexních nehladkých optimalizačních problémů velkého rozsahu s omezeními.

Klíčová slova: Lineární programování, Sestup po blocích souřadnic, Lokální konzistence, Propagace podmínek, Vážené CSP

Překlad názvu: Sestup po blocích souřadnic a lokální konzistence v lineárním programování

Acknowledgement

I am thankful to my supervisor Tomáš Werner for his guidance, frequent discussions, collaboration, and useful advice during my studies. I would also like to thank Daniel Průša and Simon de Givry for collaboration on joint papers. Furthermore, I appreciate the possibility to discuss ongoing research with other members of the Machine Learning group at the department. Last but not least, I wish to express my sincere gratitude to my family and friends for their support.

The support of the Grant Agency of the Czech Technical University in Prague (grants SGS19/170/OHK3/3T/13 and SGS22/061/OHK3/1T/13), the Czech Science Foundation (grant 19-09967S), and OP VVV project CZ.02.1.01/0.0/0.0/16_019/0000765 is gratefully acknowledged.

Contents

Appendix

Introduction

Optimization is nowadays ubiquitous in machine learning, computer vision, and artificial intelligence in general. Optimization problems also frequently emerge in industrial applications and, in the era of big data, may feature a very large number of variables and constraints. Even if the constraints are sparse, using classical off-the-shelf solvers is usually not suitable in practice which results in demand for specialized algorithms that could tackle such large-scale problems by utilizing their (usually regular) structure and sparsity.

To illustrate this, despite linear programming (LP) problems are solvable in polynomial time, even verifying feasibility of LP relaxations of some hard combinatorial problems may take significant amount of time, maybe even exceed the overall time limit for solving the original (non-relaxed) problem [47, §3.2]. As another example, LP instances originating in computer vision may have millions of constraints and variables [154, 119, Example 4.2]. Off-the-shelf LP solvers typically cannot be applied to such large instances [154, 129, 128, 73, 114] due to their super-linear time and/or space complexity. Since LP relaxations of many classical NP-hard problems are as hard to solve as any linear program [114], designing more efficient exact methods for these relaxations may result in improving upon the best known general-purpose LP solvers, which is unlikely.

One of scalable methods is block-coordinate descent (BCD, a.k.a. block-coordinate minimization). This is an iterative method for (approximate [1]) optimization of a multivariate function which, in each iteration, chooses a subset (also called a block) of variables and optimally solves the problem over this subset of variables while keeping the other variables constant. By repeating this iteration for different blocks, the method can eventually converge to a 'local' minimum (understood w.r.t. block-coordinate moves) which is optimal w.r.t. all blocks of variables, or even a global minimum. In the simplest setting, the blocks correspond to single variables, which is usually called coordinate-wise minimization. In this case, the optimization subproblems are univariate and can be easily solved, sometimes even in closed-form, which results in simple and efficient algorithms.

BCD has been successfully applied to a number of optimization problems, such as support vector machine training [77, 112], non-negative least squares [60], non-negative matrix factorization [78], regression [152, 63], or semidefinite programs with diagonal constraints [143, 144].

However, except for special cases [155, 15, §2.7, 12, 137], BCD may not even converge to the set of global minima for general constrained or non-differentiable convex optimization problems and the BCD local minima may be arbitrarily far from global minima. Despite this fundamental limitation, BCD (a.k.a. convergent message passing in this context) is a powerful heuristic to approximately optimize unconstrained convex piecewise-affine (hence non-differentiable) functions emerging as various forms of the dual LP relaxation of the weighted constraint satisfaction problem (WCSP) [65, 134, 88, 87, 135]. These methods have been generalized and applied to LP relaxations of other large-scale combinatorial problems, such as minimum cost multicut problem [128] or graph matching [130], within

[1] Throughout the thesis, whenever we write 'approximately optimize', we mean 'attempt to find some (hopefully good) solution that can be however arbitrarily bad in theory', i.e., there are no formal guarantees on its quality.

a general framework of BCD applied to a Lagrange dual decomposition of combinatorial problems [129]. Other related approaches are [73] and [97].

Let us now return to the WCSP. The WCSP is a combinatorial NP-hard optimization problem where the task is to maximize a function of many discrete variables that is expressed as the sum of weight functions where each weight function depends only on a (small) subset of the variables. Its dual LP relaxation can be interpreted as minimizing an upper bound on the optimal value of a WCSP over a subset of its reparametrizations (i.e., WCSPs with the same objective value for all assignments) and the fixed points of various BCD methods applied to this linear program can be characterized by certain local consistencies. On the other hand, there are also methods that (approximately) optimize this dual LP relaxation by directly enforcing (possibly soft) local consistencies [43, 98, 33, 107, 95] and, in case of [33, 95], attain fixed points of the same nature as the BCD algorithms mentioned above. Although these algorithms may not solve the LP relaxation exactly, they provide a bound on its optimal value which is essential for pruning the search space during branch-and-bound search which is the usually accepted method for optimally solving WCSPs.

A similar connection between coordinate-wise minimization and local consistencies was identified in [148] for general unconstrained convex piecewise-affine functions. In detail, [148] found a connection between a certain local consistency condition and fixed points of coordinate-wise minimization with a special update rule. This was further developed in the author's master thesis [48a].

The above-mentioned update rule from [148] is in fact a special case of the relative-interior rule, which was proposed in [150, 151a] and is one of the baselines for this work. To motivate this rule, realize that the set of block-minimizers in BCD may generally contain multiple elements and, in such case, one has to choose a single element from this set to perform the BCD update. The relative-interior rule [150, 151a] additionally requires that the minimizer is chosen from the relative interior of the set of block-minimizers. Although this rule is not worse than any other update rule for choosing non-unique block-minimizers, the local minima of BCD following this rule can still be arbitrarily bad.

In this dissertation, we aim to extend the aforementioned constraint-propagation-based methods to the more general setting of optimizing arbitrary LP problems, investigate applicability of BCD to linear programs, and, still focusing on linear programs, link theory of BCD to constraint programming.

Structure and Contributions

Let us now overview the structure and aims of the thesis which is divided into five main chapters:

- **§1:** We begin with providing an overview of the necessary background and related work that spans over five areas:
  - §1.1: Linear programs and inference in systems of linear inequalities and equalities.
  - §1.2: BCD applied to convex optimization problems with focus on linear programs, relative-interior rule, properties of BCD, and types of local minima occurring in BCD.
  - §1.3: Order theory which forms the theoretical basis for constraint propagation.

- §1.4: Constraint satisfaction problem (CSP), local consistencies, and their enforcing by propagators.
- §1.5: WCSP and LP-based methods for bounding its optimal value.

- **§2:** Using constraint propagation to optimize linear programs (or even other convex optimization problems) is rare aside from the aforesaid approaches for bounding the optimal value of the WCSP. We propose a theoretical framework for (approximate) optimization of large-scale linear programs using constraint propagation, which generalizes the previously mentioned approaches. In this framework, we assume that an initial feasible point is available which we then try to improve, ideally make it optimal. We apply this framework to approximately optimize the dual LP relaxation of weighted Max-SAT and experimentally verify the quality of its stopping points.

  This part follows the explanation given by Dlask and Werner in the conference paper [52a] with some new insights.

- **§3:** It is usually not easy to generalize local consistencies from ordinary CSPs to WCSPs so that their enforcing by reparametrizing the WCSP improves the bound on its optimal value. Such a generalization may be even impossible because the level of local consistency that can be achieved via reparametrizations without introducing new weight functions (likely of higher arity) is limited. In cases when this is possible, the method for enforcing a (possibly soft) local consistency in the WCSP typically needs to be tailored to the particular chosen kind of local consistency.

  In contrast, we propose a method that is able to improve the bound on the WCSP optimal value using any kind of constraint propagation without introducing new weight functions. For this, we use an LP formulation (i.e., not a relaxation) of the WCSP which can be interpreted as minimizing an upper bound over its super-reparametrizations (i.e., WCSPs whose objective value is the same or greater for all assignments). Although this formulation was already proposed [92], we newly show that it can be approximately optimized using any method that can (at least sometimes) detect unsatisfiability of a CSP. The resulting algorithm can be seen as an instance of the general framework for optimizing large-scale linear programs (in this case, with an exponential number of constraints) by constraint propagation from §2.

  The properties of this optimization problem and of super-reparametrizations have not been thoroughly studied in the literature and we aim to fill in this gap. As already mentioned, one of the benefits of this optimization problem is that it allows us to enforce arbitrarily strong local consistencies without introducing higher-arity weight functions, thus saving memory. The cost for this is that our method provides only a bound on the optimal value but may not preserve other properties of the input WCSP, such as the objective value for the individual assignments or the set of optimal assignments.

  Our results in §3 are based on the journal submission [56a] which is an improved version of the conference paper [55a], both authored by Dlask, Werner, and de Givry.

- **§4:** Seeing that the stopping points of some constraint-propagation-based algorithms for bounding the WCSP optimal value are related to the fixed points of several BCD algorithms (applied to its dual LP relaxation), we explain this connection and generalize it to arbitrary linear programs with any blocks of variables. As a by-product, we characterize the types of local minima encountered in BCD by local consistency conditions, thus link BCD to constraint propagation.

More generally, we define a class of optimization methods based on enforcing a suitable local consistency whose stopping points are related to fixed points of BCD in an analogous way. In other words, we identify the precise constraint propagation rule that corresponds to BCD. Depending on the problem being optimized, this can be interpreted, e.g., as enforcing arc consistency or performing unit propagation.

The results described in §4 are an improved version of the conference paper [54a] by Dlask and Werner.

- **§5:** The classes of convex optimization problems for which the BCD fixed points are global optima are currently not much broader than unconstrained smooth convex functions. Such classes also include the dual LP relaxation of acyclic, supermodular, or pairwise Boolean WCSPs.

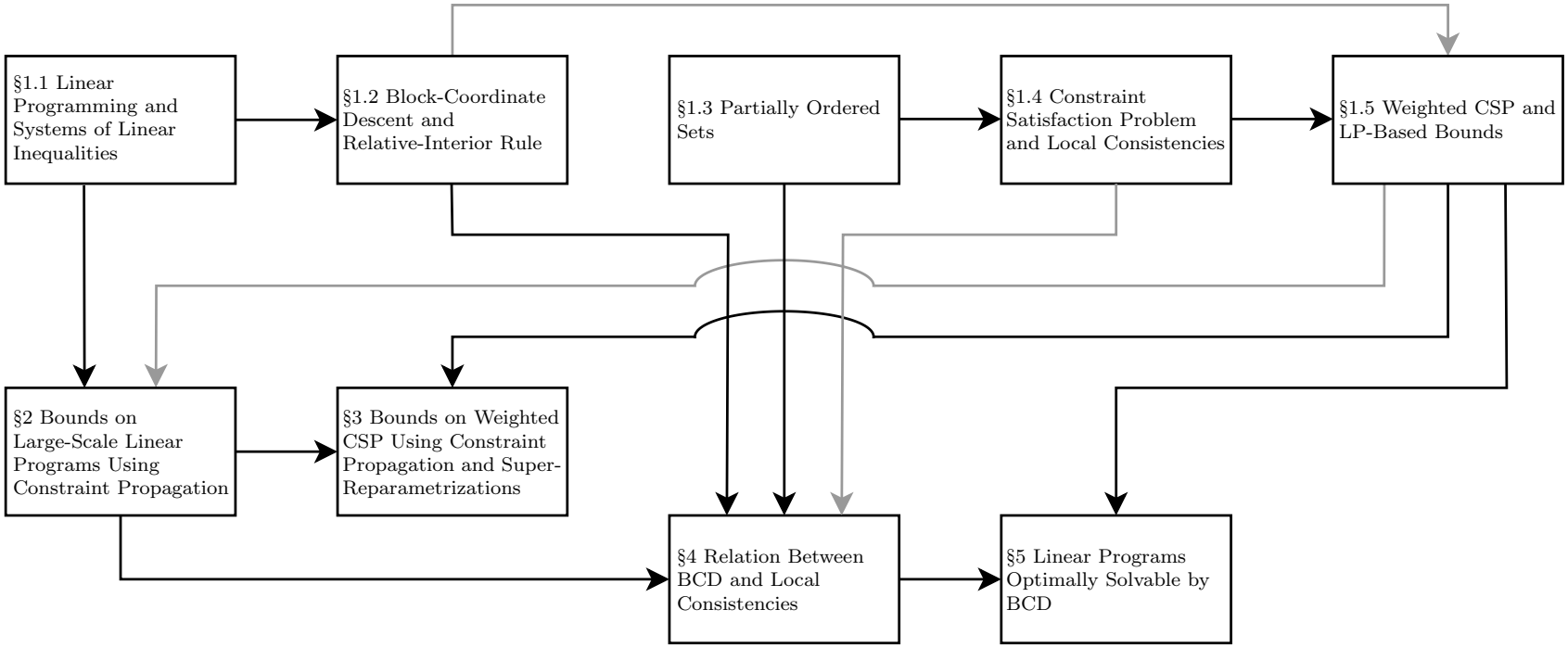
  Focusing on linear programs, we identify new classes of such problems. In fact, we even provide a characterization of linear programs that are optimally solvable by BCD in terms of constraint propagation. To be precise, the question of optimality of BCD fixed points for linear programs can be translated to the question whether a precisely defined constraint propagation rule can always detect infeasibility of a certain class of systems of linear inequalities and equalities. The newly identified classes of linear programs include, e.g., a suitable formulation of the maximum flow problem, LP relaxations of certain combinatorial problems, or the aforementioned LP formulation of a WCSP. Finally, we also explain why applicability of BCD depends on the precise formulation of the optimization problem and exemplify this phenomenon on non-trivial optimization problems, namely maximum flow, LP relaxation of weighted vertex cover, and a special LP relaxation of pairwise WCSP with Potts interactions.

  This chapter is mainly based on the results of Dlask and Werner in the journal paper [53a] (which is an improved version of the conference paper [51a] by the same authors) combined with a few insights from the conference papers [54a] (by Dlask and Werner) and [151a] (by Werner, Průša, and Dlask), and some new parts.

We conclude the thesis by summarizing the achieved results and discussing possible directions for future research. The diagram on the next page visualizes the dependencies among sections in §1 and subsequent chapters in the body of the thesis.

Let us have a few remarks on the style of the presentation. Our contributions (in §2-§5) are organized in a 'method-oriented' (rather than 'application-oriented') manner: an idea/approach is always first explained for the case of general linear programs and, after that, we exemplify how it manifests itself in specific applications. The applications include not only linear programs connected to the WCSP but also LP relaxations/formulations of other problems. This implies that we may encounter the same optimization problem multiple times throughout the thesis, each time seen from a different perspective.

Finally, let us note that the topic of this dissertation is interdisciplinary – we base our research on local consistencies from the field of constraint programming, BCD methods from optimization, and message-passing algorithms from computer vision and machine learning. This implies that there are typically multiple options for notation, nomenclature etc. that were developed independently in different fields. Therefore, e.g., a reader coming from the field of optimization may find some notation from constraint programming unusual and vice versa. We provide an overview of our notation and used abbreviations in the appendix.



Dependencies among the sections in §1 and subsequent chapters. Black arrows indicate that a section/chapter significantly depends on another whereas gray arrow is only a slight dependence. The full set of dependencies is obtained as the transitive closure of this diagram.

Chapter 1

Background

In this chapter, we provide the necessary background for linear programming, systems of linear inequalities, block-coordinate descent, partially ordered sets, and constraint satisfaction problems. We also review the weighted constraint satisfaction problem and approaches that are used for obtaining an upper bound on its optimal value. Up to a few minor insights, we do not present any novel results in this chapter but only overview already known pieces of knowledge which also help us introduce and exemplify our notation.

1.1 Linear Programming and Systems of Linear Inequalities

Linear programming is well known and finds its use in various applications [103, 24, §4.3.1]. In general, a linear program seeks to minimize or maximize a linear objective function of a finite number of variables over a set determined by a finite number of linear inequalities and equalities (i.e., a polyhedron). To each linear program, one can write its dual linear program and these optimization problems are connected by duality theorems. This construction is symmetric, so we can talk about being mutually dual.

Although linear programs come in various forms (e.g., containing both inequalities and equalities, non-negative and real-valued variables etc.), we consider the following form of a primal-dual pair

$$\begin{aligned} \max\ & c^\top x && \min\ b^\top y && \text{(1.1a)}\\ & Ax = b && y \in \mathbb{R}^m && \text{(1.1b)}\\ & x \geq 0 && A^\top y \geq c && \text{(1.1c)} \end{aligned}$$

where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ are constants and $x \in \mathbb{R}^n, y \in \mathbb{R}^m$ are variables. We denote by x_j the j-th component of vector x (similarly for y, b, c) and by A^i and A_j the i-th row and j-th column of A where $i \in [m] = \{1, \ldots, m\}$ and $j \in [n] = \{1, \ldots, n\}$, respectively. $A^\top$ is the transpose of A. We will refer to the left-hand problem (1.1) as the primal and to the right-hand problem (1.1) as the dual.

Any linear program can be easily transformed into the form of the primal or the dual (1.1) (see [103, §1.1 and §4.1]). We note that whenever we write a pair of mutually dual linear programs, we always write a constraint and the corresponding Lagrange multiplier on the same line, as in (1.1). For any primal-dual pair, *strong duality* holds:

Theorem 1.1 (Strong duality [123, §7.4, 103, §6.1])**.** *For any primal-dual pair, only one of the following cases can happen:*

(a) Both the primal and the dual are feasible and their optimal values coincide.

(b) Both the primal and the dual are infeasible.

(c) The primal is unbounded and the dual is infeasible or vice versa.

Optimality conditions for feasible solutions of linear programs are given by the well-known *complementary slackness* conditions that are stated in the following theorem.

Theorem 1.2 (Complementary slackness [103, 110])**.** *Let $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ be feasible for the primal and the dual* (1.1)*, respectively. The following are equivalent:*

(a) x and y are optimal for the primal and the dual, respectively,

(b) $\forall j \in [n]\colon x_j(A_j^\top y - c_j) = 0$, i.e., $\forall j \in [n]\colon (x_j = 0) \vee (A_j^\top y = c_j)$.

For brevity of notation, we define the mappings $\sigma\colon \mathbb{R}^n \to 2^{[n]}$ and $\tau\colon \mathbb{R}^m \to 2^{[n]}$ by

$$\sigma(x) = \{j \in [n] \mid x_j = 0\} \tag{1.2a}$$

$$\tau(y) = \{j \in [n] \mid A_j^\top y = c_j\}, \tag{1.2b}$$

so that $\sigma(x)$ is the index set of the primal constraints (1.1c) that are *active* (i.e., satisfied with equality [2]) at x. Similarly, $\tau(y)$ is the index set of the dual constraints (1.1c) that are active at y. Using this notation, statement (b) in Theorem 1.2 can be expressed as $\tau(y) \cup \sigma(x) = [n]$.

1.1.1 Relative Interior and Strict Complementarity

In the sequel, we will frequently utilize the notion of *relative interior* of a convex set. We now recall its definition and important properties.

Definition 1.1 ([100, Definition 2.1.1])**.** *Let $S \subseteq \mathbb{R}^n$ be a convex set. The* relative interior *of S, denoted by* $\operatorname{ri} S$*, is the topological interior of S relative to the affine hull of S, i.e.,*

$$\operatorname{ri} S = \{x \in S \mid \exists r > 0\colon B_r(x) \cap \operatorname{aff} S \subseteq S\} \tag{1.3}$$

where $B_r(x) = \{y \in \mathbb{R}^n \mid \|x - y\| \leq r\}$ is the ball centered at x with radius $r > 0$ and $\operatorname{aff} S$ *is the affine hull of S.*

Example 1.1 ([100, Table 2.1.1, 150, §4, 151a, §3])**.** *Let $x, x' \in \mathbb{R}^n$. We have that* $\operatorname{ri}\{x\} = \{x\}$*, i.e., the relative interior of a singleton set is the set itself. Furthermore, if $x \neq x'$, then* $\operatorname{ri}[x, x'] = [x, x'] - \{x, x'\}$ *where $[x, x'] = \{\alpha x + (1 - \alpha)x' \mid 0 \leq \alpha \leq 1\}$ is the line segment between x and x'.* △

It follows directly from Definition 1.1 that, for a convex set S, $\operatorname{ri} S \subseteq S$, and it is known [100, Theorem 2.1.3] that $\operatorname{ri} S = \emptyset$ if and only if $S = \emptyset$.

Consequently, since the set of optimal solutions of a linear program is always a convex set, its non-emptiness is equivalent to non-empty relative interior. The *strict complementary slackness* condition can be used to determine whether a primal solution x and a dual solution y lie in the relative interior of the set of optimal solutions of a primal and a dual linear program, respectively.

Theorem 1.3 (Strict complementary slackness [66, 80, 69, 158])**.** *Let $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ be feasible for the primal and the dual* (1.1)*, respectively. The following are equivalent:*

[2] In general, an inequality $c^\top x \geq d$ is called *active* for some x if $c^\top x = d$ [122, §5.5, 62, 24, §4.1.1].

(a) *x and y are in the relative interior of the set of optimizers of the primal and the dual, respectively,*

(b) *$\{\sigma(x), \tau(y)\}$ is a partition of $[n]$, i.e., $\forall j \in [n]\colon (x_j = 0) \oplus (A_j^\top y = c_j)$ where $\oplus$ denotes exclusive disjunction.*

As a corollary, for any x and y in the relative interior of the set of optimal solutions of the primal and the dual, respectively, the partition $\{\sigma(x), \tau(y)\}$ is the same and is typically referred to as the optimal partition of $[n]$ [80, 4, 69, 104].

1.1.2 Convex Piecewise-Affine Objective

In multiple places throughout this thesis, we will utilize the well-known trick [16, §1.3, 24, §4.3.1] that allows us to formulate the problem of minimizing a convex piecewise-affine function over a polyhedron as a linear program and vice versa.

Formally, we address the optimization problem

$$\min \sum_{k \in K} \max_{l \in L_k} (c_{k,l}^\top x + d_{k,l}) \tag{1.4a}$$

$$x \in X \tag{1.4b}$$

where $X \subseteq \mathbb{R}^n$ is a polyhedron, K is a finite set, L_k a finite non-empty set for each $k \in K$, and $c_{k,l} \in \mathbb{R}^n$ and $d_{k,l} \in \mathbb{R}$ are given vectors and real numbers for each $k \in K,\ l \in L_k$. The objective (1.4a) is a convex piecewise-affine [3] function of x.

It is easy to see that (1.4) can be reformulated as the linear program

$$\min \sum_{k \in K} z_k \tag{1.5a}$$

$$z_k \geq c_{k,l}^\top x + d_{k,l} \qquad \forall k \in K,\ l \in L_k \tag{1.5b}$$

$$x \in X \tag{1.5c}$$

$$z \in \mathbb{R}^K \tag{1.5d}$$

where we introduced auxiliary variables $z \in \mathbb{R}^K$. The reason is that for any $x \in X$, we can define

$$z_k = \max_{l \in L_k} (c_{k,l}^\top x + d_{k,l}) \qquad \forall k \in K \tag{1.6}$$

so that (x, z) is feasible for (1.5) and the objective values for both (1.4) and (1.5) coincide. On the other hand, for any (x, z) feasible for (1.5), x is also feasible for (1.4) and the objective (1.4a) is lower than or equal to the objective (1.5a) due to $z_k \geq \max_{l \in L_k} (c_{k,l}^\top x + d_{k,l})$ for all $k \in K$ by (1.5b). Consequently, the optimal values of (1.4) and (1.5) are equal.

1.1.3 Systems of Linear Inequalities and Linear Inference

Let us now point our attention to the logical view of linear inequalities. We briefly overview the basic properties of inference in a system of linear inequalities (and possibly equalities) and Farkas' lemma.

[3] This is called *piecewise-linear* in [16, §1.3]. However, the terminology is not unified [24, Example 3.5] as a linear function may not contain a constant term in some formalisms, so we use the unambiguous term *piecewise-affine*, as in [148, 48a].

In the sequel, we will call a system of linear inequalities (and equalities) *feasible* if it has a solution (i.e., the polyhedron defined by this system is non-empty). Otherwise, it is *infeasible*.

Moreover, for $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, and $d \in \mathbb{R}$, we will say that a system $Ax \geq b$ *implies* $c^\top x \geq d$ if $\forall x \in \mathbb{R}^n\colon Ax \geq b \implies c^\top x \geq d$ (i.e., if $c^\top x \geq d$ holds for all $x \in \mathbb{R}^n$ satisfying $Ax \geq b$). Analogously, a system $Ax \geq b$ *implies* $c^\top x = d$ if $\forall x \in \mathbb{R}^n\colon Ax \geq b \implies c^\top x = d$.[4]

As we will discuss, the logic of linear inequalities is simple in the sense that any implied inequality can be derived as a suitable combination of the existing inequalities (up to certain technical details). In particular, a contradictory inequality can be derived from any infeasible system in this way [76, §17.2-§17.3, 103, §6.4]. In contrast to the discrete setting of constraint propagation, which is considered later in §1.4.1, deciding whether a system of linear inequalities implies another given linear inequality is solvable in polynomial time by posing the problem as a linear program.

Formally, the logic of linear inequalities is described by the *affine form of Farkas' lemma*:

Theorem 1.4 (Affine form of Farkas' lemma [123, §7.6])**.** *Let $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, and $d \in \mathbb{R}$. The following are equivalent:*

(a) $Ax \geq b$ implies $c^\top x \geq d$,

(b) $Ax \geq b$ is infeasible or there is $y \geq 0$ such that $A^\top y = c$ and $b^\top y \geq d$.

Proof. Direction (b) $\implies$ (a) is immediate: for y satisfying (b) and any x satisfying $Ax \geq b$, we have $c^\top x = y^\top Ax \geq y^\top b \geq d$.

The other direction (a) $\implies$ (b) follows from[5] strong duality applied to the primal-dual pair (1.1) after changing A to $A^\top$, interchanging b with c and x with y. In particular, if $A^\top y \geq c$ implies $b^\top y \geq d$, then the dual (1.1) is infeasible or it is feasible and bounded and its optimal value is at least d. If it is feasible and bounded, the primal is too, and its optimal solution x satisfies $Ax = b, x \geq 0$ and $c^\top x \geq d$. □

Clearly, $c^\top x \geq d'$ implies $c^\top x \geq d$ if and only if $d' \geq d$ (or if $c^\top x \geq d'$ is infeasible). Thus, one can interpret Theorem 1.4 as follows: a feasible system $Ax \geq b$ implies $c^\top x \geq d$ if and only if a non-negative combination of the inequalities in $Ax \geq b$ implies $c^\top x \geq d$ (where the non-negative combination of inequalities is $c^\top x \geq d'$ where $c = A^\top y$ and $d' = b^\top y$). The vector y from Theorem 1.4 contains the coefficients of the non-negative combination and thus stores how the implied inequality was derived. Therefore, such a vector y can be called the *certificate* or the *cause vector* of the implied inequality.

Example 1.2. *Let the system $Ax \geq b$ be given by the constraints*

$$2x_1 + x_2 \geq 0 \tag{1.7a}$$
$$x_2 \geq 2 \tag{1.7b}$$
$$-x_1 + x_2 \geq 1. \tag{1.7c}$$

[4] For completeness, we note that $Ax \geq b$ does *not* imply $c^\top x \geq d$ if $\exists x \in \mathbb{R}^n$ such that $Ax \geq b$ and $c^\top x < d$. Also, $Ax \geq b$ does *not* imply $c^\top x = d$ if $\exists x \in \mathbb{R}^n$ such that $Ax \geq b$ and $c^\top x \neq d$.

[5] Conversely, Theorem 1.4 can be used to prove strong duality in linear programming [76, §17.3.1].

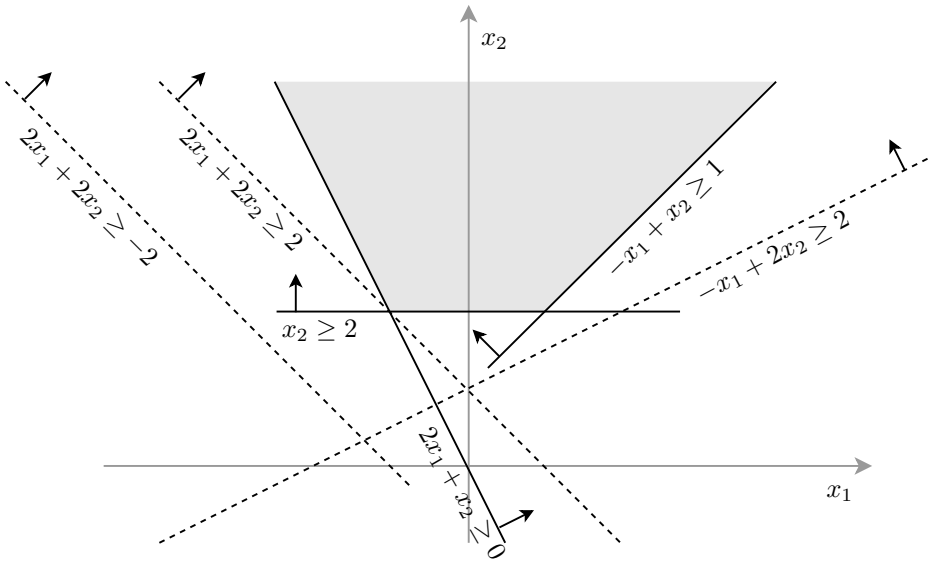


Figure 1.1: Illustration to Example 1.2. The half-planes determined by the constraints (1.7) are indicated by full lines with arrows and their intersection is shaded. The half-planes determined by the three implied inequalities from Example 1.2 are indicated by dashed lines with arrows.

This system implies $-x_1 + 2x_2 \geq 2$*, i.e.,* $c^\top x \geq d$ *for* $c = (-1, 2)$ *and* $d = 2$ *because for* $y = (\frac{1}{4}, \frac{1}{4}, \frac{3}{2})$*, we have that* $A^\top y = (-1, 2) = c$ *and* $b^\top y = 2 = d$*. Symbolically, this can be expressed as*

$$\tfrac{1}{4} \cdot (2x_1 + x_2 \geq 0) + \tfrac{1}{4} \cdot (x_2 \geq 2) + \tfrac{3}{2} \cdot (-x_1 + x_2 \geq 1) = (-x_1 + 2x_2 \geq 2). \tag{1.8}$$

Another implied inequality is, e.g., $2x_1 + 2x_2 \geq 2$ *whose cause vector is* $y' = (1, 1, 0)$*. In detail, we have* $A^\top y' = (2, 2)$ *and* $b^\top y' = 2$*. Thus,* (1.7) *also implies* $2x_1 + 2x_2 \geq d$ *for any* $d \leq 2$ *(e.g.,* $d = -2$*) because for the same* y'*, we have* $A^\top y' = (2, 2)$ *and* $b^\top y' = 2 \geq d$*.*

The set defined by (1.7) *along with the three mentioned implied inequalities is depicted in Figure 1.1.* △

The famous *Farkas' lemma* can be obtained as a corollary of Theorem 1.4:

Corollary 1.1 (Farkas' lemma [123, §7.3, 103, §6.4])**.** *Let* $A \in \mathbb{R}^{m \times n}$ *and* $b \in \mathbb{R}^m$*. Exactly one of the following systems is feasible:*

(a) $Ax \geq b$*,*

(b) $y \geq 0,\ A^\top y = 0,\ b^\top y > 0$*.*

Proof. We proceed analogously to the proof in [76, Corollary 62]. Let us define $u \in \mathbb{R}^m$ by $u_i = 1$ for all $i \in [m]$ (so that vector u contains only ones as its components). First, note that the system $Ax + tu \geq b$ (with an additional variable $t \in \mathbb{R}$) is always feasible because t can be set large enough. Second, it is easy to see that $Ax \geq b$ is infeasible if and only if $Ax + tu \geq b$ implies $t \geq d$ for any $d > 0$. Applying Theorem 1.4 to this implication yields that $Ax + tu \geq b$ implies $t \geq d$ if and only if there is y satisfying

$$y \geq 0,\ A^\top y = 0,\ u^\top y = 1,\ b^\top y \geq d. \tag{1.9}$$

Since $d > 0$ can be chosen small enough, system (b) in the corollary and system (1.9) are equisatisfiable (i.e., both systems are feasible or both are infeasible). In detail, any y satisfying (1.9) with $d > 0$ also satisfies system (b) and any y satisfying system (b) can be scaled to satisfy (1.9) for some $d > 0$. □

See that any y satisfying system (b) in Corollary 1.1 is a cause vector of inequality $0^\top x \geq d$ with $d = b^\top y > 0$ which is clearly infeasible. Thus, one can interpret Farkas' lemma as follows: system $Ax \geq b$ is infeasible if and only if a non-negative combination of the linear inequalities in $Ax \geq b$ is infeasible (i.e., if the system implies an infeasible inequality). Thus, any y satisfying system (b) in Corollary 1.1 is a *certificate of infeasibility* of system (a). System (b) is called the *Farkas alternative* to system (a).

There exist several variants of Farkas' lemma and its affine form, each corresponding to a different form of the system, possibly including also equality constraints or non-negative variables [123, §7.3, 103, §6.4]. For each such system, there is a corresponding Farkas alternative system enjoying the same properties as in Corollary 1.1. [6] We show an example below for the system

$$Ax = b \tag{1.10a}$$
$$x \geq 0. \tag{1.10b}$$

Theorem 1.5. *Let $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, and $d \in \mathbb{R}$. The following are equivalent:*

(a) $Ax = b,\ x \geq 0$ implies $c^\top x \geq d$,

(b) $Ax = b,\ x \geq 0$ is infeasible or there is $y \in \mathbb{R}^m$ such that $b^\top y \geq d$ and $A^\top y \leq c$.

In case that (1.10) is feasible and implies $c^\top x \geq d$, any y satisfying the conditions in statement (b) in Theorem 1.5 is a cause vector of $c^\top x \geq d$ and it encodes how this inequality can be obtained from the original system (1.10). Indeed, y contains the coefficients with which the equalities in (1.10a) can be combined to obtain the equality $(y^\top A)x = y^\top b$. By non-negativity of x, we have $c^\top x \geq (y^\top A)x$ due to $c \geq A^\top y$, so $c^\top x \geq (y^\top A)x = y^\top b \geq d$ holds for any x feasible for (1.10). The corresponding form of Farkas' lemma for system (1.10) is the following:

Corollary 1.2 ([123, Corollary 7.1d, 103, Proposition 6.4.1])**.** *Let $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. Exactly one of the following systems is feasible:*

(a) $Ax = b,\ x \geq 0$,

(b) $A^\top y \geq 0,\ b^\top y < 0$.

Consequently, any y satisfying system (b) in Corollary 1.2 is a certificate of infeasibility of system (a). Note, such y encodes an equality in the form $(y^\top A)x = y^\top b$ where the left-hand side is a non-negative sum of non-negative variables but the right-hand side is negative. Finally, system (b) in Corollary 1.2 is called the Farkas alternative to system (a).

Remark 1.1. *Theorem 1.5 and Corollary 1.2 apply to the system* (1.10) *which is the set of feasible solutions of the primal* (1.1)*. We chose to present Theorem 1.4 and Corollary 1.1 (which consider a system in the form of the dual constraints* (1.1b)-(1.1c)*) earlier because the aforementioned concepts are easier to illustrate for such a form.*

[6] This is analogous to the fact that (strong) duality, complementary slackness etc. in linear programming hold for problems in any form (e.g., including both equalities and inequalities, non-negative and real-valued variables etc.) even though we presented the theory only for the particular case of (1.1).

1.1.3.1 Always-Active Inequalities

For some systems of linear inequalities, it may happen that an inequality constraint is active for any solution of the system, i.e., it always holds as an equality constraint. Such constraints are called *always active* [62] (a.k.a. *implied equalities* [68] or *implicit equalities* [131, 123, §8.1, 122, §5.6]). A formal definition is given below.

Definition 1.2. *Let $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, and $i \in [m]$. The inequality $A^i x \geq b_i$ is* always active *within the system $Ax \geq b$ if $Ax \geq b$ implies $A^i x = b_i$.*

All always-active inequalities in a system of linear inequalities (and possibly equalities) can be characterized using relative interior. We present this result for systems in the form $Ax = b,\ x \geq 0$ and $Ax \geq b$ in Theorem 1.6 and Theorem 1.7, respectively.

Theorem 1.6 (cf. [68, Theorem 8])**.** *Let $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $j \in [n]$, and*[7]

$$x^* \in \operatorname{ri}\{\, x \in \mathbb{R}^n \mid Ax = b,\ x \geq 0 \,\}. \tag{1.11}$$

System $Ax = b,\ x \geq 0$ implies $x_j = 0$ if and only if $j \in \sigma(x^*)$, i.e., $x^*_j = 0$.

Proof. Consider the primal-dual pair (1.1) with $c = 0$, i.e., zero objective function in the primal. Trivially, feasibility is equivalent to optimality for the primal, so the primal (and hence also the dual) is feasible and bounded.

Let y^* be from the relative interior of optimizers of the dual (1.1). Any x feasible for the primal necessarily satisfies complementary slackness with dual-optimal y^*, i.e., $\tau(y^*) \cup \sigma(x) = [n]$. Therefore, the system implies $x_j = 0$ for all $j \in [n] - \tau(y^*) = \sigma(x^*)$ where the set equality $[n] - \tau(y^*) = \sigma(x^*)$ is given by strict complementary slackness. Since x^* satisfies $Ax^* = b,\ x^* \geq 0$ and $x^*_j > 0$ for all $j \in [n] - \sigma(x^*)$, the inequality $x_j \geq 0$ for $j \in [n] - \sigma(x^*)$ is not always active. □

Theorem 1.7 ([68, Theorem 8])**.** *Let $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $x^* \in \operatorname{ri}\{x \in \mathbb{R}^n \mid Ax \geq b\}$, and $i \in [m]$. System $Ax \geq b$ implies $A^i x = b_i$ if and only if $A^i x^* = b_i$.*

Proof. Analogous to Theorem 1.6. □

1.2 Block-Coordinate Descent and Relative-Interior Rule

In this section, we will formally define block-coordinate descent (BCD) [8], state some of its properties, and pay particular attention to the *relative-interior rule* that was proposed in [150] and later developed in [151a]. Some parts of this section are based on the review that was given in [54a].

For the purposes of the sequel, we define new notation. For $B = \{i_1, ..., i_{|B|}\} \subseteq [m]$ and $y \in \mathbb{R}^m$, $y|_B$ stands for the restriction of y onto the set B, i.e., $y|_B = (y_{i_1}, ..., y_{i_{|B|}})$ where the order of the components is defined by the total order on B inherited from the natural total order on $[m]$ given by $\leq$.

[7] By assuming the existence of such x^*, we implicitly assume that $Ax = b,\ x \geq 0$ is feasible.

[8] Also called *block-coordinate minimization*. In case that the objective is to be maximized, one usually speaks about *block-coordinate ascent*.

Suppose we minimize a continuous convex function $f\colon Y \to \mathbb{R}$ on a non-empty closed convex set $Y \subseteq \mathbb{R}^m$. Here, we assume for simplicity that the set $\operatorname{argmin}_{y\in Y} f(y)$ is non-empty, hence f is bounded from below on Y. For brevity of notation, we formulate this optimization problem as the unconstrained minimization of the extended-valued function $\bar{f}\colon \mathbb{R}^m \to \mathbb{R}_{+\infty}$ (where $\mathbb{R}_{+\infty} = \mathbb{R} \cup \{+\infty\}$) defined by

$$\bar{f}(y) = \begin{cases} f(y) & \text{if } y \in Y \\ +\infty & \text{if } y \notin Y \end{cases}. \tag{1.12}$$

Furthermore, it is assumed that a set $\mathcal{B} \subseteq 2^{[m]}$ of blocks of variables and an initial feasible solution $y^1 \in Y$ are provided. BCD in each iteration chooses a single block $B \in \mathcal{B}$ and minimizes the function $\bar{f}$ over variables $y|_B$ while keeping the remaining variables $y|_{[m]-B}$ fixed, i.e., updates y^k to some y^{k+1} satisfying

$$y^{k+1}|_B \in \operatorname*{argmin}_{y'\in\mathbb{R}^B} \bar{f}\big(y', y^k|_{[m]-B}\big) \tag{1.13a}$$

$$y_i^{k+1} = y_i^k \qquad \forall i \in [m] - B. \tag{1.13b}$$

By repeatedly performing updates (1.13) with different blocks $B \in \mathcal{B}$, the points y^k remain in the feasible set Y and the sequence of objective values $f(y^k)$ is non-increasing.

Remark 1.2. *To avoid any ambiguity, let us comment on a slight abuse of notation in* (1.13a)*. There, we have a vector $y' \in \mathbb{R}^B$ and a vector $y^k|_{[m]-B} \in \mathbb{R}^{[m]-B}$. The components of these vectors can be arranged into a single vector $y^* \in \mathbb{R}^m = \mathbb{R}^{[m]}$ and $\big(y', y^k|_{[m]-B}\big)$ then corresponds to y^*. For example, if $m = 5$ and $B = \{2,3\}$, we have $y' = (y'_2, y'_3)$, $y^k|_{[m]-B} = (y_1^k, y_4^k, y_5^k)$, and $y^* = \big(y', y^k|_{[m]-B}\big) = (y_1^k, y'_2, y'_3, y_4^k, y_5^k)$.*

Example 1.3. *To illustrate the iterations* (1.13)*, let $f(y) = 5y_1 + 2y_2 + y_3$, $\mathcal{B} = \{\{1\}, \{2,3\}\}$, $Y = \{y \in \mathbb{R}^3 \mid y_1 \geq 0, y_2 \geq 2, y_2 + y_3 = 1\}$, and $y^1 = (4, 7, -6) \in Y$. In the first iteration, we update along $B = \{1\}$ which means that we optimize over y_1 while keeping $y|_{\{2,3\}}$ (i.e., y_2 and y_3) constant. This yields the updated point $y^2 = (0, 7, -6)$ (which satisfies* (1.13) *for $k = 1$ and $B = \{1\}$). In the second iteration, we update along $B = \{2,3\}$, i.e., we optimize over $y|_{\{2,3\}}$ and keep y_1 constant which results in the updated point $y^3 = (0, 2, -1)$ (which satisfies* (1.13) *for $k = 2$ and $B = \{2,3\}$).* △

The class of convex optimization problems for which BCD provably converges to global minima (or where the fixed points are global minima) is currently quite narrow, including, e.g., unconstrained continuously differentiable convex functions with unique univariate minima [15, §2.7], unconstrained differentiable pseudoconvex functions [155], or continuously differentiable functions with a single linear equality constraint and box constraints [12]. For other examples, we refer to [137] and references therein. For general convex non-differentiable or constrained functions, the fixed points of this approach may not be global minima.

1.2.1 Relative-Interior Rule

The set $\operatorname{argmin}_{y'\in\mathbb{R}^B} \bar{f}\big(y', y^k|_{[m]-B}\big)$ of block-wise minimizers from (1.13a) is a non-empty convex set that may in general contain more than one element. In practical implementations, one needs to choose a single element from this set.

It was proposed in [150, 151a] to additionally require that $y^{k+1}|_B$ is chosen from the relative interior of this set, i.e.,

$$y^{k+1}|_B \in \operatorname{ri} \operatorname*{argmin}_{y' \in \mathbb{R}^B} \bar{f}\big(y', y^k|_{[m]-B}\big) \tag{1.14a}$$

$$y_i^{k+1} = y_i^k \qquad \forall i \in [m] - B. \tag{1.14b}$$

As discussed previously in §1.1.1, the relative interior of a non-empty convex set is non-empty, so an update satisfying (1.14) is always possible.

We will now summarize the main results of [150, 151a].

Definition 1.3 ([150, 151a])**.** *A point $y \in Y$ is*

- *a* local minimum *(LM) of f on Y w.r.t. $\mathcal{B}$ if*

$$y|_B \in \operatorname*{argmin}_{y' \in \mathbb{R}^B} \bar{f}\big(y', y|_{[m]-B}\big) \tag{1.15}$$

  (i.e., (1.13) *for $y^{k+1} = y^k = y$) holds for all $B \in \mathcal{B}$,*

- *an* interior local minimum *(ILM) of f on Y w.r.t. $\mathcal{B}$ if*

$$y|_B \in \operatorname{ri} \operatorname*{argmin}_{y' \in \mathbb{R}^B} \bar{f}\big(y', y|_{[m]-B}\big) \tag{1.16}$$

  (i.e., (1.14) *for $y^{k+1} = y^k = y$) holds for all $B \in \mathcal{B}$,*

- *a* pre-interior local minimum *(pre-ILM) of f on Y w.r.t. $\mathcal{B}$ if*[9] *there is an ILM y' of f on Y w.r.t. $\mathcal{B}$ such that y is in a face of the set Y containing y' in its relative interior.*

For brevity, when f, Y, or $\mathcal{B}$ is known from context, we omit 'of f on Y' or 'w.r.t. $\mathcal{B}$'. In §5.4, we will also address maximization problems where the analogous terms *local maximum*, *interior local maximum*, and *pre-interior local maximum* are used. In case of *coordinate-wise minimization*, i.e., when $\mathcal{B} = \{\{i\} \mid i \in [m]\}$, we say that we optimize $\min_{y \in Y} f(y)$ *coordinate-wise* and simply write, e.g., 'ILM w.r.t. individual variables' instead of 'ILM w.r.t. $\mathcal{B} = \{\{i\} \mid i \in [m]\}$'. By a minimum/maximum without adjectives, we always mean global minimum/maximum.

Clearly, the fixed points of BCD algorithm following the updates (1.13) are local minima. We emphasise that this is different from the usual notion of a local minimum: here (by Definition 1.3), the objective in a local minimum cannot be improved by any single update (1.13) instead of an arbitrary update within some neighborhood. The fixed points of BCD with the relative-interior rule (1.14) are interior local minima.

Note, every ILM is a pre-ILM and every pre-ILM is an LM [150, 151a]. Crucial properties of BCD with and without the relative-interior rule are given below.

Theorem 1.8 ([150, 151a])**.** *Let $(B_k)_{k=1}^\infty$ be a sequence of blocks $B_k \in \mathcal{B}$ that contains each element of $\mathcal{B}$ an infinite number of times. Let $(y^k)_{k=1}^\infty$ be a sequence produced by the BCD method, where the blocks are visited in the order given by $(B_k)_{k=1}^\infty$.*
(a) If $(y^k)_{k=1}^\infty$ satisfies (1.14) *and y^1 is an ILM, then y^k is an ILM for all k.*

[9] As in [150, 151a], this definition of a pre-ILM applies only when f is linear. Nevertheless, as we do not use the definition of a pre-ILM explicitly, we omit a general definition of a pre-ILM. Instead, we will rely on the characterization of pre-ILMs that is given by Theorem 1.8 stated later.

(b) If $(y^k)_{k=1}^\infty$ satisfies (1.14) and y^1 is a pre-ILM, then y^k is an ILM for some k.

(c) If $(y^k)_{k=1}^\infty$ satisfies (1.13) and y^1 is a pre-ILM, then $f(y^k) = f(y^1)$ for all k.

(d) If $(y^k)_{k=1}^\infty$ satisfies (1.14) and y^1 is not a pre-ILM, then $f(y^k) < f(y^1)$ for some k.

By Theorem 1.8c, if y^1 is a pre-ILM, the objective cannot be improved by any further BCD iterations (1.13), even with the relative-interior rule (1.14). On the other hand, if y^1 is not a pre-ILM, BCD with the relative-interior rule (1.14) inevitably improves the objective after a finite number of iterations by Theorem 1.8d. In this sense, the relative-interior rule is not worse than any other update rule for choosing non-unique block-minimizers.

Even with the relative-interior rule, the fixed points of BCD need not be global minima, as the following example shows.

Example 1.4 (cf. [148, Example 2, 119, Figure 7.3, 63, Figure 4])**.** *Let $Y = \mathbb{R}^2$ and $f\colon Y \to \mathbb{R}_+$ be defined by $f(y_1, y_2) = \max\{2y_2 - y_1, 2y_1 - y_2, 0\}$. Even though f is convex, any point $y \in \mathbb{R}^2$ satisfying $y_1 = y_2 > 0$ is an ILM w.r.t. individual variables (i.e., w.r.t. $\mathcal{B} = \{\{1\}, \{2\}\}$) but not a global minimum. Figure 1.2a depicts several contours of f and also a non-optimal ILM. Indeed, the highlighted point is an ILM w.r.t. individual variables because any movement from this point along any single coordinate increases the objective.* △

Example 1.5 (cf. [150, §2, 151a, §2])**.** *Let $Y = \{y \in [0,2]^2 \mid y_1 + y_2 \geq 1\}$ and $f(y_1, y_2) = y_2$. Suppose that we initialize BCD with point y^1, as indicated in Figure 1.2b, and optimize it coordinate-wise, i.e., we apply BCD with blocks $\mathcal{B} = \{\{1\}, \{2\}\}$.*

First, we update y^1 to y^2 by decreasing coordinate y_2 to improve the objective and attain the unique coordinate-wise optimum. Point y^2 is a local minimum because both components are coordinate-wise optimal. However, the choice for the optimizer w.r.t. coordinate y_1 is not unique and y_1^2 is not in the relative interior of optimizers, so we change the y_1 coordinate of y^2 and move from y^2 to y^3. Now, coordinate y_2 can be again decreased to improve the objective. After two similar iterations, we attain the point y^6 which is an ILM w.r.t. individual variables and also a global minimum.

Denoting by $[y, y']$ the line segment between y and y' (as in Example 1.1), the set of all LMs of f on Y w.r.t. individual variables is $[(0,1),(1,0)] \cup [(1,0),(2,0)]$. Pre-ILMs are $[(1,0),(2,0)]$ and ILMs are $\operatorname{ri}[(1,0),(2,0)]$. Global minima coincide with pre-ILMs. △

Although the following corollary was not explicitly stated in [150, 151a], it is an immediate consequence of Theorem 1.8.

Corollary 1.3. *The following are equivalent:*

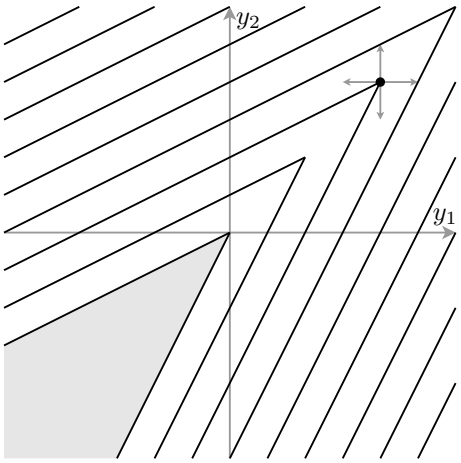
(a) every ILM of f on Y w.r.t. $\mathcal{B}$ is a global minimum,

(b) every pre-ILM of f on Y w.r.t. $\mathcal{B}$ is a global minimum.

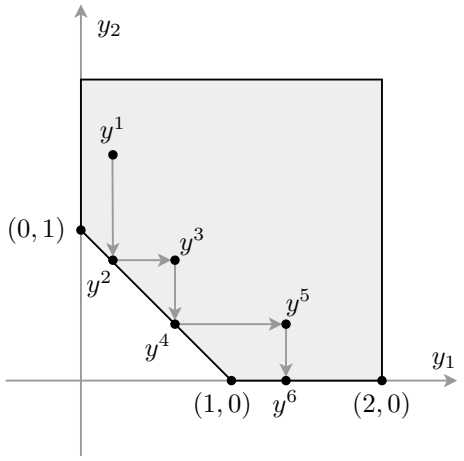
Moreover, if these statements hold, then the set of global minima (i.e., $\operatorname{argmin}_{y \in Y} f(y)$) coincides with the set of pre-ILMs of f on Y w.r.t. $\mathcal{B}$.

Proof. The implication (b) $\implies$ (a) is clear because every ILM is also a pre-ILM. For the other direction, let y^1 be a pre-ILM. By Theorem 1.8b and 1.8c, after performing a finite number of relative-interior updates (1.14) from y, we attain an ILM with the same objective.

One inclusion in the last statement follows already from (b). We prove the remaining part by contradiction: if a global minimum is not a pre-ILM, then, by Theorem 1.8d, the objective must improve after a finite number of updates, which is impossible. □



(a) Contours of function f from Example 1.4. The set of optimizers is shaded in gray and a non-optimal ILM is highlighted.



(b) Five iterations of coordinate-wise minimization with the relative-interior rule starting from the initial point y^1.

Figure 1.2: Illustrations to Examples 1.4 and 1.5.

As a consequence of Corollary 1.3, we will be able to say that '(pre-)ILMs of f on Y w.r.t. $\mathcal{B}$ are global minima'.

Remark 1.3. *The version of BCD considered in [150, 151a] is more general. In detail, one has a set $\mathcal{I}$ of subspaces of $\mathbb{R}^m$ and instead of* (1.13)*, the updates are formulated as*

$$y^{k+1} \in \operatorname{argmin}\{f(y') \mid y' \in (y^k + I) \cap Y\} \tag{1.17}$$

where $I \in \mathcal{I}$ is a chosen subspace along which we update and $y^k + I = \{y^k + y' \mid y' \in I\}$. This subsumes not only the previously described block-coordinate formulation but also optimization along a set of directions (i.e., when all subspaces from $\mathcal{I}$ have dimension 1*). For our purposes, we do not need such a general formalism.*

1.2.2 Convergence

The convergence properties of BCD with the relative-interior rule were discussed and analyzed in [150, §4.3, 151a, §5.2]. Here, we overview the parts that are most important for the sequel.

Formally, suppose that the block-coordinate updates (1.14) are determined by some mappings $u_B \colon Y \to Y$ for each $B \in \mathcal{B}$, i.e., for any $B \in \mathcal{B}$ and $y \in Y$,

$$u_B(y)|_B \in \operatorname{ri} \operatorname*{argmin}_{y' \in \mathbb{R}^B} \bar{f}\big(y', y|_{[m]-B}\big) \tag{1.18a}$$

$$u_B(y)_i = y_i \qquad \forall i \in [m] - B. \tag{1.18b}$$

In other words, the update of $y^k \in Y$ along a block of variables $B \in \mathcal{B}$ satisfying the relative-interior rule yields a point $y^{k+1} = u_B(y^k)$ that satisfies (1.14).

Furthermore, let us fix an ordering on the elements of $\mathcal{B}$ so that $\mathcal{B} = \{B_1, \ldots, B_n\}$ and define the composed mapping $U\colon Y \to Y$ by

$$U = \underbrace{\overbrace{(u_{B_n} \circ \ldots \circ u_{B_1})}^{\text{a single cycle of updates (1.14) in the specified order}} \circ \ldots \circ (u_{B_n} \circ \ldots \circ u_{B_1})}_{m+1 \text{ cycles of updates}}. \tag{1.19}$$

The mapping U performs $m+1$ cycles of updates along individual blocks from $\mathcal{B}$ in the specified order. For this setting, we have the following results from [150, 151a]:

Theorem 1.9 ([150, Theorem 18, 151a, Theorem 20])**.** *Let $y \in Y$ and $\mathcal{B} \subseteq 2^{[m]}$. If $f(U(y)) = f(y)$, then y is a pre-ILM of f on Y w.r.t. $\mathcal{B}$ and $U(y)$ is an ILM of f on Y w.r.t. $\mathcal{B}$.*

Theorem 1.10 ([150, Corollary 25, 151a, Corollary 22])**.** *Let the mappings u_B, $B \in \mathcal{B}$ be continuous, Y^* be the set of pre-ILMs of f on Y w.r.t. $\mathcal{B}$, $y^1 \in Y$, and the sequence $(y^k)_{k=1}^{\infty}$ be defined by $y^{k+1} = U(y^k)$, $k \in \mathbb{N}$. If the sequence $(y^k)_{k=1}^{\infty}$ is bounded, then*

$$\lim_{k\to\infty} d(Y^*, y^k) = 0 \tag{1.20}$$

where $d(Y^*, y^k) = \inf_{y^* \in Y^*} d(y^*, y^k)$ is the distance of y^k to the set Y^* w.r.t. any metric $d\colon Y \times Y \to \mathbb{R}_+$. I.e., the sequence $(y^k)_{k=1}^{\infty}$ converges to the set of pre-ILMs.

1.2.3 Reformulations of Problems

It is known that optimization problems come in different forms that are equivalent in the sense that a solution of one formulation can be easily constructed from the solution of a different formulation [24, §4.1.3] (precisely, they can be reduced to each other in linear time). Such changes may include, e.g., a reformulation of constraints or a change of variables. One such transformation was already shown in §1.1.2.

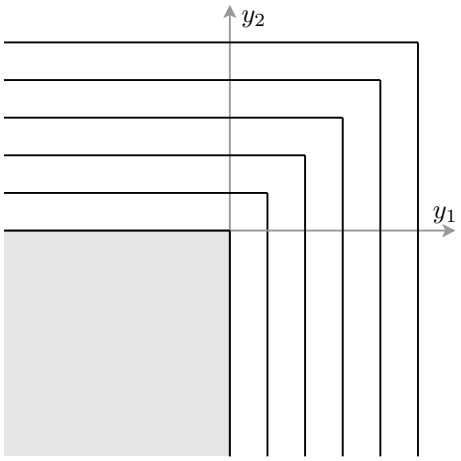
Although this was not explicitly mentioned in the literature before, it is easy to see that the quality of fixed points of BCD highly depends not only on the choice of blocks of variables, but also on the precise formulation of the optimization problem [51a, 53a]. This phenomenon is illustrated by the following examples.

Example 1.6 ([53a, Example 2])**.** *The linear program $\min\{y_1 + y_2 \mid y_1, y_2 \geq 0\}$ has one ILM w.r.t. individual variables which coincides with the unique global minimum, namely $(y_1, y_2) = (0, 0)$. If the redundant constraint $y_1 = y_2$ is added to the linear program, then any feasible point becomes an ILM w.r.t. individual variables because the redundant constraint blocks changing the variable y_1 without changing y_2 and vice versa.* △

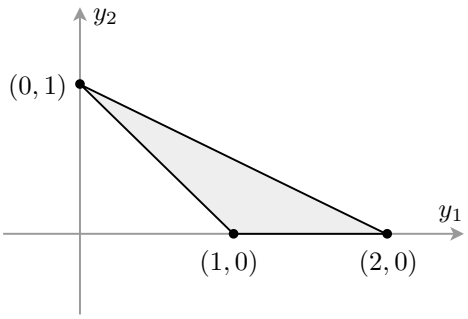
Example 1.7. *Let f and Y be as in Example 1.4 and $g\colon \mathbb{R}^2 \to \mathbb{R}^2$ be the bijection defined by $g(y_1, y_2) = (y_1 + 2y_2, 2y_1 + y_2)$. After transforming the variables by g, we obtain $f(g(y_1, y_2)) = \max\{3y_1, 3y_2, 0\}$. Several contours of function $f \circ g$ are shown in Figure 1.3a.*

Any point $y \in Y = \mathbb{R}^2$ with $y_1 = y_2 > 0$ is a local minimum of $f \circ g$ on Y w.r.t. individual variables, but no longer an ILM because the components of such points are not in the relative interior of coordinate-wise optimizers. In fact, any ILM of $f \circ g$ on Y w.r.t. individual variables is a global minimum.[10] △

[10] Optimizing $f \circ g$ coordinate-wise is in correspondence with optimizing f along subspaces $\operatorname{span}\{(1, 2)\}$ and $\operatorname{span}\{(2, 1)\}$ (recall Remark 1.3) which follows from the definition of g.



(a) Contours of function $f \circ g$ from Example 1.7. The set of optimizers is shaded.



(b) Feasible region Y' of the problem defined in Example 1.8.

Figure 1.3: Illustrations to Examples 1.7 and 1.8.

Example 1.8 (cf. [150, §2, 151a, §2])**.** *Let f and Y be as in Example 1.5. Adding the redundant constraint $y_1 + 2y_2 \leq 2$ reduces the feasible set to $Y' = \{y \in Y \mid y_1 + 2y_2 \leq 2\}$, which is depicted in Figure 1.3b. This change preserves the set of global minima, but the set of pre-ILMs and ILMs is extended by a non-optimal point $(0, 1)$ where no coordinate-wise moves within the feasible set Y' are possible.* △

Example 1.9 (cf. [53a, Example 3])**.** *Finally, we analyze the transformation from §1.1.2. Suppose that we minimize the unconstrained univariate convex piecewise-affine function $f(x_1) = \max\{x_1, 0\} + \max\{-2x_1, -2\}$. When optimizing in the form* (1.4)*, any ILM w.r.t. individual variables is trivially a global minimum. However, in the LP formulation* (1.5)*, there are ILMs w.r.t. individual variables that are not globally optimal, such as $x_1 = z_1 = z_2 = 0$.* △

Fact 1.1 ([53a, Example 3])**.** *Optimizing* (1.4) *coordinate-wise (i.e., applying BCD along individual variables) is in a precise sense equivalent to BCD applied to* (1.5) *along blocks of variables, each containing a single x_i variable and all the z variables. Formally, any ILM x of* (1.4) *w.r.t. individual variables yields an ILM (x, z) of* (1.5) *w.r.t. the above-defined blocks where z is defined by* (1.6)*. On the other hand, for any ILM (x, z) of* (1.5) *w.r.t. the above-defined blocks, x is an ILM of* (1.4) *w.r.t. individual variables. We will use this property multiple times in the sequel.*

This was discussed for the special case of $|K| = 1$ in [150, §5, 151a, §2]. However, if $|K| = 1$, coordinate-wise minimization of (1.4) *is equivalent to coordinate-wise minimization of* (1.5) *and one need not consider blocks of variables in* (1.5)*.*

1.3 Partially Ordered Sets

Let us review the basic concepts of order theory in this section. We begin by recalling the notion of a lattice, semilattice, and complete lattice. Then, we focus on the connection between complete lattices and closure operators. We conclude by analyzing iterative applications of isotone and intensive mappings. This part is based mainly on [22] and [42].

Let S be a set and $\preceq$ be a *partial order* on S, i.e., $\preceq$ is a binary relation on S that is reflexive, anti-symmetric, and transitive. Firstly, recall the *duality principle* in partially ordered sets [22, §1, 42, §1.20]: for any property that concerns the partially ordered set $(S, \preceq)$, there is a corresponding property that concerns its dual ordered set $(S, \succeq)$ where $\succeq$ is the inverse order (a.k.a. dual order), i.e., $s_1 \succeq s_2 \iff s_2 \preceq s_1$ for all $s_1, s_2 \in S$. The corresponding dual property is obtained by replacing all (both explicit and implicit) occurrences of $\preceq$ in the property by $\succeq$.

Inspired by the notation in [22], for $Q \subseteq S$, we define

$$Q^{\uparrow}_S = \{s \in S \mid \forall q \in Q : q \preceq s\} \tag{1.21a}$$

$$Q^{\downarrow}_S = \{s \in S \mid \forall q \in Q : s \preceq q\} \tag{1.21b}$$

where $Q^{\uparrow}_S$ and $Q^{\downarrow}_S$ denote the set of all *upper bounds* and *lower bounds* on Q in S, respectively.

Furthermore, if $q^* \in Q^{\uparrow}_S$ satisfies $q^* \preceq q$ for all $q \in Q^{\uparrow}_S$, then it is called the *least upper bound* on Q in S and we denote it by $q^* = \bigvee_S Q$. Analogously, if $q^* \in Q^{\downarrow}_S$ satisfies $q^* \succeq q$ for all $q \in Q^{\downarrow}_S$, then it is called the *greatest lower bound* on Q in S and is denoted by $q^* = \bigwedge_S Q$. The operations $\bigvee_S$ and $\bigwedge_S$ are called the *join* and the *meet*, respectively. We emphasise that, in general, a partially ordered set need not have the least upper bound or greatest lower bound for each of its subsets.

For convenience, for two-element sets $Q = \{q_1, q_2\}$, we may write $q_1 \vee_S q_2$ instead of $\bigvee_S \{q_1, q_2\}$ and $q_1 \wedge_S q_2$ instead of $\bigwedge_S \{q_1, q_2\}$.

In particular, we have that $S^{\uparrow}_S$ is either empty or a singleton set because if $s_1, s_2 \in S^{\uparrow}_S$, then, by definition (1.21a), $s_1 \preceq s_2$ and $s_2 \preceq s_1$, hence $s_1 = s_2$ by anti-symmetry. If $S^{\uparrow}_S$ is non-empty, then $S^{\uparrow}_S = \{\top\}$ where $\top = \bigvee_S S \in S$ is the *top element* of S. By the duality principle, $S^{\downarrow}_S$ is also either empty or a singleton set. If $S^{\downarrow}_S = \{\bot\}$, then $\bot = \bigwedge_S S \in S$ is the *bottom element* of S.[11] Again, a partially ordered set need not contain the top or the bottom element in general.

In the sequel, we simplify our notation as follows: if the set S w.r.t. which the upper bounds, lower bounds, joins, and meets are taken is clear from context, then we omit the subscript S, i.e., we simplify $Q^{\uparrow}_S$, $Q^{\downarrow}_S$, $\bigvee_S$, and $\bigwedge_S$ to $Q^{\uparrow}$, $Q^{\downarrow}$, $\bigvee$, and $\bigwedge$.

1.3.1 Lattices

Definition 1.4. *A partially ordered set $(S, \preceq)$ is:*

- *a* meet-semilattice *if for any $s_1, s_2 \in S$, $s_1 \wedge s_2$ exists in S,*
- *a* join-semilattice *if for any $s_1, s_2 \in S$, $s_1 \vee s_2$ exists in S,*
- *a* lattice *if it is a meet-semilattice and a join-semilattice,*

[11] For the top and the bottom element, we adopt notation from [42, §1.21] to avoid using 0 and 1 which we reserve for their numerical meaning.

- *a* complete lattice *if for any* $Q \subseteq S$*, both* $\bigwedge Q$ *and* $\bigvee Q$ *exist in* S*.*

It is easily shown by induction [42, §2.11] that for a meet-semilattice $(S, \preceq)$ and any non-empty finite $Q \subseteq S$, $\bigwedge Q$ exists in S. Consequently, any non-empty finite meet-semilattice $(S, \preceq)$ has a bottom element, namely $\bigwedge S$. By the duality principle, for any join-semilattice $(S, \preceq)$ and any non-empty finite $Q \subseteq S$, $\bigvee Q$ exists in S. In particular, any non-empty finite join-semilattice $(S, \preceq)$ has a top element, $\bigvee S$. A complete lattice always has both the top and the bottom element [22, Theorem 2.10].

Lemma 1.1 ([106, Theorem 2.5, 22, Theorem 2.11, 42, Theorem 2.31])**.** *Let* $(S, \preceq)$ *be a partially ordered set. The following are equivalent:*

(a) $\bigwedge Q$ *exists in* S *for any non-empty* $Q \subseteq S$ *and* $(S, \preceq)$ *has a top element* $\top$*,*

(b) $\bigvee Q$ *exists in* S *for any non-empty* $Q \subseteq S$ *and* $(S, \preceq)$ *has a bottom element* $\bot$*,*

(c) $(S, \preceq)$ *is a complete lattice.*

Proof. The implication (c) $\implies$ (a) is clear, so we proceed to prove (a) $\implies$ (c). Since $(S, \preceq)$ has top element $\top$, we have that $\bigwedge \emptyset = \top$. Consequently, for any $Q \subseteq S$, $\bigwedge Q$ exists in S.

Now, it remains to show that $\bigvee Q$ exists in S for any $Q \subseteq S$. We define $q^* = \bigwedge Q^{\uparrow}$ and claim that $q^* = \bigvee Q$. First, notice that $q_1 \preceq q_2$ for all $q_1 \in Q$ and all $q_2 \in Q^{\uparrow}$ (by definition of an upper bound). Consequently, $q_1 \preceq \bigwedge Q^{\uparrow} = q^*$ holds for all $q_1 \in Q$, so q^* is an upper bound on Q. By definition of $\bigwedge$, q^* is the least upper bound on Q in S.

The equivalence (b) $\iff$ (c) is obtained dually. □

Theorem 1.11 ([106, Theorem 2.4, 42, Corollary 2.25])**.** *Let* $(S, \preceq)$ *be a non-empty finite partially ordered set. The following are equivalent:*

(a) $(S, \preceq)$ *is a meet-semilattice with top element* $\top$*,*

(b) $(S, \preceq)$ *is a join-semilattice with bottom element* $\bot$*,*

(c) $(S, \preceq)$ *is a complete lattice,*

(d) $(S, \preceq)$ *is a lattice.*

Proof. Together with the previously stated facts, Lemma 1.1 yields (a) $\iff$ (b) $\iff$ (c). The implication (c) $\implies$ (d) is clear by definition of a complete lattice. To prove (d) $\implies$ (a), notice that any finite lattice has the top element $\bigvee S$ and is also a meet-semilattice. □

By Theorem 1.11, it is possible to augment any finite meet-semilattice by introducing a new artificial top element to obtain a complete lattice. Dually, one can include a bottom element in a finite join-semilattice which is known as the *lifting* operation [42, §1.22].

1.3.2 (Dual) Closure Operators and Chaotic Iterations

Let us now recall the connection between closure operators and complete lattices. In detail, any closure operator defined on a complete lattice defines its subset which is also a complete lattice and, under additional assumptions, such a subset defines a closure operator.

Before we state these results formally, we overview useful properties of mappings on partially ordered sets in Definition 1.5.

Definition 1.5 ([22, §1.4]). *Let $(S, \preceq)$ be a partially ordered set. Mapping $f\colon S \to S$ is*

- extensive *if* $\forall s \in S\colon\ s \preceq f(s)$,
- intensive *if* $\forall s \in S\colon\ f(s) \preceq s$,
- idempotent *if* $\forall s \in S\colon\ f(s) = f(f(s))$,
- isotone *if* $\forall s_1, s_2 \in S\colon\ s_1 \preceq s_2 \implies f(s_1) \preceq f(s_2)$,
- *a* closure operator *if it is extensive, idempotent, and isotone,*
- *a* dual closure operator *if it is intensive, idempotent, and isotone.*

Let us denote the *image* of a mapping $f\colon S \to S$ by

$$\operatorname{im} f = \{f(s) \mid s \in S\}. \tag{1.22}$$

Notice that it follows directly from Definition 1.5 that the set of fixed points of an idempotent mapping f coincides with its image, i.e., $\{s \in S \mid f(s) = s\} = \operatorname{im} f$. In detail, inclusion in the $\supseteq$ direction follows from idempotence and the other direction is clear from the definition of a fixed point [22, §1.4]. In particular, this holds for any (dual) closure operator f.

For the purpose of the following theorem, we need to present an auxiliary lemma and also introduce additional terminology: for complete lattices $(S, \preceq)$ and $(Q, \preceq)$ with $Q \subseteq S$, we say that their *meet operations coincide* if for any $Q' \subseteq Q$, we have $\bigwedge_S Q' = \bigwedge_Q Q'$. In words, this is the case if for any $Q' \subseteq Q$, the greatest lower bound on Q' in S is the same as the greatest lower bound on Q' in Q. Analogously, their *join operations coincide* if for any $Q' \subseteq Q$, we have $\bigvee_S Q' = \bigvee_Q Q'$.

Lemma 1.2 ([42, Lemma 2.22(v)]). *Let $(S, \preceq)$ be a complete lattice and $S_1 \subseteq S_2 \subseteq S$. Then, $\bigwedge S_1 \succeq \bigwedge S_2$ and $\bigvee S_2 \succeq \bigvee S_1$.*

Proof. The first claim follows from the fact that $\bigwedge S_2 = \bigwedge S_1 \wedge \bigwedge (S_2 - S_1)$ is a lower bound on $\bigwedge S_1$. The second claim follows from the duality principle. □

Theorem 1.12 (cf. [42, Theorem 7.3]). *Let $(S, \preceq)$ and $(Q, \preceq)$ be complete lattices such that $Q \subseteq S$.*

(a) If the meet operations in $(S, \preceq)$ and $(Q, \preceq)$ coincide, then the mapping $f\colon S \to Q$ defined by $f(s) = \bigwedge \{s\}^{\uparrow}_Q$ is a closure operator.

(b) If the join operations in $(S, \preceq)$ and $(Q, \preceq)$ coincide, then the mapping $g\colon S \to Q$ defined by $g(s) = \bigvee \{s\}^{\downarrow}_Q$ is a dual closure operator.

Proof. We show only (a) since (b) then follows from the duality principle. Notice that we do not need to distinguish $\bigwedge_S$ and $\bigwedge_Q$ in the definition of f because $\{s\}^{\uparrow}_Q$ is a subset of Q and the meet operations coincide.

We begin by extensivity: for $s \in S$, $s \in \{s\}^{\uparrow}_S$ holds by reflexivity of $\preceq$ and also $\{s\}^{\uparrow}_Q \subseteq \{s\}^{\uparrow}_S$ due to $Q \subseteq S$. Lemma 1.2 yields $f(s) = \bigwedge \{s\}^{\uparrow}_Q \succeq \bigwedge \{s\}^{\uparrow}_S = s$.

For isotony, let $s_1, s_2 \in S$ such that $s_1 \preceq s_2$. Then, $\{s_1\}^{\uparrow}_Q \supseteq \{s_2\}^{\uparrow}_Q$ by transitivity of $\preceq$ and we obtain $f(s_1) = \bigwedge \{s_1\}^{\uparrow}_Q \preceq \bigwedge \{s_2\}^{\uparrow}_Q = f(s_2)$ by Lemma 1.2.

Finally, to prove idempotency, we have that $f(s) \in \{f(s)\}^{\uparrow}_Q$, which yields $f(f(s)) = \bigwedge \{f(s)\}^{\uparrow}_Q = f(s)$. □

inputs: partially ordered set $(S, \preceq)$, initial element $s \in S$, isotone and intensive mappings $f_1, ..., f_n \colon S \to S$.

1 $s' := s$
2 **while** $\exists i \in [n] : f_i(s') \neq s'$ **do**
3 | Find such i.
4 | $s' := f_i(s')$
5 **return** s'

Algorithm 1.1: Iterations of a given set of mappings applied to an initial element $s \in S$ of a partially ordered set $(S, \preceq)$.

Focusing on the definition of f and g in Theorem 1.12, for any $s \in S$, we have that $f(s)$ is the least upper bound on s in Q and $g(s)$ is the greatest lower bound on s in Q. Theorem 1.12 has the following practical corollary.

Corollary 1.4. *Let $(S, \preceq)$ and $(Q, \preceq)$ be complete lattices with $Q \subseteq S$. Define the mappings $f, g \colon S \to Q$ by $f(s) = \bigwedge_Q \{s\}^{\uparrow}_Q$ and $g(s) = \bigvee_Q \{s\}^{\downarrow}_Q$. For any $s \in S$, it holds that $s \in Q \iff f(s) = s \iff g(s) = s$.*

Proof. If $s \in Q$, $s \in \{s\}^{\uparrow}_Q$ and $s \preceq q$ holds for any $q \in \{s\}^{\uparrow}_Q$ by definition, so $s = \bigwedge_Q \{s\}^{\uparrow}_Q = f(s)$. If $s \notin Q$, $f(s) \neq s$ due to $\operatorname{im} f = Q$. The part with g is obtained dually. □

Remark 1.4. *The converse connection between (dual) closure operators and complete lattices also holds. In detail, for any complete lattice $(S, \preceq)$ and any (dual) closure operator $f \colon S \to S$, $(\operatorname{im} f, \preceq)$ is a complete lattice [22, Theorem 2.14, 42, §7].*

Finally, suppose that we are given a set of isotone and intensive mappings on some finite partially ordered set $(S, \preceq)$ and an element $s \in S$. To find the greatest common fixed point s' of these mappings such that $s' \preceq s$, one can use Algorithm 1.1 whose correctness is given by Theorem 1.13.

Theorem 1.13 (cf. [6, Theorem 1])**.** *Let $(S, \preceq)$ be a finite partially ordered set, $s \in S$, and $f_1, ..., f_n \colon S \to S$ be isotone and intensive mappings. Algorithm 1.1 terminates in a finite number of steps and returns the greatest common fixed point s' of mappings $f_1, ..., f_n$ that satisfies $s' \preceq s$.*

Proof. In each iteration of the algorithm, the current s' strictly decreases w.r.t. $\preceq$ by intensivity of f_i. This implies that $s' \preceq s$ and, by finiteness of S, that the algorithm terminates after a finite number of iterations. The fact that s' is a common fixed point of all the mappings f_i follows directly from the condition on line 2 of the algorithm.

To prove that s' is the greatest common fixed point, we proceed by induction. In detail, we show that $s^* \preceq s'$ holds during the whole run of the algorithm for any common fixed point $s^* \in S$ such that $s^* \preceq s$. The base case is clear: $s^* \preceq s = s'$. For the inductive step, it follows from isotony of f_i that $s^* = f_i(s^*) \preceq f_i(s')$ where the equality is given by s^* being a common fixed point. □

Note that, by Theorem 1.13, the value returned by Algorithm 1.1 is independent of the way of choosing i on line 3. Similar algorithms are known as *chaotic iterations* [5, §2].

1.4 Constraint Satisfaction Problem and Local Consistencies

In this section, we recall the constraint satisfaction problem (CSP), the notion of local consistency, and constraint propagation. Furthermore, we also revisit the connection between local consistencies, closure operators, and iterations of propagators. Throughout the thesis, we follow our notation for CSPs from [55a, 56a] and we reuse some parts of these papers to define the notation and well-known terms in the beginning of this section.

The *structure* of a CSP is defined by a triple (V, D, C) where V is a finite set of *variables*, D is a common finite *domain* of each variable, and $C \subseteq 2^V$ is a non-empty set of non-empty *scopes* of constraints so that (V, C) can be interpreted as a hypergraph. The structure (V, D, C) gives rise to the set of all *tuples*

$$T = \{\, (S, k) \mid S \in C,\ k \in D^S \,\}, \tag{1.23}$$

which can be naturally partitioned into the sets

$$T_S = \{\, (S, k) \mid k \in D^S \,\} \tag{1.24}$$

for each $S \in C$.

In machine learning, hypergraphs are often equivalently represented by so-called factor graphs which is convenient as it allows us to avoid the more complex hypergraph terminology. In analogy to [83], the *factor graph* of a hypergraph (V, C) (or, factor graph of a CSP with a set of variables V and constraint scopes C) is the bipartite graph whose nodes correspond to variables in V and scopes in C. The factor graph contains an edge between a variable node $i \in V$ and a scope node $S \in C$ if $i \in S$.[12] We show an example of a factor graph in Figure 1.4.

Even though most of the results in the thesis do not rely on this, we will for simplicity of presentation generally assume that $\{i\} \in C$ for each $i \in V$ and that the factor graph of (V, C) is connected (unless specified otherwise). We also implicitly assume that there is at most one constraint with each scope because C is a set (instead of a multiset).

An instance of the *constraint satisfaction problem (CSP)* is defined by the quadruple (V, D, C, A) where (V, D, C) is the structure and $A \subseteq T$ is the set of *allowed tuples* (while the tuples $T - A$ are *forbidden*). In the sequel, we will not need to change the structure (V, D, C) of the CSP, so we will refer to CSP instances only as A for brevity. In other words, we identify CSP instances with subsets of T.

In analogy to §1.2, for an *assignment*[13] $x \in D^V$ and $S \subseteq V$, $x|_S$ stands for the restriction of x onto the set S, i.e., for $S = \{i_1, ..., i_{|S|}\}$ we have $x|_S = (x_{i_1}, ..., x_{i_{|S|}}) \in D^S$ (where the order of the components is defined by the total order on S inherited from some arbitrary fixed total order on V). An assignment $x \in D^V$ *uses* a tuple $(S, k) \in T$ if $x|_S = k$. An assignment $x \in D^V$ is a *solution* to CSP A if it uses only allowed tuples,

[12] The factor graph is isomorphic to the primal constraint graph [46, §2.1.3] of the hidden transformation [8, Definition 7] of the CSP. Also recall that a sequence $(i_1, S_1, i_2, S_2, \ldots, S_n, i_{n+1})$ with $n \geq 2$ and $i_1 = i_{n+1}$ is a *Berge cycle* [14, §17] of a hypergraph (V, C) if $i_1, \ldots, i_n$ are distinct vertices from V, $S_1, \ldots, S_n$ are distinct scopes from C, and $i_k, i_{k+1} \in S_k$ holds for all $k \in [n]$. A hypergraph is *Berge-acyclic* if it does not contain a Berge cycle. It is easy to see that the factor graph of (V, C) is acyclic if and only if the hypergraph (V, C) is Berge-acyclic.

[13] As usual, D^V denotes the set of all mappings from V to D, so $x \in D^V$ is the same as $x \colon V \to D$.

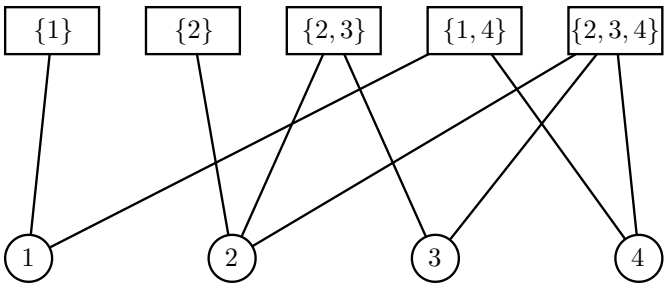


Figure 1.4: Factor graph of the hypergraph (V, C) where $V = \{1, 2, 3, 4\}$ and $C = \{\{1\}, \{2\}, \{2, 3\}, \{1, 4\}, \{2, 3, 4\}\}$. The factor graph contains 5 nodes corresponding to elements of C (shown as rectangles), 4 nodes corresponding to elements of V (shown as circles), and 9 edges in total.

i.e., $\big(S, x|_S\big) \in A$ for all $S \in C$. Note that each assignment $x \in D^V$ uses exactly one tuple from each T_S, $S \in C$.

As usual [5, 17, 81, 46, 67], the solution set of a CSP A is denoted by $\mathrm{SOL}(A) \subseteq D^V$. We say that CSPs A, $A' \subseteq T$ are *equivalent* if $\mathrm{SOL}(A) = \mathrm{SOL}(A')$ [5, 6, 105]. CSP A is *satisfiable* if $\mathrm{SOL}(A) \neq \emptyset$ and *unsatisfiable* otherwise. The problem of deciding whether a CSP is satisfiable is NP-complete (e.g., by reduction from 3-coloring [116, §8.6.1, 84]).

Note that one can interpret SOL as a mapping $\mathrm{SOL}\colon 2^T \to 2^{(D^V)}$. Since allowing more tuples does not make the solution set smaller, the mapping SOL is isotone, i.e., $A \subseteq A'$ implies $\mathrm{SOL}(A) \subseteq \mathrm{SOL}(A')$.

To avoid any ambiguity, we define a CSP to be *Boolean* if $|D| = 2$. On the other hand, a CSP is *pairwise* if $|S| \leq 2$ for all $S \in C$.[14] Finally, we define $C_{\geq 2} = \{S \in C \mid |S| \geq 2\}$ to be the set of all non-unary scopes.

Example 1.10 (cf. [56a, Example 2])**.** *Let* $V = \{1, 2\}$*,* $D = \{\mathsf{a}, \mathsf{b}\}$*, and* $C = \{\{1\}, \{2\}, \{1, 2\}\}$*. For this structure, the set of tuples is*

$$\begin{aligned} T = \{&(\{1\}, \mathsf{a}),\ (\{1\}, \mathsf{b}),\ (\{2\}, \mathsf{a}),\ (\{2\}, \mathsf{b}),\\ &(\{1, 2\}, (\mathsf{a}, \mathsf{a})),\ (\{1, 2\}, (\mathsf{a}, \mathsf{b})),\ (\{1, 2\}, (\mathsf{b}, \mathsf{a})),\ (\{1, 2\}, (\mathsf{b}, \mathsf{b}))\}, \end{aligned} \tag{1.25}$$

i.e., $T = T_{\{1\}} \cup T_{\{2\}} \cup T_{\{1,2\}}$ *where, e.g.,* $T_{\{2\}} = \{(\{2\}, \mathsf{a}), (\{2\}, \mathsf{b})\}$*.*

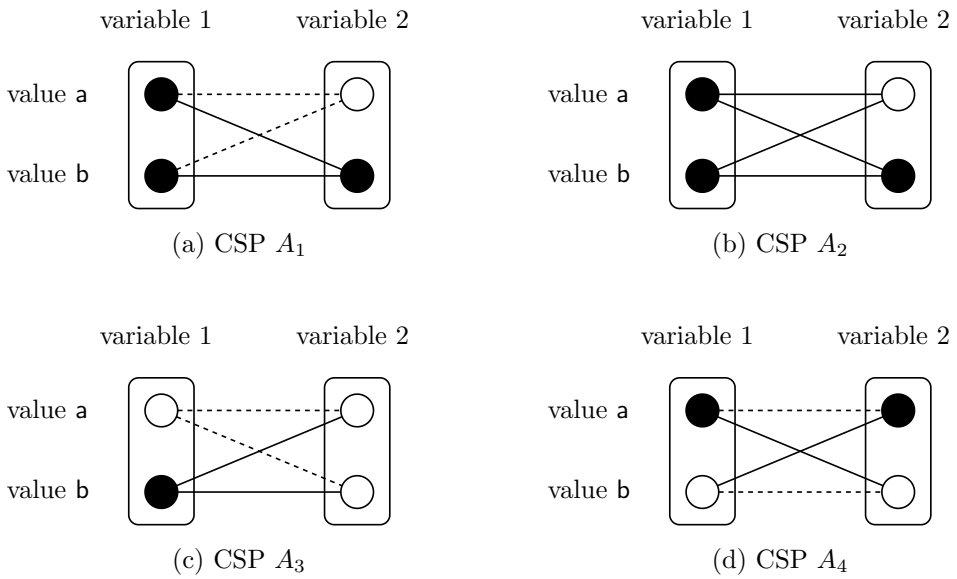
The assignment $x = (\mathsf{a}, \mathsf{b}) \in D^V$ *(i.e.,* $x_1 = \mathsf{a}$*,* $x_2 = \mathsf{b}$*) uses tuples* $(\{1\}, \mathsf{a})$*,* $(\{2\}, \mathsf{b})$*, and* $(\{1, 2\}, (\mathsf{a}, \mathsf{b}))$*. CSP* A_1 *from Figure 1.5a is defined by*

$$A_1 = \{\, (\{1\}, \mathsf{a}),\ (\{1\}, \mathsf{b}),\ (\{2\}, \mathsf{b}),\ (\{1, 2\}, (\mathsf{a}, \mathsf{b})),\ (\{1, 2\}, (\mathsf{b}, \mathsf{b}))\,\} \subseteq T \tag{1.26}$$

and is equivalent to A_2 *in Figure 1.5b since* $\mathrm{SOL}(A_1) = \mathrm{SOL}(A_2) = \{(\mathsf{a}, \mathsf{b}), (\mathsf{b}, \mathsf{b})\}$*.*

CSPs A_3 *and* A_4 *in Figures 1.5c and 1.5d are both unsatisfiable (and therefore equivalent). Since the structure of all these CSPs is the same, they are all Boolean and pairwise.* △

[14] We intentionally avoid the term 'binary' that sometimes refers to 'Boolean' whereas in other works it means 'pairwise'.



(a) CSP A_1 (b) CSP A_2

(c) CSP A_3 (d) CSP A_4

Figure 1.5: Visualisations of four CSPs with the same structure. Variables (elements of V) are depicted as rounded rectangles, tuples (elements of T) as circles and line segments. Black circles and full lines indicate allowed tuples, whereas white nodes and dashed lines indicate forbidden tuples. Our visualisation convention for (W)CSPs is similar to the ones considered in [149, 146, 134, 119, 92, 46] [15]– for CSPs, this in fact depicts its micro-structure [82].

1.4.1 Local Consistencies and Constraint Propagation

Let us now focus on the connection of local consistencies, consistency closures, and propagators. Local consistencies are a key concept in the constraint programming community that is typically used to prune the search space when searching for a solution of a CSP. The topic of local consistencies is broad and well-studied [17, 46, §3, 99, §3-§7].

A *local consistency* is a (usually simple) condition that is necessary for allowed tuples of a CSP to be used by (some of) its solutions. Enforcing a local consistency means finding an equivalent CSP that satisfies the given condition. For this, *constraint propagation* is used. In general, constraint propagation explicitly infers some knowledge that was only implicit before. This could mean, e.g., forbidding tuples that are not used by any solution or introducing new constraints that are satisfied by all solutions. In the sequel, we focus only on the former case, i.e., reducing the constraints while maintaining equivalence which is known as constraint reduction process [5, §3.2]. In other words, we develop formalism only for a subset of local consistencies that preserve the structure of a CSP and are applied to constraints that are given in extension (i.e., by enumerating the allowed tuples).

In this part, we first formally define the properties of local consistencies sufficient for defining the associated (dual) closure operators. Then, we point our attention to how this closure can be obtained using a set of propagators and exemplify the shown notions on arc consistency. Finally, we discuss that some classes of CSPs can be solved by enforcing local consistencies without any search. We base our description mainly on [5] and [17].

[15] On the other hand, our convention is in contrast to some other papers, such as [136, 107, 98, 43, 33, 70] where forbidden non-unary tuples are emphasised whereas we emphasise the allowed tuples.

Remark 1.5. *In some formalisms, e.g., [17, 5, 8, 99], the domains of the individual variables are explicitly reduced during constraint propagation, i.e., the set D is different for each variable and is gradually made smaller. However, we do not reduce the domains of individual variables explicitly in our notation and set D is kept unchanged. Instead, we expect that there is a unary constraint on each variable and removing value $k \in D$ from the domain of variable $i \in V$ is performed by forbidding the tuple $(\{i\}, k)$. Both approaches are semantically equivalent [17, §3.1].*

To be more formal, consider a local consistency Φ and the following properties.

Property 1.1 ([17])**.** *If CSPs A and A' are Φ-consistent, $A \cup A'$ is also Φ-consistent.*

Property 1.2 ([17])**.** *CSP $\emptyset$ is Φ-consistent.*

Property 1.3. *If a CSP A satisfies that for all $(S, k) \in A$ there exists $x \in \mathrm{SOL}(A)$ such that $x|_S = k$, then A is Φ-consistent.*

Property 1.1 is called *stability under union* (cf. [17, Definition 3.17]) and is satisfied by typical local consistencies [16]. Property 1.2 can be assumed by convention, as stated in [17, Theorem 3.19]. Property 1.3 is a formalization of the statement that Φ is a necessary condition for the allowed tuples to be extendable to a solution, i.e., we should not forbid tuples if such an action changes the solution set. Note, Property 1.3 implies Property 1.2. [17] For now, we analyze the implications of the first two properties and we will focus on the importance of Property 1.3 later.

For a local consistency Φ satisfying Properties 1.1 and 1.2, the set of all Φ-consistent CSPs (with the fixed structure (V, D, C)) equipped by the partial order given by the set inclusion is by Theorem 1.11 a complete lattice since it contains the bottom element $\emptyset$ and its join operation is the set union. By Theorem 1.12b [18], this gives rise to a dual closure operator $\mathscr{C}_\Phi \colon 2^T \to 2^T$ defined by

$$\mathscr{C}_\Phi(A) = \bigcup \{A' \subseteq A \mid A' \text{ is } \Phi\text{-consistent}\} \tag{1.27}$$

which is the greatest Φ-consistent CSP that is a subset of A. In addition, Corollary 1.4 yields that CSP A is Φ-consistent if and only if $\mathscr{C}_\Phi(A) = A$.

Remark 1.6. *Although* (1.27) *is sometimes only referred to as a closure (e.g., as in arc consistency closure [8, 33, 17]), it is indeed a* dual *closure in the sense of §1.3.2 because it is intensive, i.e., it reduces the set of allowed tuples. This distinction is only technical as it can be easily corrected by either considering the dual setting (as in [5]) where the ordering is formally reversed, or by defining the CSP by the set of forbidden (rather than allowed) tuples. To be consistent with the literature, we will sometimes omit 'dual' and say, e.g., 'arc consistency closure' instead of 'arc consistency dual closure'.*

[16] For an example of a local consistency that is not stable under union, see [17, Example 3.18].

[17] On the other hand, Property 1.3 is not implied by Properties 1.1 and 1.2. As an example, if $\emptyset$ is the only Φ-consistent CSP, then this notion of Φ-consistency satisfies Properties 1.1 and 1.2 but does not satisfy Property 1.3.

[18] In more detail, the set of Φ-consistent CSPs is a subset of 2^T which is also partially ordered by the set inclusion and $(2^T, \subseteq)$ is a complete lattice. Both of these complete lattices have the same join operation, namely the set union $\cup$.

We emphasise that, for the partially ordered set of Φ-consistent CSPs to be a complete lattice, we required only Properties 1.1 and 1.2, not Property 1.3. The same holds for the existence of the dual closure operator $\mathscr{C}_\Phi$. However, we will show that if a local consistency Φ does not satisfy Property 1.3, then the corresponding dual closure operator $\mathscr{C}_\Phi$ may not return an equivalent CSP.

To concisely describe this result, let us recall positive consistency [7, §III, 20, Definition 3] (related terms are *minimal network* and *minimal CSP* from [105, §3, 46, §2.3.2, 101, §7.4, 57, Definition 2, 67, §1.1]).

Definition 1.6 ([7, 20])**.** *A CSP A is* positively consistent *if for all $(S, k) \in A$, there is $x \in \mathrm{SOL}(A)$ such that $x|_S = k$.*

It is easy to see that forbidding any tuple in a positively consistent CSP changes its solution set. This new notion allows us to restate Property 1.3 equivalently as follows: *any positively consistent CSP is also Φ-consistent*. Clearly, positive consistency is stable under union (i.e., satisfies Property 1.1) and CSP $\emptyset$ is positively consistent, hence the corresponding dual closure operator (1.27) is defined and will be denoted by $\mathscr{C}_{\mathrm{pos}}$. The following proposition states an expected property of the mapping $\mathscr{C}_{\mathrm{pos}}$.

Proposition 1.1. *Let $A \subseteq T$. $\mathrm{SOL}(A) = \mathrm{SOL}(\mathscr{C}_{\mathrm{pos}}(A))$.*

Proof. Let $A' = A - \{(S, k) \in T \mid \forall x \in \mathrm{SOL}(A) \colon x|_S \neq k\}$. CSP A' is clearly positively consistent and $A' \subseteq A$, so $A' \subseteq \mathscr{C}_{\mathrm{pos}}(A) \subseteq A$. Moreover, $\mathrm{SOL}(A') = \mathrm{SOL}(A)$ holds by definition of A'. Applying isotony of SOL results in $\mathrm{SOL}(A') \subseteq \mathrm{SOL}(\mathscr{C}_{\mathrm{pos}}(A)) \subseteq \mathrm{SOL}(A)$, hence $\mathrm{SOL}(\mathscr{C}_{\mathrm{pos}}(A)) = \mathrm{SOL}(A)$. □

Using Proposition 1.1, we are now in position to formalize the importance of Property 1.3 in Theorem 1.14.

Theorem 1.14. *Let Φ be a local consistency satisfying Properties 1.1 and 1.2. The following are equivalent:*

(a) Φ-consistency satisfies Property 1.3,

(b) $\forall A \subseteq T : \mathrm{SOL}(\mathscr{C}_\Phi(A)) = \mathrm{SOL}(A)$.

Proof. (a) $\Longrightarrow$ (b): Since $\mathscr{C}_{\mathrm{pos}}(A)$ is Φ-consistent and $\mathscr{C}_{\mathrm{pos}}(A) \subseteq A$, it follows that $\mathscr{C}_{\mathrm{pos}}(A) \subseteq \mathscr{C}_\Phi(A)$ by definition of $\mathscr{C}_\Phi$ in (1.27). Proposition 1.1 yields $\mathrm{SOL}(\mathscr{C}_{\mathrm{pos}}(A)) = \mathrm{SOL}(A)$. Combining this with $\mathscr{C}_{\mathrm{pos}}(A) \subseteq \mathscr{C}_\Phi(A) \subseteq A$ (where we used intensivity of $\mathscr{C}_\Phi$) and isotony of SOL, we obtain $\mathrm{SOL}(\mathscr{C}_\Phi(A)) = \mathrm{SOL}(A)$ analogously to the proof of Proposition 1.1.

(b) $\Longrightarrow$ (a): By contrapositive: let Property 1.3 not be satisfied and let A be the positively consistent CSP that is not Φ-consistent. Since A is not Φ-consistent, we have $\mathscr{C}_\Phi(A) \neq A$ and thus $\mathscr{C}_\Phi(A) \subsetneq A$ by intensivity of $\mathscr{C}_\Phi$. This implies $\mathrm{SOL}(\mathscr{C}_\Phi(A)) \subsetneq \mathrm{SOL}(A)$ because forbidding any tuple in a positively consistent CSP changes its solution set. □

To summarize, if a local consistency Φ satisfies Properties 1.1 and 1.3 (and thus also satisfies Property 1.2), then $\mathscr{C}_\Phi$ is a dual closure operator and for any CSP A, $\mathscr{C}_\Phi(A)$ and A are equivalent, i.e., $\mathrm{SOL}(A) = \mathrm{SOL}(\mathscr{C}_\Phi(A))$. In the following parts, we will assume that Φ satisfies these properties.

inputs: CSP A, local consistency Φ.
1 $A' := A$
2 **while** A' is not Φ-consistent **do**
3 Find $R \subseteq A'$, such that $R \neq \emptyset$ and $\mathrm{SOL}(A' - R) = \mathrm{SOL}(A')$.
4 $A' := A' - R$
5 **return** A'

Algorithm 1.2: Propagation algorithm enforcing Φ-consistency of CSP A.

1.4.1.1 Propagation Algorithm and Propagators

In practice, the dual closure (1.27) is not computed in a single step, but instead by iteratively applying multiple propagators [17, §3.7]. Typically, there is a *propagation algorithm* that gradually forbids some tuples that are identified to be inconsistent (which implies that they are not used by any solution). After these tuples are forbidden, the algorithm may detect that other tuples became inconsistent and thus, forbids them too. The algorithm eventually stops when it is unable to forbid any other tuples, i.e., the instance satisfies the local consistency condition.

This is the principle of constraint propagation applied to a CSP that we (for now slightly informally) outline in Algorithm 1.2. The input of the propagation algorithm is a CSP A and a local consistency Φ. First, the algorithm initializes $A' := A$ and then, in each iteration, the algorithm finds a non-empty subset R of allowed tuples in A' that were identified not to be used by any solution of the CSP by the local consistency Φ and forbids them.[19] Note that such a subset always exists by Property 1.3: if a CSP is not Φ-consistent, then there exists at least one allowed tuple that is not used by any solution. Such updates are repeated until A' becomes Φ-consistent. Note that the returned CSP A' is equivalent to the input CSP A. In particular, if CSP A' is empty, then the input CSP A is unsatisfiable.

We will now show an alternative formalism for enforcing a local consistency that is based on propagators. Let p_i, $i \in \mathcal{P}$ be *propagators* indexed by a finite set $\mathcal{P}$, i.e., for all $i \in \mathcal{P}$, $p_i \colon 2^T \to 2^T$ is an isotone and intensive mapping such that $\mathrm{SOL}(A) = \mathrm{SOL}(p_i(A))$ holds for all $A \subseteq T$.[20] Suppose that we repeatedly apply these propagators to an input CSP, as outlined in Algorithm 1.3. Clearly, this algorithm is an instantiation of Algorithm 1.1, so, by Theorem 1.13, the output of Algorithm 1.3 is the greatest CSP (w.r.t. $\subseteq$) that is a subset of A and a fixed point of all propagators p_i, $i \in \mathcal{P}$.

Typically [6], the set of propagators is defined so that their common fixed points characterize the desired local consistency Φ, i.e., for all $A \subseteq T$, we have that

$$A \text{ is } \Phi\text{-consistent} \iff \forall i \in \mathcal{P} \colon p_i(A) = A. \tag{1.28}$$

In such case, Algorithm 1.3 computes $\mathscr{C}_\Phi(A)$ and is a more formal version of the previously shown Algorithm 1.2. In detail, the conditions on line 2 of both algorithms become equivalent due to (1.28). Also, the task of finding a non-empty subset R of Φ-inconsistent tuples

[19] As stated in [99, §3.2], usual local consistencies Φ are nogood-identifying, i.e., whenever a CSP is not Φ-consistent, it is because at least one allowed tuple is found not to be used by any solution of the CSP by the local consistency Φ. Such tuples are called Φ-inconsistent [99, Definition 3.12].

[20] Up to the requirement of isotony, this is a *constraint reduction function* [5, Definition 3.5].

inputs: CSP A, set of propagators p_i, $i \in \mathcal{P}$, i.e., each $p_i \colon 2^T \to 2^T$ is an isotone and intensive mapping satisfying $\forall A' \subseteq T \colon \mathrm{SOL}(A') = \mathrm{SOL}(p_i(A'))$.

1 $A' := A$
2 **while** $\exists i \in \mathcal{P} \colon p_i(A') \neq A'$ **do**
3 Find such i.
4 $A' := p_i(A')$
5 **return** A'

Algorithm 1.3: Propagation algorithm based on application of individual propagators to the input CSP.

from A' boils down to finding $i \in \mathcal{P}$ such that $p_i(A) \subsetneq A$. If such a propagator p_i exists, the set $R = A' - p_i(A')$ satisfies the required conditions stated on line 3 of Algorithm 1.2.

1.4.1.2 Example: Arc Consistency

We now exemplify the previously discussed concepts on arc consistency (AC). Recall that a CSP A is *(generalized[21]) arc consistent* [149, §4.1, 146, §3, 119, §6.2.2] if

$$(\{i\}, k) \in A \iff \exists (S, \ell) \in A : \ell_i = k \tag{1.29}$$

holds for all $S \in C_{\geq 2}$, $i \in S$, and $k \in D$. It is readily verified that AC satisfies Properties 1.1 and 1.3.

Remark 1.7. *In constraint programming [17, 141, 8, 101, 46, 99], a more common definition requires*

$$(\{i\}, k) \in A \implies \exists (S, \ell) \in A \colon (\ell_i = k \wedge \forall j \in S \colon (\{j\}, \ell_j) \in A) \tag{1.30}$$

instead of (1.29)*. To enforce AC in this sense, it is not necessary to forbid tuples corresponding to non-unary constraints. Both definitions are equivalent in terms of the forbidden unary tuples (i.e., the reduced domains) and have the same strength. In detail, any CSP that is arc consistent in the sense of* (1.29) *is also arc consistent in the sense of* (1.30) *(e.g., CSP in Figure 1.5a). Moreover, it can be easily shown that for any CSP A that is arc consistent in the sense of* (1.30) *(e.g., CSP in Figure 1.5b), the CSP*

$$A' = A - \{\, (S, k) \in A \mid S \in C_{\geq 2},\ \exists i \in S \colon (\{i\}, k_i) \notin A \,\} \tag{1.31}$$

is arc consistent in the sense of (1.29)*. Note that we only forbid some tuples of the non-unary constraints in* (1.31)*.*

To obtain the AC closure, one uses AC propagators. Formally, for $S \in C_{\geq 2}$, $i \in S$, and $k \in D$, we define the propagator $p_{S,i,k} \colon 2^T \to 2^T$ by

$$p_{S,i,k}(A) = \begin{cases} A & \text{if (1.29) is satisfied} \\ A - \{(\{i\}, k)\} & \text{if (1.29) is not satisfied and } (\{i\}, k) \in A\,. \\ A - \{(S, \ell) \in T_S \mid \ell_i = k\} & \text{if (1.29) is not satisfied and } (\{i\}, k) \notin A \end{cases} \tag{1.32}$$

[21] Sometimes [17, §3.3, 33] (but not always), arc consistency for CSPs with higher-order constraints is called generalized arc consistency. We do not use this name in the thesis and simply call the local consistency 'arc consistency'.

Clearly, $A \subseteq T$ is arc consistent if and only if $p_{S,i,k}(A) = A$ holds for all such triples (S, i, k). Moreover, the propagators always return an equivalent CSP and are intensive, isotone, and even idempotent.

Also, for any $S \in C_{\geq 2}$, $i \in S$, $k \in D$, and $A \subseteq T$, it is easy to verify that the CSP $p_{S,i,k}(A)$ satisfies (1.29) for this triple (S, i, k). By repeated application of different propagators $p_{S,i,k}$, as in Algorithm 1.3, more and more tuples become forbidden. Eventually, when $p_{S,i,k}(A) = A$ holds for all triples (S, i, k), CSP A is arc consistent and this result coincides with the AC closure of the initial CSP.

1.4.1.3 CSPs Solved by Enforcing Local Consistencies

In general, a local consistency Φ is neither a necessary nor a sufficient condition of satisfiability [5, 8] – e.g., an arc consistent CSP need not be satisfiable, but a CSP that is not arc consistent may be satisfiable.

For a local consistency Φ satisfying the properties from §1.4.1 and $A \subseteq T$, if $\mathscr{C}_\Phi(A)$ is empty, then CSP A is unsatisfiable because an empty CSP is unsatisfiable and $\mathrm{SOL}(A) = \mathrm{SOL}(\mathscr{C}_\Phi(A))$. On the other hand, non-emptiness of $\mathscr{C}_\Phi(A)$ does *not* in general imply that A is satisfiable.

However, for some CSP instances, enforcing a local consistency is sufficient to decide whether they are satisfiable (or even find a solution). Formally, we say that a local consistency Φ is *refutation complete* for a class of CSPs $\mathcal{A}$ if for any $A \in \mathcal{A}$ with non-empty $\mathscr{C}_\Phi(A)$, A is satisfiable.[22] Otherwise, we say that the local consistency is *refutation incomplete* (for some class of CSPs).

Example 1.11. *Restricting ourselves to pairwise CSPs, it is well known [61] that AC is refutation complete for CSPs where the graph $(V, C_{\geq 2})$ is a tree (or, more generally, a forest). For CSPs with higher-order constraints, AC is refutation complete if the factor graph of this CSP is acyclic (i.e., if the hypergraph is Berge-acyclic) and this is the only structural restriction where AC is refutation complete [31, Theorem 1].*

Recall that a CSP A with a total order $\preceq$ on its domain D is max-closed *[81, Definition 2.5] if for each scope $S \in C$, $(S, k) \in A$ and $(S, \ell) \in A$ implies $(S, k \vee \ell) \in A$ where $k \vee \ell \in D^S$ contains the element-wise maximal elements of k and ℓ w.r.t. the total order given by $\preceq$. Enforcing AC is refutation complete for max-closed CSPs [30, Example 6.39]. The same results also hold for* min-closed *CSPs that are defined dually.*

For other classes and more details, we refer the interested reader to [31, 161, 30] and references therein. △

Example 1.12. *Positive consistency is refutation complete for all CSPs. Indeed, by Definition 1.6, we have that $\mathscr{C}_{\mathrm{pos}}(A)$ is non-empty if and only if A is satisfiable. However, enforcing positive consistency, deciding whether a CSP is positively consistent, or even finding a solution to a positively consistent CSP is likely intractable [67, 20, 57].* △

[22] Synonymous terms are that enforcing Φ is a *decision procedure* for CSPs from $\mathcal{A}$ [33, 161, 133] or that Φ *decides* CSPs from $\mathcal{A}$ [31, Definition 2]. Alternatively, one can say that enforcing Φ is a *complete refutation method* [76, §3.2.1] (for a class of problems).

1.5 Weighted CSP and LP-Based Bounds

In this part, we formally define the *weighted constraint satisfaction problem* (WCSP) [23] which is an NP-hard combinatorial optimization problem [146]. Although there exist also other approaches [74], successful exact WCSP solvers usually rely on branch-and-bound search [136] which creates demand for good upper bounds that could efficiently prune the search space. Thus, we also give an overview of LP-based methods that can be used for obtaining such bounds. Throughout this section, we generally follow the notation that we used in [55a] or [56a] and also reuse certain parts of these papers.

The structure of a WCSP is the same as of a CSP, i.e., a triple (V, D, C). Similarly as with CSPs, we will assume that the structure is fixed and a WCSP is thus defined only by a collection of its *weight functions* $g_S \colon D^S \to \mathbb{R}$, $S \in C$. The task is to find an assignment $x \in D^V$ maximizing the objective function

$$F(x \,|\, g) = \sum_{S \in C} g_S\big(x|_S\big), \tag{1.33}$$

or to find the optimal value of WCSP g, i.e., $\max_{x \in D^V} F(x \,|\, g)$.

Notice that the weights of all the weight functions can be stacked into a single real-valued vector $g \in \mathbb{R}^T$ where T is the set of all tuples, as defined in (1.23). Analogously to CSPs, we will identify WCSP instances with vectors from $\mathbb{R}^T$. The components of $g \in \mathbb{R}^T$ can thus be indexed by $t \in T$, e.g., $g_t = g_S(k)$ if $t = (S, k)$.

Example 1.13 ([56a, Example 1])**.** *Let $V = \{1, 2, 3, 4\}$, $D = \{\mathsf{a}, \mathsf{b}\}$, and $C = \{\{1\}, \{2\}, \{2, 3\}, \{1, 4\}, \{2, 3, 4\}\}$. In such a setting, we want to maximize the expression*

$$g_{\{1\}}(x_1) + g_{\{2\}}(x_2) + g_{\{2,3\}}(x_2, x_3) + g_{\{1,4\}}(x_1, x_4) + g_{\{2,3,4\}}(x_2, x_3, x_4)$$

over $x_1, x_2, x_3, x_4 \in \{\mathsf{a}, \mathsf{b}\}$. In analogy to CSP terminology introduced in §1.4, this WCSP is Boolean but not pairwise. The factor graph of this WCSP is depicted in Figure 1.4. △

Remark 1.8. *In some formalisms, the objective* (1.33) *is minimized. For our purposes, these settings are equivalent as one can invert the sign of all weights and maximize instead.*

We emphasise that we make no assumption on the sign of the weights, as opposed to, e.g., [33, §7, 107, §2, 133, §II, 36, §3] where minimization and non-negative weights are assumed. When only non-negative weights are allowed, it is usual to assume that $\emptyset \in C$ since the weight $g_\emptyset$ then constitutes a bound on the optimal value. In contrast, we will need both positive and negative weights later in §3, so we require $\emptyset \notin C$ for simplicity of notation (the empty scope would not yield any bound by itself anyway).

In more general setting, hard constraints can be introduced by allowing $g \in \mathbb{R}^T_{-\infty}$ where $\mathbb{R}_{-\infty} = \mathbb{R} \cup \{-\infty\}$. We remark (without proof) that we see no obstacle to generalizing our results from later chapters to WCSPs with such constraints but we do not allow hard constraints for the sake of simplicity.

[23] This problem is also known as finite-valued CSP [132, 133], partial CSP [94], discrete energy minimization [88, 83, 119, 74, 135], max-sum (or min-sum) labeling problem [146, 120], or maximum a posteriori inference in graphical models (or Markov random fields) [125, 139, 146, 119, 134, 135, 87, 74]. It is also the main task in cost function networks [39]. Some of these formalisms however also allow infinite weights (i.e., hard constraints).

1.5.1 Linearity and Marginal Polytope

The objective (1.33) is linear in the weight vector g, i.e., for any $f, g \in \mathbb{R}^T$ and any $\alpha, \beta \in \mathbb{R}$, we have $F(x\,|\,\alpha f + \beta g) = \alpha F(x\,|\,f) + \beta F(x\,|\,g)$ for all $x \in D^V$. This is made explicit by using a different notation for the WCSP objective that is common in machine learning [140, §3, 119, §4, 88, §2, 139, §1.1]. We introduce this notation only for the purposes of this subsection, Example 1.14, and Remark 3.1 given later.

Let us define an indicator map $\phi\colon D^V \to \{0,1\}^T$ by

$$\phi(x)_t = [\![x|_S = k]\!] \qquad \forall t = (S,k) \in T \tag{1.34}$$

where $[\![\cdot]\!]$ denotes the *Iverson bracket* which equals 1 if the logical expression in the bracket is true and 0 if it is false. In other words, for any $x \in D^V$, we have that $\phi(x)_t = 1$ if and only if x uses tuple $t \in T$.

The WCSP objective (1.33) can be written as the dot product of vectors $g, \phi(x) \in \mathbb{R}^T$, namely

$$F(x\,|\,g) = \sum_{S \in C} g_S\big(x|_S\big) = \sum_{t \in T} g_t \phi(x)_t = g^\top \phi(x) \tag{1.35}$$

which makes explicit that the objective (1.33) is linear in the weight vector g. The optimal value of a WCSP can be thus also expressed as

$$\max_{x \in D^V} F(x\,|\,g) = \max_{x \in D^V} g^\top \phi(x) = \max_{\mu \in M} g^\top \mu = \max_{\mu \in \operatorname{conv} M} g^\top \mu \tag{1.36}$$

where conv denotes the convex hull operator [24, §2.1.4, 10, §1] and

$$M = \{\, \phi(x) \mid x \in D^V \,\} \subseteq \{0,1\}^T. \tag{1.37}$$

The last equality in (1.36) follows from the well-known fact that a linear function on a polytope attains its maximum in at least one vertex of the polytope [119, §3.3].

Note that M is defined only by the structure (V, D, C). The set $\operatorname{conv} M \subseteq [0,1]^T$ is known as the *marginal polytope* and has the central role in approaches to WCSP based on linear programming (see [139, 119, 127, 140, 142] and references therein).

1.5.2 Active Tuples and Upper Bound

A tuple $t = (S,k) \in T$ is *active* for WCSP $g \in \mathbb{R}^T$ if

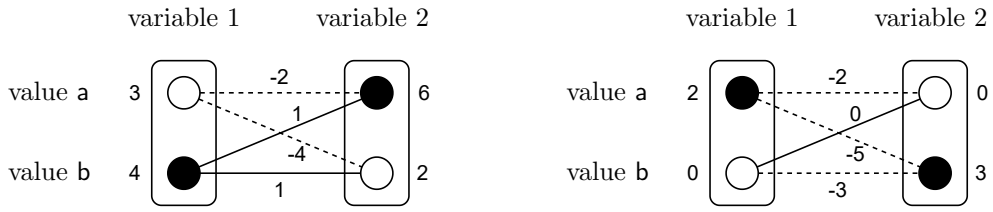
$$g_S(k) = \max_{\ell \in D^S} g_S(\ell), \qquad \text{i.e.,} \qquad g_t = \max_{t' \in T_S} g_{t'} \tag{1.38}$$

and is *inactive* otherwise. The set of all active tuples for g will be denoted[24] by $A^*(g)$. Since $A^*(g) \subseteq T$, $A^*(g)$ will be interpreted as a CSP (the *active-tuple CSP*).

We can define a tractable upper bound $B\colon \mathbb{R}^T \to \mathbb{R}$ on the optimal value of a WCSP g by

$$B(g) = \sum_{S \in C} \max_{k \in D^S} g_S(k) = \sum_{S \in C} \max_{t \in T_S} g_t \tag{1.39}$$

[24] The characteristic vector of the set $A^*(\cdot)$ is denoted by $\bar{\cdot}$ in [134, 146], by $\lceil\cdot\rceil$ in [149], and by mi$[\cdot]$ in [119]. CSP $A^*(\cdot)$ is analogous to CSP Bool$(\cdot)$ in [33, 107, 136].



(a) WCSP g (b) WCSP f

Figure 1.6: Visualisations of two WCSPs g and f with structure as in Examples 1.10 and 1.14. We depict WCSPs in analogy to CSPs: the active tuples are shown as black circles and full lines (because they are allowed in the active-tuple CSP) whereas inactive tuples are shown as white circles and dashed lines (because they are forbidden in the active-tuple CSP). The weights g_t (and f_t) are written next to the circles and line segments.

which is a convex piecewise-affine function. Notice that the maxima in (1.39) are attained precisely by the active tuples[25] $t \in A^*(g)$. The properties that link function B to the set of active tuples are stated formally in the next theorem and corollary.

Theorem 1.15 ([146, §4, 149, Theorem 2])**.** *Let $g \in \mathbb{R}^T$. For any $x \in D^V$, we have that*

(a) $B(g) \geq F(x\,|\,g)$,

(b) $B(g) = F(x\,|\,g)$ if and only if $x \in \mathrm{SOL}(A^*(g))$.

Proof. Statement (a) is immediate after comparing expressions (1.33) and (1.39). Statement (b) follows from the definition of an active tuple and a solution of a CSP. □

Corollary 1.5 ([146, 33])**.** *Let $g \in \mathbb{R}^T$. The upper bound is tight, i.e., $\max_{x \in D^V} F(x\,|\,g) = B(g)$, if and only if $A^*(g)$ is satisfiable.*

Proof. By Theorem 1.15, we have that $B(g) \geq \max_{x \in D^V} F(x\,|\,g)$ which holds as equality if and only if $\exists x \in D^V : x \in \mathrm{SOL}(A^*(g))$, which means that $A^*(g)$ is satisfiable. □

Example 1.14. *Recall the structure $V = \{1, 2\}$, $D = \{\mathsf{a}, \mathsf{b}\}$, and $C = \{\{1\}, \{2\}, \{1, 2\}\}$ from Example 1.10.*

An example of a WCSP g with this structure is shown in Figure 1.6a. The weight vector reads $g = (3, 4, 6, 2, -2, -4, 1, 1) \in \mathbb{R}^T$ where the order of the tuples is given by (1.25). The objective value of WCSP g for $x = (\mathsf{a}, \mathsf{b})$ is $F(x\,|\,g) = 3+2-4 = 1$ that can be also expressed as $g^\top \phi(x) = 1$ where $\phi(x) = (1, 0, 0, 1, 0, 1, 0, 0) \in \{0, 1\}^T$ (and the order of the tuples is again given by (1.25)).

The upper bound equals $B(g) = 4 + 6 + 1 = 11$ and is tight because CSP $A^*(g)$ is satisfiable. In particular, $F((\mathsf{b}, \mathsf{a})\,|\,g) = 11$. On the other hand, for WCSP f from Figure 1.6b, $A^*(f)$ is unsatisfiable, so the upper bound $B(f) = 2 + 0 + 3 = 5$ is not tight. △

[25] The term 'active tuple' thus comes from the term 'active inequality' (Footnote 2). Following §1.1.2, (1.39) can be calculated as the minimum of $\sum_{S \in C} z_S$ subject to $z_S \geq g_t \; \forall S \in C, t \in T_S$. At optimum, we have $z_S = \max_{t \in T_S} g_t$ and an inequality $z_S \geq g_t$ is active if and only if tuple t is active.

1.5.3 Reparametrizations and LP Relaxation

Definition 1.7 ([88, 87, 136, 117, 91])**.** *Let $f, g \in \mathbb{R}^T$. WCSP f is a* reparametrization *of WCSP g if $F(x\,|\,f) = F(x\,|\,g)$ for all $x \in D^V$.*[26]

The binary relation 'is a reparametrization of' on the set $\mathbb{R}^T$ is an equivalence, i.e., it is symmetric, reflexive, and transitive. It is easy to see that if f is a reparametrization of g, then $B(f)$ is not only an upper bound on the optimal value of f, but also on the optimal value of g. This suggests minimizing the upper bound on WCSP g over its reparametrizations, i.e.,

$$\min B(f) \tag{1.40a}$$
$$F(x\,|\,f) = F(x\,|\,g) \qquad \forall x \in D^V \tag{1.40b}$$
$$f \in \mathbb{R}^T. \tag{1.40c}$$

Although this optimization problem has an exponential number of constraints (1.40b), these constraints are linear in f by §1.5.1, hence the set of feasible solutions to (1.40) is an affine subspace of $\mathbb{R}^T$.

We will now briefly focus on describing the set of all reparametrizations of a WCSP. Let $g \in \mathbb{R}^T$ and $\varphi_{S,i}(k) \in \mathbb{R}$ for each $S \in C_{\geq 2}$, $i \in S$, and $k \in D$. WCSP $g^\varphi \in \mathbb{R}^T$ defined by

$$g^\varphi_{\{i\}}(k) = g_{\{i\}}(k) - \sum_{\substack{S \in C_{\geq 2} \\ i \in S}} \varphi_{S,i}(k) \qquad \forall i \in V,\ k \in D \tag{1.41a}$$
$$g^\varphi_S(k) = g_S(k) + \sum_{i \in S} \varphi_{S,i}(k_i) \qquad \forall S \in C_{\geq 2},\ k \in D^S \tag{1.41b}$$

is a reparametrization of g [149, §3.2].[27] If the WCSP is pairwise and the graph $(V, C_{\geq 2})$ is connected, one can obtain any reparametrization of g by an appropriate choice of φ [146, 88, 120]. In general, if the graph is not connected or the WCSP is not pairwise, not every reparametrization of g can be obtained by some choice of φ in (1.41).

Remark 1.9. *There exist larger subsets of reparametrizations that can be described by means of a* coupling scheme *[149, §3] which is a subset $\mathscr{S} \subseteq \{\,(S, S') \mid S, S' \in C,\ S \supsetneq S'\,\}$. For a coupling scheme $\mathscr{S}$, let $\varphi_{S,S'}(k) \in \mathbb{R}$ be scalars for each $(S, S') \in \mathscr{S}$ and $k \in D^{S'}$. Then, WCSP $f \in \mathbb{R}^T$ defined by*[28]

$$f_S(k) = g_S(k) - \sum_{(S',S) \in \mathscr{S}} \varphi_{S',S}(k) + \sum_{(S,S') \in \mathscr{S}} \varphi_{S,S'}\big(k|_{S'}\big) \qquad \forall S \in C,\ k \in D^S \tag{1.42}$$

is a reparametrization of $g \in \mathbb{R}^T$ [149, §3.2]. See that the transformation (1.41) *is a special case of* (1.42) *for the coupling scheme $\mathscr{S} = \{\,(S, \{i\}) \mid S \in C_{\geq 2},\ i \in S\,\}$.*

[26] It is also often said that f is *equivalent* to g [107, 134, 33, 135, 149, 146, 136, 36, 120, 98, 108]. Other related terms are that f is an equivalence-preserving (or equivalent) transformation of g.

[27] Especially in machine learning, variables φ are sometimes referred to as 'messages' [146, 88, 125, 149].

[28] We will not define a specific notation for WCSPs obtained using (1.42) as we use only the (simpler and less general) transformation (1.41) in the sequel.

Replacing variables f in (1.40) by g^φ from (1.41) while introducing φ as variables results in the optimization problem $\min_\varphi B(g^\varphi)$ which can be interpreted as an unconstrained minimization of a convex piecewise-affine function. Written explicitly, this is

$$\min \sum_{i\in V} \max_{k\in D} \big(g_{\{i\}}(k) - \sum_{\substack{S\in C_{\geq 2}\\ i\in S}} \varphi_{S,i}(k)\big) + \sum_{S\in C_{\geq 2}} \max_{k\in D^S} \big(g_S(k) + \sum_{i\in S} \varphi_{S,i}(k_i)\big) \tag{1.43a}$$

$$\varphi_{S,i}(k) \in \mathbb{R} \qquad \forall S \in C_{\geq 2},\ i \in S,\ k \in D, \tag{1.43b}$$

which can be formulated as a linear program, as discussed in §1.1.2.

The dual linear program corresponds to the *basic LP relaxation* of the WCSP g [146] that can be stated as [29]

$$\max \sum_{S\in C} \sum_{k\in D^S} g_S(k)\mu_S(k) \tag{1.44a}$$

$$\sum_{\substack{\ell\in D^S\\ \ell_i=k}} \mu_S(\ell) = \mu_{\{i\}}(k) \qquad \forall S \in C_{\geq 2},\ i \in S,\ k \in D \tag{1.44b}$$

$$\sum_{k\in D} \mu_{\{i\}}(k) = 1 \qquad \forall i \in V \tag{1.44c}$$

$$\mu_S(k) \geq 0 \qquad \forall (S,k) \in T. \tag{1.44d}$$

Indeed, this is an LP relaxation of WCSP g since there is a bijection between assignments $x \in D^V$ and the integral vectors μ feasible for (1.44). This LP relaxation was proposed independently a number of times [121, 94, 28, 139, 38] and it is a powerful tool in the sense that it solves all WCSPs defined by a tractable constraint language (i.e., the set of allowed weight functions) [132].

Remark 1.10. *In more detail, depending on the set of allowed weight functions, the resulting class of WCSPs can be either NP-hard or polynomially solvable [132].* [30] *In the latter case, the optimal value of the basic LP relaxation of g coincides with the optimal value of WCSP g [132].*

If the WCSP is Boolean and pairwise, the LP relaxation (1.44) is half-integral and can be reduced to the minimum st-cut problem [119, §12, 117, 23, 146]. On the other hand, even for pairwise structure with $|D| \geq 3$, solving the problem (1.44) (or (1.43)) is as hard as solving a general linear program [115].

1.5.4 Methods for Obtaining Bounds Using Reparametrizations

Although linear programs can be solved in polynomial time, our ability to find optimal solutions is limited by the fact that the time complexity of current off-the-shelf LP solvers is super-linear which makes them impractical for large-scale instances which occur, e.g.,

[29] Formally, there should also be the constraint $\sum_{k\in D^S} \mu_S(k) = 1$ for each $S \in C_{\geq 2}$ but these constraints are already implied by (1.44b) together with (1.44c) and are typically not included in the basic LP relaxation [133, 132, §3.1].

[30] An analogous statement (the Feder-Vardi conjecture [58]) was proved for CSPs independently in [159] and [27]. As stated in [89], this implies that general-valued CSPs (i.e., WCSPs that additionally allow $-\infty$ weights) also exhibit a dichotomy.

in computer vision [154, 88, 127, 134, 115]. We will now give an overview of methods for bounding the optimal value of (1.43) – such methods were developed, to some extent independently, in computer vision/machine learning and constraint programming communities.

1.5.4.1 Methods Based on BCD / Message Passing

To obtain good upper bounds while avoiding solving the LP relaxation to optimality, a competitive approach is to apply BCD (in this context also called 'message passing') to the problem (1.43). There exists a wide class of convergent message-passing algorithms that optimize different forms of a dual LP relaxation of WCSP by BCD. This family of algorithms originates from the field of computer vision.

For pairwise WCSPs, such algorithms include, e.g., MPLP [65], max-sum diffusion [96, 146], MPLP++ [134], or SPAM [135]. The fixed points of these methods are related to non-empty AC closure[31] of $A^*(g^\varphi)$. To be more precise, if the current solution is φ and the AC closure of $A^*(g^\varphi)$ is non-empty, then the aforementioned algorithms will not be able to improve the objective further. On the other hand, if the AC closure of $A^*(g^\varphi)$ is empty, the objective will be improved after a finite number of BCD iterations.

For instances with weight functions of higher arity, there exist specialized algorithms, such as [87, 149] whose stopping conditions are also based on local consistencies.

Since the optimized objective is a convex piecewise-affine function, the fixed points of these algorithms need not be global minima. Indeed, non-empty AC closure of $A^*(g^\varphi)$ is only a necessary condition for optimality of φ for (1.43) (or optimality of $f = g^\varphi$ for (1.40)) but not sufficient in general [146].

Fact 1.2. *Consider any algorithm for (approximate) optimization of* (1.43) *that returns points φ such that $A^*(g^\varphi)$ has non-empty AC closure. If, upon termination, $A^*(g^\varphi)$ is in some class of CSPs for which AC is refutation complete (recall Example 1.11), then $A^*(g^\varphi)$ is satisfiable and $B(g^\varphi)$ is the optimal value of WCSP g by Corollary 1.5 [33, §10].*

This is the case, e.g., when the factor graph of the WCSP is acyclic. Another example are instances where each g_S, $S \in C$ is supermodular because then $A^*(g^\varphi)$ is both max-closed and min-closed [33, §10, 149, §7] and AC is in such case refutation complete. Note that the transformation (1.41) *preserves supermodularity [36, §4, 149, §7, 120, §2.3]. We also remark that pairwise WCSPs with supermodular weight functions can be solved by a reduction to minimum st-cut [29, 120, 119, §11].*

Fact 1.3. *In Boolean pairwise WCSPs, non-empty AC closure of $A^*(g^\varphi)$ is a sufficient condition for optimality of φ for* (1.43) *[146] and the optimal value of the LP relaxation* (1.43) *can be computed by reduction to minimum st-cut [119, §12, 117, 23, 146] (also see [91]). However, the LP relaxation need not be tight.*

Since non-empty AC closure of the active-tuple CSP need not be sufficient for its satisfiability (hence optimality of the obtained bound by Corollary 1.5), one can include zero weight functions of higher arity to improve the bound even further, as it is done in [127, 11, 149, 147, 126] and corresponds to a fine-grained version of the Sherali-Adams hierarchy [124] for WCSP. Even though adding such factors can strengthen the LP relaxation

[31] Called node-edge agreement in [119, 134] and non-empty kernel in [146].

or improve the fixed points of BCD algorithms [127, 149, 11, 126], this approach may significantly increase the memory requirements of the method.

Aside from the previously mentioned algorithms, there are also approaches that optimize an LP relaxation based on acyclic decompositions. However, the optimal value of such a relaxation is the same as of (1.43) [119, 93, 88]. This is, e.g., the TRW-S algorithm [88] whose fixed points satisfy a local consistency condition called *weak tree agreement*. Moreover, any feasible solution satisfying weak tree agreement cannot be improved by subsequent iterations of TRW-S. Based on the study [83], TRW-S is typically the fastest method for (approximately) optimizing the LP relaxation but seems to be recently surpassed by SPAM [135].

Remark 1.11. *For pairwise WCSPs, the formulation* (1.43) *can be interpreted as a decomposition based on individual vertices and edges of the graph* $(V, C_{\geq 2})$ *and non-empty AC closure is then a special case of weak tree agreement [134]. Moreover, the weak tree agreement condition is in a precise sense equivalent to non-empty AC closure (up to reformulation) [119, 88] (also mentioned in [146]). In [129], several BCD algorithms were presented in a unified view within the framework of Lagrange dual decomposition of combinatorial problems and the fixed point conditions of this approach generalize both arc consistency and weak tree agreement.*

The connections among different formulations of (1.43) along with a BCD method on trees is given in [125]. Some of the mentioned BCD methods are also studied in a common framework in [142]. A more recent overview and taxonomy is presented in [135].

1.5.4.2 Methods Based on Enforcing (Soft) Local Consistencies

A different class of algorithms for obtaining an upper bound is based on enforcing (soft) local consistencies (instead of performing BCD). Such algorithms generally originated in the constraint programming community.

This is, e.g., the Virtual Arc Consistency (VAC) algorithm [33] that enforces non-empty AC closure of the active-tuple CSP and if the AC closure is found to be empty, the algorithm traces the operations of the AC propagator in anti-chronological order to find a reparametrization with a better bound. VAC algorithm is closely related to the Augmenting DAG algorithm [95, 146] that also enforces non-empty AC closure but was defined only for pairwise WCSPs. Because the terminating condition of VAC is non-empty AC closure of the active-tuple CSP, Facts 1.2 and 1.3 are also applicable.

Aside from VAC, there are also other methods based on arc consistency, such as EDAC [43], FDAC, or DAC [37, 98], which are faster at the cost of weaker bounds in general. A stronger local consistency is Optimal Soft Arc Consistency (OSAC) [38, 33] that is (by definition) satisfied by g^{φ} where φ is an optimal solution of (1.43).[32] OSAC is however limited to preprocessing, as it is too expensive to be maintained during search [33]. Higher-order consistencies for weighted CSP have been also studied in the constraint programming community, e.g., in [35].

However, it is known that for a given WCSP, there need not exist its reparametrization (with the same structure) such that its active-tuple CSP satisfies a given local consistency. Thus, stronger local consistencies (and thus better bounds) can be again obtained at the

[32] Assuming that there is only a single weight function for each scope, (dual of) the basic LP relaxation coincides with the OSAC formulation. In more general settings, this need not hold [90, Footnote 1, 133].

cost of introducing weight functions of higher arity (if such weight functions were not already present in the problem), e.g., these are ternary weight functions if one wants to enforce triangle-based consistencies, as it is done in [107].

Remark 1.12. *As a technical note, the aforementioned algorithms based on enforcing local consistencies do not obtain a reparametrization using* (1.41) *explicitly. Instead, the weights are shifted between the individual weight functions by applying the operations* project *and* extend *[98, 43, 33, 108]. However, for soft arc consistencies, these operations are in one-to-one correspondence with increasing or decreasing individual values of φ in* (1.41)*. In more general settings, the relation is analogous except that one uses a different coupling scheme (recall Remark 1.9) between the weight functions.*

1.5.4.3 Applications of Reparametrizations and Approximate Solutions

The listed algorithms are not only useful for pruning the search space in branch-and-bound search by providing upper bounds, but can also provide heuristics on which variables to branch (see, e.g., [136]). An alternative approach for reducing the search space is to divide the initial problem into two parts: a 'hard' part that is solved by an exact solver and an 'easy' part where the LP relaxation (1.43) is tight [74]. This reduces the size of the problem that needs to be solved by a combinatorial algorithm. Moreover, to solve the 'easy' part, it is not necessary to rely on an exact LP solver, but one can instead utilize the approximate approaches based on BCD or enforcing local consistencies [74].

Aside from exact solving, one can also use the reparametrized instance to obtain solutions that are acceptable in practice. As an example of such a primal heuristic, one can define an arbitrary total order on the variables V and then, in this order, choose for each variable such a value from D so that the objective (1.33) (where we sum only over the scopes whose variables are already instantiated) is the greatest. This greedy technique was proposed in [88, §4.3] and used in many computer vision applications [93, §4.2, 83, §4.5, 134, Equation (6), 129, §4.2, 92, §3.1], possibly with slight modifications.

1.5.5 Super-Reparametrizations

We now point our attention to the approach from [92] where super-reparametrizations [33] were introduced in order to reach tighter upper bounds on the WCSP optimal value.

Definition 1.8 ([92])**.** *Let $f, g \in \mathbb{R}^T$. WCSP f is a* super-reparametrization *of WCSP g if $F(x\,|\,f) \geq F(x\,|\,g)$ for all $x \in D^V$.*

In analogy to (1.40), it was proposed in [92, §2] to minimize the upper bound (1.39) over super-reparametrizations, i.e.,

$$\min B(f) \tag{1.45a}$$

$$F(x\,|\,f) \geq F(x\,|\,g) \qquad \forall x \in D^V \tag{1.45b}$$

$$f \in \mathbb{R}^T. \tag{1.45c}$$

We note that this formulation does not belong to the classical hierarchy of LP relaxations based on introducing additional weight functions of higher arity, mentioned in §1.5.4.

[33] Originally, super-reparametrizations were called *virtual potentials* in [92] and *sup-reparametrizations* in [125]. We introduced the more descriptive term *super-reparametrization* in [55a].

Also, in contrast to (1.40), it is unlikely that there is a compact (i.e., polynomially sized) representation of the optimization problem (1.45) since its optimal value coincides with the optimal value of WCSP g, as given by the following theorem.

Theorem 1.16 ([92, Theorem 1, 125])**.** *The optimal value of* (1.45) *is* $\max_{x \in D^V} F(x \,|\, g)$.

Proof. We have that $B(f) \geq F(x \,|\, f) \geq F(x \,|\, g)$ for any $x \in D^V$, so $B(f) \geq \max_x F(x \,|\, f) \geq \max_x F(x \,|\, g)$. Consequently, the optimal value of (1.45) is at least $m = \max_x F(x \,|\, g)$.

To show that this value is attained, define f by

$$f_t = m/|C| \qquad \forall t \in T. \tag{1.46}$$

It can be checked from (1.33) and (1.39) that $B(f) = F(x \,|\, f) = m$ for all $x \in D^V$. Due to $m \geq F(x \,|\, g)$ for all $x \in D^V$ by definition of m, f is feasible and optimal. □

To approximately solve (1.45), iterations of Augmenting DAG algorithm were interleaved with a so-called cycle-repair procedure in [92]. In detail, this procedure found inconsistent cycles (i.e., cycles that do not allow an assignment satisfying all constraints in the cycle) in the CSP $A^*(f)$ and, if such an inconsistent cycle was found, the weights of the tuples along the edges in this cycle were manipulated so that some of the tuples became inactive and the bound could be improved by Augmenting DAG algorithm. We note that this approach was defined only for pairwise WCSPs and its extension to higher-order WCSPs or other notions of local consistencies is not straightforward.

For certain instances, this method was able to obtain bounds superior to TRW-S and in some cases reported reaching optimal value of the original WCSP.

To the best of our knowledge, super-reparametrizations were not utilized nor analyzed except for [92] and [125]. However, [125] focuses almost solely on the relation between different formulations of reparametrizations, instead of super-reparametrizations.

Chapter 2

Bounds on Large-Scale Linear Programs Using Constraint Propagation

Although linear programs are solvable in polynomial time, solving very large sparse instances can be challenging in practice. Such linear programs occur in many areas, a prominent example being the computation of bounds in branch-and-bound search by LP relaxation. In such a setting, the applicability of classical off-the-shelf LP solvers is limited [154, 129, 128, 73, 47] which calls for the development of specialized (possibly approximate) methods. In this chapter, we present and exemplify our approach to approximately solving large-scale linear programs that we originally proposed in [52a].

Based on §1.1, we start in §2.1 by explaining how to practically perform constraint propagation (in this case, infer new implied inequalities) in a system of linear inequalities. Then, we review our general framework for approximate optimization of linear programs using constraint propagation, including a variant of capacity scaling that ensures its finiteness. Furthermore, we show that the VAC algorithm [33] can be (up to technical details) interpreted as an algorithm belonging to this framework. Finally, we exemplify this approach on the LP relaxation of weighted Max-SAT.

Some of our motivations for this approach were the logic-theoretic view on LP duality [103, §6] and the concept of inference duality [76, §17]. However, our approach is not a straightforward application of inference duality as we put inference into an iterative optimization scheme and do it in a refutation-incomplete way. Some parts of this chapter were published in [52a].

Here, we broaden the notion of refutation-completeness so that it can be applied to settings where we try to detect infeasibility of a system of linear inequalities and equalities by specific constraint propagation rules. Thus, we will more generally say that a propagation method is refutation complete for a given class of problems if it detects a contradiction in any infeasible/unsatisfiable problem from the class and is refutation incomplete otherwise.

2.1 Constraint Propagation for Linear Inequalities

Inference in a system of linear inequalities (and equalities) was discussed in §1.1.3. Here, we would like to determine whether such a system is feasible or not using constraint propagation. [34] Suppose that a fixed set of inference rules is at our disposal. Using these rules (which are problem-dependent), we generate new linear inequalities until either no new inequality can be generated or a contradiction (e.g., the inequality $0 \geq 1$) is found.

[34] As mentioned in §1.4.1, constraint propagation can be also seen as inference of new constraints. In this part, we consider adding implied constraints (i.e., implied inequalities) to the system instead of tightening the original constraints (which was discussed in §1.4.1 for the CSP). We analyze a possible analogy of constraint tightening in §4. However, tightening a constraint $A^i x \leq b_i$ to $A^i x = b_i$ is in correspondence to inferring that $A^i x \geq b_i$.

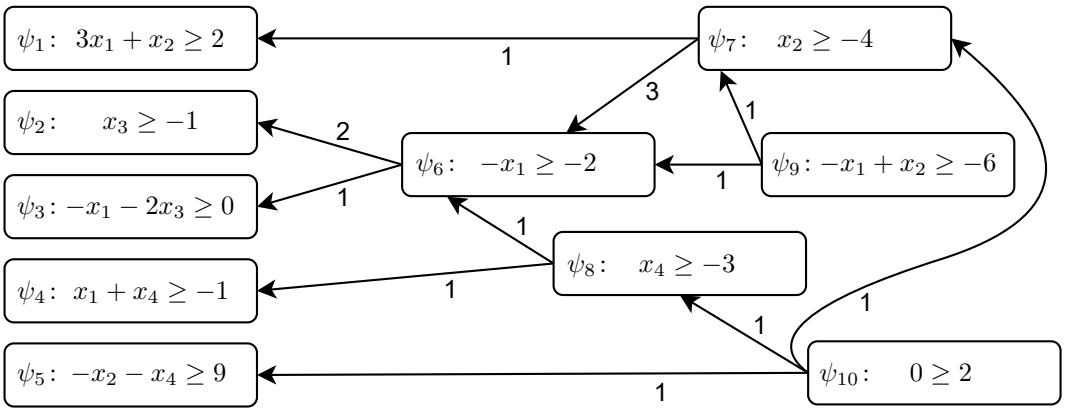


Figure 2.1: Propagation in a system of linear inequalities. The inequalities are indexed by 1–10. Inequalities 1–5 are initial, inequalities 6–10 are inferred. Edge weights indicate the coefficients of non-negative combinations.

Each time a new inequality is generated, its cause vector is stored, encoding how the inequality was obtained from the existing inequalities. In detail, the cause vector is given by the affine form of Farkas' lemma (Theorem 1.4) and contains the coefficients with which the original inequalities were combined to obtain the new inequality, as discussed in §1.1.3. When a contradiction is found, a certificate of infeasibility (i.e., a vector satisfying the Farkas alternative system) corresponds to its cause vector. However, as we will discuss next, one need not store all the cause vectors explicitly because, if a contradiction is found, the certificate of infeasibility can be computed efficiently by tracking the newly generated inequalities back to the original system.

Remark 2.1. *The procedure explained above is independent of the specific form of the system to which it is applied. In the rest of this section, we choose to exemplify it on a system in the form $Ax \geq b$. Although any system of linear inequalities (and possibly equalities) can be transformed to this form in linear time, such a transformation is not needed as one can always use the appropriate version of affine Farkas' lemma to derive new inequalities. E.g., we showed Farkas' lemma and its affine form for a system in the form $Ax = b$, $x \geq 0$ in Corollary 1.2 and Theorem 1.5, respectively.*

Let us focus on obtaining the certificate of infeasibility. As a running example, consider the system $Ax \geq b$ of $m = 5$ initial inequalities on the left in Figure 2.1. From inequalities ψ_2 and ψ_3, we infer inequality $\psi_6 = 2\psi_2 + \psi_3$.[35] Next, we gradually infer inequalities $\psi_7 = \psi_1 + 3\psi_6$, $\psi_8 = \psi_4 + \psi_6$, $\psi_9 = \psi_6 + \psi_7$, and finally $\psi_{10} = \psi_5 + \psi_7 + \psi_8$. Since ψ_{10} reads $0 \geq 2$, the initial system $\psi_1, \ldots, \psi_5$ is infeasible.

The history of propagation is represented by a directed acyclic graph (DAG) (V, E) where V is the set of all (initial and inferred) inequalities and $E \subseteq V \times V$ is a set of directed edges with weights $w\colon E \to \mathbb{R}_+$ so that each inferred inequality is given by $\psi_i = \sum_{j \in N_i^+} w_{ij}\psi_j$ where $N_i^+ = \{\, j \in V \mid (i,j) \in E \,\}$. By composing the inferences, each inequality ψ_i can be expressed in terms of the initial inequalities as $\psi_i = \sum_{j=1}^{m} y_j^i \psi_j$ where

[35] As in Example 1.2, we 'sum' inequalities in the following informal (yet natural) way: $2 \cdot (x_3 \geq -1) + (-x_1 - 2x_3 \geq 0) = (-x_1 \geq -2)$.

$y^i = (y^i_1, \ldots, y^i_m) \in \mathbb{R}^m_+$ is the cause vector of ψ_i. For $i \leq m$, we have $y^i = e^i$ where e^i is the i-th standard-basis vector of $\mathbb{R}^m$. For $i > m$, we have

$$y^i = \sum_{\substack{j \in N_i^+ \\ N_j^+ = \emptyset}} w_{ij} e^j + \sum_{\substack{j \in N_i^+ \\ N_j^+ \neq \emptyset}} w_{ij} y^j. \tag{2.1}$$

Note that $N_j^+ = \emptyset$ holds only for the initial inequalities ψ_j, $j \leq m$.

In the example, $V = \{1, \ldots, 10\}$, $y^6 = 2e^2 + e^3 = (0, 2, 1, 0, 0)$, $y^7 = e^1 + 3y^6$, $y^8 = e^4 + y^6$, $y^9 = y^6 + y^7$, and $y^{10} = e^5 + y^7 + y^8 = (1, 8, 4, 1, 1)$. Since $b^\top y^{10} = 2$ and $A^\top y^{10} = 0$, vector y^{10} is a certificate of infeasibility of the system $Ax \geq b$ by Farkas' lemma (Corollary 1.1).

More generally, the cause vector $y^i \in \mathbb{R}^m_+$ for inequality ψ_i given by $(c^i)^\top x \geq d_i$ satisfies $A^\top y^i = c^i$ and $b^\top y^i \geq d_i$ (recall Theorem 1.4).

2.1.1 Computing Certificate of Infeasibility

As we need the cause vector only for the final (contradictory) inequality (ψ_{10} in the example), storing all cause vectors explicitly in the memory is wasteful. In addition, some inferred inequalities may not be needed for the proof of infeasibility (ψ_9 in the example). We show that any single cause vector can be computed more efficiently by dynamic programming.

Let ψ_i, $i \in [m]$ be an initial inequality and ψ_k, $k > m$ be an inferred inequality. Then, y^k_i is the sum of weight-products [36] of all directed paths from node k to node i in the DAG. Suppose we want to compute y^k for some single k. We can consider only the subgraph of the DAG reachable from node k along directed paths. We introduce auxiliary variables z_j, which are to equal the sum of weight-products of all directed paths from node k to node j. Initially, we set $y^k = 0$, $z_k = 1$, and $z_j = 0$ for all $j \neq k$. Then, we process the nodes i of the subgraph in a topological order as follows: if $N_i^+ = \emptyset$ then set $y^k_i := z_i$, 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 $i \in [m]$. The time and space complexity of this algorithm is linear in the size of the graph.

2.2 Bounding the Optimal Value of Linear Programs

In §2.1, we explained how constraint propagation in a system of linear inequalities can be performed and, if infeasibility is detected, how to compute a certificate of infeasibility. Here, we will show how constraint propagation can be used to improve a non-optimal solution of a linear program.

For this, suppose that we have a primal-dual pair of linear programs and a dual-feasible solution y. For this fixed y, we construct the complementary slackness conditions in terms of the primal variables, which is a system of linear inequalities and equalities that is feasible if and only if y is optimal for the dual. We will show that, if infeasibility of the complementary slackness conditions is detected, then any certificate of infeasibility constitutes a dual-improving direction that can be used to improve the current dual-feasible y.

[36] We define the weight-product of a path to be the product of all edge weights along the path.

In the sequel, we show this approach for the primal-dual pair in the form (1.1).[37] To this end, suppose that we have a feasible solution y of the dual (1.1) (by which we implicitly assume that the dual is feasible). We would like to determine whether it is optimal and, if it is not optimal, find a dual-feasible solution with an improved objective.

By Theorem 1.2 (complementary slackness), a feasible $y \in \mathbb{R}^m$ is optimal for the dual (1.1) if and only if there exists $x \in \mathbb{R}^n$ feasible for the primal (1.1) that satisfies complementary slackness conditions with y, i.e., if

$$Ax = b \tag{2.2a}$$
$$x_j \geq 0 \qquad \forall j \in J \tag{2.2b}$$
$$x_j = 0 \qquad \forall j \in [n] - J \tag{2.2c}$$

with $J = \tau(y)$ (defined in (1.2b)) is feasible. By Farkas' lemma (recall Corollary 1.2), system (2.2) is infeasible if and only if system

$$b^\top \bar{y} < 0 \tag{2.3a}$$
$$A_j^\top \bar{y} \geq 0 \qquad \forall j \in J \tag{2.3b}$$

is feasible. Any vector $\bar{y}$ satisfying (2.3) is a certificate of infeasibility for (2.2). Note, the set of $\bar{y}$ feasible for (2.3) is a convex cone.

Moreover, it is easy to verify that any $\bar{y}$ feasible for (2.3) with $J = \tau(y)$ constitutes a dual-improving direction from y, i.e., one can update $y := y + \alpha\bar{y}$ for a suitable step size $\alpha > 0$ and improve the dual objective, as shown in Proposition 2.1.

Proposition 2.1 (cf. [110, Theorem 5.2])**.** *Let y be feasible for the dual* (1.1)*, $\bar{y}$ satisfy* (2.3) *for $J = \tau(y)$, and*

$$\alpha^* = \min_{\substack{j \in [n] \\ A_j^\top \bar{y} < 0}} \frac{c_j - A_j^\top y}{A_j^\top \bar{y}} > 0. \tag{2.4}$$

Then, $y' = y + \alpha\bar{y}$ is feasible for the dual (1.1) *and $b^\top y > b^\top y'$ if and only if $0 < \alpha \leq \alpha^*$.*

Proof. First, note that $\alpha^* > 0$ which follows from the fact that if $A_j^\top \bar{y} < 0$, then $j \notin J = \tau(y)$ by (2.3b), hence $c_j - A_j^\top y < 0$ by definition of $\tau(y)$ in (1.2b). This together with $A_j^\top \bar{y} < 0$ in (2.4) implies that each term in the minimum (2.4) is positive.[38]

We begin by the 'if' direction. To prove feasibility of y', i.e., $A_j^\top y' \geq c_j$ for all $j \in [n]$, we consider two cases. If $A_j^\top \bar{y} \geq 0$, then $A_j^\top y' = A_j^\top y + \alpha A_j^\top \bar{y} \geq A_j^\top y \geq c_j$ where we used $\alpha > 0$ and the assumption that y is feasible. If $A_j^\top \bar{y} < 0$, then $\alpha \leq (c_j - A_j^\top y)/A_j^\top \bar{y}$ by definition of α, which is equivalent to $A_j^\top y' = A_j^\top y + \alpha A_j^\top \bar{y} \geq c_j$. Also, $b^\top y' = b^\top y + \alpha b^\top \bar{y} < b^\top y$ due to (2.3a) and $\alpha > 0$.

For the 'only if' direction: if $\alpha \leq 0$, we have $b^\top y \leq b^\top y'$. If $\alpha > \alpha^*$, then there is $j \in [n]$ such that $A_j^\top \bar{y} < 0$ and $(c_j - A_j^\top y)/A_j^\top \bar{y} < \alpha$. This implies $A_j^\top y' = A_j^\top y + \alpha A_j^\top \bar{y} < c_j$, i.e., y' is infeasible. □

[37] In analogy to Remark 2.1, such an approach is applicable to a primal-dual pair in any form. The differences again lie only in the form of the complementary slackness conditions that determines the form of the certificate of infeasibility (i.e., improving direction).

[38] If there is no $j \in [n]$ with $A_j^\top \bar{y} < 0$, the dual (1.1) is unbounded and α can be chosen arbitrarily large.

inputs: instance of problem (1.1), dual-feasible solution y, constraint propagation rules for (2.2).

repeat
  Compute $J = \tau(y)$.
  Try to detect infeasibility of (2.2) by constraint propagation.
  if (2.2) is proved to be infeasible **then**
    Find an improving direction $\bar{y}$ satisfying (2.3).
    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 that (2.2) is infeasible.)

Algorithm 2.1: Iterative scheme for approximate optimization of the dual (1.1).

Remark 2.2. *Step size α^* given by* (2.4) *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 a linear program if an initial dual-feasible solution y is provided. In detail, one can construct system (2.2) and, if it is infeasible, find an improving direction $\bar{y}$ satisfying (2.3), update $y := y+\alpha\bar{y}$ using Proposition 2.1, 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 (1.1). Eventually, if (2.2) with $J = \tau(y)$ becomes feasible for current y, y is optimal for the dual. Note that, 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 2.3. *This optimization scheme is related to the primal-dual*[39] *algorithm [110, §5] (referred to as primal-dual method in [103, §7]) where complementary slackness conditions are not enforced strictly, but their violation is minimized instead. This change ensures convergence and finiteness of the algorithm. Practical algorithms for solving certain combinatorial problems that can be formulated as linear programs (e.g., shortest path problem or maximum flow) can be seen as instantiations of the primal-dual algorithm [110, 103].*

However, since determining feasibility of (2.2) is in general as hard as solving a linear program, we propose to do it in a refutation-incomplete way by constraint propagation that may detect infeasibility only sometimes. In other words, we propose to apply some (problem-dependent) constraint propagation rules to the complementary slackness conditions (2.2) and, whenever infeasibility is detected, construct the certificate of infeasibility that turns out to be a dual-improving direction. The iterative scheme based on constraint propagation is outlined in Algorithm 2.1. In analogy to the primal-dual algorithm (Remark 2.3), we called this iterative scheme *primal-dual approach* in [54a].

We emphasise that points y returned by Algorithm 2.1 need not be optimal since one may not detect infeasibility of (2.2) even if (2.2) is infeasible. However, even non-optimal solutions can be useful in practice. As an example, if the primal (1.1) is an LP relaxation

[39] As remarked in [110, §5.2], this is a misnomer because only a dual-feasible solution is maintained.

of a hard combinatorial problem (which is to be maximized), then any solution feasible for the dual (1.1) provides an upper bound on the optimal value of the original combinatorial problem. As we discussed earlier, such bounds can be used in branch-and-bound search where it may not be necessary or efficient to solve the LP relaxation to optimality as there is a trade-off between the time spent in computing the bound and the time spent in search.

2.2.1 Finiteness and Capacity Scaling

As already indicated, Algorithm 2.1 need not generally terminate in a finite number of iterations. In this section, we state sufficient conditions for its finiteness.

First of all, notice that the improving direction $\bar{y}$ generated on line 5 of Algorithm 2.1 can be chosen as any vector satisfying (2.3). 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 2.1 is from this set. This condition is not satisfied trivially because the same set J may be encountered repeatedly during the run of Algorithm 2.1 and, in each such an iteration, the improving direction $\bar{y}$ may be chosen as a different vector satisfying (2.3).

The following theorem shows that if the dual is in addition bounded and there exists a positive lower bound on the step sizes used in Algorithm 2.1, then the algorithm terminates after a finite number of iterations.

Theorem 2.1. *Let the dual* (1.1) *be feasible and bounded. Algorithm 2.1 terminates after a finite number of iterations (i.e., updates of y) provided that there exists $\alpha_{\min} > 0$ and a finite set $\bar{Y} \subseteq \mathbb{R}^m$ such that in every iteration:*

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

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

Proof. We follow a proof technique analogous to [48a, §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 in y, it terminates after at most $\lfloor (b^\top y - b^\top y^*)/\Delta \rfloor$ iterations.

Without loss of generality, we assume that $b^\top \bar{y} < 0$ for each $\bar{y} \in \bar{Y}$. If some $\bar{y} \in \bar{Y}$ violates this property, then it can be removed from $\bar{Y}$ as it is never used by the algorithm because it does not satisfy (2.3) for any $J \subseteq [n]$.

With this assumption, the objective improves in each iteration at least by $\Delta = \min_{\bar{y} \in \bar{Y}}(-\alpha_{\min} b^\top \bar{y})$ because, after any update from y to $y + \alpha\bar{y}$, $b^\top y - b^\top(y + \alpha\bar{y}) = -\alpha b^\top \bar{y} \geq -\alpha_{\min} b^\top \bar{y} \geq \Delta$.

For completeness, if $\bar{Y} = \emptyset$, Δ is not well-defined. But, in such case, Algorithm 2.1 always terminates already with the initial point as it is not capable of providing any improving direction. □

We will now discuss how the assumptions of Theorem 2.1 can be satisfied in practice. We begin by commenting on condition (a).

Because set $[n]$ has only a finite number of subsets, the existence of a finite set $\bar{Y}$ satisfying condition (a) in Theorem 2.1 is equivalent to the following: for each $J \subseteq [n]$, there exists a finite set $\bar{Y}(J)$ satisfying

$$\bar{Y}(J) \subseteq \{\, \bar{y} \in \mathbb{R}^m \mid \overbrace{b^\top \bar{y} < 0,\ \forall j \in J\colon A_j^\top \bar{y} \geq 0}^{\text{conditions (2.3)}} \,\} \tag{2.5}$$

such that, in any iteration, the improving direction chosen on line 5 in Algorithm 2.1 is from the set $\bar{Y}(\tau(y))$ where y is the current dual-feasible solution. Note that $\bar{Y}(J) = \emptyset$ if (2.2) is feasible because then (2.3) is infeasible. Also, if we are unable to prove infeasibility of (2.2) for some J, then one can set $\bar{Y}(J) = \emptyset$.

In particular, if the way of constructing the improving direction on line 5 in Algorithm 2.1 is deterministic and depends only on J, then each set $\bar{Y}(J)$ is either empty or a singleton, so $\bar{Y} = \bigcup_{J \subseteq [n]} \bar{Y}(J)$ satisfies condition (a) in Theorem 2.1.

Next, in order to guarantee the existence of a positive lower bound on the step size α, we introduce a heuristic analogous to capacity scaling in maximum flow algorithms [102, §7.3]. This heuristic was used to guarantee finiteness of the Augmenting DAG algorithm [95, 146], VAC algorithm [33, §11.1] [40], and later in [148, Equation (21)] and [48a, §3.1.7] in a more general setting of minimizing unconstrained convex piecewise-affine functions.

Formally, we consider the set of dual constraints which are 'almost' active. For this purpose, we define

$$\tau_\epsilon(y) = \{\, j \in [n] \mid A_j^\top y \leq c_j + \epsilon \,\} \tag{2.6}$$

where $\epsilon \geq 0$.

If $\tau(y)$ is replaced by $\tau_\epsilon(y)$ on line 2 in Algorithm 2.1, the scheme remains valid. In detail, if (2.2) is infeasible for $J = \tau_\epsilon(y)$, then it is also infeasible for $J = \tau(y)$ due to $\tau(y) \subseteq \tau_\epsilon(y)$ for any $\epsilon \geq 0$. This is equivalent to the fact that any $\bar{y}$ feasible for (2.3) with $J = \tau_\epsilon(y)$ is also feasible for (2.3) with $J = \tau(y)$.

Next, we prove that, if the computed step size on line 6 of Algorithm 2.1 is optimal and $\tau(y)$ on line 2 is replaced by $\tau_\epsilon(y)$ for some constant $\epsilon > 0$, then there exists a positive lower bound on the computed step sizes. Thus, condition (b) in Theorem 2.1 is implied by the aforementioned conditions which yields the next theorem.

Theorem 2.2. *Let the dual* (1.1) *be feasible and bounded. Algorithm 2.1 terminates after a finite number of iterations (i.e., updates of y) provided 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) $\tau(y)$ on line 2 is replaced by $\tau_\epsilon(y)$ where $\epsilon > 0$ is a constant,

(c) the step size computed on line 6 is optimal, i.e., $\alpha = \alpha^*$ where α^* is (2.4).

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 2.1. Analogously to the proof of Theorem 2.1, we assume (without loss of generality) that for any $\bar{y} \in \bar{Y}$, there exists dual-feasible y such that $\bar{y}$ satisfies (2.3) for $J = \tau_\epsilon(y)$.

Let us define

$$\delta = \max_{\bar{y} \in \bar{Y}} \max_{\substack{j \in [n] \\ A_j^\top \bar{y} < 0}} -A_j^\top \bar{y} \tag{2.7}$$

so that $\delta \geq -A_j^\top \bar{y} > 0$ for any $\bar{y}$ used by Algorithm 2.1 and any j considered in definition of step size (2.4). 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 2.1. On

[40] We comment on capacity scaling used in VAC in more detail in §2.3.

inputs: instance of problem (1.1), dual-feasible solution y, initial $\epsilon > 0$, constraint propagation rules for (2.2).

1 **while** ϵ is not small enough or time limit is not reached **do**

2 Improve y by Algorithm 2.1 with $\tau(y)$ replaced by $\tau_\epsilon(y)$ on line 2.

3 Decrease ϵ while keeping $\epsilon > 0$ (e.g., $\epsilon := \epsilon/10$).

4 **return** y

Algorithm 2.2: Approximate optimization of the dual (1.1) with gradually decreasing ϵ.

the other hand, if $\bar{Y} \neq \emptyset$, but for all $\bar{y} \in \bar{Y}$, we have $\forall j \in [n]\colon A_j^\top \bar{y} \geq 0$, then the dual is unbounded (see Footnote 38), which violates our assumption on its boundedness.

Now, we claim that $\alpha_{\min} = \epsilon/\delta$ is positive (because $\epsilon > 0$ and $\delta > 0$) and constitutes a lower bound on any step size computed by the algorithm. For this, let y be any feasible solution for the dual and $\bar{y} \in \bar{Y}$ satisfy (2.3) for $J = \tau_\epsilon(y)$. We claim that $\alpha^* \geq \alpha_{\min}$, i.e., for all $j \in [n]$ with $A_j^\top \bar{y} < 0$, $(c_j - A_j^\top y)/(A_j^\top \bar{y}) \geq \alpha_{\min}$. Indeed, as discussed in the proof of Proposition 2.1, if $A_j^\top \bar{y} < 0$, then $j \notin J = \tau_\epsilon(y)$ due to (2.3b), so $A_j^\top y - c_j > \epsilon$ by definition of $\tau_\epsilon(y)$. Combined with $\delta \geq -A_j^\top \bar{y}$, the claim follows. □

Note, instead of using a single fixed $\epsilon > 0$, one can run the iterative algorithm multiple times, each time with a lower value of ϵ, thus gradually improving the current feasible solution y even further, as it was done in [33, §11.1] or later in [48a, Algorithm 14]. In this way, we obtain an anytime algorithmic scheme outlined in Algorithm 2.2. It is experimentally observed that this modification results in larger step sizes and faster decrease of the objective [33, 48a], which is why it was utilized in our implementations [52a, 55a] that will be described later in §2.4 and §3.

In the following sections, we exemplify Algorithm 2.1 on LP relaxations of two combinatorial problems. In detail, in §2.3, we show that the VAC algorithm [33] is its special case and, in §2.4, we newly apply the approach to the LP relaxation of weighted Max-SAT.

2.3 Example: Basic LP Relaxation and Arc Consistency

In §1.5.4.2, we mentioned the VAC algorithm. Using notation from §1.4 and §1.5, we will focus on this algorithm here in detail and proceed to show that it is in a precise sense subsumed by our previously outlined iterative scheme.

To this end, suppose that we aim to compute an upper bound on WCSP $g \in \mathbb{R}^T$ by approximately minimizing (1.43) over variables φ. Recall from §1.5.4 that non-empty AC closure is necessary (but not sufficient) for optimality of φ for (1.43). Up to technical details (see Remark 1.12), the VAC algorithm improves a current solution φ of (1.43) by enforcing AC in the CSP $A^*(g^\varphi)$. In detail, if the AC closure of $A^*(g^\varphi)$ is empty, the algorithm traces the AC operations in anti-chronological order and changes the values of some components of φ to improve the objective. If the AC closure of $A^*(g^\varphi)$ turns out to be non-empty, the algorithm terminates.

By complementary slackness [146, 149, 119], φ is optimal for (1.43) if and only if there exists μ feasible for (1.44) such that $\mu_t = 0$ for all $t \in T - A^*(g^\varphi)$. Written explicitly, this

is

$$\sum_{\substack{\ell\in D^S\\ \ell_i=k}} \mu_S(\ell) - \mu_{\{i\}}(k) = 0 \qquad \forall S \in C_{\geq 2},\ i \in S,\ k \in D \tag{2.8a}$$

$$\sum_{k\in D} \mu_{\{i\}}(k) = 1 \qquad \forall i \in V \tag{2.8b}$$

$$\mu_t \geq 0 \qquad \forall t \in A^*(g^\varphi) \tag{2.8c}$$

$$\mu_t = 0 \qquad \forall t \in T - A^*(g^\varphi) \tag{2.8d}$$

which can be interpreted as an LP relaxation of CSP $A^*(g^\varphi)$ [146]. For $t = (S, k) \in T$, we write $\mu_S(k)$ and μ_t interchangeably, analogously to the notation used in §1.5.

Remark 2.4. *Observe that the basic LP relaxation* (1.44) *is a linear program in the form of the primal* (1.1)*, so complementary slackness conditions* (2.8) *are a special case of* (2.2)*. The Farkas alternative system to* (2.8) *is therefore* (2.3)*. Note that we changed* (1.44b) *to* (2.8a) *so that matrix A in* (2.3) *and* (1.1) *is defined unambiguously.*

We will now apply a certain propagation rule to the system (2.8), in analogy to what was explained in §2.1. This propagation rule will be inferring zero values of the primal variables μ in (2.8) using the marginalization constraint (2.8a). Next, we will show how to formally infer this by deriving new equalities implied by system (2.8) and also argue that this process is equivalent to enforcing AC in the CSP $A^*(g^\varphi)$.

On a high level, proving that variable μ_t is zero for some $t \in T$ will be done indirectly [41] by inferring an equality $\mu_t + p_t = 0$ where

$$p_t = \sum_{t'\in T} \beta_{t'}\mu_{t'} \quad \text{where constants } \beta_{t'} \text{ satisfy } \begin{cases} \beta_{t'} \geq 0 & \text{if } t' \in A^*(g^\varphi) \\ \beta_{t'} \in \mathbb{R} & \text{if } t' \notin A^*(g^\varphi) \end{cases}. \tag{2.9}$$

Due to constraints (2.8c)-(2.8d) we have $p_t \geq 0$, so $\mu_t + p_t = 0$ implies $\mu_t = 0$. The equality $\mu_t + p_t = 0$ will be obtained as a linear combination of the equalities (2.8a)-(2.8b) and the coefficients of this linear combination will form its cause vector y^t. Eventually, if for some $i \in V$ we infer that variables $\mu_{\{i\}}(k)$ are zero for all $k \in D$, this is contradictory with (2.8b) and we are able to construct a certificate $\bar{y}$ of infeasibility of system (2.8). Such a certificate satisfies the corresponding (recall Remark 2.4) system (2.3).

We now describe the propagation rules in detail, including the computation of cause vectors y^t. We note that all cause vectors y^t have the same dimension, equal to the number of constraints (2.8a)-(2.8b). We denote the standard-basis vector of this space corresponding to (2.8a) and (2.8b) by $e^{S,i,k}$ and e^i, respectively.

For $t \notin A^*(g^\varphi)$, y^t is initialized to be the zero vector that thus represents equality $0 = 0$; indeed, this corresponds to the form $\mu_t + p_t = 0$ as one can interpret it as $\mu_t - \mu_t = 0$ where $p_t = -\mu_t$ conforms to (2.9). If $t \in A^*(g^\varphi)$, y^t will be a non-zero vector representing an equality $\mu_t + p_t = 0$, as discussed earlier. The propagation rules are as follows:

[41] In theory, we might infer directly $\mu_t = 0$ if (1.44d) (and (2.8c)) were seen as constraints over *real* variables μ (instead of interpreting (1.44d) as the definition of *non-negative* variables μ). This change would introduce additional dual variables and the derivation would be slightly more technical and involved.

- If $\mu_{\{i\}}(k) = 0$ for some $i \in V$ and $k \in D$, constraints (2.8a) (together with non-negativity of μ variables) imply $\mu_S(\ell) = 0$ for all $S \in C_{\geq 2}$ and $\ell \in D^S$ such that $i \in S$ and $\ell_i = k$. For every such (S, ℓ), we can infer this formally as:

$$\overbrace{\big(\mu_{\{i\}}(k) + p_{\{i\}}(k) = 0\big)}^{\text{equality certifying } \mu_{\{i\}}(k) = 0} + \overbrace{\big(\sum_{\substack{\ell \in D^S \\ \ell_i = k}} \mu_S(\ell) - \mu_{\{i\}}(k) = 0\big)}^{\text{marginalization constraint (2.8a)}} = \overbrace{\big(\sum_{\substack{\ell \in D^S \\ \ell_i = k}} \mu_S(\ell) + p_{\{i\}}(k) = 0\big)}^{\text{new inferred equality}}.$$

  Let us now fix a single tuple $t = (S, \ell)$ satisfying the conditions above. The new inferred equality can be simply transformed into the form $\mu_t + p_t = 0$ by moving the other μ variables into the term p_t. Assuming that $p_{\{i\}}(k)$ is in the form (2.9), p_t is in such a form too. The cause vector of this equality is $y^t = y^{(\{i\},k)} + e^{S,i,k}$. Note the similarity of this inference rule to the third case in the definition of the AC propagator (1.32).

- If for some $S \in C_{\geq 2}$, $i \in V$, and $k \in D$ we have $\mu_S(\ell) = 0$ for all $\ell \in D^S$ with $\ell_i = k$, constraint (2.8a) implies $\mu_{\{i\}}(k) = 0$. Inference in terms of equalities:

$$\sum_{\substack{\ell \in D^S \\ \ell_i = k}} \overbrace{\big(\mu_S(\ell) + p_S(\ell) = 0\big)}^{\text{equality certifying } \mu_S(\ell) = 0} - \overbrace{\big(\sum_{\substack{\ell \in D^S \\ \ell_i = k}} \mu_S(\ell) - \mu_{\{i\}}(k) = 0\big)}^{\text{marginalization constraint (2.8a)}} = \overbrace{\big(\mu_{\{i\}}(k) + \sum_{\substack{\ell \in D^S \\ \ell_i = k}} p_S(\ell) = 0\big)}^{\text{new inferred equality}}.$$

  The new inferred equality is clearly in the form $\mu_t + p_t = 0$ for $t = (\{i\}, k)$ where $p_t = \sum_{\substack{\ell \in D^S \\ \ell_i = k}} p_S(\ell)$ is in the form (2.9) if this is the case for each $p_S(\ell)$. The cause vector of the new inferred equality is $y^t = \sum_{\substack{\ell \in D^S \\ \ell_i = k}} y^{(S,\ell)} - e^{S,i,k}$. Again, one can notice the similarity of this inference rule to the second case in the definition of the AC propagator (1.32).

- If for some $i \in V$ we have $\mu_{\{i\}}(k) = 0$ for all $k \in D$, constraint (2.8b) is contradictory (domain wipeout). Inference in terms of equalities:

$$\sum_{k \in D} \overbrace{\big(\mu_{\{i\}}(k) + p_{\{i\}}(k) = 0\big)}^{\text{equality certifying } \mu_{\{i\}}(k) = 0} - \overbrace{\big(\sum_{k \in D} \mu_{\{i\}}(k) = 1\big)}^{\text{simplex constraint (2.8b)}} = \overbrace{\big(\sum_{k \in D} p_{\{i\}}(k) = -1\big)}^{\text{inferred contradiction}}.$$

  The inferred equality is contradictory since every term p_t in the form (2.9) is non-negative under conditions (2.8c)-(2.8d). The cause vector of this contradictory equality is $\bar{y} = \sum_{k \in D} y^{(\{i\},k)} - e^i$.

By properties of p_t and the fact that $\bar{y}$ encodes an equality in the form $\sum_{t \in T_{\{i\}}} p_t = -1$, it is not hard to show that $\bar{y}$ is feasible for the Farkas alternative system (2.3) (recall Remark 2.4). In particular, the inferred contradictory equality is $\sum_{t \in T} \beta_t \mu_t = -1$ where $\beta_t \geq 0$ for all $t \in A^*(g^\varphi)$ by (2.9). Since the vector $\bar{y}$ stores the coefficients with which this equality was inferred, this inequality is in fact $\bar{y}^\top A \mu = \bar{y}^\top b$ where $\bar{y}^\top b = -1$ and $\bar{y}^\top A = \beta^\top$.

Consequently, $\bar{y}$ certifies infeasibility of (2.8) and constitutes an improving direction for the dual linear program to (1.44). However, let us note a subtlety. Strictly speaking,

optimization problem (1.43) is not the dual linear program to (1.44) because it is obtained by eliminating some of the dual variables (cf. §1.1.2), hence a subvector of $\bar{y}$ (where only the components corresponding to the φ variables are present [42]) is an improving direction for (1.43) from the current point φ.

The described algorithm is 'almost' equivalent to the VAC / Augmenting DAG algorithm [33, 95, 146]. The stopping points of both algorithms are characterized by the same property, namely non-empty AC closure of $A^*(g^\varphi)$. However, improving directions $\bar{y}$ constructed as above are in general different from the ones in [33, 95, 146], sometimes having larger absolute values of their components (and thus possibly allowing smaller step size α). The reason is that our algorithm does not take into account the values of the individual coefficients in the cause vectors y^t. A finer-grained version is possible, given that the cause vectors are computed in an anti-chronological order after the contradiction is detected instead of fixing their values already when inferring individual equalities. Such an approach would be analogous to what we described in [54a, Appendix B] or what will be explained later in §3.2.2.2.

It is known [33, Appendix A] that the VAC algorithm without capacity scaling can enter an infinite loop. In order to avoid this, it was proposed [33, §11.1] to replace the CSP $A^*(g^\varphi)$ by the CSP $A^*_\epsilon(g^\varphi)$ formed by 'almost' active tuples. In our notation [43], this reads

$$A^*_\epsilon(f) = \left\{ (S,k) \in T \;\middle|\; f_S(k) + \epsilon \geq \max_{\ell \in D^S} f_S(\ell) \right\} \tag{2.10}$$

where $\epsilon \geq 0$.

Despite finiteness of the VAC algorithm was already ensured in [33], we would like to point out that Theorem 2.2 is easily applicable here. First, under our assumption on finite weights in WCSP g (see §1.5), the basic LP relaxation (1.44) is clearly feasible and bounded. By strong duality, the dual is also feasible and bounded. Second, despite there might be many orders in which the AC operations can be applied to the CSP $A^*(g^\varphi)$, the set of tuples T is finite, so there are only finitely many ways in which the AC operations can be applied to each CSP. For each such order, the improving direction $\bar{y}$ is constructed deterministically (both by the VAC algorithm or by our propagation rules above). Finally, using $\epsilon > 0$ and optimal step size, both of these approaches will terminate in finite time.

Remark 2.5. *Since non-empty AC closure of $A^*(g^\varphi)$ is in general not sufficient for optimality of φ for* (1.43) *[146, §5], applying the aforementioned rules to problem* (2.8) *is a refutation-incomplete method. In other words, these rules may not detect infeasibility of* (2.8) *in some cases. Consequently, VAC (or the algorithm just sketched above) may terminate in a non-optimal point φ.*

2.4 Example: LP Relaxation of Weighted Max-SAT

In this section, we outline how the iterative scheme based on constraint propagation from §2.2 applies to a problem different from WCSP. In particular, we focus on the

[42] To be precise, recall that the components of $\bar{y}$ are in one-to-one correspondence with the constraints (2.8a)-(2.8b). Partitioning $\bar{y}$ into two vectors, $\bar{\varphi}$ (whose components correspond to constraints (2.8a)) and $\bar{z}$ (whose components correspond to (2.8b)), yields the improving direction $\bar{\varphi}$ for (1.43) from the current point φ.

[43] CSP $A^*_\epsilon(\cdot)$ is denoted by $\mathrm{Bool}_\theta(\cdot)$ in [33, §11.1], by $\mathcal{O}^\epsilon(\cdot)$ in [134, Appendix B.1], and by $\mathrm{mi}_\epsilon[\cdot]$ in [119, §6.2.4].

weighted Max-SAT problem. We note that the symbols V and C have a different meaning in this part than what was considered in the (W)CSP setting (in §1.4, §1.5, and §2.3).

In the weighted Max-SAT problem [138, §16, 21, §19.2], we are given a finite set V of Boolean variables and a finite set C of clauses with positive weights $w \in \mathbb{R}^C_{++}$. The task is to find an assignment to the Boolean variables such that the sum of the 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 denote the number of negated variables in a clause $c \in C$ by $n_c = |V_c^-|$ so that $n \in \mathbb{N}_0^C$. For any $S \subseteq V$, we will use the shortcut $x(S) = \sum_{i \in S} x_i$. [44]

The classical LP relaxation of weighted Max-SAT [138, §16.3, 21, §19.2] is the left-hand problem of the primal-dual pair

$$\begin{aligned}
&\max w^\top z && && && (2.11a)\\
&z_c \le x(V_c^+) + n_c - x(V_c^-) && && y_c \ge 0 && \forall c \in C && (2.11b)\\
&x_i \ge 0 && && p_i - y(C_i^+) + y(C_i^-) \ge 0 && \forall i \in V && (2.11c)\\
&x_i \le 1 && && p_i \ge 0 && \forall i \in V && (2.11d)\\
&z_c \ge 0 && && q_c + y_c \ge w_c && \forall c \in C && (2.11e)\\
&z_c \le 1 && && q_c \ge 0 && \forall c \in C. && (2.11f)
\end{aligned}$$

As with (1.1), we will refer to the left-hand problem (2.11) as primal and to the right-hand problem (2.11) as dual. This LP relaxation was shown to be as hard to solve as any linear program [114].

Note that the primal variables x_i represent the (relaxed) original Boolean variables. To better understand the meaning of the primal constraint (2.11b), it is easy to verify that for any $c \in C$ and $x \in \{0,1\}^V$, the expression

$$x(V_c^+) + n_c - x(V_c^-) = \sum_{i \in V_c^+} x_i + \sum_{i \in V_c^-} (1 - x_i) \tag{2.12}$$

is zero if and only if clause c is not satisfied by the corresponding assignment of truth values and is at least 1 otherwise. Thus, for any fixed assignment $x \in \{0,1\}^V$, setting the auxiliary z variables as large as possible (while maintaining feasibility) results in $z_c = 0$ if clause c is not satisfied and $z_c = 1$ otherwise.

Let us now point our attention to the dual (2.11). Analogously to the notation in the primal, for any subset of clauses $S \subseteq C$, we use the shortcut $y(S) = \sum_{c \in S} y_c$ and similarly for the other dual variables. Clearly (cf. §1.1.2), at dual optimum we have

$$\begin{aligned}
&p_i = \max\{y(C_i^+) - y(C_i^-),\, 0\} && \forall i \in V && (2.13a)\\
&q_c = \max\{w_c - y_c,\, 0\} && \forall c \in C. && (2.13b)
\end{aligned}$$

Substituting (2.13) into the dual objective together with $n^\top y = \sum_{i \in V} y(C_i^-)$ results in a simpler form of the dual, namely

$$\begin{aligned}
&\min \sum_{c \in C} \max\{w_c - y_c, 0\} + \sum_{i \in V} \max\{y(C_i^+), y(C_i^-)\} && && (2.14a)\\
&\qquad y_c \ge 0 \qquad \forall c \in C && && (2.14b)
\end{aligned}$$

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

[44] This shortcut will be used only in sections related to Max-SAT and SAT.

2.4.1 Employing Constraint Propagation

We will now show how the iterative scheme with constraint propagation from §2.2 is applied.

We begin by stating the complementary slackness conditions in terms of the primal variables based on fixed dual variables: a vector $y \in \mathbb{R}^C_+$ is optimal for (2.14) if and only if there exists $x \in \mathbb{R}^V$ satisfying

$$\begin{aligned}
x(V_c^+) + n_c - x(V_c^-) &\geq 1 && \forall c \in C_{\geq 1}(y) = \{\, c \in C \mid y_c = 0 \,\} && \text{(2.15a)}\\
x(V_c^+) + n_c - x(V_c^-) &= 1 && \forall c \in C_{=1}(y) = \{\, c \in C \mid 0 < y_c < w_c \,\} && \text{(2.15b)}\\
x(V_c^+) + n_c - x(V_c^-) &\leq 1 && \forall c \in C_{\leq 1}(y) = \{\, c \in C \mid y_c = w_c \,\} && \text{(2.15c)}\\
x(V_c^+) + n_c - x(V_c^-) &= 0 && \forall c \in C_{=0}(y) = \{\, c \in C \mid y_c > w_c \,\} && \text{(2.15d)}\\
x_i &= 1 && \forall i \in X_1(y) \;= \{\, i \in V \mid y(C_i^+) > y(C_i^-) \,\} && \text{(2.15e)}\\
x_i &= 0 && \forall i \in X_0(y) \;= \{\, i \in V \mid y(C_i^+) < y(C_i^-) \,\} && \text{(2.15f)}\\
0 \leq x_i &\leq 1 && \forall i \in X_U(y) = \{\, i \in V \mid y(C_i^+) = y(C_i^-) \,\}. && \text{(2.15g)}
\end{aligned}$$

We note that the z variables were eliminated from system (2.15) to simplify it, which resulted in the 4 types of constraints (2.15a)-(2.15d). Also, notice that $\{C_{\geq 1}(y), C_{=1}(y), C_{\leq 1}(y), C_{=0}(y)\}$ is a partition of C and $\{X_1(y), X_0(y), X_U(y)\}$ is a partition of V.

We now define propagation rules for system (2.15). These rules set the values of some of the undecided variables x_i, $i \in X_U(y)$ to 0 or 1. Precisely, we iteratively visit each constraint (2.15a)-(2.15d) and look whether with the already decided variables it permits only a single value of some so-far undecided variable. If so, we fix the value of this variable (i.e., make it decided). If some constraint (2.15a)-(2.15d) cannot be satisfied by any assignment subject to the already decided variables, (2.15) is infeasible. During propagation, we store the cause vector for each decided variable, so that if infeasibility is detected, we are able to construct an improving direction $\bar{y}$ for (2.14).

The propagation rules are listed in Table 2.1, divided into 3 groups based on the types of constraints (2.15a)-(2.15d). For each rule, we also specify how to construct the cause vector y^i for each decided variable x_i. For $i \in X_1(y) \cup X_0(y)$, we define $y^i = 0$ so that it can be referred to in the definition of other cause vectors y^j or $\bar{y}$. To simplify the explanation of the rules, for any $c \in C$, we define $V_c = V_c^+ \cup V_c^-$. Moreover, for any $c \in C$ and $i \in V_c$, we denote

$$x_i^c = \begin{cases} x_i, & \text{if } i \in V_c^+, \\ 1 - x_i, & \text{if } i \in V_c^- \end{cases} \tag{2.16}$$

so that, e.g., (2.12) can be written as $\sum_{i \in V_c} x_i^c$. Finally, we define $e^c \in \mathbb{R}^C$ to be the standard-basis vector of $\mathbb{R}^C$ with 1 in the place corresponding to clause c.

Example 2.1. *To show how the rules outlined in Table 2.1 are applied, let* $C = \{1, 2, 3, 4\}$, $V = \{1, 2, 3, 4, 5\}$, *and system* (2.15a)-(2.15d) *be*

$$\begin{aligned}
x_1 \;+\; (1 - x_2) &= 0 && \text{(2.17a)}\\
x_1 \;+\; x_3 &\geq 1 && \text{(2.17b)}\\
x_2 \;+\; x_4 \;+\; (1 - x_5) &\leq 1 && \text{(2.17c)}\\
x_3 \;+\; x_5 &= 1 && \text{(2.17d)}
\end{aligned}$$

Applicable to		Rule
$C_{\geq 1}(y)$ and $C_{=1}(y)$	A1	If there is $k \in V_c$ such that x_k is undecided and for all $i \in V_c - \{k\}$, x_i is decided and satisfies $x_i^c = 0$, then we set $x_k = [\![k \in V_c^+]\!]$ and $y^k = e^c + \sum_{i \in V_c - \{k\}} y^i$.
	A2	If for all $i \in V_c$, x_i is decided and satisfies $x_i^c = 0$, then we obtain a contradiction and set $\bar{y} = e^c + \sum_{i \in V_c} y^i$.
$C_{=1}(y)$ and $C_{\leq 1}(y)$	B1	If there is exactly one $i \in V_c$ such that x_i is decided and $x_i^c = 1$, then we set $x_k = [\![k \in V_c^-]\!]$ and $y^k = -e^c + y^i$ for all undecided variables x_k, $k \in V_c$.
	B2	If there are two (or more) decided variables x_i, x_j for $i, j \in V_c$ with $x_i^c = x_j^c = 1$, then we obtain a contradiction and set $\bar{y} = -e^c + y^i + y^j$.
$C_{=0}(y)$	C1	If there is no $i \in V_c$ such that x_i is decided and $x_i^c = 1$, then we set $x_k = [\![k \in V_c^-]\!]$ and $y^k = -e^c$ for all undecided variables x_k, $k \in V_c$.
	C2	If there is $i \in V_c$ such that x_i is decided and $x_i^c = 1$, then we obtain a contradiction and set $\bar{y} = -e^c + y^i$.

Table 2.1: Propagation rules for system (2.15). The first column determines the types of constraints to which the rule applies.

where all x variables are initially undecided, i.e., $X_U(y) = V$.

First, it is clear that condition (2.17a) *(together with $x_1, x_2 \in [0,1]$) implies $x_1 = 0$ and $x_2 = 1$ by rule C1 and we set $y^1 = y^2 = -e^1$. Second, since $x_1 = 0$, we can apply rule A1 to* (2.17b) *to infer that $x_3 = 1$ with $y^3 = e^2 + y^1$. Next, because we have $x_2 = 1$, applying rule B1 to* (2.17c) *results in $x_4 = 0$ and $x_5 = 1$ with $y^4 = y^5 = -e^3 + y^2$. Finally, since $x_3 = 1$ and $x_5 = 1$, rule B2 detects that condition* (2.17d) *is contradictory and sets $\bar{y} = -e^4 + y^3 + y^5$.* △

The derivation of the inequalities corresponding to the cause vectors $y^i \in \mathbb{R}^C$ is technical and must be done for each rule separately. This is more complicated when compared to the case presented in §2.3 because here we have more types of constraints (which also include inequalities in different directions) and we set the x variables not only to 0, but also to 1, resulting in a larger number of propagation rules.

Remark 2.6. *For general weighted Max-SAT, the propagation rules listed in Table 2.1 need not always detect infeasibility of* (2.15) *and thus are refutation incomplete. As an example, let $V = \{1,2,3\} = X_U(y)$ and* (2.15a)-(2.15d) *be*

$$x_1 + x_2 + x_3 = 1 \tag{2.18a}$$
$$x_1 + x_2 = 1 \tag{2.18b}$$
$$x_1 + x_3 = 1 \tag{2.18c}$$
$$x_2 + x_3 = 1. \tag{2.18d}$$

No rule from Table 2.1 is applicable, but (2.18) *is infeasible.*

However, for weighted Max-2SAT (i.e., instances where $|V_c| \leq 2$ for all $c \in C$), the rules are refutation complete. In particular, if no more propagation is possible and no

contradiction is detected, setting all undecided variables x_i to $\frac{1}{2}$ satisfies all constraints of (2.15)*. This is simply verified by case analysis while assuming that none of the listed propagation rules is applicable (cf. more general Lemma 5.7a given later).*

Remark 2.7. *If any of the propagation rules A1, B1, or C1 is applied to some constraint $c \in C$ from* (2.15a)-(2.15d)*, then all variables x_i, $i \in V_c$ become decided and the corresponding constraint from* (2.15a)-(2.15d) *is satisfied. Thus, no rule from Table 2.1 is applicable to this constraint anymore. Consequently, after at most $|C|$ rules are applied, one either detects a contradiction or finds out that no more rules are applicable.*

Moreover, see that if a single rule sets the values of multiple variables at once (i.e., rules B1 and C1), then the cause vectors for these newly decided variables are identical.

Remark 2.8. *One can ask whether it is possible to infer other values of undecided variables than 0 or 1, such as $\frac{1}{2}$. Assuming that inference is done only from a single constraint from* (2.15a)-(2.15d)*, this is impossible because the polyhedron defined by a single (in)equality from* (2.15a)-(2.15d) *subject to $0 \leq x_i \leq 1$ (where some of the variables may be already set to 0 or 1) has integral vertices. This was proved for constraint in the form* (2.15a) *in [76, Theorem 45] and the other cases* (2.15b)-(2.15d) *could be shown analogously (see Lemma 5.5 given later).*

2.4.2 Finding Step Size by Approximate Line Search

If a contradiction is detected in (2.15) and improving direction $\bar{y}$ from the current point y is constructed, we need to find a step size $\alpha > 0$ in order to update $y := y + \alpha\bar{y}$ as in §2.2. The optimal way (exact line search) would be to minimize the univariate function $g(\alpha) = f(y + \alpha\bar{y})$ over $\alpha > 0$ subject to $y + \alpha\bar{y} \geq 0$ where $f(y)$ is the objective (2.14a). Since this is too costly for large instances, we do only approximate line search: we find the first breakpoint $\alpha > 0$ (i.e., non-differentiable point) of the univariate convex piecewise-affine function g, i.e., the smallest $\alpha > 0$ at which at least one previously inactive affine function becomes active.[45] This value of α may be further reduced to ensure feasibility, i.e., $y + \alpha\bar{y} \geq 0$. More formally, such α is the maximum number satisfying the following constraints[46]:

- To stay within the feasible set, we need $y_c + \alpha\bar{y}_c \geq 0$, therefore $\alpha \leq -y_c/\bar{y}_c$ for all $c \in C$ with $\bar{y}_c < 0$.
- For terms $\max\{w_c - y_c, 0\}$, if $w_c - y_c > 0$ and $\bar{y}_c > 0$ (or with both inequalities inverted), then we need $\alpha \leq (w_c - y_c)/\bar{y}_c$ where the bound is the point where $w_c - (y_c + \alpha\bar{y}_c) = 0$. So, this is for all $c \in C$ such that $(w_c - y_c)\bar{y}_c > 0$.
- For terms $\max\{y(C_i^+), y(C_i^-)\}$, if $y(C_i^+) > y(C_i^-)$ and $\bar{y}(C_i^+) < \bar{y}(C_i^-)$ (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 formulating the Farkas alternative system to the complementary slackness conditions and noting the substitution (2.13), it can be shown analogously to Proposition 2.1 that there always exists $\alpha > 0$ satisfying these bounds.

[45] In formalism of §1.1.2, for a convex piecewise-affine function $\sum_{k \in K} \max_{l \in L_k} \left(c_{kl}^\top x + d_{kl}\right)$, an affine function $c_{kl^*}^\top x + d_{kl^*}$ is *active* (for some x) if $c_{kl^*}^\top x + d_{kl^*} = \max_{l \in L_k} \left(c_{kl}^\top x + d_{kl}\right)$, cf. Footnote 25.

[46] This choice of α is analogous to the *first-hit strategy* in [48a, §3.1.4].

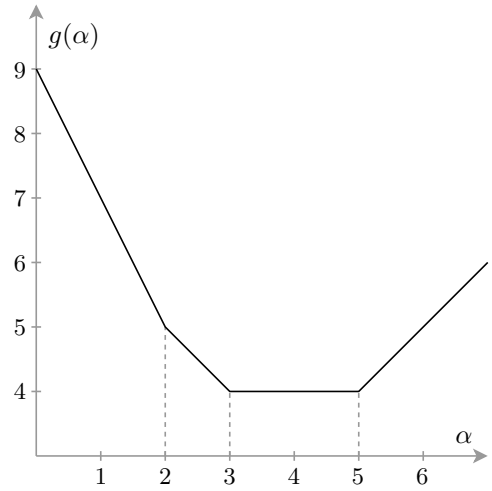


Figure 2.2: Graph of function g from Example 2.2.

Example 2.2. *Let* $C = \{1, 2, 3\}$, $V = \{1, 2\}$, $C_1^+ = \{1, 2\}$, $C_1^- = \emptyset$, $C_2^+ = \{1\}$, $C_2^- = \{2, 3\}$, *and* $w = (1, 1, 1)$. *For this setting, the objective* (2.14a) *is given by*

$$\begin{aligned} f(y) = &\max\{1 - y_1, 0\} + \max\{1 - y_2, 0\} + \max\{1 - y_3, 0\} \\ &+ \max\{y_1 + y_2, 0\} + \max\{y_1, y_2 + y_3\}. \end{aligned} \tag{2.19}$$

Next, let the current point be $y = (0, 3, 2)$. *So, system* (2.15a)-(2.15d) *reads*

$$x_1 + x_2 \geq 1 \tag{2.20a}$$
$$x_1 + (1 - x_2) = 0 \tag{2.20b}$$
$$(1 - x_2) = 0 \tag{2.20c}$$

and (2.15e)-(2.15f) *sets* $x_1 = 1$ *and* $x_2 = 0$. *Rule C2 applied to* (2.20b) *detects contradiction and constructs the improving direction* $\bar{y} = (0, -1, 0)$. *We will now compute step size* α *according to the principle outlined above.*

By the first point, we have to ensure $\alpha \leq -y_2/\bar{y}_2 = 3$ *to satisfy* $y + \alpha\bar{y} \geq 0$ *because* $\bar{y}_2 < 0$. *Since* $\bar{y}_1, \bar{y}_3 \geq 0$, *there are no other bounds given by the first point. Following the second point, we need* $\alpha \leq (w_2 - y_2)/\bar{y}_2 = 2$ *due to* $w_2 - y_2 < 0$ *and* $\bar{y}_2 < 0$. *Since* $(w_1 - y_1)\bar{y}_1 = (w_3 - y_3)\bar{y}_3 \leq 0$, *no other bounds are enforced by the second point. The third point yields conditions* $\alpha \leq 3$ *and* $\alpha \leq 5$.

The maximum number α *satisfying the above derived upper bounds is* $\alpha = 2$ *and* y *is therefore updated to* $y + \alpha\bar{y} = (0, 1, 2)$.

For clarity, we show the graph of

$$\begin{aligned} g(\alpha) &= f(y + \alpha\bar{y}) \\ &= 1 + \max\{\alpha - 2, 0\} + 0 + \max\{3 - \alpha, 0\} + \max\{0, 5 - \alpha\} \end{aligned} \tag{2.21}$$

in Figure 2.2. Notice that the bounds given by the second and third point, i.e., $\{2, 3, 5\}$, *are precisely the breakpoints* α *of function* g *with* $\alpha > 0$. *Moreover, see that the chosen step size* $\alpha = 2$ *is the first (in increasing order) breakpoint of* g *with* $\alpha > 0$ *and also the point where the affine function* $\alpha - 2$ *from* (2.21) *becomes active (while it is inactive for* $\alpha < 2$*).*

Based on Figure 2.2, 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.* △

2.4.3 Algorithm Overview and Implementation Details

Let us summarize the algorithm that follows the general iterative scheme previously shown in Algorithm 2.1. We start with $y = 0$ (which is dual-feasible) and repeat the following iteration: For the current y, construct system (2.15). Apply rules listed in Table 2.1 to fix values of undecided variables. During that, construct the DAG defining each y^i (recall §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, similarly as in §2.1.1. Calculate step size α as in §2.4.2 and update $y := y + \alpha\bar{y}$.

Remark 2.9. *System* (2.15) *can be interpreted as an LP relaxation of a CSP with Boolean variables $x_i \in \{0, 1\}$, $i \in V$ and constraints* (2.15a)-(2.15f)*. The propagation corresponds to enforcing AC in this CSP (in the sense of* (1.30)*). The whole algorithm seeks to find a feasible dual solution of the LP relaxation of weighted Max-SAT that enforces this CSP to have a non-empty AC closure. Compare this with the WCSP case, where* (2.8) *is an LP relaxation of the CSP formed by the active tuples and the VAC algorithm seeks to find a reparametrization that makes this CSP have a non-empty AC closure. Note that, 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 that was introduced in §2.2.1. In particular, we redefine the sets in (2.15) up to a tolerance $\epsilon > 0$: we replace $y_c > 0$ by $y_c > \epsilon$, $y(C_i^+) < y(C_i^-)$ by $y(C_i^+) + \epsilon < y(C_i^-)$, $y_c = w_c$ by $w_c - \epsilon \leq y_c \leq w_c + \epsilon$ etc. We follow the general scheme outlined in Algorithm 2.2 where we initialize $\epsilon = w(C) = \sum_{c \in C} w_c$ and whenever the algorithm cannot detect infeasibility with the current ϵ, we keep the current y and update $\epsilon := \epsilon/10$. We continue until ϵ is not very small (10^{-6}).

All data structures used by the algorithm need space that is linear in the input size, i.e., in the number $\sum_{c \in C} |V_c|$ of non-zeros in linear program (2.11). In particular, it can be shown that the DAG (used to calculate $\bar{y}$) can be conveniently stored as an oriented subgraph of the clause-variable incidence graph.[47] Following Remark 2.7, it is only necessary to store for each $c \in C$ what rule was applied to this clause and a partition of V_c that encodes which variables were decided before the rule was applied and which variables were decided by the rule.

Remark 2.10. *We argue that this algorithm terminates after a finite number of iterations (recall Theorem 2.1). First, the primal* (2.11) *is always feasible and bounded by $w(C)$, so the dual* (2.11) *is also feasible and bounded. Second, based on Remark 2.7, there are only finitely many options in which the rules from Table 2.1 can be applied to system* (2.15) *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* (2.14)*), it can be shown that there exists a positive lower bound on the step size by analyzing the bounds listed in §2.4.2 and noting that $\epsilon > 0$ (this is analogous to the proof of Theorem 2.2).*

[47] 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 $i \in V_c$. If V_c is unique for each $c \in C$, then the clause-variable incidence graph is isomorphic to the factor graph of $(V, \{V_c \mid c \in C\})$.

Alternatively, one can apply Theorem 2.2 directly: if we used the dual formulation (2.11) *where variables p and q are set as in* (2.13)*, then one could show that the computed step size α is in fact optimal for updating $(p, q, y) := (p, q, y) + \alpha(\bar{p}, \bar{q}, \bar{y})$ where the improving direction $(\bar{p}, \bar{q}, \bar{y})$ for the dual* (2.11) *is obtained in a natural way from the already existing improving direction $\bar{y}$.*

2.4.4 Experimental Results

We compared the upper bound on the optimal value of (2.11) obtained by our algorithm with the exact optimal value of (2.11) computed by an off-the-shelf LP solver (we used Gurobi [72] with default parameters) on the Max-SAT Evaluations 2018 benchmark [9]. This benchmark contains 2591 instances of weighted Max-SAT. Gurobi was able to optimize (without memory overflow) the smallest 2100 instances, the largest of which had up to 600 thousand clauses, 300 thousand variables and 1.6 million non-zeros.[48] The largest instances in the benchmark have up to 27 million clauses, 19 million variables and 77 million non-zeros and were still manageable by our algorithm.

From the smallest 2100 instances, 154 instances were Max-2SAT and 91 instances did not contain any unit clause. As discussed in Remark 2.6, our algorithm attained the exact optimum of the LP relaxation on instances of Max-2SAT. Similarly, if an instance does not contain any unit clause, then setting $x_i = \frac{1}{2}$, $i \in V$ and $z_c = 1$, $c \in C$ yields an optimal solution of the primal (2.11) with objective value $w(C)$ [76, §13.1.1]. Our algorithm also attains optimality on these instances because $y = 0$ is already optimal for the dual. We exclude these instances from further evaluation.

Each of the remaining 1855 instances contains a clause of length at least 3 and also contains a unit clause, thus 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 by two criteria, namely

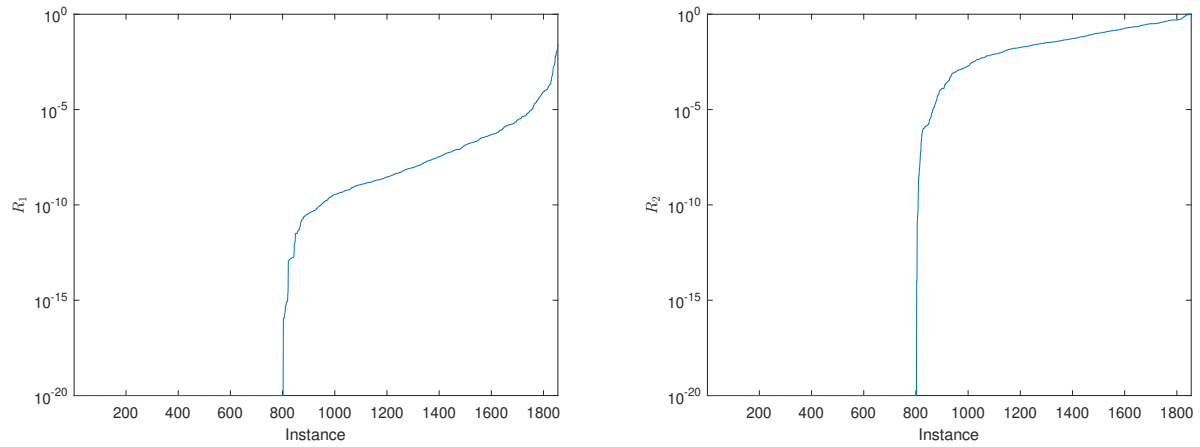
$$R_1 = \frac{U - U^*}{U^*} \qquad \text{and} \qquad R_2 = \frac{U - U^*}{w(C) - U^*} \tag{2.22}$$

where U^* is the optimal value of (2.11) and U is the upper bound computed by our algorithm. Both criteria are invariant to scaling the weights. Criterion R_1 is the relative difference between the optimal value and the provided upper bound whereas criterion R_2 shows how tight the bound is relative to the trivial bound $w(C)$.

The sorted numbers R_1 and R_2 for the 1855 instances are plotted in Figure 2.3. For 802 instances, we obtained $U = U^*$. Due to this, the vertical logarithmic axes in Figures 2.3a and 2.3b are trimmed, starting from 10^{-20}. Based on the linear plots, the obtained upper bound U is comparable to U^* in at least 1000-1100 cases. In detail, R_1 was lower than 10^{-6} and 10^{-8} on 1644 and 1308 from the 1855 instances, respectively, and was always lower than 0.029. Also, R_2 was higher than 0.6 only in 35 instances.

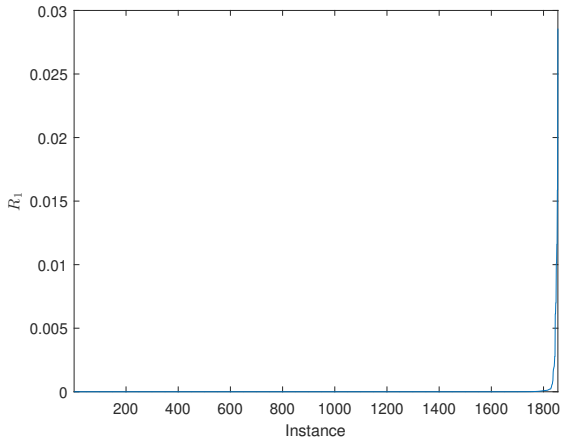
For 152 out of the 2100 considered instances, the LP relaxation is known to be tight (i.e., its optimal value coincides with the optimal value of the weighted Max-SAT problem). In 133 of them, our algorithm attained this optimum. Only 2 of these were Max-2SAT and each contained a unit clause, so optimality was not guaranteed trivially.

[48] By the number of non-zeros, we mean $\sum_{c \in C} |V_c|$.

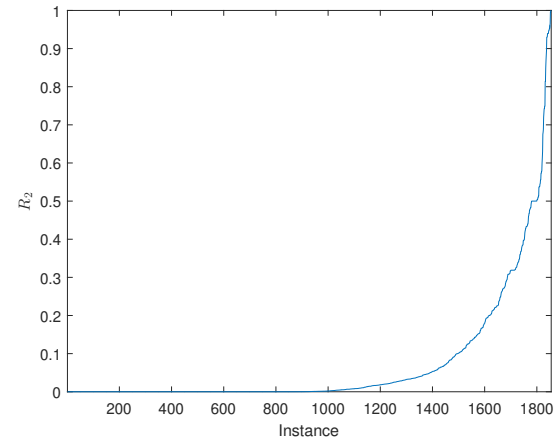


(a) R_1 (log scale).

(b) R_2 (log scale).



(c) R_1 (linear scale).



(d) R_2 (linear scale).

Figure 2.3: Sorted values of R_1 (left) and R_2 (right) with linear (down) and logarithmic (up) scale.

An unoptimized implementation of our algorithm was on average 3.3 times faster [49] than Gurobi. We believe a significant speed-up could be achieved by warm-starting. The part of the DAG needed to explain the found contradiction (see §2.1) is usually very small. If the DAG is built in every iteration from scratch, most of it is therefore thrown away. Since the system (2.15) 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 in [108] and for the Augmenting DAG algorithm in [145] with significant speed-ups.

2.4.5 Tightness of the Bound on Tractable Max-SAT Classes

We show that our constraint propagation rules in system (2.15) 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 (2.11) is tight and any point returned by our algorithm is an optimizer of the LP relaxation.

It was shown in [85, 41] 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 optimum of the linear program 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 [85, Lemma 3, 41] and the optimum of its LP formulation equals (up to a trivial recalculation) the optimum of the LP relaxation of Max-SAT which is thus tight. Since this is an instance of Max-2SAT, any point returned by our algorithm is optimal by Remark 2.6.

Following Remark 2.9, if we view (2.15) as the LP relaxation of a CSP with Boolean variables, then the propagation rules in Table 2.1 enforce AC of this CSP (in the sense of (1.30)). If the factor graph of this CSP is acyclic, AC solves this CSP exactly (recall Example 1.11). Equivalently, if the clause-variable incidence graph (i.e., the factor graph of this CSP) is acyclic, our constraint propagation rules are refutation complete and the points returned by our algorithm are optimal. Additionally, if no contradiction is detected, an integral solution to (2.15) can be constructed, so the LP relaxation is tight.

2.5 Discussion

In this chapter, we reviewed a technique that we originally proposed in [52a] to bound the optimal value of large-scale linear programs. To summarize, given a dual-feasible solution, infeasibility of the complementary slackness conditions (a system of linear inequalities and equalities in the primal variables) is detected by constraint propagation. If the system

[49] The average speed-up of 3.3 is the overall runtime of the LP solver divided by overall runtime of our algorithm. The geometric and arithmetic mean of speed-ups for the individual instances are 4.4 and 19.0, respectively.

is proved to be infeasible, a certificate of infeasibility of this system turns out to be a dual-improving direction that can be used to improve the current solution. In general, the constraint propagation method may be refutation incomplete, hence the feasible solutions returned by the algorithm may not be global optima of the linear program. We do not solve the system given by complementary slackness exactly because this is not practical for large instances.

Although constraint propagation for systems of constraints (here, linear inequalities and equalities) with continuous variables has been studied [13], our novelty lies in constructing the infeasibility certificate which constitutes a dual-improving direction.

This technique can be seen as a generalization of the VAC / Augmenting DAG algorithm [33, 95, 146] for WCSP. Newly, we applied it to the LP relaxation of weighted Max-SAT. Recall from §1.5.4 that the main purpose of (soft) local consistencies in WCSP, such as EDAC [43], FDAC, DAC [37, 98], or OSAC [38], is to bound the optimal value of WCSP during search. Each local consistency has a different trade-off point between bound tightness and computational complexity. In this view, our approach can be seen as a (soft) local consistency technique for other problems than WCSP.

Though in principle our approach can be also applied to other linear programs (if an initial dual-feasible solution is available), the quality of the obtained bounds depends on the possibility to design suitable (possibly problem-dependent) propagation rules.

The propagation rules that we used in §2.3 and §2.4 can be interpreted as examples of a single general rule that infers whether some inequalities in the system given by complementary slackness are always active (recall Definition 1.2). Indeed, in (2.8), we inferred whether some of the constraints (2.8c) are always active (i.e., if some of the primal variables μ are implied to be zero). Similarly, in case of (2.15), we inferred whether some variables x_i are implied to be 0 or 1, which corresponds to one of the inequalities in (2.15g) being always active. We will precisely define and thoroughly analyze this general propagation rule, which is applicable to any linear program, later in §4. In §4, we also show that the fixed points of BCD are related to the stopping points of Algorithm 2.1 with this rule. Before we do that, we show another application of our approach from §2.2 in §3.

Chapter 3

Bounds on Weighted CSP Using Constraint Propagation and Super-Reparametrizations

As we reviewed in §1.5, a popular approach for obtaining bounds on the optimal value of a WCSP is to compute a feasible (ideally optimal) solution of its dual LP relaxation. For large instances, such solutions are obtained by methods based on BCD (§1.5.4.1) or constraint propagation (§1.5.4.2) whose stopping points are typically characterized by local consistencies of the active-tuple CSP of the reparametrized WCSP. This approach is limited in that it cannot enforce an arbitrary level of local consistency, unless new weight functions are introduced. Moreover, the methods for improving the bound by enforcing local consistencies in the active-tuple CSP needed to be specifically designed based on the chosen kind of local consistency.

In contrast, in this chapter, we propose a method that is able to improve the bound on the WCSP optimal value using any kind of constraint propagation without introducing new weight functions. To this end, we recall from §1.5.5 the problem of minimizing an upper bound on the optimal value of a WCSP over its super-reparametrizations (1.45) and show that it can be approximately optimized using any method that can (at least sometimes) detect unsatisfiability of a CSP. On the other hand, a super-reparametrization of a WCSP need not preserve the objective values of the individual assignments or even the set of optimal assignments, but, as we will show, it is capable of providing a (possibly tighter) bound on the optimal value.

Super-reparametrizations were not utilized nor analyzed in the literature except for [92] and [125]. Yet, they are mentioned only briefly in [125] where the main focus is on reparametrizations. Thus, to fill in this gap, we also provide additional theoretical results connected to the optimization problem (1.45) and to super-reparametrizations.

Optimization problem (1.45) has an exponential number of constraints (1.45b), equal to the number of assignments, i.e., $|D^V|$. However, the suitable structure of this problem allows us to apply the approach that we outlined in §2.2. Indeed, the method that we present in this chapter can be interpreted as an instantiation of the previously shown Algorithm 2.1. However, we chose not to include this method as another example in §2 because it is more involved and its description is extensive.

The structure of this chapter is as follows. We begin in §3.1 by extending the notation that was previously defined in §1.5 and use it to state the necessary and sufficient conditions of optimality for the problem (1.45). Next, we present our approach for approximate minimization of the upper bound using constraint propagation in §3.2. After that, we theoretically analyze the optimization problem and also identify further properties of the active-tuple CSPs and the sets of optimal (and also non-optimal) super-reparametrizations in §3.3. Unsurprisingly, we prove in §3.4 that some decision problems connected to our approach and super-reparametrizations are NP-complete.

This chapter contains (in some places reformulated or rewritten) text and figures from

the submitted journal version [56a] of our earlier conference paper [55a].

3.1 Notation and Optimality Conditions

Throughout this chapter, we will use the notation for CSPs and WCSPs that was defined in §1.4 and §1.5. Nevertheless, to concisely explain our approach, we additionally define the set

$$M^{\perp} = \{\, d \in \mathbb{R}^T \mid F(x\,|\,d) = 0 \;\forall x \in D^V \,\}, \tag{3.1}$$

i.e., $M^{\perp}$ is the set of WCSPs whose objective values are zero for all assignments (called zero problems in [146]). See that the set $M^{\perp}$ is defined only by the structure (V, D, C). An example of a WCSP belonging to $M^{\perp}$ is in Figure 1.6b.

By linearity of $F(x\,|\,\cdot)$ (see §1.5.1), set $M^{\perp}$ is defined as the solution set of a system of homogeneous linear equalities and thus constitutes a linear subspace of $\mathbb{R}^T$. Moreover, it is clear that WCSP f is a reparametrization of WCSP g if and only if $f - g \in M^{\perp}$ due to $F(x\,|\,f) = F(x\,|\,g) \iff F(x\,|\,f-g) = 0$ for all $x \in D^V$. The affine subspace of all reparametrizations of g is thus [50] $g + M^{\perp} = \{\, g + d \mid d \in M^{\perp} \,\}$ and the optimization problem of minimizing the upper bound over reparametrizations (1.40) can be stated as

$$\min\{\, B(f) \mid f \text{ is a reparametrization of } g \,\} \;=\; \min\{\, B(f) \mid f \in g + M^{\perp} \,\}. \tag{3.2}$$

Analogously, we define the set of all WCSPs (with the fixed structure) whose objective values are non-negative for all assignments, i.e.,

$$M^{*} = \{\, d \in \mathbb{R}^T \mid F(x\,|\,d) \geq 0 \;\forall x \in D^V \,\}. \tag{3.3}$$

Set M^{*} is a polyhedral convex cone which is however not pointed (i.e., it contains a line [24, §2.4]) because $M^{\perp} \subseteq M^{*}$ and the subspace $M^{\perp}$ is non-trivial (assuming $|V| > 1$). For a given $d \in \mathbb{R}^T$, deciding whether $d \notin M^{*}$ is NP-complete, as we show later in Corollary 3.2.

Again, WCSP f is a super-reparametrization of WCSP g if and only if $f - g \in M^{*}$. Therefore, the set of all super-reparametrizations of g is the translated cone $g + M^{*} = \{\, g + d \mid d \in M^{*} \,\}$. The binary relation 'is a super-reparametrization of' (on the set of WCSPs with a fixed structure) is reflexive and transitive, hence a preorder. It is not anti-symmetric: $f - g \in M^{*}$ and $g - f \in M^{*}$ does *not* imply $f = g$ but merely $f - g \in M^{\perp}$, i.e., that f is a reparametrization of g. This is because the cone M^{*} is not pointed (see [79, §2] and [24, §2.4]).

Remark 3.1. *To explain our notation for the sets $M^{\perp}$ and M^{*}, recall the mapping ϕ from* (1.34) *and the set M from* (1.37)*. By expressing the set $M^{\perp}$ as*

$$M^{\perp} = \{\, d \in \mathbb{R}^T \mid d^{\top}\phi(x) = 0 \;\forall x \in D^V \,\} = \{\, d \in \mathbb{R}^T \mid d^{\top}\mu = 0 \;\forall \mu \in M \,\}, \tag{3.4}$$

it becomes clear that $M^{\perp} \subseteq \mathbb{R}^T$ is the orthogonal space [156, §1.1] of the set $M \subseteq \mathbb{R}^T$.

Similarly, we have that

$$M^{*} = \{\, d \in \mathbb{R}^T \mid d^{\top}\phi(x) \geq 0 \;\forall x \in D^V \,\} = \{\, d \in \mathbb{R}^T \mid d^{\top}\mu \geq 0 \;\forall \mu \in M \,\} \tag{3.5}$$

is the dual cone [156, §1.1] to M. It is easy to show that

$$M^{*} = (\operatorname{conv} M)^{*} = (\operatorname{cone} M)^{*} \tag{3.6}$$

[50] Note, the symbol '+' in the expression $g + M^{\perp}$ denotes the sum of a vector and a set of vectors.

where cone *denotes the conic hull operator and* * *the dual cone operator [24, §2.1.5 and §2.6.1, 10, §1]. Thus,* M^* *can be also seen as the dual cone to the marginal polytope. We made this observation in [56a, Remark 1] and, to the best of our knowledge, this has not been mentioned before.*

Using the set M^*, one can rewrite the optimization problem (1.45) as

$$\min\{\, B(f) \mid f \text{ is a super-reparametrization of } g \,\} \;=\; \min\{\, B(f) \mid f \in g + M^* \,\}. \quad (3.7)$$

It is important to note that every f feasible for (3.7) (i.e., every super-reparametrization of g) satisfies

$$B(f) \geq F(x\,|\,f) \geq F(x\,|\,g) \qquad \forall x \in D^V, \quad (3.8)$$

and thus provides an upper bound on the optimal value of g. This is also an immediate consequence of the previously given Theorem 1.16.

The next theorem characterizes optimal solutions of (3.7).

Theorem 3.1. *Let* f *be feasible for* (3.7)*. The following are equivalent:*

(a) f *is optimal for* (3.7)*,*

(b) $B(f) = \max\limits_{x \in D^V} F(x\,|\,f) = \max\limits_{x \in D^V} F(x\,|\,g)$*,*

(c) CSP $A^*(f)$ *has a solution* x *satisfying* $F(x\,|\,f) = F(x\,|\,g)$*.*

Proof. (a) $\Longleftrightarrow$ (b): This is a corollary of Theorem 1.16 together with (3.8).

(b) $\Longrightarrow$ (c): Since every feasible f satisfies (3.8), (b) implies $B(f) = F(x\,|\,f) = F(x\,|\,g)$ for some x. By Theorem 1.15b, this implies (c).

(c) $\Longrightarrow$ (b): By Theorem 1.15b together with (3.8), (c) implies $B(f) = F(x\,|\,f) = F(x\,|\,g)$ for some x. Statement (b) now follows from (3.8). □

We remark that for $f \in g + M^*$, deciding whether f is optimal for (3.7) is NP-complete, as we discuss later in Corollary 3.3. Although one part of Theorem 3.1 was already proved in [92, Theorem 1] (reviewed in §1.5.5, see Theorem 1.16), it is crucial to identify statement (c) in Theorem 3.1 as it has a simple but useful consequence:

Theorem 3.2. *Let* $g \in \mathbb{R}^T$*. CSP* $A^*(g)$ *is satisfiable if and only if* $B(g) \leq B(f)$ *for every* $f \in g + M^*$*.*

Proof. By Theorem 3.1, $A^*(g)$ is satisfiable if and only if (3.7) attains its optimum at the point $f = g$, i.e., $B(g) \leq B(f)$ for every $f \in g + M^*$. □

3.2 Iterative Method to Improve the Bound

In this section, we present an iterative method for approximately solving (3.7). Starting from a feasible solution to (3.7), every iteration finds a new feasible solution with a lower objective, which by (3.8) corresponds to decreasing the upper bound on the optimal value of the initial WCSP.

input: WCSP $g \in \mathbb{R}^T$.
Initialize $f^0 := g$, $k := 0$.
while CSP $A^*(f^k)$ is unsatisfiable **do**
  Find $f^{k+1} \in f^k + M^*$ such that $B(f^{k+1}) < B(f^k)$.
  $k := k + 1$
return $B(f^k)$

Algorithm 3.1: Iterative scheme for upper bounding the optimal value of WCSP g assuming the ability to decide satisfiability of a CSP.

3.2.1 Outline of the Method

Consider a WCSP f feasible for (3.7), i.e., $f \in g + M^*$. By Theorem 3.1, a necessary (but not sufficient) condition for f to be optimal for (3.7) is that CSP $A^*(f)$ is satisfiable. By Theorem 3.2, $A^*(f)$ is satisfiable if and only if $B(f) \leq B(f')$ for all $f' \in f + M^*$. In summary, we have the following implications and equivalences:

$$\begin{array}{ccc} f \text{ is optimal for (3.7)} & \Longrightarrow & \text{CSP } A^*(f) \text{ is satisfiable} \\ \Updownarrow & & \Updownarrow \\ B(f) \leq B(f') \ \ \forall f' \in g + M^* & \Longrightarrow & B(f) \leq B(f') \ \ \forall f' \in f + M^* \end{array} \tag{3.9}$$

The left-hand equivalence is just the definition of the optimum of (3.7), the right-hand equivalence is Theorem 3.2, and the top implication follows from Theorem 3.1. The bottom implication independently follows from transitivity of super-reparametrizations, which means that $f' \in f + M^*$ implies $f' \in g + M^*$ (assuming $f \in g + M^*$).

Suppose for the moment that we have an oracle that, for a given $f \in \mathbb{R}^T$ feasible for (3.7) (i.e., $f \in g + M^*$), decides if $A^*(f)$ is satisfiable and if it is not, finds some $f' \in f + M^*$ such that $B(f') < B(f)$ (which exists by Theorem 3.2). By transitivity of super-reparametrizations, such f' is also feasible for (3.7). This suggests an iterative scheme to improve feasible solutions to (3.7) that we outline in Algorithm 3.1 and analyze next. Note that transitivity of super-reparametrizations implies $f^k \in f^0 + M^*$ for every k, so every f^k is feasible for (3.7) as expected. An example of a single iteration of Algorithm 3.1 is shown in Figure 3.1a and 3.1b.

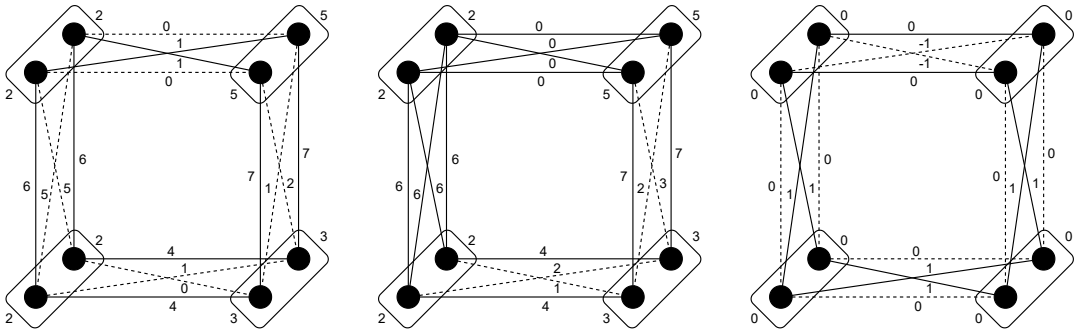
3.2.1.1 Properties of the Method

This iterative scheme can be interpreted as a local search method to (approximately) solve (3.7): having a current feasible estimate f^k, we search for the next estimate f^{k+1} with a strictly better objective within a neighborhood $f^k + M^*$ of f^k. We can define *local optima* of problem (3.7) with respect to this method to be super-reparametrizations f of g such that $A^*(f)$ is satisfiable. [51]

By transitivity of super-reparametrizations, for every k we have

$$f^{k+1} + M^* \subseteq f^k + M^* \tag{3.10}$$

[51] For clarity, we of course do *not* apply local search to improve some assignment x w.r.t. the objective of the WCSP. Instead, our variables are the components of the weight vector f and we try to improve $B(f)$ by local search. Furthermore, the local optima mentioned here are different from the ones considered in Definition 1.3.



(a) WCSP f^0, $A^*(f^0)$ unsatisfiable. (b) WCSP $f^1 \in f^0 + M^*$, $B(f^1) < B(f^0)$. (c) Certificate d of unsatisfiability of $A^*(f^0)$, $f^1 = f^0 + d$.

Figure 3.1: Example of one iteration on a pairwise WCSP whose graph $(V, C_{\geq 2})$ is a cycle of length 4.

which holds with equality if and only if $f^{k+1} \in f^k + M^\perp$ (i.e., f^{k+1} is a reparametrization of f^k). This shows that the search space of the method may shrink with increasing k, in other words, a larger and larger part of the feasible set $f^0 + M^*$ of (3.7) is cut off and becomes forever inaccessible. If, for some k, all global optima of (3.7) happen to lie in the cut-off part, the method has lost any chance to find a global optimum. This is illustrated in Figure 3.2.

This has the following consequence. By Theorems 1.15a and 3.1, every f^k satisfies

$$B(f^k) \;\geq\; \min_{f \in f^k + M^*} B(f) = \max_{x \in D^V} F(x \,|\, f^k). \tag{3.11}$$

In every iteration, the left-hand side of inequality (3.11) decreases and the right-hand side increases or stays the same due to (3.10). If both sides meet for some k, the CSP $A^*(f^k)$ becomes satisfiable by Theorem 1.15b and the method stops. Monotonic increase of the right-hand side can be seen as 'greediness' of the method: if we could choose f^{k+1} from the initial feasible set $f^0 + M^*$ rather than from its subset $f^k + M^*$, the right-hand side could also decrease. Any increase of the right-hand side is undesirable because the bounds $B(f^k)$ in future iterations will never be able to get below it. This is illustrated in Figures 3.3 and 3.4. Unlike the case of reparametrizations, note that not every optimal assignment for WCSP $f \in g + M^*$ is optimal for WCSP g. We will return to this in §3.3.

If $A^*(f^k)$ is unsatisfiable, there are usually many vectors $f^{k+1} \in f^k + M^*$ satisfying $B(f^{k+1}) < B(f^k)$. We should choose among them the one that does not cause 'too much' shrinking of the search space and/or increase of the right-hand side of (3.11). Inclusion (3.10) holds with equality if and only if $f^{k+1} \in f^k + M^\perp$, so whenever possible we should choose f^{k+1} to be a reparametrization (rather than just a super-reparametrization) of f^k. Unfortunately, we know of no other useful theoretical results to help us choose f^{k+1}, so we must recourse to heuristics. One natural heuristic is to choose f^{k+1} such that the vector $f^{k+1} - f^k$ is sparse and its positive components are small. Unfortunately, this can sometimes be too restrictive because, e.g., vectors from $M^\perp$ can be dense and their components have unbounded magnitudes. [52]

[52] To see this, recall (1.41) where one can set $g = 0$ and choose φ arbitrarily. Then, g^φ may be dense and have arbitrarily large components g^φ_t, yet $g^\varphi \in M^\perp$.

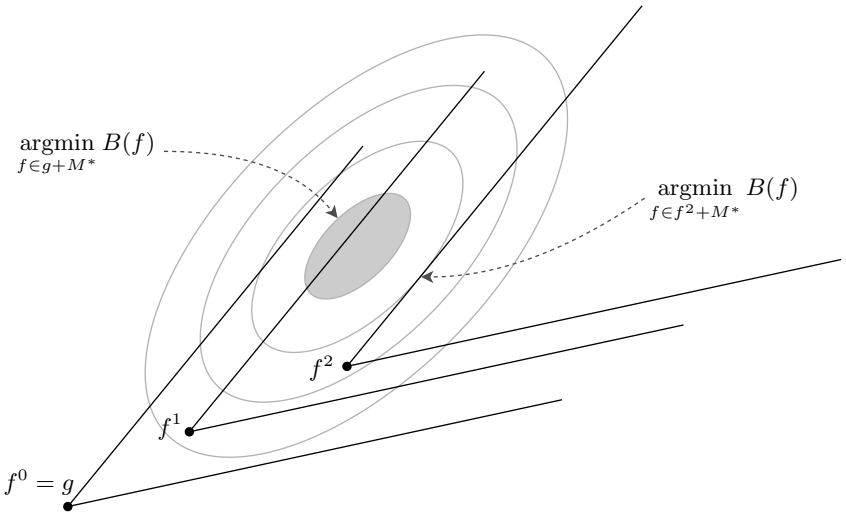


Figure 3.2: The shrinking of the search space of the iterative method. The figure illustrates the translated cones $f^i + M^*$ and several contours of the objective $B(f)$. After the second iteration, all global minima of the original problem (marked in gray) become inaccessible as the right-hand side of (3.11) increases. Note that the picture is only illustrative.

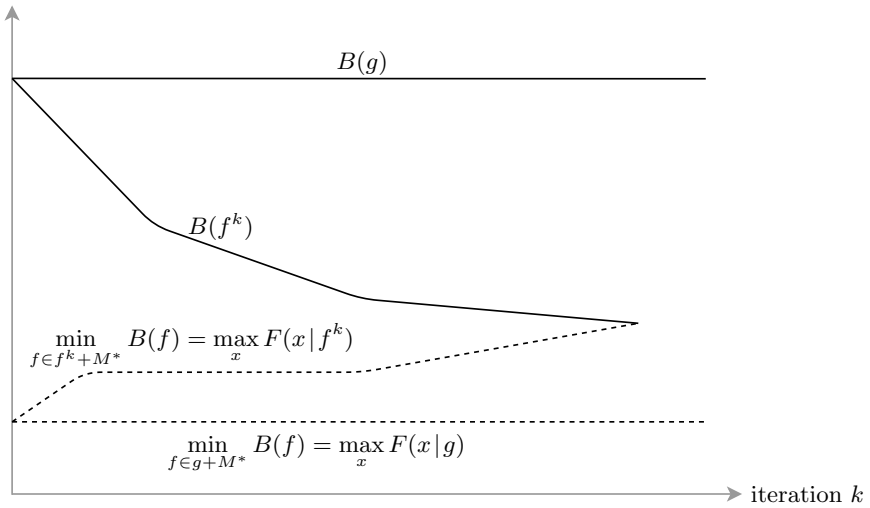
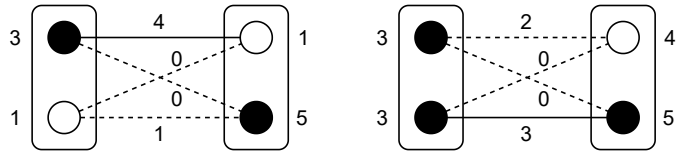


Figure 3.3: Illustration to the iterative scheme: $B(g)$ and $B(f^k)$ are shown by the full lines, $\max_x F(x\,|\,g)$ and $\max_x F(x\,|\,f^k)$ are represented by the dashed lines.



(a) WCSP g (b) WCSP f

Figure 3.4: WCSP f is a super-reparametrization of WCSP g and this pair of WCSPs satisfies $B(f) = 11 < B(g) = 12$ and $\max_{x \in D^V} F(x|f) = 11 > \max_{x \in D^V} F(x|g) = 8$. Assignment $x = (\mathsf{b}, \mathsf{b})$ (where b is the lower value, as in Figure 1.6) is not optimal for g even though $B(f) = F(x|f)$.

input: WCSP $g \in \mathbb{R}^T$.
1 Initialize $f^0 := g$, $k := 0$.
2 **repeat**
3 Try to prove that CSP $A^*(f^k)$ is unsatisfiable (e.g., using constraint propagation).
4 **if** CSP $A^*(f^k)$ is proved to be unsatisfiable **then**
5 Find $f^{k+1} \in f^k + M^*$ such that $B(f^{k+1}) < B(f^k)$.
6 $k := k + 1$
7 **else**
8 **return** $B(f^k)$

Algorithm 3.2: Iterative scheme for upper bounding the optimal value of WCSP g using constraint propagation.

3.2.1.2 Employing Constraint Propagation

So far, we assumed that we can always decide if CSP $A^*(f)$ is satisfiable. This is unrealistic because the CSP is NP-complete. Yet the approach remains applicable even if we detect unsatisfiability of $A^*(f)$ only sometimes, e.g., using constraint propagation. We outline this procedure in Algorithm 3.2. In this case, stopping points of the method will be even weaker local minima of (3.7), but they nevertheless might be still non-trivial and useful.

In the sequel, we develop this approach in detail. In particular, we show, if $A^*(f^k)$ is unsatisfiable, how to find a vector $f^{k+1} \in f^k + M^*$ satisfying $B(f^{k+1}) < B(f^k)$. We will do it in two steps. First (in §3.2.2), given the CSP $A^*(f^k)$ we find a direction $d \in M^*$ using constraint propagation. This direction is a *certificate of unsatisfiability* of the CSP $A^*(f^k)$ and, at the same time, an improving direction for (3.7). Second (in §3.2.3), given d and f^k, we find a step size $\alpha > 0$ such that $f^{k+1} = f^k + \alpha d$ and $B(f^{k+1}) < B(f^k)$. An example of such a certificate of unsatisfiability is shown in Figure 3.1c.

3.2.1.3 Relation to Existing Approaches

The Augmenting DAG algorithm [95, 146] and the VAC algorithm [33] are (up to the precise way of computing certificates d and step sizes α) an example of the described approach, which uses arc consistency to prove unsatisfiability of $A^*(f^k)$. In this favorable case, there exist certificates $d \in M^\perp$, so we are, in fact, approximately solving (3.2) rather than (3.7). Such certificates do not generally exist for stronger local consistencies (i.e., inevitably $F(x|d) > 0$ for some x).

The algorithm proposed in [92] can be also seen as an example of our approach. It interleaves iterations using arc consistency (in fact, the Augmenting DAG algorithm) and iterations using cycle consistency.

As an alternative to our approach, stronger local consistencies can be achieved by introducing new weight functions (of possibly higher arity) into the WCSP objective (1.33) and minimizing an upper bound over reparametrizations, as in [127, 11, 149, 147, 107]. In our particular case, after each update $f^{k+1} = f^k + \alpha d$ we could introduce [53] a new weight function with scope

$$S' = \bigcup \{ S \mid (S,k) \in T,\ d_S(k) \neq 0 \} \tag{3.12}$$

and weights

$$f_{S'}(k) = -\alpha \sum_{\substack{S \in C \\ S \subseteq S'}} d_S\big(k|_S\big) \tag{3.13}$$

where $k \in D^{S'}$. In this view, our approach can be seen as enforcing stronger local consistencies but omitting these compensatory higher-order weight functions, thus saving memory.

Finally, the described approach is an example of the iterative scheme to optimize linear programs using constraint propagation from §2.2. In this particular case, if (3.7) is formulated as a linear program, then the complementary slackness conditions (expressed in terms of the dual variables) can be interpreted as the optimality condition (c) stated in Theorem 3.1 expressed as a set of linear equalities with an exponential number of non-negative variables (we will discuss this in detail later in §5.2). Applying constraint propagation to this system is in correspondence with constraint propagation in a CSP.

3.2.2 Certificates of Unsatisfiability of CSP

Recall from §1.4.1 that constraint propagation [54] is an iterative algorithm, which in each iteration (executed by a propagator) infers that some allowed tuples $R \subseteq A$ of a current CSP $A \subseteq T$ can be forbidden without changing its solution set, i.e., $\mathrm{SOL}(A) = \mathrm{SOL}(A - R)$, and forbids these tuples, i.e., sets $A := A - R$. The algorithm terminates when it is no longer able to forbid any tuples or when it becomes explicit that the current CSP is unsatisfiable. The former usually happens when the CSP satisfies some desired local consistency Φ. The latter happens if $A \cap T_S = \emptyset$ for some $S \in C$, which implies unsatisfiability of A. [55]

In this section, we show how to augment constraint propagation so that if it proves a CSP unsatisfiable, it also provides its *certificate of unsatisfiability* $d \in M^*$. This certificate is needed as an improving direction for (3.7), as was mentioned in §3.2.1.2. First, in §3.2.2.1, we introduce a more general concept, *deactivating directions*. One iteration of constraint propagation constructs an R-deactivating direction for the current CSP A,

[53] Notice that such an added weight function would not increase the bound (1.39) since its weights are non-positive due to the fact that it needs to decrease the objective value of some assignments.

[54] We speak only about constraint propagation but the approach outlined in this section is applicable to any method that proves unsatisfiability of a CSP by iteratively forbidding subsets of tuples. In theory, as a stronger alternative one could also use any CSP solver that is augmented to provide a certificate of unsatisfiability (which is always possible, as we will discuss later in this section).

[55] This is because, as stated in §1.4, each assignment uses exactly one tuple from each scope. In addition, if $A \cap T_S = \emptyset$ for some $S \in C$ with $|S| = 1$, this is usually called domain wipeout [33, 31].

which certifies that $\mathrm{SOL}(A) = \mathrm{SOL}(A - R)$. Then, in §3.2.2.2, we show how to compose the deactivating directions obtained from individual iterations of constraint propagation into a single deactivating direction for the initial CSP. If the initial CSP has been proved unsatisfiable by the propagation, this composed deactivating direction is then its certificate of unsatisfiability.

Remark 3.2. *Deactivating directions correspond to cause vectors from §1.1.3 and §2. Indeed, cause vectors certify that a linear inequality is implied by a system of linear inequalities, whereas deactivating directions certify that some tuples can be forbidden in a CSP while its solution set is preserved. As in §2, where cause vectors were combined to compute an improving direction, deactivating directions will be composed to obtain an improving direction here. Although it is also possible to interpret deactivating directions as cause vectors of certain equalities (implied by an LP formulation of a CSP), using CSP terminology without referring to systems of linear (in)equalities simplifies the explanation given here.*

3.2.2.1 Deactivating Directions

Definition 3.1. *Let $A \subseteq T$ and $R \subseteq A$, $R \neq \emptyset$. An* R-deactivating direction for CSP A *is a vector $d \in M^*$ satisfying*

(a) $d_t < 0$ for all $t \in R$,

(b) $d_t = 0$ for all $t \in A - R$.

For fixed A and R, all R-deactivating directions for A form a convex cone. Moreover, note that if $A \subseteq A' \subseteq T$ and d is an R-deactivating direction for A', then d is an R-deactivating direction also for A. Taking this observation into account, the following result shows a way how to obtain a particular class of R-deactivating directions for A:

Theorem 3.3. *Let $R \subseteq A \subseteq A' \subseteq T$ be such that $\mathrm{SOL}(A') = \mathrm{SOL}(A' - R)$ and $R \neq \emptyset$. Denote*[56] *$\delta = |\{\, S \in C \mid T_S \cap R \neq \emptyset \,\}|$. Then, vector $d \in \mathbb{R}^T$ defined by*

$$d_t = \begin{cases} -1 & \text{if } t \in R \\ \delta & \text{if } t \in T - A' \\ 0 & \text{otherwise (i.e., } t \in A' - R\text{)} \end{cases} \qquad \forall t \in T \tag{3.14}$$

is an R-deactivating direction for A.

Proof. Conditions (a) and (b) of Definition 3.1 are clearly satisfied, so it only remains to show that $d \in M^*$. We have

$$F(x \,|\, d) = \sum_{S \in C} d_S\big(x|_S\big) = \sum_{\substack{S \in C \\ (S, x|_S) \in R}} -1 + \sum_{\substack{S \in C \\ (S, x|_S) \in T - A'}} \delta = -n_1(x) + \delta n_2(x) \tag{3.15}$$

where $n_1(x) = |\{\, S \in C \mid \big(S, x|_S\big) \in R \,\}|$ and $n_2(x) = |\{\, S \in C \mid \big(S, x|_S\big) \in T - A' \,\}|$.

[56] The quantity δ is the number of scopes S such that T_S contains at least one tuple from R. In other words, for every assignment $x \in D^V$, $\big(S, x|_S\big) \in R$ holds for at most δ scopes. We remark that the value of δ could be in some cases decreased while (3.14) remains an R-deactivating direction, thus decreasing also the objective values $F(x \,|\, d)$. However, deciding whether (3.14) is not an R-deactivating direction for A for a given value δ is an NP-complete problem (see Theorem 3.13).

For contradiction, let $x \in D^V$ satisfy $F(x|d) < 0$. This implies $n_1(x) > 0$ and $n_2(x) = 0$, where the latter is because $n_1(x) \leq \delta$ by the definition of δ. That is, we have $\left(S^*, x|_{S^*}\right) \in R$ for some $S^* \in C$ and $\left(S, x|_S\right) \in A'$ for all $S \in C$. But the latter means $x \in \mathrm{SOL}(A')$ and the former implies $x \notin \mathrm{SOL}(A' - R)$, a contradiction. □

Theorem 3.4. *Let $A \subseteq T$ and $R \subseteq A$. If there exists an R-deactivating direction for A, then* $\mathrm{SOL}(A) = \mathrm{SOL}(A - R)$.

Proof. Recall from §1.4 that SOL is an isotone mapping, so $\mathrm{SOL}(A) = \mathrm{SOL}(A - R)$ is equivalent to $\mathrm{SOL}(A) \subseteq \mathrm{SOL}(A - R)$ because forbidding tuples may only remove solutions.

By contradiction: let d be an R-deactivating direction for A and let $x \in \mathrm{SOL}(A) - \mathrm{SOL}(A - R)$, so $\left(S, x|_S\right) \in R$ for some $S \in C$. By (1.33), we have $F(x|d) < 0$ because $d_S\left(x|_S\right) = 0$ for all $\left(S, x|_S\right) \in A - R$ by condition (b) in Definition 3.1 and $d_S\left(x|_S\right) < 0$ for all $\left(S, x|_S\right) \in R$ by condition (a). This contradicts $d \in M^*$. □

Combining Theorem 3.3 (with $A' = A$) and Theorem 3.4 yields the following result: for any $R \subseteq A$ with $R \neq \emptyset$, an R-deactivating direction for A exists if and only if $\mathrm{SOL}(A) = \mathrm{SOL}(A - R)$. Thus, any R-deactivating direction for A is a *certificate* of $\mathrm{SOL}(A) = \mathrm{SOL}(A - R)$.

Unfortunately, vectors d calculated naively from (3.14) with $A' = A$ can have many positive components, which is undesirable as we explained earlier in §3.2.1.1. However, we are allowed to have $A' \supseteq A$ in Theorem 3.3, which gives us some freedom in choosing the deactivating direction. In particular, (3.14) shows that larger sets A' give rise to sparser vectors d – more precisely, vectors d with fewer positive components. This offers us a possibility to obtain a sparser deactivating direction if we can provide a superset $A' \supseteq A$ of the allowed tuples satisfying $\mathrm{SOL}(A') = \mathrm{SOL}(A' - R)$.

Given $A \subseteq T$ and $R \subseteq A$, finding a maximal (w.r.t. the partial ordering by set inclusion) superset $A' \supseteq A$ such that $\mathrm{SOL}(A') = \mathrm{SOL}(A' - R)$ is closely related to finding a minimal unsatisfiable core and minimally unsatisfiable set of tuples [57] of an unsatisfiable CSP. While finding a maximal such subset is likely intractable (see [70, 71]), for obtaining a 'sparse enough' vector d it suffices to find a 'large enough' such superset A'. Such a superset is often cheaply available as a side result of executing the propagator. Namely, we take $A' = T - P$ where P is the set of forbidden tuples that were visited during the run of the propagator. Clearly, tuples not visited by the propagator could not be needed to infer $\mathrm{SOL}(A) = \mathrm{SOL}(A - R)$. Note that P need not be the same for each CSP instance, even for a fixed level of local consistency: for example, if the AC closure of A is empty, then A is unsatisfiable but a domain wipeout may occur sooner or later depending on A, which affects which tuples needed to be visited.

Let us emphasize that an R-deactivating direction for A need not be always obtained using formula (3.14), any other method can be used as long as d satisfies Definition 3.1. We will now give examples of deactivating directions corresponding to some popular constraint propagation rules.

[57] Given an unsatisfiable CSP $A \subseteq T$, finding a maximal set $A' \supseteq A$ such that A' is still unsatisfiable corresponds to finding a minimally unsatisfiable set of tuples [70]. This is a finer-grained (tuple-based rather than constraint-based) version of finding a minimal unsatisfiable core of a CSP [71]. Note that we are looking here for a *maximal* superset A' in contrast to a *minimal* unsatisfiable core/set of tuples because we define CSP instances by *allowed* tuples while cores are CSP instances defined by *forbidden* tuples.

Example 3.1. *As already stated in §1.4.1.2, CSP A is arc consistent if for all $S \in C_{\geq 2}$, $i \in S$, and $k \in D$, we have the equivalence*

$$(\{i\}, k) \in A \iff \exists (S, \ell) \in A : \ell_i = k. \tag{3.16}$$

If, for some $S \in C_{\geq 2}$, $i \in S$, and $k \in D$, the left-hand statement in (3.16) *is true and the right-hand statement is false, AC propagator* (1.32) *infers* $\mathrm{SOL}(A) = \mathrm{SOL}(A - R)$ *where $R = \{(\{i\}, k)\}$. To infer this, it suffices to know that the tuples $P = \{\, (S, \ell) \mid \ell \in D^S,\ \ell_i = k \,\}$ are all forbidden. An R-deactivating direction d for A can be chosen as in* (3.14)*, where $\delta = |\{\, S' \in C \mid T_{S'} \cap R \neq \emptyset \,\}| = 1$ and $A' = T - P$. Note that then we have $d \in M^\perp$.*

If the left-hand statement in (3.16) *is false and the right-hand statement is true, AC propagator* (1.32) *infers* $\mathrm{SOL}(A) = \mathrm{SOL}(A - R)$ *where $R = \{\, (S, \ell) \mid \ell \in D^S,\ \ell_i = k \,\} \cap A$. To infer this, it suffices to know that the tuple $P = \{(\{i\}, k)\}$ is forbidden. In this particular case, rather than using* (3.14) *it is better to choose d as*

$$d_t = \begin{cases} -1 & \text{if } t \in \{\, (S, \ell) \mid \ell \in D^S,\ \ell_i = k \,\} \\ 1 & \text{if } t = (\{i\}, k) \\ 0 & \text{otherwise} \end{cases} \qquad \forall t \in T. \tag{3.17}$$

Vector (3.17) *satisfies $d \in M^\perp$, in contrast to vector* (3.14) *which satisfies only $d \in M^*$. Thus, the update $f^{k+1} = f^k + \alpha d$ is a mere reparametrization*[58]*, which is desirable as explained in §3.2.1.1.* △

Example 3.2. *We now consider* cycle consistency *as defined in [92].*[59] *As this local consistency was defined only for pairwise CSPs, we assume that $|S| \leq 2$ for each $S \in C$. Let $\mathcal{L}$ be a (polynomially sized) set of cycles in the graph $(V, C_{\geq 2})$. A CSP A is cycle consistent if for each tuple $(\{i\}, k) \in A$ (where $i \in V$ and $k \in D$) and each cycle $L \in \mathcal{L}$ that passes through node $i \in V$, there exists an assignment x with $x_i = k$ that uses only allowed tuples in cycle L. It can be shown that the cycle-repair procedure in [92] constructs a deactivating direction whenever an inconsistent cycle is found. Moreover, the constructed direction in this case coincides with* (3.14) *for a suitable set P (i.e., $A' = T - P$) which contains a subset of the forbidden tuples within the cycle.* △

Example 3.3. *Recall that a CSP A is* singleton arc consistent *if for every tuple $t = (\{i\}, k) \in A$ (where $i \in V$ and $k \in D$), the CSP*[60] *$A|_{x_i = k} = A - (T_{\{i\}} - \{t\})$ has non-empty AC closure. Good (i.e., sparse) deactivating directions for singleton arc consistency (SAC) can be obtained as follows. For some $(\{i\}, k) \in A$, we enforce arc consistency of CSP $A|_{x_i = k}$, during which we store the causes for forbidding each tuple. If $A|_{x_i = k}$ is found to have empty AC closure, we trace back the AC operations and identify only those tuples that were necessary to prove the empty AC closure. These tuples form the set P. The deactivating direction is then constructed as in Theorem 3.3 with $R = \{(\{i\}, k)\}$ and $A' = T - P$. Note that SAC does not have bounded support [19] as many other local consistencies do, so the size of P can significantly vary for different CSP instances.* △

[58] Such reparametrizations correspond to soft arc consistency operations *extend* and *project* that we mentioned in Remark 1.12.

[59] This is different from *cyclic consistency* as defined in [34]. E.g., reparametrizations are sufficient to enforce cyclic consistency, whereas super-reparametrizations are needed for cycle consistency.

[60] This can be also stated as $A|_{x_i = k} = A - \{\, (\{i\}, k') \mid k' \in D - \{k\} \,\}$. In other words, the solutions of CSP $A|_{x_i = k}$ are the solutions x of CSP A satisfying $x_i = k$. This notation is used, e.g., in [18, 99].

```
1 procedure (S, (R_i)_{i=0}^n, (d^i)_{i=0}^n) = propagate(A)
2 Initialize n := 0, A_0 := A.
3 while A_n is not Φ-consistent do
4 |   Find a set R_n ⊆ A_n and an R_n-deactivating direction d^n for A_n.
5 |   A_{n+1} := A_n − R_n
6 |   if ∃S ∈ C: A_{n+1} ∩ T_S = ∅ then
7 |   |   return (S, (R_i)_{i=0}^n, (d^i)_{i=0}^n)
8 |   n := n + 1
9 return (∅, (R_i)_{i=0}^{n−1}, (d^i)_{i=0}^{n−1}))
```

Algorithm 3.3: The procedure `propagate` applies constraint propagation to CSP $A \subseteq T$ and returns the sequence $(R_i)_{i=0}^n$ of tuple sets that were forbidden and the corresponding deactivating directions $(d^i)_{i=0}^n$. If all tuples in some scope $S \in C$ become forbidden during propagation, `propagate` returns also S, otherwise it returns $S = \emptyset$.

3.2.2.2 Composing Deactivating Directions

Consider now a propagator which, for a current CSP $A \subseteq T$, returns a set $R \subseteq A$ such that $\mathrm{SOL}(A) = \mathrm{SOL}(A - R)$ and an R-deactivating direction for A. This propagator is applied iteratively, each time forbidding a different set of tuples, until the current CSP achieves the desired local consistency Φ or it becomes explicit that the CSP is unsatisfiable (due to $A \cap T_S = \emptyset$ for some $S \in C$). This is outlined in Algorithm 3.3, which stores the generated sets R_i of tuples being forbidden and the corresponding R_i-deactivating directions d^i. By line 5 of the algorithm, we have $A_i = A - \bigcup_{j=0}^{i-1} R_j$ for every $i \in \{0, \ldots, n+1\}$. Therefore, by Theorem 3.4, we have $\mathrm{SOL}(A) = \mathrm{SOL}(A_1) = \mathrm{SOL}(A_2) = \ldots = \mathrm{SOL}(A_{n+1})$, which implies that if A_{n+1} is unsatisfiable, then so is A. Note that Algorithm 3.3 is an extension of the previously shown Algorithm 1.2 that did not use deactivating directions.

Next, we show how to compose the generated sequence of R_i-deactivating directions d^i for A_i into a single $\left(\bigcup_{i=0}^n R_i\right)$-deactivating direction for A. This can be done using the following composition rule:

Proposition 3.1. *Let $A \subseteq T$ and $R, R' \subseteq A$ where $R \cap R' = \emptyset$. Let d be an R-deactivating direction for A. Let d' be an R'-deactivating direction for $A - R$. Let*

$$\delta = \begin{cases} 0 & \text{if } d'_t \leq -1 \text{ for all } t \in R, \\ \max\{\, (-1 - d'_t)/d_t \mid t \in R,\ d'_t > -1 \,\} & \text{otherwise.} \end{cases} \tag{3.18}$$

Then $d'' = d' + \delta d$ is an $(R \cup R')$-deactivating direction for A.

Proof. First, if $d'_t \leq -1$ for all $t \in R$, then $d'' = d'$ satisfies the required condition immediately. Otherwise, $\delta > 0$ since $d_t < 0$ for all $t \in R$ by definition and $-1 - d'_t < 0$ due to $d'_t > -1$ in the definition of δ. We will show that d'' satisfies the conditions in Definition 3.1.

For $t \in R$ with $d'_t \leq -1$, $d''_t = d'_t + \delta d_t < d'_t \leq -1$ because $\delta d_t < 0$. If $t \in R$ and $d'_t > -1$, then $\delta \geq (-1 - d'_t)/d_t$, so $d''_t = d'_t + \delta d_t \leq -1$. Summarizing, we have $d''_t < 0$ for all $t \in R$.

```
1 procedure (R*, d*) = compose((R_i)_{i=0}^n, (d^i)_{i=0}^n, I)
2 Initialize i := max I, d* := d^i, R* := R_i.
3 while i > 0 do
4 |   i := i − 1
5 |   if i ∈ I or ∃t ∈ R_i: d*_t ≠ 0 then
6 |   |   d* := d* + δd^i (where δ is given by (3.18) with d, d', R replaced by d^i, d*, R_i)
7 |   |   R* := R* ∪ R_i
8 return (R*, d*)
```

Algorithm 3.4: The procedure `compose` takes the sequences $(R_i)_{i=0}^n$ and $(d^i)_{i=0}^n$ (generated by the procedure `propagate` in Algorithm 3.3) and a non-empty index set $I \subseteq \{0, \dots, n\}$ and composes them into an R^*-deactivating direction d^* for A.

For $t \in R'$, $d'_t < 0$ and $d_t = 0$ holds by definition due to $R' \subseteq A - R$, thus $d''_t = d'_t + \delta d_t = d'_t < 0$ which together with the previous paragraph yields condition (a).

Due to $A - R \supseteq (A - R) - R' = A - (R \cup R')$, for any $t \in A - (R \cup R')$ we have $d_t = 0$ and $d'_t = 0$, which implies $d''_t = d' + \delta d = 0$, thus verifying condition (b).

Finally, we have $d'' \in M^*$ because $d, d' \in M^*$ and $\delta \geq 0$. □

Proposition 3.1 allows us to combine R_i-deactivating direction d^i for $A_i = A_{i-1} - R_{i-1}$ with R_{i-1}-deactivating direction d^{i-1} for A_{i-1} into a single $(R_{i-1} \cup R_i)$-deactivating direction for A_{i-1}. Iteratively, we can thus gradually build a $\big(\bigcup_{i=0}^n R_i\big)$-deactivating direction for A, which certifies unsatisfiability of A whenever Algorithm 3.3 detects on line 6 that A_{n+1} (and thus also A) is unsatisfiable.

However, it is not always necessary to construct a full $\big(\bigcup_{i=0}^n R_i\big)$-deactivating direction because not every iteration of constraint propagation may have been necessary to prove unsatisfiability of A. Instead, we can use the scope $S \in C$ satisfying $A_{n+1} \cap T_S = \emptyset$ (where $A_{n+1} = A - \bigcup_{i=0}^n R_i$, as mentioned above) returned by Algorithm 3.3 on line 7 and construct an R^*-deactivating direction d^* for a (usually smaller) set $R^* \subseteq \bigcup_{i=0}^n R_i$ such that $(A - R^*) \cap T_S = \emptyset$. Such a direction d^* still certifies unsatisfiability of A and can be sparser and/or may have lower objective values $F(x \,|\, d^*)$ than a $\big(\bigcup_{i=0}^n R_i\big)$-deactivating direction, which is desirable as explained in §3.2.1.1.

This is outlined in Algorithm 3.4, which composes only a subsequence of directions d^i based on a given set of indices $I \subseteq \{0, \dots, n\}$ and constructs an R^*-deactivating direction with $R^* \supseteq \bigcup_{i \in I} R_i$. Although Algorithm 3.4 is applicable to any set I, in our case I is obtained by taking a scope $S \in C$ such that $A_{n+1} \cap T_S = \emptyset$ and then setting

$$I = \{\, i \in \{0, \dots, n\} \mid R_i \cap T_S \neq \emptyset \,\} \tag{3.19}$$

so that $(A - R^*) \cap T_S = \emptyset$ due to the following fact:

Proposition 3.2. *Let $S \in C$ be such that $(A - \bigcup_{i=0}^n R_i) \cap T_S = \emptyset$. Let I be given by* (3.19)*. Then, $(A - \bigcup_{i \in I} R_i) \cap T_S = \emptyset$.*

Proof. For any sets $A, R, T' \subseteq T$, we have $(A - R) \cap T' = (T' - R) \cap A$. In particular, $(A - \bigcup_{i=0}^n R_i) \cap T_S = (T_S - \bigcup_{i=0}^n R_i) \cap A$. But $T_S - \bigcup_{i=0}^n R_i = T_S - \bigcup_{i \in I} R_i$ because for each $i \notin I$ we have $R_i \cap T_S = \emptyset$ which is equivalent to $T_S - R_i = T_S$. □

Correctness of Algorithm 3.4 is given by the following theorem:

Theorem 3.5. *Algorithm 3.4 returns an R^*-deactivating direction d^* for A such that $\bigcup_{i\in I} R_i \subseteq R^* \subseteq \bigcup_{i=0}^{n} R_i$.*

Proof. The fact that $R^* \supseteq \bigcup_{i\in I} R_i$ is obvious due to $R_{\max I} \subseteq R^*$ by initialization on line 2 and $R_i \subseteq R^*$ for any $i \in I$ such that $i < \max I$ because in such case the update on line 7 is performed. Similarly, $R^* \subseteq \bigcup_{i=0}^{n} R_i$ holds by initialization of R^* on line 2 and updates on line 7.

It remains to show that d^* is an R^*-deactivating direction for A, which will be done by induction. We claim that vector d^* is always an R^*-deactivating direction for A_i on line 3 and an R^*-deactivating direction for A_{i+1} on line 5.

Initially, we have $d^* = d^i$, so d^* is R_i-deactivating (i.e., R^*-deactivating since $R^* = R_i$ before the loop is entered) for A_i. Also, when vector d^* is first queried on line 5, i decreased by 1 due to the update on line 4, so d^* is R^*-deactivating for A_{i+1}. The required property thus holds when the condition on line 5 is first queried with $i = \max I - 1$.

We proceed with the inductive step. If the condition on line 5 is not satisfied, then necessarily $d^*_t = 0$ for all $t \in R_i$. So, if d^* is R^*-deactivating for A_{i+1}, then it is also R^*-deactivating for $A_i = A_{i+1} \cup R_i$, as seen from Definition 3.1.

If the condition on line 5 is satisfied, d^* is R^*-deactivating for A_{i+1} before the update on lines 6-7. Since $A_{i+1} = A_i - R_i$ and d^i is R_i-deactivating for A_i, Proposition 3.1 can be applied to d^i and d^* to obtain an $(R^* \cup R_i)$-deactivating direction for A_i. After updating R^* on line 7, it becomes R^*-deactivating for A_i.

When eventually $i = 0$, d^* is R^*-deactivating for $A_0 = A$ by line 2 in Algorithm 3.3. □

Remark 3.3. *This is similar to what the VAC [33] or Augmenting DAG algorithm [95, 146] do for arc consistency. To attempt to disprove satisfiability of CSP $A^*(f)$, these algorithms enforce AC of $A^*(f)$, during which the causes for forbidding tuples are stored. If empty AC closure of $A^*(f)$ is detected (which corresponds to $T_S \cap A_{n+1} = \emptyset$ for some $S \in C$), these algorithms do not iterate through all previously forbidden tuples but only trace back the causes for forbidding the elements of the wiped-out domain (here, the elements of T_S).*

3.2.3 Line Search

In §3.2.2, we showed how to construct an R-deactivating direction d for CSP A, which certifies unsatisfiability of A whenever $(A - R) \cap T_S = \emptyset$ for some $S \in C$. Given a WCSP $f \in \mathbb{R}^T$ with $A^*(f) = A$, to obtain $f' \in f + M^*$ with $B(f') < B(f)$ (as in Theorem 3.2), we need to find a step size $\alpha > 0$ so that $f' = f + \alpha d$, as discussed in §3.2.1.2. That means, we need to find $\alpha > 0$ such that $B(f + \alpha d) < B(f)$.

Finding the best step size (i.e., exact line search) would require finding a global minimum of the univariate convex piecewise-affine function $\alpha \mapsto B(f + \alpha d)$. As this would be too expensive for large WCSP instances, we perform only approximate line search, i.e., find some non-zero step size α by the following theorem. [61]

[61] In detail, the step size $\min\{\beta, \gamma\}$ computed in Theorem 3.6c corresponds to the first breakpoint of the univariate function with a lower objective. Similarly as in §2.4.2, this is analogous to the *first-hit strategy* in [48a, §3.1.4].

Theorem 3.6. *Let $f \in \mathbb{R}^T$. Let d be an R-deactivating direction for $A^*(f)$. Denote*[62]

$$\beta = \min\left\{ \frac{\max_{t\in T_{S'}} f_t - f_{t'}}{d_{t'}} \;\middle|\; S' \in C,\ t' \in T_{S'},\ d_{t'} > 0 \right\},$$
$$\gamma = \min\left\{ \frac{f_t - f_{t'}}{d_{t'} - d_t} \;\middle|\; S \in C,\ (A^*(f) - R) \cap T_S = \emptyset,\ t \in T_S \cap R,\ t' \in T_S - R,\ d_{t'} > d_t \right\}.$$

Then, $\beta, \gamma > 0$ and for every $S \in C$ and $\alpha \in \mathbb{R}$, WCSP $f' = f + \alpha d$ satisfies:

(a) If $(A^*(f) - R) \cap T_S \neq \emptyset$ and $0 \leq \alpha \leq \beta$, then $\max_{t\in T_S} f'_t = \max_{t\in T_S} f_t$.

(b) If $(A^*(f) - R) \cap T_S \neq \emptyset$ and $0 < \alpha < \beta$, then $A^*(f') \cap T_S = (A^*(f) - R) \cap T_S$.

(c) If $(A^*(f) - R) \cap T_S = \emptyset$ and $0 < \alpha \leq \min\{\beta, \gamma\}$, then $\max_{t\in T_S} f'_t < \max_{t\in T_S} f_t$.

Proof. We have $\beta > 0$ because $d_{t'} > 0$ implies that t' is an inactive tuple, so $\max_{t\in T_S} f_t > f_{t'}$. We have $\gamma > 0$ because in $f_t - f_{t'}$ tuple t is always active and t' is inactive, hence $f_t > f_{t'}$.

To prove (a), let $t^* \in (A^*(f) - R) \cap T_S$. Hence, by Definition 3.1, $d_{t^*} = 0$ and the value $\max_{t\in T_S} f'_t$ does not decrease for any α since $f'_{t^*} = f_{t^*} + \alpha d_{t^*} = f_{t^*}$. To show the maximum does not increase, consider a tuple $t' \in T_S$ such that $d_{t'} > 0$ (due to $\alpha \geq 0$, tuples with $d_{t'} \leq 0$ cannot increase the maximum). It follows that $\alpha \leq \beta \leq \frac{\max_{t\in T_S} f_t - f_{t'}}{d_{t'}}$, so $f'_{t'} = f_{t'} + \alpha d_{t'} \leq \max_{t\in T_S} f_t$.

To prove (b), let $(A^*(f) - R) \cap T_S \neq \emptyset$. As in (a), we have $\max_{t\in T_S} f_t = \max_{t\in T_S} f'_t$. If $t \in (A^*(f) - R) \cap T_S$, then $d_t = 0$ and such tuples remain active by $f'_t = f_t$. Tuples $t \in R \cap T_S$ become inactive since $f'_t = f_t + \alpha d_t < f_t = \max_{t'\in T_S} f_{t'}$ by $d_t < 0$ and $\alpha > 0$. Tuples $t \notin A^*(f)$ either satisfy $d_t \leq 0$ and cannot become active or satisfy $d_t > 0$ and by $\alpha < \beta \leq \frac{\max_{t'\in T_S} f_{t'} - f_t}{d_t}$, $f'_t = f_t + \alpha d_t < \max_{t'\in T_S} f_{t'}$, so $t \notin A^*(f')$.

To prove (c), let $(A^*(f) - R) \cap T_S = \emptyset$. For all $t \in T_S \cap R$, we have $f'_t = f_t + \alpha d_t < f_t$ by $d_t < 0$ and $\alpha > 0$, i.e., $\max_{t\in T_S\cap R} f'_t < \max_{t\in T_S\cap R} f_t$. We proceed to show that $f'_{t'} \leq \max_{t\in T_S\cap R} f'_t$ for every $t' \in T_S - R$. Let $t^* \in T_S \cap R$ satisfy $f'_{t^*} = \max_{t\in T_S\cap R} f'_t$. If $d_{t'} > d_{t^*}$, $\alpha \leq \gamma \leq \frac{f_{t^*} - f_{t'}}{d_{t'} - d_{t^*}}$ implies $f'_{t^*} = f_{t^*} + \alpha d_{t^*} \geq f_{t'} + \alpha d_{t'} = f'_{t'}$. If $d_{t'} \leq d_{t^*}$, then also $\alpha d_{t'} \leq \alpha d_{t^*}$ and $f'_{t'} = f_{t'} + \alpha d_{t'} \leq f_{t^*} + \alpha d_{t^*} = f'_{t^*}$ holds for any $\alpha \geq 0$ since $f_{t'} < f_{t^*}$. As a result, $\max_{t'\in T_S - R} f'_{t'} \leq \max_{t\in T_S\cap R} f'_t < \max_{t\in T_S\cap R} f_t = \max_{t\in T_S} f_t$. □

If d is an R-deactivating direction for CSP $A^*(f)$ and $(A^*(f) - R) \cap T_S \neq \emptyset$ for all $S \in C$, then there is $\alpha > 0$ such that $f' = f + \alpha d$ satisfies $B(f') = B(f)$ and $A^*(f') = A^*(f) - R$ by Theorem 3.6a and 3.6b. This justifies why such a direction d is called R-deactivating: a suitable update of f along this direction makes tuples R inactive.

Remark 3.4. *This might suggest that to improve the current bound $B(f)$, we need not use Algorithm 3.4 to construct an R^*-deactivating direction d^* with $(A^*(f) - R^*) \cap T_S = \emptyset$ for some $S \in C$, but instead, perform steps using the intermediate R_i-deactivating directions d^i to create a sequence $f^{i+1} = f^i + \alpha_i d^i$ satisfying $B(f^0) = B(f^1) = \ldots = B(f^n) > B(f^{n+1})$. Unfortunately, it is hard to make this work reliably as there are many choices for the intermediate step sizes $0 < \alpha_i < \beta_i$. We empirically found Algorithm 3.5 to be preferable.*

[62] β is always defined: by Definition 3.1 we have $F(x \mid d) \geq 0$ for all x, hence $\exists t\colon d_t < 0 \implies \exists t'\colon d_{t'} > 0$. γ is defined and needed only in (c), where we assume that $(A^*(f) - R) \cap T_S = \emptyset$ for some $S \in C$. If the set in the definition of γ is empty, then $\gamma = +\infty$ by convention and $\min\{\beta, \gamma\} = \beta$.

input: WCSP $g \in \mathbb{R}^T$.
1 Initialize $f := g$.
2 **repeat**
3 $\quad (S, (R_i)_{i=0}^n, (d^i)_{i=0}^n) := \texttt{propagate}(A^*(f))$ (see Algorithm 3.3)
4 $\quad$ **if** $S \neq \emptyset$ **then**
5 $\quad\quad$ Define I as in (3.19).
6 $\quad\quad$ $(R^*, d^*) := \texttt{compose}((R_i)_{i=0}^n, (d^i)_{i=0}^n, I)$ (see Algorithm 3.4)
7 $\quad\quad$ Update $f := f + \min\{\beta, \gamma\} d^*$ following Theorem 3.6.
8 $\quad$ **else**
9 $\quad\quad$ **return** $B(f)$

Algorithm 3.5: The final algorithm to iteratively improve feasible solutions to (3.7).

If d is an R-deactivating direction for $A^*(f)$ and we have $(A^*(f) - R) \cap T_S = \emptyset$ for some $S \in C$, then there is $\alpha > 0$ such that $f' = f + \alpha d$ satisfies $B(f') < B(f)$ by Theorem 3.6a and 3.6c. The following corollary of Theorem 3.6 finally justifies why the certificate d of unsatisfiability of CSP $A^*(f)$ is an improving direction for (3.7):

Corollary 3.1. *CSP $A \subseteq T$ is unsatisfiable if and only if there is $d \in M^*$ such that for every $f \in \mathbb{R}^T$ with $A = A^*(f)$ there exists $\alpha > 0$ such that $B(f + \alpha d) < B(f)$.*

Proof. First, if for some $S \in C$ we have that $A \cap T_S = \emptyset$, A is unsatisfiable and no $f \in \mathbb{R}^T$ satisfies $A = A^*(f)$, so the second condition is trivially satisfied by choosing any $d \in M^*$.

If $A \cap T_S \neq \emptyset$ for all $S \in C$ but A is unsatisfiable, let d be any A-deactivating direction (which exists by Theorem 3.3). It follows from Theorem 3.6 that for any $f \in \mathbb{R}^T$ with $A^*(f) = A$, we can compute a suitable step size $\alpha > 0$ such that $B(f + \alpha d) < B(f)$. The case when A is satisfiable follows from Theorem 3.2. □

3.2.4 Final Algorithm

Having certificates of unsatisfiability from §3.2.2 and step sizes from §3.2.3, we can now precisely formulate in Algorithm 3.5 the iterative method that was previously sketched in §3.2.1.2 (Algorithm 3.2). First, constraint propagation is applied to CSP $A^*(f)$ by Algorithm 3.3 until either $A^*(f)$ is proved unsatisfiable or no more propagation is possible. In the latter case, the algorithm halts and returns $B(f)$ as the best achieved upper bound on the optimal value of WCSP g. Otherwise, if $A^*(f)$ is proved unsatisfiable due to $A_{n+1} \cap T_S = \emptyset$ for some $S \in C$, define I as in (3.19) so that $(A^*(f) - \bigcup_{i \in I} R_i) \cap T_S = \emptyset$ and compute an R^*-deactivating direction d^* where $R^* \supseteq \bigcup_{i \in I} R_i$ using Theorem 3.5. Since $(A^*(f) - R^*) \cap T_S = \emptyset$, we can update WCSP f using Theorem 3.6. Consequently, the bound $B(f)$ strictly improves after each update on line 7.

Although our theoretical results are more general, our implementation is limited only to pairwise WCSPs. In our implementation, we again use the trick similar to capacity scaling, i.e., we replace the active tuples $A^*(f)$ with 'almost' active tuples $A^*_\epsilon(f)$ defined in (2.10) and proceed as in Algorithm 2.2. Initially, ϵ is set to a high value and whenever we are unable to disprove satisfiability of $A^*_\epsilon(f)$, the current ϵ is decreased as $\epsilon := \epsilon/10$.

The process continues until ϵ becomes very small.[63] This heuristic forces the algorithm to disprove satisfiability using tuples that are far from being active, thus hopefully leading to larger step sizes and faster decrease of the bound.

We implemented two versions of Algorithm 3.5 (including capacity scaling), differing in the local consistency used to attempt to disprove satisfiability of CSP $A^*(f)$:

- *Virtual singleton arc consistency via super-reparametrizations (VSAC-SR)* uses singleton arc consistency (SAC). Precisely, we alternate between AC and SAC propagators: whenever a single tuple $(\{i\}, k)$ (where $i \in V$ and $k \in D$) is removed by SAC, we step back to enforcing AC until no more AC propagations are possible, and repeat.
- *Virtual cycle consistency via super-reparametrizations (VCC-SR)* is the same as VSAC-SR except that SAC is replaced by cycle consistency (CC). Though our implementation is different from [92] (we compose deactivating directions rather than alternate between the cycle-repair procedure and the Augmenting DAG algorithm), it has the same stopping points.

The procedures for generating deactivating directions for AC, SAC and CC were implemented as described in Examples 3.1, 3.3, and 3.2, respectively. In SAC and CC, it is useful to step back to AC whenever possible because deactivating directions of AC correspond to reparametrizations (that are more favorable, recall §3.2.1.1) rather than super-reparametrizations.

Remark 3.5. *In analogy to [33, 107], let us call a WCSP instance f* virtual Φ-consistent *(e.g., virtual AC or virtual RPC) if $A^*(f)$ has non-empty Φ-consistency closure. Then, a* virtual Φ-consistency algorithm *naturally refers to an algorithm to transform a given WCSP instance to a virtual Φ-consistent WCSP instance. In the VAC algorithm, this transformation is equivalence-preserving, i.e., a reparametrization. But in our case, it is a super-reparametrization, which is why we call our algorithms VSAC-SR and VCC-SR.*

Since we restricted ourselves to pairwise WCSPs, $(V, C_{\geq 2})$ is an undirected graph. The cycles in VCC-SR were chosen as follows: if $2|C_{\geq 2}|/|V| \leq 5$ (i.e., the average degree of the nodes in $(V, C_{\geq 2})$ is at most 5), then all cycles of length 3 and 4 present in the graph $(V, C_{\geq 2})$ are used. If $2|C_{\geq 2}|/|V| \leq 10$, then all cycles of length 3 present in the graph are used. If $2|C_{\geq 2}|/|V| > 10$ or the above method did not result in any cycles, we use all fundamental cycles w.r.t. a spanning tree of the graph $(V, C_{\geq 2})$ [116, §9]. No additional edges are added to the graph. Note, [92] experimented with grid graphs (where cycles of length 4 and 6 of the grid were used) and complete graphs (where cycles of length 3 were used).

Since both VSAC-SR and VCC-SR start by enforcing VAC (i.e., making $A^*(f)$ have non-empty AC closure by reparametrizations), before running these methods we used toulbar2 [1] to reparametrize the input WCSP instance to a VAC state (because a specialized algorithm is faster than the more general Algorithm 3.5). We employed specialized data structures for storing the sequences $(R_i)_{i=0}^n$ and $(d^i)_{i=0}^n$ from Algorithm 3.3, which utilize the property that the sets $(R_i)_{i=0}^n$ are disjoint and make easier sequential querying

[63] In detail, we initialized $\epsilon = \max_{k_i,k_j} g_{\{i,j\}}(k_i,k_j) - \min_{k_i,k_j} g_{\{i,j\}}(k_i,k_j) + \max_k g_{i'}(k) - \min_k g_{i'}(k)$ where $\{i,j\} \in C_{\geq 2}$ and $i' \in V$ is the edge and variable with the lowest index (based on indexing in the input instance). The terminating condition was $\epsilon \leq 10^{-6}$. In order to improve the efficiency of our method, we also decreased ϵ whenever the bound did not improve by more than 10^{-15} in 20 consecutive iterations (cf. VAC_ϵ in [33, §11.1]).

of (sparse) vectors $(d^i)_{i=0}^n$ in Algorithm 3.4. Note that the sequence $(A_i)_{i=0}^{n+1}$ need not be stored and is only needed for theoretical analysis. Moreover, sparse representations were used when composing deactivating directions in Algorithm 3.4. To avoid working with 'structured' tuples (1.23), we employed a bijection between T and $\{1, \ldots, |T|\}$ to work with numerical indices instead.

Besides the above improvements, we did not fine-tune our implementation for efficiency. Thus, the set $A^*(f)$ was always calculated by iterating through all tuples (which could be made faster if sparsity of the improving direction was taken into account). The hyper-parameters of our algorithm (e.g., the decrease schedule of ϵ or constants mentioned in Footnote 63) were not learned nor systematically optimized. SAC was checked on all active tuples without warm-starting or using any faster SAC algorithm than SAC1 [18, 45, Figure 2]. Perhaps most importantly, we did not implement inter-iteration warm-starting as in [145, 48a], i.e., after updating the weights on line 7 of Algorithm 3.5, some deactivating directions in the sequence that were not used in computing the improving direction may be preserved for the next iteration instead of being computed from scratch. Except for computing the deactivating directions, the code was the same for VSAC-SR and VCC-SR. We implemented everything in Java.

3.2.5 Experimental Results

We compared the bounds calculated by VSAC-SR and VCC-SR with the bounds provided by EDAC [43], VAC [33], pseudo-triangles (option `-t=8000` in toulbar2, adds up to 8 GB of ternary weight functions), PIC, EDPIC, maxRPC, and EDmaxRPC [107], which are implemented in toulbar2 [1].

We did the comparison on the Cost Function Library benchmark [2]. Due to limited computation resources, we used only the smallest 16500 instances (out of 18132). Of these, we omitted instances containing weight functions of arity 3 or higher. Moreover, to avoid easy instances, we omitted instances that were solved by VAC without search (i.e., toulbar2 with options `-A -bt=0` found an optimal solution). We also omitted the *validation* instances that are used for testing and debugging. Overall, 5371 instances were left for our comparison.

For each instance and each method, we only calculated the upper bound and did not do any search. Then, for each instance and method, we computed the normalized bound $\frac{B_w - B_m}{B_w - B_b}$ where B_w and B_b are the worst and the best bound for the instance among all the methods, respectively, and B_m is the bound computed by the method for the instance. Thus, the best bound[64] transforms to 1 and the worst bound to 0, i.e., greater is better.

For 26 instances, at least one method was not able to finish in the prespecified 1-hour CPU-time limit. These timed-out methods were omitted from the calculation of the normalized bounds for these instances. From the point of view of the method, the instance was not incorporated into the average of the normalized bounds of this particular method. We note that implementations of VSAC-SR and VCC-SR provide a bound when terminated at any time, whereas the implementations of the other methods provide a bound only when they are left to finish. Time-out happened 5, 2, 3, 6, and 24 times for pseudo-triangles, PIC, EDPIC, maxRPC, and EDmaxRPC, respectively. This did not affect the results much as there were 5731 instances in total.

[64] To avoid numerical precision issues, bounds B_m within $B_b \pm 10^{-4} B_b$ or $B_b \pm 0.01$ are also normalized to 1. If $B_w = B_b$, then the normalized bounds for all methods are equal to 1 on this instance.

Instance Group	Instances	EDAC	VAC	VSAC-SR	VCC-SR	Pseudo-tr.	PIC	EDPIC	maxRPC	EDmaxRPC
/biqmaclib/	157	0.02	0.11	0.90	0.22	**0.92**	0.83	0.81	0.79	0.81
/crafted/academics/	8	0.88	0.88	0.97	0.95	0.88	0.88	0.88	0.88	**1.00**
/crafted/auction/paths/	420	0.00	0.09	0.91	0.35	**0.99**	0.45	0.68	0.64	0.57
/crafted/auction/regions/	411	0.00	0.05	**0.99**	0.10	0.98	0.08	0.18	0.23	0.13
/crafted/auction/scheduling/	419	0.00	0.02	**1.00**	0.09	0.80	0.41	0.38	0.41	0.24
/crafted/coloring/	33	0.94	0.94	0.99	0.97	0.98	**1.00**	**1.00**	**1.00**	0.99
/crafted/feedback/	6	0.00	0.00	0.54	0.58	0.71	0.49	0.53	0.51	**0.72**
/crafted/kbtree/	1800	0.25	0.29	0.60	0.67	0.80	0.73	0.81	0.76	**0.89**
/crafted/maxclique/dimacs_maxclique/	49	0.06	0.24	**0.98**	0.39	0.87	0.39	0.50	0.51	0.55
/crafted/maxcut/spinglass_maxcut/unweighted/	5	0.00	0.00	**1.00**	0.42	0.15	0.15	0.15	0.15	0.15
/crafted/maxcut/spinglass_maxcut/weighted/	5	0.00	0.00	**1.00**	0.38	0.17	0.17	0.17	0.17	0.17
/crafted/modularity/	6	0.17	0.19	0.38	0.25	**0.99**	0.96	0.94	0.96	0.97
/crafted/planning/	65	0.00	0.54	**0.94**	0.72	0.32	0.07	0.09	0.07	0.17
/crafted/sumcoloring/	43	0.04	0.15	0.47	0.50	**0.81**	0.53	0.63	0.64	0.61
/crafted/warehouses/	49	0.35	0.99	**1.00**	0.99	0.35	0.42	0.42	0.42	0.42
/qaplib/	5	0.40	0.40	0.40	0.41	**0.99**	0.97	0.97	0.98	0.97
/qplib/	23	0.00	0.10	**0.96**	0.38	0.27	0.25	0.25	0.24	0.25
/random/maxcsp/completeloose/	50	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
/random/maxcsp/completetight/	50	0.00	0.12	0.57	0.72	0.88	0.94	**0.99**	0.69	0.76
/random/maxcsp/denseloose/	50	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
/random/maxcsp/densetight/	50	0.02	0.14	0.52	**1.00**	0.68	0.48	0.49	0.52	0.60
/random/maxcsp/sparseloose/	90	0.96	0.96	**1.00**	0.96	0.96	0.96	0.96	0.96	0.96
/random/maxcsp/sparsetight/	50	0.01	0.12	0.54	**1.00**	0.64	0.40	0.40	0.43	0.51
/random/maxcut/random_maxcut/	400	0.00	0.00	0.77	0.13	0.95	0.98	0.98	0.97	**0.99**
/random/mincut/	500	0.09	**1.00**	**1.00**	**1.00**	0.10	0.10	0.10	0.10	0.10
/random/randomksat/	493	0.01	0.02	0.75	0.22	**0.95**	0.91	0.89	0.86	0.87
/random/wqueens/	6	0.00	0.52	**0.96**	0.94	0.48	0.12	0.29	0.13	0.72
/real/celar/	23	0.00	0.05	0.08	0.16	**0.97**	0.66	0.66	0.78	0.95
/real/maxclique/protein_maxclique/	1	0.00	0.00	**1.00**	0.03	0.93	0.04	0.04	0.08	0.04
/real/spot5/	1	0.00	0.08	**1.00**	0.49	**1.00**	0.74	0.66	0.41	0.74
/real/tagsnp/tagsnp_r0.5/	23	0.04	0.86	**0.95**	0.86	0.31	0.31	0.33	0.29	0.46
/real/tagsnp/tagsnp_r0.8/	80	0.13	0.66	**0.91**	0.68	0.29	0.39	0.38	0.33	0.47
Average over all groups	5371	0.20	0.36	**0.82**	0.58	0.72	0.56	0.58	0.56	0.62
Average over groups with ≥ 5 instances	5369	0.21	0.38	**0.80**	0.60	0.71	0.57	0.59	0.58	0.63

Table 3.1: Results on instances from Cost Function Library: Average normalized bounds.

Instance Group	Instances	EDAC	VAC	VSAC-SR	VCC-SR	Pseudo-tr.	PIC	EDPIC	maxRPC	EDmaxRPC
/biqmaclib/	157	**0.11**	0.12	180.07	34.60	83.25	1240.00	1241.29	1242.16	1271.86
/crafted/academics/	8	**0.11**	**0.11**	28.61	1.04	29.08	121.44	120.86	108.08	104.47
/crafted/auction/paths/	420	**0.04**	**0.04**	1.96	0.83	1.92	0.19	0.23	0.48	0.64
/crafted/auction/regions/	411	**0.20**	0.32	32.14	9.45	673.42	49.85	51.37	102.61	110.48
/crafted/auction/scheduling/	419	**0.10**	0.12	16.22	2.03	49.85	26.90	26.89	32.06	32.30
/crafted/coloring/	33	**0.09**	0.10	4.99	1.40	0.20	545.50	545.50	545.51	545.50
/crafted/feedback/	6	**0.70**	**0.70**	3588.39	3600.11	11.64	1860.89	1874.08	1875.93	1873.07
/crafted/kbtree/	1800	**0.02**	**0.02**	3.13	11.25	0.10	0.04	0.05	0.06	0.07
/crafted/maxclique/dimacs_maxclique/	49	**0.71**	1.32	279.08	126.90	955.60	1345.67	1342.14	1429.73	1428.12
/crafted/maxcut/spinglass_maxcut/unweighted/	5	0.02	0.02	0.82	0.44	0.02	**0.01**	**0.01**	**0.01**	**0.01**
/crafted/maxcut/spinglass_maxcut/weighted/	5	0.02	0.02	1.09	0.53	0.02	**0.01**	**0.01**	**0.01**	**0.01**
/crafted/modularity/	6	**0.19**	0.29	1023.48	127.39	66.25	706.30	783.02	741.91	1442.57
/crafted/planning/	65	**0.16**	0.29	638.85	60.62	7.41	0.93	0.96	2.33	4.73
/crafted/sumcoloring/	43	**1.29**	1.94	727.49	963.61	255.72	1508.37	1508.36	1509.34	1512.68
/crafted/warehouses/	49	4.10	9.48	735.80	735.83	**4.09**	29.48	29.54	28.80	29.82
/qaplib/	5	**0.08**	0.09	119.05	278.53	7.38	1448.63	1444.95	1450.09	1449.22
/qplib/	23	**0.13**	0.14	255.85	43.11	195.32	626.25	626.24	626.27	626.36
/random/maxcsp/completeloose/	50	**0.06**	**0.06**	1.31	0.16	0.48	0.09	0.10	0.19	0.18
/random/maxcsp/completetight/	50	**0.02**	0.03	6.35	12.68	0.47	0.21	0.25	0.31	0.33
/random/maxcsp/denseloose/	50	**0.02**	**0.02**	166.78	0.06	0.11	0.03	0.03	0.03	0.03
/random/maxcsp/densetight/	50	**0.02**	**0.02**	4.20	17.38	0.10	0.06	0.07	0.07	0.08
/random/maxcsp/sparseloose/	90	**0.03**	**0.03**	611.38	0.05	0.06	0.04	0.04	0.04	0.04
/random/maxcsp/sparsetight/	50	**0.02**	**0.02**	11.00	9.74	0.06	0.04	0.05	0.05	0.05
/random/maxcut/random_maxcut/	400	**0.01**	**0.01**	0.73	0.15	0.04	0.03	0.03	0.05	0.07
/random/mincut/	500	1.09	2.43	14.40	86.22	1.12	0.88	**0.87**	**0.87**	**0.87**
/random/randomksat/	493	**0.02**	**0.02**	3.42	0.17	0.13	0.07	0.10	0.16	0.31
/random/wqueens/	6	**1.33**	1.49	992.85	502.42	644.87	1800.15	1800.20	1800.18	1800.60
/real/celar/	23	**0.27**	0.28	1798.51	2972.69	66.56	300.76	219.91	495.26	1066.87
/real/maxclique/protein_maxclique/	1	**0.26**	0.44	25.24	6.77	1196.62	114.62	114.99	215.30	220.81
/real/spot5/	1	**0.01**	**0.01**	0.62	0.08	0.11	0.03	0.03	0.04	0.04
/real/tagsnp/tagsnp_r0.5/	23	**4.83**	378.77	3338.53	2897.83	239.38	3155.96	3148.66	3172.58	3295.19
/real/tagsnp/tagsnp_r0.8/	80	**1.52**	22.82	1239.73	858.83	90.05	195.12	206.76	359.55	409.88
Average over all groups	5371	**0.55**	13.17	495.38	417.59	143.17	471.21	471.49	491.88	538.35
Average over groups with ≥ 5 instances	5369	**0.58**	14.04	527.54	445.20	112.82	498.80	499.08	517.49	566.88

Table 3.2: Results on instances from Cost Function Library: Average CPU time in seconds.

The results in Table 3.1 show that no method is best for all instance groups, instead, each method is suitable for a different group. However, VSAC-SR performed best for most groups and otherwise was often competitive to the other strong consistency methods. VSAC-SR seems particularly good at spinglass_maxcut [3], planning [40] and qplib [64] instances. Taking the overall unweighted average of group averages (giving the same importance to each group), VSAC-SR achieved the greatest average value. We also evaluated the ratio to worst bound, B_m/B_w, for instances with $B_w \neq 0$; the results were qualitatively the same: VSAC-SR again achieved the best overall average of 3.93 (or 4.15 if only groups with ≥ 5 instances are considered) compared to second-best pseudo-triangles with 2.71 (or 2.84).

The runtimes (on a laptop with i7-4710MQ processor at 2.5 GHz and 16GB RAM) are reported in Table 3.2. Again, the results are group-dependent and one can observe that the methods explore different trade-offs between bound quality and runtime. However, the strong consistencies are comparable in terms of runtime on average, except for pseudo-triangles, which is a faster method that however needs significantly more memory.

3.3 Additional Properties of Super-Reparametrizations

In this section, we present a more detailed study of properties of WCSPs that are preserved by (possibly optimal) super-reparametrizations. Although we did not need these properties for the previously described method, they may be valuable for future research. Here, we first revisit in §3.3.1 the notion of a minimal CSP for a set of assignments that was studied, e.g., in [105, §3, 46, §2.3.2]. The key result of §3.3 is presented in §3.3.2, where we study the relation of the set of optimal assignments of some WCSP to the set of optimal assignments of its super-reparametrization optimal for (3.7), showing that they need not coincide in general. In §3.3.3, we analyze the case of general (i.e., not necessarily optimal for (3.7)) super-reparametrizations.

3.3.1 Minimal CSP

Let us ask when, for a given set $X \subseteq D^V$ of assignments (i.e., a $|V|$-ary relation over D), does there exist $A \subseteq T$ such that $X = \mathrm{SOL}(A)$, i.e., when is X representable as the solution set of a CSP with a given structure (V, D, C). For that, denote

$$A_{\min}(X) = \bigcap \mathcal{A}(X) \quad \text{where} \quad \mathcal{A}(X) = \{\, A \subseteq T \mid X \subseteq \mathrm{SOL}(A) \,\}. \tag{3.20}$$

Thus, $\mathcal{A}(X)$ is the set of all CSPs whose solution set includes X and $A_{\min}(X)$ is the intersection of these CSPs. We call $A_{\min}(X)$ the *minimal CSP* for X. For pairwise CSPs, this concept was studied in [105, §3] and [46, §2.3.2].

Proposition 3.3. *The mapping* SOL *preserves intersections*[65]*, i.e., for any* $A_1, A_2 \subseteq T$ *we have* $\mathrm{SOL}(A_1 \cap A_2) = \mathrm{SOL}(A_1) \cap \mathrm{SOL}(A_2)$.

[65] We remark that isotony of SOL (that we noted already in §1.4) does not in general imply preserved intersections. A weaker result than our Proposition 3.3 is [105, Theorem 3.2]: in our notation, it says that $\mathrm{SOL}(A_1) = \mathrm{SOL}(A_2)$ implies $\mathrm{SOL}(A_1 \cap A_2) = \mathrm{SOL}(A_1)$.

Proof. For any $x \in D^V$, we have

$$\begin{aligned} x \in \mathrm{SOL}(A_1) \cap \mathrm{SOL}(A_2) &\iff x \in \mathrm{SOL}(A_1),\ x \in \mathrm{SOL}(A_2) \\ &\iff \forall S \in C\colon \big(S, x|_S\big) \in A_1,\ \big(S, x|_S\big) \in A_2 \\ &\iff \forall S \in C\colon \big(S, x|_S\big) \in A_1 \cap A_2 \\ &\iff x \in \mathrm{SOL}(A_1 \cap A_2). \end{aligned}$$

□

Proposition 3.4. *For any $X \subseteq D^V$, the set $\mathcal{A}(X)$ is closed under intersections, i.e., for any $A_1, A_2 \subseteq T$ we have $A_1, A_2 \in \mathcal{A}(X) \implies A_1 \cap A_2 \in \mathcal{A}(X)$.*

Proof. Suppose that $X \subseteq \mathrm{SOL}(A_1)$ and $X \subseteq \mathrm{SOL}(A_2)$, then $X \subseteq \mathrm{SOL}(A_1) \cap \mathrm{SOL}(A_2) = \mathrm{SOL}(A_1 \cap A_2)$, where the equality holds by Proposition 3.3. □

Proposition 3.4 implies $A_{\min}(X) \in \mathcal{A}(X)$, i.e., $X \subseteq \mathrm{SOL}(A_{\min}(X))$. This shows that $A_{\min}(X)$ is the smallest CSP whose solution set includes X. It follows that $X = \mathrm{SOL}(A_{\min}(X))$ if and only if $X = \mathrm{SOL}(A)$ for some $A \subseteq T$.

The minimal CSP for X can be equivalently defined in terms of tuples:

Proposition 3.5 ([105, 46])**.** *We have $A_{\min}(X) = \{\, (S,k) \in T \mid \exists x \in X\colon x|_S = k \,\}$.*

Proof. Denote $A' = \{\, (S,k) \in T \mid \exists x \in X\colon x|_S = k \,\}$. By definition of A', we have $\mathrm{SOL}(A') \supseteq X$, so $A' \in \mathcal{A}(X)$ and $A_{\min}(X) = \bigcap \mathcal{A}(X) \subseteq A'$.

It remains to show that $A_{\min}(X) \supseteq A'$. For contradiction, suppose there is a tuple $(S^*, k^*) \in A' - A_{\min}(X)$. By definition of A', there exists $x \in X$ such that $x|_{S^*} = k^*$. However, since $\big(S^*, x|_{S^*}\big) = (S^*, k^*) \notin A_{\min}(X)$, we have $x \notin \mathrm{SOL}(A_{\min}(X))$. By $x \in X$, this contradicts $X \subseteq \mathrm{SOL}(A_{\min}(X))$. □

Note that Proposition 3.5 shows that $A_{\min}(X)$ is positively consistent (recall Definition 1.6) for every $X \subseteq D^V$.

Theorem 3.7. *For any $X \subseteq D^V$ and $A \subseteq T$, we have*

$$A_{\min}(X) \subseteq A \iff X \subseteq \mathrm{SOL}(A). \tag{3.21}$$

Proof. If $A_{\min}(X) \subseteq A$, then by isotony of SOL we have $\mathrm{SOL}(A_{\min}(X)) \subseteq \mathrm{SOL}(A)$. Since $X \subseteq \mathrm{SOL}(A_{\min}(X))$, we have $X \subseteq \mathrm{SOL}(A)$.

If $X \subseteq \mathrm{SOL}(A)$, i.e., $A \in \mathcal{A}(X)$, then $A_{\min}(X) = \bigcap \mathcal{A}(X) \subseteq A$. □

Comparing (3.21) with (3.20) shows that the set $\mathcal{A}(X)$ is just an interval (w.r.t. the partial ordering given by the set inclusion):

$$\mathcal{A}(X) = \{\, A \mid A_{\min}(X) \subseteq A \subseteq T \,\} \tag{3.22}$$

Theorem 3.7 further reveals that the maps $A_{\min}$ and SOL form a *Galois connection* [42, §7] between sets $2^{(D^V)}$ and 2^T, partially ordered by the set inclusion [66]. Associated with the Galois connection are the closure operator $\mathrm{SOL} \circ A_{\min}$ and the dual closure operator $A_{\min} \circ \mathrm{SOL}$. We have already seen their meaning:

[66] To our knowledge, we were the first to notice this in [56a, §5.1].

- For any CSP $A \subseteq T$, the CSP $A_{\min}(\mathrm{SOL}(A))$ is the positive consistency (dual) closure $\mathscr{C}_{\mathrm{pos}}(A)$ from §1.4.1.
- For any set of assignments $X \subseteq D^V$, $\mathrm{SOL}(A_{\min}(X))$ is the smallest (possibly non-strict) superset of X which is the solution set of some CSP $A \subseteq T$. We have $X = \mathrm{SOL}(A_{\min}(X))$ if and only if $X = \mathrm{SOL}(A)$ for some $A \subseteq T$.

Remark 3.6. *Following [42, §7.27], it is easy to see in this case that the mappings* $A_{\min}$ *and* SOL *are mutually inverse bijections (even order-isomorphisms) if we restrict ourselves only to positively consistent CSPs, i.e.,* $\{\, A \subseteq T \mid A_{\min}(\mathrm{SOL}(A)) = A \,\}$ *and sets of assignments representable as solution sets of some CSP* $A \subseteq T$*, i.e.,* $\{\, \mathrm{SOL}(A) \mid A \subseteq T \,\}$.

3.3.2 Optimal Assignments of Optimal Super-Reparametrizations

Theorem 3.1 (or already Theorem 1.16) says that the optimal value of (3.7) equals the optimal value of WCSP g. We now focus on the optimal assignments (rather than value) of WCSP g. For brevity, we denote the set of all optimal assignments of WCSP g by

$$\mathrm{OPT}(g) = \operatorname*{argmax}_{x \in D^V} F(x \,|\, g) \subseteq D^V. \tag{3.23}$$

Theorem 3.8. *If* f *is optimal for* (3.7)*, then* $\mathrm{OPT}(g) \subseteq \mathrm{OPT}(f) = \mathrm{SOL}(A^*(f))$.

Proof. To show $\mathrm{OPT}(g) \subseteq \mathrm{OPT}(f)$, let $x^* \in \mathrm{OPT}(g)$. By statement (b) in Theorem 3.1, $F(x^* \,|\, g) = B(f)$. Since $B(f) \geq F(x^* \,|\, f) \geq F(x^* \,|\, g)$, we have that $B(f) = F(x^* \,|\, f) = F(x^* \,|\, g)$, thus x^* is optimal for WCSP f.

The equality $\mathrm{OPT}(f) = \mathrm{SOL}(A^*(f))$ follows from Theorem 1.15b together with $B(f) = \max_{x \in D^V} F(x \,|\, f)$ which is given by optimality of f (see statement (b) in Theorem 3.1). □

Our main goal in §3.3 is to characterize when the inclusion in Theorem 3.8 holds with equality, which is given by a later stated Theorem 3.10.

Proposition 3.6. *For every* $g \in \mathbb{R}^T$ *and* $A \subseteq T$ *such that* $\mathrm{OPT}(g) \subseteq \mathrm{SOL}(A)$*, there exists* $f \in \mathbb{R}^T$ *optimal for* (3.7) *such that* $A = A^*(f)$.

Proof. Define the vector f by

$$f_t = \begin{cases} F_1/|C| & \text{if } t \in A \\ F_2/|C| & \text{if } t \notin A \end{cases} \qquad \forall t \in T \tag{3.24}$$

where

$$F_1 = \max_{x \in D^V} F(x \,|\, g) \qquad \text{and} \qquad F_2 = \max\{\, F(x \,|\, g) \mid x \in D^V,\ F(x \,|\, g) < F_1 \,\} \tag{3.25}$$

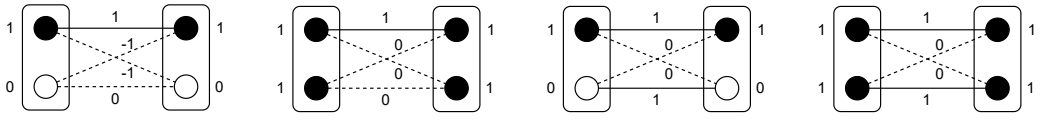
are the best and the second-best objective value of WCSP g, respectively. Note, if $\mathrm{OPT}(g) = D^V$, then F_2 is undefined but it does not matter because it is never used in (3.24).

Since $\emptyset \neq \mathrm{OPT}(g) \subseteq \mathrm{SOL}(A)$, CSP A is satisfiable. Therefore for each $S \in C$ we have $A \cap T_S \neq \emptyset$, hence

$$\max_{t \in T_S} f_t = F_1/|C|. \tag{3.26}$$

Equality $A = A^*(f)$ now follows from (3.24).

To show that f is feasible for (3.7), we distinguish two cases:



(a) WCSP g (b) WCSP f^1 (c) WCSP f^2 (d) WCSP f^3

Figure 3.5: Examples of four WCSPs with the same structure. WCSPs f^1, f^2, and f^3 are optimal super-reparametrizations of WCSP g.

- If $x \in \mathrm{OPT}(g)$, i.e., $F(x\,|\,g) = F_1$, then $x \in \mathrm{SOL}(A) = \mathrm{SOL}(A^*(f))$. Therefore for all $S \in C$ we have $\big(S, x|_S\big) \in A^*(f)$, hence $f_S\big(x|_S\big) = F_1/|C|$ by (3.26). Substituting into (1.33) yields $F(x\,|\,f) = F_1$ and $F(x\,|\,f) = F_1 = F(x\,|\,g)$.
- If $x \notin \mathrm{OPT}(g)$, we have $f_S\big(x|_S\big) \geq F_2/|C|$ for all $S \in C$, hence $F(x\,|\,f) \geq F_2$ by (1.33). By (3.25) we also have $F(x\,|\,g) \leq F_2$, so $F(x\,|\,f) \geq F_2 \geq F(x\,|\,g)$.

To show that f is optimal for (3.7), we use (3.26) to obtain $B(f) = \sum_{S\in C} F_1/|C| = F_1 = \max_x F(x\,|\,g)$ and apply Theorem 3.1. □

Example 3.4. *To exemplify Proposition 3.6, consider WCSP g from Figure 3.5a. We have that $|C| = 3$, $F_1 = 3$ and $F_2 = 0$. Each of the WCSPs shown in Figures 3.5b, 3.5c, and 3.5d has a different set of active tuples, is a super-reparametrization of g, and is optimal for* (3.7). △

Theorem 3.9. *For every $g \in \mathbb{R}^T$, we have*

$$\mathcal{A}(\mathrm{OPT}(g)) = \{\, A^*(f) \mid f \text{ is optimal for } (3.7) \,\}. \tag{3.27}$$

Proof. The inclusion $\supseteq$ says that for every optimal f we have $\mathrm{OPT}(g) \subseteq \mathrm{SOL}(A^*(f))$, which was proved in Theorem 3.8. The inclusion $\subseteq$ was proved in Proposition 3.6. □

Now we combine the results of §3.3.1 and §3.3.2 to obtain the main result of §3.3. First observe that, by (3.22), the set (3.27) is just the interval $\{\, A \mid A_{\min}(\mathrm{OPT}(g)) \subseteq A \subseteq T \,\}$.

Theorem 3.10. *For every $g \in \mathbb{R}^T$, the following statements are equivalent:*

(a) $\mathrm{OPT}(g) = \mathrm{SOL}(A)$ *for some* $A \subseteq T$,

(b) $\mathrm{OPT}(g) = \mathrm{OPT}(f)$ *for some f optimal for* (3.7).

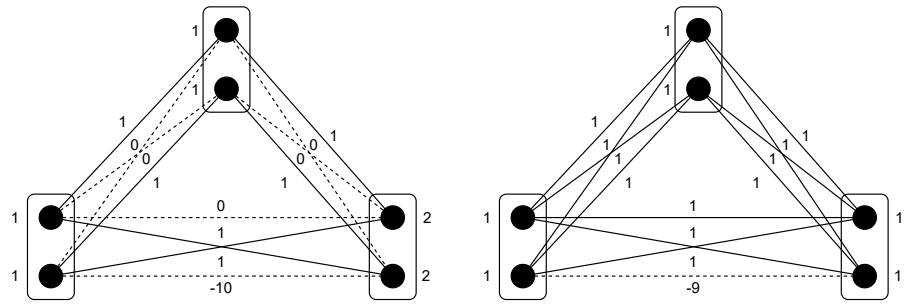
If both statements are true, then statement (a) holds, e.g., for $A = A_{\min}(\mathrm{OPT}(g))$ and statement (b) holds, e.g., if $A^*(f) = A_{\min}(\mathrm{OPT}(g))$.

Proof. Let $g \in \mathbb{R}^T$. By Theorem 3.9, there exists f optimal for (3.7) satisfying $A^*(f) = A_{\min}(\mathrm{OPT}(g))$. By Theorem 3.8, this f satisfies $\mathrm{OPT}(f) = \mathrm{SOL}(A^*(f))$.

By the results of §3.3.1, statement (a) is equivalent to $\mathrm{OPT}(g) = \mathrm{SOL}(A_{\min}(\mathrm{OPT}(g)))$. Therefore, if (a) holds, then (b) holds for the above f. In the other direction, if (b) holds for the above f, then (a) holds. □

Theorem 3.10 shows that the set inclusion in Theorem 3.8 holds with equality for some optimal f if and only if the set $\mathrm{OPT}(g)$ is representable as a solution set of some CSP with the same structure. If no such CSP exists, then $\mathrm{OPT}(g) \subsetneq \mathrm{OPT}(f)$ for all optimal f. An example of WCSP g for which no such CSP exists is in Figure 3.6.

It is natural to ask which WCSPs possess this property. Though we are currently unable to provide a full characterization of such WCSPs, we identify two such classes.



(a) WCSP g with $B(g) = 7$ (b) WCSP f with $B(f) = 6$

Figure 3.6: WCSP f is an optimal super-reparametrization of WCSP g. It is easy to verify that $A^*(f) = A_{\min}(\mathrm{OPT}(g))$ but $\mathrm{OPT}(f) = \mathrm{SOL}(A^*(f)) \supsetneq \mathrm{OPT}(g)$.

Proposition 3.7 ([121, 146])**.** *If the LP relaxation* (3.2) *(i.e.,* (1.40)*) of a WCSP* $g \in \mathbb{R}^T$ *is tight, then* $\mathrm{OPT}(g) = \mathrm{SOL}(A)$ *for some* $A \subseteq T$*.*

Proof. If the LP relaxation (3.2) is tight, then there exists a vector $f \in \mathbb{R}^T$ such that $B(f) = \max_{x \in D^V} F(x\,|\,g)$ and f is a reparametrization of g, i.e., $F(x\,|\,f) = F(x\,|\,g)$ for all $x \in D^V$, thus, f is also optimal for (3.7). It follows that the sets of optimal assignments for f and g coincide. By Theorem 3.8, $A^*(f)$ is the required CSP. □

It is clear that Proposition 3.7 also applies to the basic LP relaxation (1.44), i.e., if the basic LP relaxation of a WCSP g is tight, then $\mathrm{OPT}(g) = \mathrm{SOL}(A)$ for some $A \subseteq T$.

Proposition 3.8. *If a WCSP* $g \in \mathbb{R}^T$ *has a unique optimal assignment, then* $\mathrm{OPT}(g) = \mathrm{SOL}(A)$ *for some* $A \subseteq T$*.*

Proof. Let $\mathrm{OPT}(g) = \{x\}$. We claim that $A = \{\, (S, x|_S) \mid S \in C \,\}$ is the required CSP, i.e., $\mathrm{SOL}(A) = \{x\}$. For contradiction, suppose that $x' \in \mathrm{SOL}(A)$ and $x' \neq x$. By $x' \in \mathrm{SOL}(A)$, we necessarily have that $x'|_S = x|_S$ for all $S \in C$. By definition (1.33), this implies $F(x'\,|\,g) = F(x\,|\,g)$. Thus, $\{x', x\} \subseteq \mathrm{OPT}(g)$, which is contradictory with $|\,\mathrm{OPT}(g)| = 1$. □

3.3.3 General Super-Reparametrizations

Finally, we present one property of general super-reparametrizations f of a fixed WCSP g, i.e., f is only feasible (but possibly not optimal) for (3.7).

Theorem 3.11. *For every* $g \in \mathbb{R}^T$ *we have*

$$\{\, A^*(f) \mid f \in \mathbb{R}^T \text{ is a super-reparametrization of } g \,\} = \{\, A^*(f) \mid f \in \mathbb{R}^T \,\}. \tag{3.28}$$

Proof. The inclusion $\subseteq$ is trivial. To prove $\supseteq$, let $f' \in \mathbb{R}^T$ be arbitrary. Define $f \in \mathbb{R}^T$ by $f_t = B(g)/|C| + [\![t \in A^*(f')]\!]$, $t \in T$. Clearly, f is a super-reparametrization of g due to $F(x\,|\,f) \geq B(g) \geq F(x\,|\,g)$ for any $x \in D^V$. In addition, by definition of f, $\max_{t \in T_S} f_t = B(g)/|C| + 1$ for any $S \in C$, hence $A^*(f') = A^*(f)$. □

Theorem 3.11 shows that the left-hand set in (3.28) does not depend on g at all. Therefore, if we approximately optimize (3.7), i.e., we find a (possibly non-optimal) super-reparametrization f of g, then there is in general no relation between sets $\mathrm{OPT}(f)$ and

$\mathrm{OPT}(g)$. However, as discussed in §3.1, an arbitrary super-reparametrization still maintains the valuable property that it provides an upper bound $B(f)$ on the optimal value of WCSP g.

3.4 Hardness Results

Unsurprisingly, a number of decision and optimization problems related to problem (3.7) is computationally hard since the optimization problem (3.7) is hard itself. We overview a number of such problems here.

Theorem 3.12. *The following problem is NP-complete: Given $f, g \in \mathbb{Q}^T$, decide whether f is* not *a super-reparametrization of g (i.e., whether f is* not *feasible for* (3.7)*).*

Proof. Membership in NP can be shown easily: first, one can non-deterministically choose any $x \in D^V$ and then in polynomial time decide whether $F(x\,|\,f) < F(x\,|\,g)$.

To show NP-hardness, we perform a reduction from CSP satisfiability which is known to be NP-complete. Let $A \subseteq T$ be a CSP. We would like to decide whether $\mathrm{SOL}(A) \neq \emptyset$.

Let us define $g \in \{0,1\}^T$ by

$$g_S(k) = [\![(S,k) \in A]\!] \qquad \forall (S,k) \in T. \tag{3.29}$$

Thus, for any $x \in D^V$, $F(x\,|\,g)$ equals the number of constraints in CSP A that are satisfied by the assignment x. So, $F(x\,|\,g) \in \{0,1,\ldots,|C|\}$ and $F(x\,|\,g) = |C|$ if and only if $x \in \mathrm{SOL}(A)$. Consequently, $\max_x F(x\,|\,g) \leq |C| - 1$ if and only if $\mathrm{SOL}(A) = \emptyset$.

We define $f \in \mathbb{Q}^T$ by $f_t = (|C| - 1)/|C|$, $t \in T$. So, $F(x\,|\,f) = |C| - 1$ for all $x \in D^V$. Hence, $\mathrm{SOL}(A) = \emptyset$ if and only if f is a super-reparametrization of g. □

Corollary 3.2. *The following problem is NP-complete: Given $d \in \mathbb{Q}^T$, decide whether $d \notin M^*$.*

Proof. Membership in NP is analogous to Theorem 3.12. The question of whether f is not a super-reparametrization of g from Theorem 3.12 reduces to whether $d = f - g \notin M^*$. □

Corollary 3.3. *The following problem is NP-complete: Given $f, g \in \{0,1\}^T$ where f is a super-reparametrization of g, decide whether f is optimal for* (3.7)*.*

Proof. Membership in NP follows from statement (c) in Theorem 3.1: as in Theorem 3.12, one can choose $x \in D^V$ and then in polynomial time decide whether $x \in \mathrm{SOL}(A^*(f))$ and $F(x\,|\,f) = F(x\,|\,g)$.

The hardness part is completely analogous to the proof of Theorem 3.12 except that we define f by $f_t = 1$, $t \in T$, so $B(f) = |C|$. Clearly, f is element-wise greater than or equal to g, so it is a super-reparametrization. Moreover, $|C| = \max_x F(x\,|\,g)$ if and only if $\mathrm{SOL}(A) \neq \emptyset$, so f is optimal for (3.7) if and only if $\mathrm{SOL}(A) \neq \emptyset$. □

Recall that in formula (3.14), the number δ had the concrete value given by Theorem 3.3. However, sometimes the value of δ can be decreased while (3.14) still remains to be an R-deactivating direction for A. Finding a small such δ is desirable because then (3.14) results in smaller objective values $F(x\,|\,d)$, as explained in §3.2.1.1. Unfortunately, finding the least value of δ is likely intractable:

Theorem 3.13. *The following problem is NP-complete: Given $\delta \in \mathbb{Q}$ and $\emptyset \neq R \subseteq A \subseteq T$ satisfying $\delta > 0$ and $\mathrm{SOL}(A) = \mathrm{SOL}(A - R)$, decide whether vector d defined by* (3.14) *with $A' = A$ is* not *an R-deactivating direction for A.*

Proof. Membership in NP is analogous to Theorem 3.12. Since conditions (a) and (b) from Definition 3.1 are satisfied, the question boils down to deciding whether $d \notin M^*$.[67]

To show hardness, we proceed by reduction from the 3-coloring problem [116, §8.6.1, 84]: given a graph $G^* = (V^*, E^*)$, decide whether it is 3-colorable. Let $G = (V, C)$ be the graph sum [116, §8.1.2] (a.k.a. disjoint union of graphs) of G^* and K_4 where K_4 is the complete graph with 4 vertices.[68]

Let CSP A have the structure (V, D, C) where $|D| = 3$ and

$$A = \{\, (\{i,j\}, (k_i, k_j)) \mid \{i,j\} \in C,\ k \in D^{\{i,j\}},\ k_i \neq k_j \,\}. \tag{3.30}$$

Hence, any $x \in D^V$ can be interpreted as an assignment of colors to the nodes of G and $x \in \mathrm{SOL}(A)$ if and only if x is a 3-coloring of G. Since G contains K_4 as its subgraph, it is not 3-colorable and A is unsatisfiable. Hence, setting $R = A$ satisfies $\mathrm{SOL}(A - R) = \mathrm{SOL}(\emptyset) = \emptyset = \mathrm{SOL}(A)$.

For the purpose of our reduction, let us define $\delta = (|C| - 2)/2 > 0$. We will show that for such a setting, d is not an R-deactivating direction for A if and only if G^* is 3-colorable.

Plugging the above-defined sets A and R into the definition of d in (3.14) yields

$$F(x\,|\,d) = \sum_{\substack{\{i,j\}\in C \\ x_i \neq x_j}} (-1) + \sum_{\substack{\{i,j\}\in C \\ x_i = x_j}} \delta = \delta(|C| - \mathrm{COL}(x)) - \mathrm{COL}(x) \tag{3.31}$$

where $\mathrm{COL}(x) = |\{\, \{i,j\} \in C \mid x_i \neq x_j \,\}|$ is the number of edges in G whose adjacent vertices have different colors in assignment $x \in D^V$.

If G^* is 3-colorable, then there is $x \in D^V$ such that $\mathrm{COL}(x) = |C| - 1$. In other words, only for a single edge in $C - E^*$ (i.e., edge of graph K_4), the adjacent vertices are assigned the same color, so $F(x\,|\,d) = -|C|/2 < 0$ by (3.31) and definition of δ. Hence, d is not an R-deactivating direction for A.

For the other case, if G^* is not 3-colorable, then for any $x \in D^V$, $\mathrm{COL}(x) \leq |C| - 2$. The reason is that for at least one edge in K_4 and at least one edge in G^*, the adjacent vertices will be assigned the same color in any assignment. By substituting the value of δ and a simple manipulation of (3.31), one obtains

$$F(x\,|\,d) = \delta|C| - (\delta + 1)\,\mathrm{COL}(x) = |C|\,(|C| - 2 - \mathrm{COL}(x))\,/2 \geq 0 \tag{3.32}$$

where the term in brackets is non-negative due to $\mathrm{COL}(x) \leq |C| - 2$ for any $x \in D^V$. So, d is an R-deactivating direction for A. □

In connection to §3.3.1, a number of decision problems concerning the minimal CSP have been also proved hard. For recent results, we refer to [67, 57].

[67] This case cannot be reduced to the case in Corollary 3.2 because d in (3.14) has a special form. On the other hand, Theorem 3.13 directly implies Corollary 3.2 and consequently also Theorem 3.12, but we preferred to include more straightforward self-contained proofs for the previous statements.

[68] Informally, G is the graph obtained from G^* by adding 4 new vertices and including an edge between each pair of these new vertices.

3.5 Discussion

We have proposed a method to compute upper bounds on the WCSP. Following [92] (reviewed in §1.5.5), the WCSP is formulated as an optimization problem with a convex piecewise-affine objective and an exponential number of linear constraints that define the set of feasible solutions to be the super-reparametrizations of the input WCSP instance. Whenever the active-tuple CSP of a current WCSP instance is unsatisfiable, there exists an improving direction (in fact, a certificate of unsatisfiability of this CSP). We showed how these improving directions can be generated by constraint propagation (or, more generally, any methods to prove unsatisfiability of a CSP). We proved that super-reparametrizations are closely related to the dual cone to the well-known marginal polytope (Remark 3.1).

Special cases of our approach are the VAC / Augmenting DAG algorithm [33, 95, 146], which uses arc consistency, and the algorithm in [92], which uses cycle consistency. We have newly implemented the approach for singleton arc consistency, resulting in VSAC-SR algorithm. When compared to existing local consistency methods on a public dataset, VSAC-SR provides comparable or better bounds for many instances.

We expect our improved bounds to be useful to prune the search space during branch-and-bound search, when solving WCSP instances to optimality. However, we have done no experiments with this, so it is open whether during search the tighter bounds would outweigh the higher complexity of the algorithm. We leave this for future research. Our approach can be also useful to solve more WCSP instances even without search (similarly as the VAC algorithm solves all supermodular WCSPs without search – recall Fact 1.2) or, given a suitable primal heuristic, to solve WCSP instances approximately.

Our approach can be straightforwardly extended to WCSPs with different domain sizes [69] and some weights equal to minus infinity (i.e., some constraints being hard). Of course, further experiments would be needed to evaluate the quality of the bounds if infinite weights are allowed.

Finally, we presented a theoretical analysis of super-reparametrizations of WCSPs, describing the properties of optimal super-reparametrizations and characterizing the set of active-tuple CSPs induced by different optimal super-reparametrizations. For example, even an optimal super-reparametrization may change the set of optimal assignments, as shown in §3.3.2. Additionally, we have shown that general (i.e., possibly non-optimal) super-reparametrizations are only weakly related to the original WCSP instance.

[69] In fact, our implementation already supports different domain sizes. We did not present our theoretical results for this generalized setting only to simplify notation.

Chapter 4

Relation Between BCD and Local Consistencies

Let us recall from §1.5.4 that the fixed points of BCD algorithms applied to the dual LP relaxation of WCSP can be characterized by local consistency conditions, typically forms of arc consistency. Although the VAC / Augmenting DAG algorithm [33, 95, 146] is not a BCD method, its stopping points are also characterized by arc consistency, namely non-empty AC closure of the active-tuple CSP. In addition, we discovered in [50a, §3] that when BCD with the relative-interior rule is applied to the dual LP relaxation of SAT, it corresponds in a precise sense to unit propagation.

These results suggest that there is a close relation between BCD applied to a linear program and constraint propagation in a system of linear inequalities (and possibly equalities). In this chapter, we identify and describe this relation precisely. While constraint propagation in such a system can be done in many ways, we define a particular kind of constraint propagation rule that infers from a subset of inequalities that some of them are always active (i.e., always hold as equalities). We show that, for any linear program, Algorithm 2.1 using this rule and BCD with the relative-interior rule have fixed points of the same quality. In other words, the stopping points of Algorithm 2.1 with this rule cannot be improved by BCD with the relative-interior rule and vice versa. Additionally, we show that the kinds of local minima encountered in BCD (Definition 1.3) can be characterized by certain local consistency conditions.

Although the aforementioned propagation rule is applicable to any system of linear inequalities (and equalities) and our results hold for linear programs in any form, we show them for the primal-dual pair (1.1). Formally, we consider applying BCD to the dual (1.1) and applying Algorithm 2.1 to the primal-dual pair (1.1), as in §2.2. We have the following assumptions:

- both linear programs (1.1) are feasible and bounded,
- a dual-feasible solution y is available,
- a finite collection $\mathcal{B} \subseteq 2^{[m]}$ of blocks of dual variables (i.e., subsets of primal constraints) is provided.

For both methods, the goal is to improve this dual-feasible solution, ideally to make it optimal. For brevity, we will assume that the set of blocks $\mathcal{B}$, matrix A, and vectors b and c are fixed in §4.1 and §4.2.

This chapter is an improved version of [54a]. Throughout our explanation, we will refer to some results from the preliminaries, especially from §1.1, §1.2, and §1.3.

4.1 Propagation Rule and Local Consistency Condition

The precise form of constraint propagation (i.e., the set of inference rules) was neither specified nor restricted in §2.2. Even though propagation in a system of linear inequalities (and equalities) can be done in many ways, here we focus on a particular propagation rule:

From a subset of linear inequalities and equalities, infer which inequalities always hold with equality (i.e., are always active within the subset) and make them equalities.	(4.1)

After repeatedly applying this rule to different subsets, if a subset of inequalities (and equalities) is infeasible, then the original system was also infeasible.

This constraint propagation rule is used, under various names, in several existing methods:

- As we explained in §2.3, in the VAC / Augmenting DAG algorithm [33, 95, 146], primal problem (1.1) is (1.44) and complementary slackness conditions are (2.8). Indeed, in §2.3, we decided whether (under the conditions (2.8c)-(2.8d)), a single equality from (2.8a) implies that some inequality from (2.8c) is always active (i.e., which variables μ_t are implied to be zero).

- The approach that we overviewed in §2.4 to upper-bound the LP relaxation of weighted Max-SAT is another example. In detail, we inferred whether (under conditions (2.15e)-(2.15g)) a single inequality (or equality) from (2.15a)-(2.15d) implies that some of the inequalities (2.15g) are always active (i.e., if some variables are implied to be 0 or 1). There is a technical subtlety: in case of inequalities, i.e, (2.15a) and (2.15c), we also need to infer whether they are always active. Thus, after making all variables in a single inequality constraint (2.15a) or (2.15c) decided (recall Remark 2.7), one needs to check if the inequality is satisfied with strict inequality or with equality. In the latter case, the inequality should be made an equality. However, in this case, this has no effect on whether the rule will detect a contradiction or not because all the variables in this constraint are already decided by Remark 2.7.

- If the minimization of a convex piecewise-affine function is expressed as a linear program, then our method subsumes the sign relaxation technique introduced in [148, §3] and further developed in [48a, §2.2]. In more detail, one infers whether a single equality constraint (over non-negative variables that may be set to zero) implies whether some of the non-negative variables are implied to be zero.

For the particular case of system (2.2), rule (4.1) means that we initialize $J = \tau(y)$, choose a subset $B \in \mathcal{B} \subseteq 2^{[m]}$ of equalities (2.2a) and decide if the system

$$A^i x = b_i \qquad \forall i \in B \tag{4.2a}$$
$$x_j \geq 0 \qquad \forall j \in J \tag{4.2b}$$
$$x_j = 0 \qquad \forall j \in [n] - J \tag{4.2c}$$

implies [70] $x_j = 0$ for some $j \in J$. If so, we remove all such indices j from J and proceed with another block from $\mathcal{B}$. Eventually, if set J becomes so small that (4.2) becomes infeasible for some $B \in \mathcal{B}$, original system (2.2) was also infeasible, so y is not optimal and can be improved. In the following parts, we will analyze this propagation rule in detail.

[70] Recall from §1.1.3 that system (4.2) implies $x_j = 0$ if $x_j = 0$ holds for all x satisfying (4.2), i.e., $x_j \geq 0$ is always active within (4.2).

Definition 4.1. *Let $B \subseteq [m]$. A set $J \subseteq [n]$ is* B-consistent *if system (4.2) is feasible and does not imply $x_j = 0$ for any $j \in J$* [71]*, i.e., if the system*

$$A^i x = b_i \qquad \forall i \in B \tag{4.3a}$$
$$x_j > 0 \qquad \forall j \in J \tag{4.3b}$$
$$x_j = 0 \qquad \forall j \in [n] - J \tag{4.3c}$$

is feasible. For $\mathcal{B} \subseteq 2^{[m]}$, J is $\mathcal{B}$-consistent if it is B-consistent for each $B \in \mathcal{B}$.

Proposition 4.1. *Let $B \subseteq [m]$. If J and J' are B-consistent, so is $J \cup J'$.*

Proof. If (4.3) is satisfied by x and x' for J and J', respectively, then it is satisfied by $(x + x')/2$ for $J \cup J'$ due to $(x_j + x'_j)/2 > 0 \iff (x_j > 0 \vee x'_j > 0)$ which follows from non-negativity of the components of x and x'. □

The property identified in Proposition 4.1 is a usual attribute of local consistencies which was mentioned earlier in Property 1.1 (in §1.4.1). In words, Proposition 4.1 shows that, for a fixed $B \subseteq [m]$, the set of all B-consistent sets is closed under union. Hence, it is a join-semilattice where the order is given by the set inclusion and its join is the set union. However, it is not a (complete) lattice as it need not have a bottom element.

In order to overcome this, we add the bottom element $\bot$ by the lifting operation (see §1.3.1). Formally, the set $\mathcal{J} = 2^{[n]} \cup \{\bot\}$ can be equipped with the partial order $\sqsubseteq$ defined by

$$J \sqsubseteq J' \iff \big(J = \bot \vee (J, J' \subseteq [n] \wedge J \subseteq J')\big) \tag{4.4}$$

where $\vee$ and $\wedge$ represent logical disjunction and conjunction, respectively, and $J, J' \in \mathcal{J}$. Consequently, for any $B \subseteq [m]$, the set

$$\mathcal{J}_B = \{J \subseteq [n] \mid J \text{ is } B\text{-consistent}\} \cup \{\bot\} \subseteq \mathcal{J}, \tag{4.5}$$

partially ordered by $\sqsubseteq$, is a complete lattice by Theorem 1.11. The join operation of this lattice is the binary operation $\sqcup$ on $\mathcal{J}$ defined by [72]

$$J \sqcup J' = \begin{cases} J & \text{if } J' \sqsubseteq J \\ J' & \text{if } J \sqsubseteq J' \\ J \cup J' & \text{otherwise} \end{cases}. \tag{4.6}$$

Following Theorem 1.12b, the complete lattice $(\mathcal{J}_B, \sqsubseteq)$ gives rise to the dual closure operator $p_B \colon \mathcal{J} \to \mathcal{J}_B$ defined by

$$p_B(J) = \bigsqcup \{J' \in \mathcal{J}_B \mid J' \sqsubseteq J\}. \tag{4.7}$$

Note that the assumptions of Theorem 1.12b are satisfied because $\mathcal{J}_B \subseteq \mathcal{J}$ and the complete lattices $(\mathcal{J}_B, \sqsubseteq)$ and $(\mathcal{J}, \sqsubseteq)$ have the same join operation, namely $\sqcup$. We outline some important properties of p_B in Proposition 4.2 and Theorem 4.1.

[71] In other words, a set J is B-consistent if (4.2) is feasible and does not contain any always-active inequality.

[72] Recall that $\bot \sqsubseteq J$ for any $J \in \mathcal{J}$, so if the elements J, J' are not comparable by $\sqsubseteq$, they are subsets of $[n]$, hence set union in the last case in (4.6) is well-defined.

Proposition 4.2. *A set $J \subseteq [n]$ is B-consistent if and only if $p_B(J) = J$.*

Proof. Apply Corollary 1.4 to the complete lattices $(\mathcal{J}, \sqsubseteq)$ and $(\mathcal{J}_B, \sqsubseteq)$ and the associated dual closure operator p_B. Note that $\bot$ is not a subset of $[n]$, so it is not B-consistent despite $p_B(\bot) = \bot$. □

Theorem 4.1. *Let $J \subseteq [n]$ and $B \subseteq [m]$. The following are equivalent:*

(a) system (4.2) is feasible,

(b) $p_B(J) \neq \bot$,

(c) $p_B(J)$ is the union of all B-consistent subsets of J, i.e., $p_B(J)$ is the greatest B-consistent subset of J,

(d) $p_B(J) = J - \{\, j \in J \mid (4.2)$ implies $x_j = 0 \,\}$, i.e., system (4.2) implies $x_j = 0$ if and only if $j \in [n] - p_B(J)$.

If these statements hold, then $p_B(J) = [n] - \sigma(x^*)$ where $x^* \in \operatorname{ri} \{\, x \in \mathbb{R}^n \mid x$ satisfies (4.2) $\}$ and σ was defined in (1.2a).

Proof. For brevity of notation in this proof, for any $J \subseteq [n]$ and $B \subseteq [m]$, we define

$$X_B(J) = \{\, x \in \mathbb{R}^n \mid x \text{ satisfies (4.2)} \,\}. \tag{4.8}$$

It is easy to see from (4.2) that if $J \subseteq J'$, then $X_B(J) \subseteq X_B(J')$.[73]

(a) $\Longrightarrow$ (b): If (4.2) is feasible, then

$$J' = J - \{\, j \in J \mid (4.2) \text{ implies } x_j = 0 \,\} \tag{4.9}$$

is B-consistent by definition. Also, $J' \sqsubseteq J$, hence $p_B(J) \neq \bot$ due to $\bot \sqcup J' = J'$.

(b) $\Longrightarrow$ (c): If $p_B(J) \neq \bot$, $p_B(J) = \bigcup \{J' \in \mathcal{J}_B \mid J' \subseteq J\}$ because $\bot$ is the identity element of $\sqcup$ and operation $\sqcup$ coincides with $\cup$ when applied to subsets of $[n]$.

(c) $\Longrightarrow$ (a): By intensivity of p_B together with $p_B(J) \subseteq [n]$, we have that $p_B(J) \subseteq J$, so $X_B(p_B(J)) \subseteq X_B(J)$. By definition of a B-consistent set, system (4.2) is feasible for $p_B(J)$, i.e., $X_B(p_B(J)) \neq \emptyset$. Consequently, $X_B(J) \neq \emptyset$, i.e., (4.2) is feasible.

(c) $\Longrightarrow$ (d) by contradiction: Let J' be defined as in (4.9) and $p_B(J) \neq J'$. As discussed previously, J' is B-consistent and $J' \subseteq J$, so $p_B(J) \supsetneq J'$. Consequently, there exists $j \in p_B(J) - J'$. By definition of J', (4.2) implies $x_j = 0$, i.e., $x_j = 0$ holds for all $x \in X_B(J) = X_B(J')$. Moreover, $p_B(J) \subseteq J$ implies $X_B(p_B(J)) \subseteq X_B(J)$, thus $x_j = 0$ holds for all $x \in X_B(p_B(J))$. Having $j \in p_B(J)$ is contradictory with $p_B(J)$ being B-consistent (recall Proposition 4.2).

(d) $\Longrightarrow$ (b): Trivial because (4.9) is not equal to $\bot$.

The last statement follows from (d) by applying Theorem 1.6 to system (4.2). □

Following statement (d) in Theorem 4.1, p_B can be interpreted as a *propagator* for the propagation rule (4.1). Recalling §1.4.1.1, mapping p_B also satisfies the typical properties of propagators, i.e., intensivity and isotony. Moreover, p_B holds on to the intuitive idea

[73] One can show that the restriction of X_B to B-consistent sets is a bijection between B-consistent sets and non-empty faces of the polyhedron $X_B([n])$. Recalling [160, §2.2] that the *face lattice* of a polyhedron is the set of its faces partially ordered by the set inclusion, it also holds that the face lattice of $X_B([n])$ is order-isomorphic to the lattice $(\mathcal{J}_B, \sqsubseteq)$. To be precise, for this result it is expected that $\emptyset$ is a face of any polyhedron (as in [160, §2.1]) and that $\emptyset$ always belongs to the face lattice – this may not be the case in some formalisms [10, §8].

inputs: $J \in \mathcal{J}$, set of blocks $\mathcal{B} \subseteq 2^{[m]}$.
1 $J' := J$
2 **while** $\exists B \in \mathcal{B} : p_B(J') \neq J'$ **do**
3 Find such B.
4 $J' := p_B(J')$
5 **return** J'

Algorithm 4.1: Propagation algorithm $p_{\mathcal{B}}$ applied to input $J \in \mathcal{J}$.

of a propagator: it makes an inference based on local information (in this case, detects whether some inequality is always active based on a subset B of all equalities). Using this propagator, we formulate the *propagation algorithm* that enforces $\mathcal{B}$-consistency in Algorithm 4.1.

Since the set $\mathcal{J}$ is finite and propagators $p_B, B \in \mathcal{B}$ are intensive and isotone, Algorithm 4.1 is an instance of the more general Algorithm 1.1. By Theorem 1.13, the value returned by Algorithm 4.1 is independent of the order in which the propagators p_B are applied. Thus, we can denote the value returned by Algorithm 4.1 as $p_{\mathcal{B}}(J)$. For any $J \in \mathcal{J}$, $p_{\mathcal{B}}(J)$ is the greatest common fixed point of the propagators $p_B, B \in \mathcal{B}$ such that $p_{\mathcal{B}}(J) \sqsubseteq J$.

Proposition 4.3. *A set $J \subseteq [n]$ is $\mathcal{B}$-consistent if and only if $p_{\mathcal{B}}(J) = J$.*

Proof. Consider the following chain of equivalences:

$$J \text{ is } \mathcal{B}\text{-consistent} \iff \forall B \in \mathcal{B}\colon J \text{ is } B\text{-consistent} \tag{4.10a}$$
$$\iff \forall B \in \mathcal{B}\colon p_B(J) = J \tag{4.10b}$$
$$\iff p_{\mathcal{B}}(J) = J \tag{4.10c}$$

where (4.10a) is given by Definition 4.1 and (4.10b) follows from Proposition 4.2. Equivalence (4.10c) follows from the definition of $p_{\mathcal{B}}$ in Algorithm 4.1. □

Remark 4.1. *It can be shown that $p_{\mathcal{B}}$ is the dual closure operator associated with (recall Theorem 1.12b) the complete lattice of $\mathcal{B}$-consistent sets extended by $\perp$, i.e.,*

$$\{\, J \subseteq [n] \mid J \text{ is } \mathcal{B}\text{-consistent} \,\} \cup \{\perp\} = \{\, J \in \mathcal{J} \mid p_{\mathcal{B}}(J) = J \,\} \tag{4.11}$$

where the partial order is $\sqsubseteq$ and its join operation is again $\sqcup$.

Example 4.1. *Let $m = 3$, $n = 4$, $J = \{2, 3, 4\}$, $\mathcal{B} = \{\{1, 2\}, \{3\}\}$, and the system $Ax = b$ from* (2.2a) *be*

$$3x_1 + x_2 = 1 \tag{4.12a}$$
$$x_2 + 2x_3 = 1 \tag{4.12b}$$
$$-2x_2 + 5x_3 - x_4 = 0 \tag{4.12c}$$

where the equalities are numbered 1–3 from top to bottom. Recall that $x_2, x_3, x_4 \geq 0$ due to (2.2b) *and $x_1 = 0$ due to* (2.2c) *(note that $1 \notin J$). We now demonstrate the run of Algorithm 4.1.*

inputs: instance of problem (1.1), dual-feasible solution y, set of blocks $\mathcal{B}$.

while $p_{\mathcal{B}}(\tau(y)) = \bot$ **do**

  Find an improving direction $\bar{y}$ satisfying (2.3).

  Compute (possibly non-optimal) step size $\alpha > 0$ so that $y + \alpha\bar{y}$ is feasible.

  Update $y := y + \alpha\bar{y}$.

return y

Algorithm 4.2: Algorithmic scheme for approximate optimization of the dual (1.1) using the constraint propagation rule (4.1).

First, see that $p_{\{3\}}(J) = J$ because equality (4.12c) *can be easily satisfied by, e.g., $x_2 = x_3 = 1$ and $x_4 = 3$. So, this equality does not (together with non-negativity of x_2, x_3, x_4) imply that any of these variables are zero and $p_{\{3\}}$ is therefore not applied.*

Second, we apply propagator $p_{\{1,2\}}$. From (4.12a)*, it follows that $x_2 = 1$ due to $x_1 = 0$. Combining this with* (4.12b) *yields that $x_3 = 0$. Hence, the equalities* (4.12a)-(4.12b) *together with $x_1 = 0$ imply $x_3 = 0$, i.e., $p_{\{1,2\}}(J) = \{2, 4\}$ and x_3 is set to zero (via* (2.2c) *and $3 \notin p_{\{1,2\}}(J)$).*[74]

Third, we return to propagator $p_{\{3\}}$. Due to $x_3 = 0$, (4.12c) *can be satisfied only with $x_2 = x_4 = 0$. In other words,* (4.12c) *together with $x_3 = 0$ and non-negativity of x_2 and x_4 implies $x_2 = x_4 = 0$, i.e., all variables are now set to zero and $p_{\{3\}}(\{2, 4\}) = \emptyset$.*

Finally, we again apply propagator $p_{\{1,2\}}$ and find that equalities (4.12a)-(4.12b) *cannot be satisfied if all variables are zero. Hence, the propagation algorithm detected a contradiction, i.e., $p_{\{1,2\}}(\emptyset) = \bot$ and $p_{\mathcal{B}}(J) = \bot$.*[75] △

Let us recall system (2.2). If $p_{\mathcal{B}}(J) = \bot$, then (2.2) is infeasible. This follows from the fact that if a subsystem (4.2) implies $x_j = 0$ for some $j \in J$, then also the whole system (2.2) implies $x_j = 0$. Furthermore, if a subsystem (4.2) is infeasible, then so is (2.2). On the other hand, $p_{\mathcal{B}}(J) \neq \bot$ in general does not imply that (2.2) is feasible.

As discussed previously, if (2.2) with $J = \tau(y)$ is infeasible, there exists an improving direction $\bar{y}$ satisfying (2.3). We note that such an improving direction can be constructed from the history of the propagation, as we outlined in [54a, Appendix B].

We show how the propagation rule defined in this section can be applied to approximately optimize the dual (1.1) in Algorithm 4.2 which is a specialized version of the more general Algorithm 2.1.

4.2 Relation Between the Approaches

Now, we proceed to show a close connection between Algorithm 4.2 (based on constraint propagation) and BCD. In detail, we will prove that the stopping points of Algorithm 4.2 are precisely pre-ILMs of the dual (1.1) w.r.t. $\mathcal{B}$ and that ILMs of the dual (1.1) w.r.t. $\mathcal{B}$ can be characterized as points y for which the set $\tau(y)$ is $\mathcal{B}$-consistent. Then, using Theorem 1.8, it follows that BCD with the relative-interior rule (and consequently, any exact BCD method) cannot improve the objective in the stopping points of Algorithm 4.2.

[74] We do *not* set $x_2 = 1$ because we infer only which variables are zero. In other words, the reason is that there is no inequality $x_2 \leq 1$ (or $x_2 \geq 1$) in system (2.2) that could be made active.

[75] In general, it does not necessarily hold that $p_B(\emptyset) = \bot$. E.g., if $b = 0$ (i.e., system (4.3a) is homogeneous), then (4.3) is feasible even with $J = \emptyset$ and $p_B(\emptyset) = \emptyset \neq \bot$.

On the other hand, Algorithm 4.2 cannot improve the objective from any ILM or even pre-ILM w.r.t. $\mathcal{B}$.

We consider applying BCD to the dual linear program (1.1), so whenever we refer to any statement from §1.2, we assume that $Y = \{y \in \mathbb{R}^m \mid A^\top y \geq c\}$, $f(y) = b^\top y$, and $\bar{f}$ is defined as in (1.12).

Next, we formulate the dual (1.1) restricted to a block of variables $y|_B$, $B \subseteq [m]$ together with the corresponding primal [76], i.e.,

$$\max d^\top x \qquad \min \sum_{i \in B} b_i y_i \tag{4.13a}$$

$$A^i x = b_i \qquad y_i \in \mathbb{R} \qquad \forall i \in B \tag{4.13b}$$

$$x_j \geq 0 \qquad \sum_{i \in B} A_{ij} y_i \geq d_j \qquad \forall j \in [n] \tag{4.13c}$$

where $d_j = c_j - \sum_{i \in [m]-B} A_{ij} y_i$ are constants and A_{ij} is the element of A in row i and column j. Notice that the complementary slackness (and strict complementary slackness) conditions for block-optimality of $y|_B$, i.e., optimality of $y|_B$ for dual (4.13), are equivalent to (4.2) (and (4.3)) being feasible for $J = \tau(y)$, respectively, due to $\sum_{i \in B} A_{ij} y_i = d_j \iff j \in \tau(y)$ by definition of d_j. This observation allows us to show the connection of local minima, interior local minima, and optimizers of the dual (1.1) to B-consistent sets.

Lemma 4.1. *Let y be feasible for the dual (1.1) and let $B \subseteq [m]$. Then,*

(a) block of variables $y|_B$ satisfies (1.15) if and only if $p_B(\tau(y)) \neq \perp$,

(b) block of variables $y|_B$ satisfies (1.16) if and only if $p_B(\tau(y)) = \tau(y)$, i.e., $\tau(y)$ is B-consistent.

Proof. (a): Clearly, $y|_B$ is optimal for the dual (4.13) if and only if there exists x feasible for the primal (4.13) satisfying complementary slackness conditions. The complementary slackness conditions are equivalent to (4.2) for $J = \tau(y)$. By statements (a) and (b) in Theorem 4.1, (4.2) for $J = \tau(y)$ is feasible if and only if $p_B(\tau(y)) \neq \perp$.

(b): Block $y|_B$ satisfies (1.16) if and only if there exists an optimal solution x for the primal problem (4.13) satisfying strict complementary slackness conditions with $y|_B$. This is equivalent to feasibility of (4.3) for $J = \tau(y)$. By definition of B-consistency, this is equivalent to $\tau(y)$ being B-consistent, i.e., $p_B(\tau(y)) = \tau(y)$ by Proposition 4.2. □

Theorem 4.2. *Let y be feasible for the dual (1.1). Then,*

(a) y is an LM of the dual (1.1) w.r.t. $\mathcal{B}$ if and only if $p_B(\tau(y)) \neq \perp$ for all $B \in \mathcal{B}$,

(b) y is an ILM of the dual (1.1) w.r.t. $\mathcal{B}$ if and only if $p_{\mathcal{B}}(\tau(y)) = \tau(y)$, i.e., $\tau(y)$ is $\mathcal{B}$-consistent,

(c) y is an optimizer of the dual (1.1) if and only if $p_{[m]}(\tau(y)) \neq \perp$,

(d) y is in the relative interior of optimizers of the dual (1.1) if and only if $\tau(y)$ is $[m]$-consistent.

[76] The pair (4.13) is mutually dual. Of course, the primal (on the left-hand side in (4.13)) is different from the primal in (1.1). Namely, it contains only a subset of the equality constraints and the objective is different.

Proof. (a): By definition, y is an LM of dual (1.1) w.r.t. $\mathcal{B}$ if (1.15) holds for all $B \in \mathcal{B}$. Applying Lemma 4.1a, this is equivalent to $p_B(\tau(y)) \neq \perp$ for all $B \in \mathcal{B}$. Claim (b) is analogous, except that we use Lemma 4.1b and recall Proposition 4.3.

(c): Dual optimality of y is equivalent to (1.15) with $B = [m]$. The claim now follows from Lemma 4.1a. Statement (d) is obtained similarly, by Lemma 4.1b. □

4.2.1 Connection Between the Propagators and BCD Updates

Before we characterize pre-ILMs using the propagation algorithm, let us focus on a connection between the propagators p_B and block-coordinate updates (1.14).

Lemma 4.2. *Let y be a feasible point for the dual* (1.1) *and $B \subseteq [m]$. Let y^* be the result of BCD iteration* (1.14) *applied to y w.r.t. block B, i.e., $y^* = \big(y^*|_B, y|_{[m]-B}\big)$ where $y^*|_B \in \operatorname{ri}\operatorname{argmin}_{y' \in \mathbb{R}^B} \bar{f}\big(y', y|_{[m]-B}\big)$. Then,*

(a) $p_B(\tau(y)) = \perp$ if and only if $b^\top y > b^\top y^*$,

(b) if $p_B(\tau(y)) \neq \perp$, then $b^\top y = b^\top y^*$ and $p_B(\tau(y)) = \tau(y^*)$.

Proof. It follows from Lemma 4.1a that $p_B(\tau(y)) = \perp$ holds if and only if condition (1.15) was not satisfied i.e., $y|_B$ was not block-optimal and updating it results in improved objective, i.e., $b^\top y > b^\top y^*$. On the other hand, if (1.15) was satisfied, $y|_B$ was block-optimal and objective does not improve, i.e., $p_B(\tau(y)) \neq \perp$ implies $b^\top y = b^\top y^*$.

For the remaining statement, suppose that $y|_B$ was block-optimal, i.e., $p_B(\tau(y)) \neq \perp$ by Lemma 4.1a. Since any optimal solution to the primal (4.13) needs to satisfy complementary slackness with any dual-optimal $y|_B$, the set of primal-optimal solutions coincides with x feasible for (4.2) where $J = \tau(y)$. Let x^* be from the relative interior of (4.2) for $J = \tau(y)$, i.e., from the relative interior of optimizers of the primal (4.13). By the last statement in Theorem 4.1 together with feasibility of (4.2) for $J = \tau(y)$, we have that $p_B(\tau(y)) = [n] - \sigma(x^*) = \tau(y^*)$ where the second equality holds by strict complementary slackness (Theorem 1.3). □

Remark 4.2. *We now describe an interpretation of the relative-interior rule based on Lemmas 4.1 and 4.2. Let y be a dual-feasible solution. When performing a single update of y over a block $B \in \mathcal{B}$ to obtain y^*, precisely one of the following options happens:*

(a) If $y|_B$ already satisfies condition (1.16)*, then $\tau(y) = \tau(y^*)$ and $b^\top y = b^\top y^*$.*

(b) If $y|_B$ satisfies (1.15) *but not* (1.16)*, then $\tau(y) \supsetneq \tau(y^*)$ and $b^\top y = b^\top y^*$.*

(c) If $y|_B$ does not satisfy (1.15)*, then $b^\top y > b^\top y^*$.*

Suppose that we try to improve y by BCD updates (1.14) *where the sequence $(B_k)_{k=1}^\infty$ is chosen such that each $B \in \mathcal{B}$ occurs in it an infinite number of times. Since the set $\tau(y)$ may shrink only a finite number of times, case (b) from the previous list can happen only a finite number of times in a row. Hence, when applying relative-interior updates* (1.14) *for consecutive $B \in \mathcal{B}$ from the sequence, either the objective $b^\top y$ improves after a finite number of iterations or one needs to attain an ILM w.r.t. $\mathcal{B}$ (cf. Theorem 1.8).*

The connection between the propagators and BCD for multiple consecutive iterations is given by the following theorem.

Theorem 4.3. *Let y^1 be a feasible point for the dual (1.1). Let $(B_k)_{k=1}^{\infty}$ be a sequence of blocks from $\mathcal{B}$. Let $(y^k)_{k=1}^{\infty}$ be a sequence satisfying (1.14) w.r.t. blocks $(B_k)_{k=1}^{\infty}$. Let $(J_k)_{k=1}^{\infty}$ be the sequence defined by $J_1 = \tau(y^1)$ and $J_{k+1} = p_{B_k}(J_k)$ for all $k \geq 1$. Then, for every k, it holds that:*

(a) if $J_k = \bot$, then $b^\top y^k < b^\top y^1$,

(b) if $J_k \neq \bot$, then $b^\top y^k = b^\top y^1$ and $\tau(y^k) = J_k$.

Proof. We proceed by induction. The base case with $k = 1$ holds by definition due to $b^\top y^1 = b^\top y^1$ and $J_1 = \tau(y^1) \neq \bot$. For the inductive step with $k \geq 1$, y^{k+1} originated from y^k by updating block B_k and $J_{k+1} = p_{B_k}(J_k)$. We consider the following cases:

- If $p_{B_k}(\tau(y^k)) \neq \bot$ and $J_k \neq \bot$, we have that $b^\top y^k = b^\top y^{k+1}$ and $p_{B_k}(\tau(y^k)) = \tau(y^{k+1})$ by Lemma 4.2b. By induction hypothesis, due to $J_k \neq \bot$, we have $b^\top y^1 = b^\top y^k$ and $\tau(y^k) = J_k$. Therefore, $\tau(y^{k+1}) = p_{B_k}(\tau(y^k)) = p_{B_k}(J_k) = J_{k+1} \neq \bot$ and $b^\top y^1 = b^\top y^k = b^\top y^{k+1}$.
- If $p_{B_k}(\tau(y^k)) = \bot$ and $J_k \neq \bot$, then by Lemma 4.2a, $b^\top y^k > b^\top y^{k+1}$. Similarly as in the previous case, by induction hypothesis: since $J_k \neq \bot$, it holds that $b^\top y^1 = b^\top y^k$ and $\tau(y^k) = J_k$. Thus, $p_{B_k}(\tau(y^k)) = p_{B_k}(J_k) = J_{k+1} = \bot$ and $b^\top y^1 = b^\top y^k > b^\top y^{k+1}$.
- If $J_k = \bot$, by induction hypothesis $b^\top y^k < b^\top y^1$ and by isotony of propagator, $J_{k+1} = p_{B_k}(J_k) = p_{B_k}(\bot) = \bot$. Since updates (1.14) never worsen the objective, $b^\top y^1 > b^\top y^k \geq b^\top y^{k+1}$. □

Example 4.2. *Let us now consider the same instance as in Example 4.1. Here, we will illustrate that BCD updates with the relative-interior rule are in correspondence with the actions of the propagators that were shown in Example 4.1.*

Let the matrix A and vector b be defined by (4.12) and $\mathcal{B} = \{\{1,2\},\{3\}\}$, i.e., as in Example 4.1. In addition, we define $c = (3,6,6,0)$ so that the dual (1.1) reads

$$\min y_1 + y_2 \tag{4.14a}$$

$$3y_1 \geq 3 \tag{4.14b}$$

$$y_1 + y_2 - 2y_3 \geq 6 \tag{4.14c}$$

$$2y_2 + 5y_3 \geq 6 \tag{4.14d}$$

$$-y_3 \geq 0 \tag{4.14e}$$

where the constraints (4.14b)-(4.14e) correspond to the primal variables x_1-x_4.

For $y^1 = (3,3,0)$, the set of active dual constraints is $J = \tau(y^1) = \{2,3,4\}$ (which is the same initial set J as in Example 4.1). We now initialize BCD at y^1. As given by the previously shown theorems, the set of active dual constraints after each update of block B will be the same as if the propagator p_B was applied (to be precise, this holds until the propagator does not return $\bot$). Moreover, the propagator returns $\bot$ if and only if the corresponding BCD update improves the objective.

For $B = \{3\}$, y^1 is block-optimal and also in the relative interior of block-optimizers which is in correspondence to $p_{\{3\}}(\tau(y^1)) = \tau(y^1)$ (cf. Lemma 4.1b for y^1 and $B = \{3\}$).

Next, y^1 is block-optimal for $B = \{1,2\}$ but not in the relative interior of block-optimizers. Updating the values of $y^1|_{\{1,2\}}$ to the relative interior of block-optimizers results in, e.g., $y^2 = (2,4,0)$. Although the relative interior contains multiple elements,

we will have $p_{\{1,2\}}(\tau(y^1)) = \tau(y^2) = \{2,4\}$ *for any of them (cf. Lemma 4.2b for* y^1 *and* $B = \{1,2\}$*).*

Now, y^2 *is block-optimal for* $B = \{3\}$ *but not in the relative interior of block-optimizers. Performing a relative-interior update for this coordinate results in, e.g.,* $y^3 = (2, 4, -\frac{1}{5})$*. So, we have* $p_{\{3\}}(\tau(y^2)) = \tau(y^3) = \emptyset$ *(cf. Lemma 4.2b for* y^2 *and* $B = \{3\}$*).*

Finally, y^3 *is not block-optimal for* $B = \{1,2\}$*. Updating this block with the relative-interior rule results in, e.g.,* $y^4 = (2-\frac{1}{5}, 4-\frac{1}{5}, -\frac{1}{5})$ *and improves the objective from* $b^\top y^3 = 6$ *to* $b^\top y^4 = 6 - \frac{2}{5}$*. Note that* $p_{\{1,2\}}(\tau(y^3)) = \perp$ *(cf. Lemma 4.2a for* y^3 *and* $B = \{3\}$*).* △

4.2.2 Pre-interior Local Minima and Overview of Results

We are now able to combine the previous results with the properties of pre-ILMs given in Theorem 1.8 to obtain their characterization using the propagation algorithm.

Theorem 4.4. *For any* y *feasible for the dual* (1.1)*,* y *is a pre-ILM of dual* (1.1) *w.r.t.* $\mathcal{B}$ *if and only if* $p_{\mathcal{B}}(\tau(y)) \neq \perp$*.*

Proof. Let $(B_k)_{k=1}^{l}$ be a finite sequence of blocks $B_k \in \mathcal{B}$ such that

$$p_{B_l}(p_{B_{l-1}}(\dots p_{B_2}(p_{B_1}(\tau(y)))\dots)) = p_{\mathcal{B}}(\tau(y)). \tag{4.15}$$

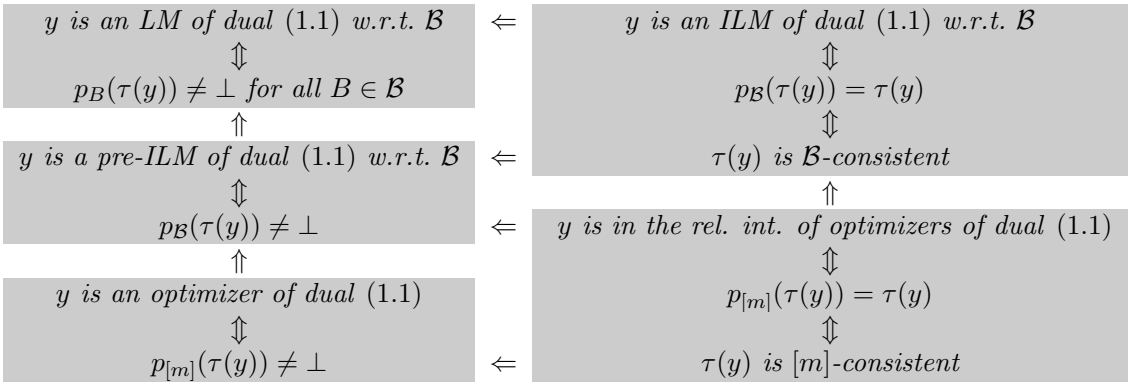
Note, $(B_k)_{k=1}^{l}$ can be obtained, e.g., by storing the individual blocks as they were applied in Algorithm 4.1. Let us extend this finite sequence into an infinite sequence $(B_k)_{k=1}^{\infty}$ so that $(B_k)_{k=1}^{\infty}$ contains each element of $\mathcal{B}$ an infinite number of times. Next, define sequences $(J_k)_{k=1}^{\infty}$ and $(y^k)_{k=1}^{\infty}$ based on the sequence $(B_k)_{k=1}^{\infty}$ and $y^1 = y$ as in Theorem 4.3.

If $J_{l+1} = p_{\mathcal{B}}(\tau(y)) = \perp$, Theorem 4.3a yields $b^\top y^1 > b^\top y^{l+1}$. This together with Theorem 1.8c implies that $y = y^1$ is not a pre-ILM because updates (1.14) cannot improve the objective from a pre-ILM.

On the other hand, if $J_{l+1} = p_{\mathcal{B}}(\tau(y)) \neq \perp$, then $p_B(J_{l+1}) = J_{l+1} \neq \perp$ for all $B \in \mathcal{B}$, hence $J_k \neq \perp$ for all k, so $b^\top y^1 = b^\top y^k$ by Theorem 4.3b. Combining this with Theorem 1.8d yields that $y = y^1$ is a pre-ILM. □

Corollary 4.1 summarizes and connects the results given by Theorems 4.2 and 4.4.

Corollary 4.1. *Let* y *be a feasible point for the dual* (1.1)*. The following implications and equivalences hold (for better readability, equivalent statements are boxed in gray):*



Proof. The equivalences are given by Theorems 4.2 and 4.4. To link the individual statements together, let $J \subseteq [n]$. We prove the following implications:

$$\begin{array}{ccccc} p_{[m]}(J) = J & \overset{\text{(a)}}{\Longrightarrow} & p_{\mathcal{B}}(J) = J & & \\ \text{(b)}\,\Big\Downarrow & & \text{(c)}\,\Big\Downarrow & & \\ p_{[m]}(J) \neq \bot & \overset{\text{(d)}}{\Longrightarrow} & p_{\mathcal{B}}(J) \neq \bot & \overset{\text{(e)}}{\Longrightarrow} & \forall B \in \mathcal{B}\colon p_B(J) \neq \bot. \end{array} \tag{4.16}$$

To show (a), $p_{[m]}(J) = J$ is equivalent to J being $[m]$-consistent by Proposition 4.2, i.e., (4.3) is feasible for $B = [m]$ by definition. This clearly implies feasibility of (4.3) for any $B \subseteq [m]$ as then (4.3) contains only a subset of all the equalities, so J is $\mathcal{B}$-consistent which is equivalent to $p_{\mathcal{B}}(J) = J$ by Proposition 4.3.

Implications (b) and (c) are trivial due to $J \neq \bot$.

For implication (d), denote $p_{[m]}(J) = J' \neq \bot$, so (4.3) is feasible for J' and $B = [m]$ because J' is $[m]$-consistent by Proposition 4.2. As in the previous case, this implies $p_{\mathcal{B}}(J') = J'$. By (4.10), J' is a common fixed point of propagators p_B, $B \in \mathcal{B}$ and a subset of J. As discussed in §4.1, $p_{\mathcal{B}}(J)$ is the greatest common fixed point of these propagators such that $p_{\mathcal{B}}(J) \sqsubseteq J$, so $\bot \neq J' \sqsubseteq p_{\mathcal{B}}(J)$. Due to $\bot \sqsubseteq J'$, we have $p_{\mathcal{B}}(J) \neq \bot$.

Implication (e) can be easily proved by contrapositive: if $p_B(J) = \bot$ for some $B \in \mathcal{B}$, then the propagation algorithm (Algorithm 4.1) terminates with $p_{\mathcal{B}}(J) = \bot$. □

4.3 Other Forms of Linear Programs

As we previously noted in §1.1 and §1.2.3, it is well known that linear programs come in different forms which can be easily transformed to each other, preserving global optima [110, §2.1]. One can ask if the propagation algorithm can be formulated and the relation with BCD holds also for different forms than (1.1). This question is non-trivial because transformations that preserve global minima do not necessarily preserve (pre-)interior local minima, as we discussed in §1.2.3. Nevertheless, we argue that, independently of the form of the LP problem, the two approaches are still related in the same way if we use the propagation rule (4.1).

4.3.1 Inequalities and Non-negative Variables

For example, consider the primal-dual pair

$$\begin{aligned} \max\ & c^\top x && \min\ b^\top y && \text{(4.17a)} \\ Ax &\leq b && y &\geq 0 && \text{(4.17b)} \\ x &\geq 0 && A^\top y &\geq c. && \text{(4.17c)} \end{aligned}$$

The primal problem (4.17) (on the left-hand side) can be equivalently reformulated [110, §2.1, 103, §4.1] by introducing non-negative slack variables $s_i \geq 0$, $i \in [m]$ which yields

the primal-dual pair

$$\max c^\top x \qquad \min b^\top y \tag{4.18a}$$
$$Ax + s = b \qquad y \in \mathbb{R}^m \tag{4.18b}$$
$$x \geq 0 \qquad A^\top y \geq c \tag{4.18c}$$
$$s \geq 0 \qquad y \geq 0 \tag{4.18d}$$

which is in the form (1.1). See that the duals (4.17) and (4.18) are identical, hence also BCD applied to them is identical.

The propagation rule (4.1) for the case of (4.18) corresponds to deciding which s_i and x_j are implied to be zero. Clearly, setting $s_i = 0$ corresponds to setting $A^i x = b_i$ and enforcing $s_i > 0$ is equivalent to $A^i x < b_i$. Thus, instead of rewriting (4.17) into (4.18), we can apply propagation directly to the primal (4.17) except that when considering system (4.2) for some $B \in \mathcal{B}$, we will instead of a single set J use two sets, $J_1 \subseteq [m]$ and $J_2 \subseteq [n]$, that indicate which of the original inequalities need to hold with equality, i.e., we will use

$$A^i x \leq b_i \qquad \forall i \in J_1 \cap B \tag{4.19a}$$
$$A^i x = b_i \qquad \forall i \in ([m] - J_1) \cap B \tag{4.19b}$$
$$x_j \geq 0 \qquad \forall j \in J_2 \tag{4.19c}$$
$$x_j = 0 \qquad \forall j \in [n] - J_2 \tag{4.19d}$$

instead of (4.2). Deciding which inequalities among (4.19a) in (4.19) are always active by considering a set $J_1 \subseteq [m]$ is in one-to-one correspondence with deciding which inequalities $s_i \geq 0$ in

$$A^i x + s_i = b_i \qquad \forall i \in B \tag{4.20a}$$
$$s_i \geq 0 \qquad \forall i \in J_1 \tag{4.20b}$$
$$s_i = 0 \qquad \forall i \in [m] - J_1 \tag{4.20c}$$
$$x_j \geq 0 \qquad \forall j \in J_2 \tag{4.20d}$$
$$x_j = 0 \qquad \forall j \in [n] - J_2 \tag{4.20e}$$

are always active. In particular, (4.19) contains an always-active inequality if and only if (4.20) contains an always-active inequality.

4.3.2 Inequalities and Real-Valued Variables

The second common form of a primal-dual pair is

$$\max c^\top x \qquad \min b^\top y \tag{4.21a}$$
$$Ax \leq b \qquad y \geq 0 \tag{4.21b}$$
$$x \in \mathbb{R}^n \qquad A^\top y = c. \tag{4.21c}$$

By complementary slackness, y is optimal for the dual if and only if there exists $x \in \mathbb{R}^n$ such that

$$A^i x \leq b_i \qquad \forall i \in \sigma'(y) \tag{4.22a}$$
$$A^i x = b_i \qquad \forall i \in [m] - \sigma'(y) \tag{4.22b}$$

where $\sigma'(y) = \{i \in [m] \mid y_i = 0\}$ (in analogy to (1.2a)).

From this point, we could completely repeat the reasoning in §4.1 and prove the same theorems as in §4.2. We would infer from

$$A^i x \leq b_i \qquad \forall i \in \sigma'(y) \cap B \tag{4.23a}$$

$$A^i x = b_i \qquad \forall i \in ([m] - \sigma'(y)) \cap B \tag{4.23b}$$

whether some of the inequalities (4.23a) are always active. As an example, for any y feasible for the dual (4.21): y is an ILM of the dual (4.21) w.r.t. $\mathcal{B}$ if and only if for each $B \in \mathcal{B}$, (4.23) is feasible and no inequality from (4.23a) is always active within the system (4.23).

4.4 Discussion

In the previous sections, we have shown that, for any $\mathcal{B} \subseteq 2^{[m]}$, pre-ILMs w.r.t. $\mathcal{B}$ of the dual (1.1) coincide with the stopping points of Algorithm 4.2. Perhaps most importantly, we characterized the types of local and global minima using the local consistency from §4.1 in Corollary 4.1. Let us now comment on how these results apply in practice to LP relaxations that we met earlier.

4.4.1 Weighted CSP

We proved in [151a, §6] that

- φ is an ILM of (1.43) (w.r.t. individual variables) if and only if $A^*(g^\varphi)$ is arc consistent (in the sense of (1.29)) and non-empty [77] and
- φ is a pre-ILM of (1.43) (w.r.t. individual variables) if and only if $A^*(g^\varphi)$ has non-empty AC closure.

These results can be now seen as a consequence of the characterization in Corollary 4.1 since propagation rule (4.1) (with appropriately defined blocks) applied to system (2.8) corresponds to enforcing arc consistency, as we discussed earlier at the beginning of §4.1.

In addition, we showed [151a, §6] that max-sum diffusion [96, 146] and even general max-marginal averaging approaches (such as TRW-S [88]) satisfy the relative-interior rule. We also found out that MPLP [65] does not satisfy the relative-interior rule (i.e., it may choose a block-optimizer from the relative boundary), but its fixed points are pre-ILMs [151a, §6.2]. As discussed in §4.1, the VAC [33] and Augmenting DAG algorithm [95, 146] are instances of Algorithm 4.2. Therefore, our results subsume the relation between these BCD methods and the VAC / Augmenting DAG algorithm.

Furthermore, we are able to show that other BCD methods applied to a dual LP relaxation of WCSP also satisfy the relative-interior rule (e.g., the generalization of max-sum diffusion to higher-order models in [149, §6.2] or, possibly under additional conditions, the convex max-product algorithm [135, §3.3, 111, Algorithm 1]) or that their fixed points are pre-ILMs (e.g., SRMP [87] or MPLP++ [134]).

Since the updates of the generalized max-sum diffusion from [149, §6.2] satisfy the relative-interior rule, it immediately follows from the discussion in [149, §6.2] (combined with our previous results here) that BCD with the relative-interior rule in fact enforces

[77] Non-emptiness is in fact not necessary here because CSP $A^*(f)$ is non-empty for any WCSP $f \in \mathbb{R}^T$. We state it in this form only for the purposes of comparison with Corollary 5.3 given later.

pairwise consistency of the active-tuple CSP if a dual LP relaxation with a different coupling scheme (recall Remark 1.9) is optimized.

Even though the constraint-propagation-based algorithm presented in §3.2 can be interpreted as detecting *some* always-active inequalities in a subsystem of the complementary slackness conditions, it does *not* find all of them even if unsatisfiability of CSP $A^*(f)$ is always detected. Consequently, constraint propagation that was used in §3.2 is weaker than the one proposed in §4.1. We will return to this in more detail later in §5.2.

4.4.2 SAT Problem

Next, we review one of our results from [50a] where we noticed a connection between BCD and unit propagation. We will argue next that this connection can be explained by the general relation between BCD and constraint propagation identified in §4.2. This subsection is to some extent based on [50a, §3].

Let us consider the SAT problem [21] (a.k.a. the Boolean satisfiability problem) for a set of clauses C. Following the notation from §2.4, this problem can be formulated [110, Example 13.4] as finding an assignment $x \in \{0,1\}^V$ such that

$$\sum_{i \in V_c^+} x_i + \sum_{i \in V_c^-} (1 - x_i) \geq 1 \qquad \forall c \in C. \tag{4.24}$$

Deciding whether such an assignment exists is well known to be NP-complete [32, 84].

A popular technique used to tackle this problem is *unit propagation* [153, §4]. Unit propagation can simplify a set of clauses by gradually assigning values to variables that can take only a single specific value in order to result in a satisfying assignment. Noting that the LP relaxation of (4.24) is (2.15) where $C_{\geq 1}(y) = C$ and $X_U(y) = V$ (while the remaining sets $C_{=1}(y)$, $C_{\leq 1}(y)$, $C_{=0}(y)$, $X_1(y)$, and $X_0(y)$ are empty), unit propagation in fact applies propagation rules A1 and A2 from Table 2.1 to make some of the variables decided or detect that the instance is unsatisfiable. Note, the other rules B1-C2 are not applicable due to $C_{=1}(y) = C_{\leq 1}(y) = C_{=0}(y) = \emptyset$. More on unit propagation can be found, e.g., in [157] or [76, §3.2.3, 21, §3.5.2] (where it is called *unit resolution*).

As already indicated, the LP relaxation of SAT is the feasibility problem of finding $x \in [0,1]^V$ satisfying (4.24). The dual LP relaxation can be expressed in a form similar to (2.14), namely

$$\min \sum_{i \in V} \max\{y(C_i^+), y(C_i^-)\} - y(C) \tag{4.25a}$$

$$y_c \geq 0 \qquad \forall c \in C. \tag{4.25b}$$

Suppose that we initialize $y = 0$ and sequentially update each y_c, $c \in C$ to be in the relative interior of the set of coordinate-wise optimizers, i.e., we apply BCD with the relative-interior rule to (4.25) where each block contains only a single variable y_c, $c \in C$. After $|C| + 1$ cycles [78] of updates, one of the following two options happens:

- The dual objective (4.25a) improves, i.e., decreases below 0 which was the initial objective for $y = 0$. This is the case if and only if unit propagation would detect contradiction if applied to this instance.

[78] By a cycle of updates, we mean performing the update (1.14) for each variable y_c, $c \in C$ in some predefined order.

- The dual objective (4.25a) stays zero. In this case, let us define the sets $X_0(y)$, $X_1(y)$, and $X_U(y)$ as in (2.15e)-(2.15g) based on the current y. Sets $X_0(y)$ and $X_1(y)$ contain precisely those variables that would be set by unit propagation to 0 and 1, respectively. The set $X_U(y)$ contains those variables whose value would not be determined by unit propagation. Note, in this case, unit propagation would not detect contradiction.

We now discuss how this phenomenon follows from our previous results. The LP relaxation of SAT has zero objective, so it can be either feasible with optimal value zero, or infeasible. To find out which case it is, one can use unit propagation: if no contradiction is detected by applying rules A1 and A2 from Table 2.1, then, for each $c \in C$, either all variables in V_c are decided and the clause is satisfied, or there are at least two undecided variables in V_c. Consequently, setting all undecided variables to $\frac{1}{2}$ results in an assignment $x \in [0,1]^V$ satisfying (4.24). All in all, unit propagation can decide the feasibility of the LP relaxation of SAT, i.e., the rules A1 and A2 are refutation complete for the system of linear inequalities given by the LP relaxation of any SAT instance.

Next, the dual (4.25) is always feasible, so it can be either feasible with zero optimal value (if the primal is feasible), or unbounded (if the primal is infeasible). The complementary slackness conditions for optimality of $y = 0$ for (4.25) are equivalent to feasibility of (4.24) subject to $x \in [0,1]^V$ (i.e., feasibility of LP relaxation of SAT). Applying the general propagation rule (4.1) to these conditions corresponds to rules A1 and A2 from Table 2.1 that in fact perform unit propagation in (4.24). By §4.2.1, these rules detect contradiction if and only if the dual objective improves. And, if the dual objective does not improve, then point y obtained after $|C| + 1$ cycles of updates is an ILM of (4.25) (w.r.t. individual variables) by Theorem 1.9 and, by §4.2.1, the sets $X_0(y)$, $X_1(y)$, and $X_U(y)$ defined by the ILM y (via (2.15e)-(2.15g)) coincide with the sets of variables that unit propagation sets to 0, 1, and does not decide, respectively.

Of course, there exist faster algorithms for unit propagation (see, e.g., [157]) than applying BCD with the relative-interior rule to the dual LP relaxation (4.25). However, this result is of theoretical importance as it provided a starting point for the more general results that were described in §4.2.

Remark 4.3. *This result is also connected to the fact that applying the rules from Table 2.1 corresponds to enforcing arc consistency of* (4.24) *seen as a CSP (recall Remark 2.9). The connection between unit propagation in the SAT problem and arc consistency in its different formulations as a CSP was studied in [141].*

4.4.3 Weighted Max-SAT

As explained in §4.1, the propagation rules outlined in §2.4.1 for the LP relaxation of weighted Max-SAT are (up to technical details) an example of the specific constraint propagation rule (4.1) applied to system (2.15). Consequently, pre-ILMs of (2.14) w.r.t. individual variables [79] coincide with the stopping points of the algorithm from §2.4.3.

[79] In detail, the system from which we infer the always-active inequalities in case of the rules in Table 2.1 contains all constraints $0 \leq x \leq 1$ and $0 \leq z \leq 1$, and a single primal constraint from (2.11b) (where some of the inequalities may in fact be equalities, as in the complementary slackness conditions (2.15)). This means that each block of dual variables in BCD consists of all p and q variables, and a single y_c variable where c is the index of the constraint from (2.11b). Recalling Fact 1.1, applying BCD to the dual (2.11) with these blocks of variables is in correspondence to applying BCD to (2.14) along individual variables y_c.

We performed experiments with coordinate-wise minimization (following the relative-interior rule) of the dual LP relaxation of weighted partial[80] Max-SAT in [53a, §4.1] and observed that the objective was frequently not too far from the optimum of the LP relaxation. The results of this chapter show that the fixed points of BCD are of the same nature as stopping points of the method from §2.4 (possibly up to the differences caused by allowing hard clauses). We theoretically analyzed BCD with the relative-interior rule applied to the dual LP relaxation of weighted partial Max-SAT in [50a, §2].

Interestingly, in [50a, §4], we identified a connection between BCD applied to the dual LP relaxation of weighted Max-SAT and a modified version of the *dominating unit-clause rule* [153, §4.3, 109, §3.1] which finds a part of some optimal solution, i.e., it finds an assignment to a subset of variables that can be extended to some optimal solution of the weighted Max-SAT problem. Similarly to §4.4.2, there is a relation between the sets of variables that are set to 0 and 1 by this rule and the sets $X_0(y)$ and $X_1(y)$ defined by (2.15f) and (2.15e) for an ILM y of (2.14), respectively. In detail, the former sets are subsets of the latter sets [50a, Theorem 2]. However, we are currently not able to link this phenomenon to our results from §4.2 or provide a simple proof and refer to [50a, §4] for details.

[80] Weighted partial Max-SAT is a generalization of both weighted Max-SAT and SAT. In weighted partial Max-SAT, there are two types of clauses – soft and hard. Each soft clause is assigned a weight and the task is to find an assignment satisfying all hard clauses such that the sum of weights of satisfied soft clauses is maximized [21, §19.2].

Chapter 5

Linear Programs Optimally Solvable by BCD

As we reviewed in §1.2, the fixed points of BCD need not be global minima for convex optimization problems (or even for linear programs). Moreover, finding new classes of linear programs that are solvable by BCD (more precisely, linear programs where (pre-)ILMs are global minima of the problem) seems to be hard. Despite these limitations, we were able to find new such classes and present our results in this chapter.

To this end, we use our previous results from §4.2 and, in §5.1, link refutation-completeness of the propagation rule (4.1) to optimality of BCD with the relative-interior rule. This provides a new proof technique for finding (classes of) linear programs optimally solvable by BCD. Next, in §5.2, we exemplify this technique on the optimization problem (1.45) and show that (pre-)ILMs of (1.45) w.r.t. individual variables are global minima – however, solving this problem (or even performing coordinate-wise updates with the relative-interior rule) is likely intractable. After that, in §5.3, we find two more classes that are defined by limiting the problem structure and form of the constraints. These classes include, e.g., the maximum flow problem or LP relaxations of certain hard combinatorial problems. Nevertheless, thanks to the special structure of these linear programs, they can be also solved by efficient combinatorial algorithms. Finally, we exemplify in §5.4 why reformulating a problem has an impact on its solvability by BCD.

This chapter is mainly based on [53a] (which is an improved version of [51a]) and contains some insights from [54a, 151a]. Results described in §5.2 have not been published before.

5.1 Refutation-Completeness and Optimality of BCD

First, let us return to the general constraint-propagation-based approach that we formulated in Algorithm 2.1. If the chosen constraint propagation rule is refutation complete, i.e., it is always able to detect infeasibility of complementary slackness conditions (2.2), then any stopping point of Algorithm 2.1 is clearly a global minimum of the dual (1.1). Applying this observation to the propagation rule (4.1), which was analyzed in §4.1, yields the following corollary.

Corollary 5.1. *Let $\mathcal{B} \subseteq 2^{[m]}$. The following are equivalent:*

(a) every ILM y of the dual (1.1) *w.r.t. $\mathcal{B}$ is a global minimum of the dual* (1.1)*,*

(b) every pre-ILM y of the dual (1.1) *w.r.t. $\mathcal{B}$ is a global minimum of the dual* (1.1)*,*

(c) for all y feasible for the dual (1.1)*: $p_{[m]}(\tau(y)) = \bot \implies p_{\mathcal{B}}(\tau(y)) = \bot$, i.e., if* (2.2) *is infeasible for $J = \tau(y)$, then the propagation algorithm detects it ($p_{\mathcal{B}}(\tau(y)) = \bot$),*

(d) for all y feasible for the dual (1.1)*: if $J = \tau(y)$ is $\mathcal{B}$-consistent, then* (2.2) *is feasible.*

Proof. The equivalence (a) $\iff$ (b) is given by Corollary 1.3 and (a) $\iff$ (d) follows from Theorem 4.2b. The contrapositive of statement (c) is $p_{\mathcal{B}}(\tau(y)) \neq \bot \implies p_{[m]}(\tau(y)) \neq \bot$.

The equivalence (b) $\iff$ (c) is now immediate from the implications and equivalences summarized in Corollary 4.1. □

This result shows that the question whether all the (pre-)ILMs are global minima for a given linear program can be reformulated as the question whether the constraint propagation rule (4.1) is refutation complete for a certain class of systems of linear inequalities and equalities where this class is given by the complementary slackness conditions.

5.2 Solving Weighted CSP by BCD

Now, we apply this observation to the optimization problem (1.45) (i.e., (3.7)) which was approximately optimized using constraint propagation in §3.2. Here, we show that (pre-)ILMs w.r.t. individual variables of (1.45) are in fact global minima. More strongly, this holds even for the LP formulation of (1.45). Next, we characterize pre-ILMs and ILMs of the LP formulation using positive consistency. Finally, we derive coordinate-wise updates satisfying the relative-interior rule, discuss their convergence, and prove that executing a single update is likely intractable.

Using the well-known transformation from §1.1.2, (1.45) can be expressed as a linear program with an exponential number of constraints. This linear program is the right-hand problem of the pair

$$\max \sum_{x \in D^V} F(x\,|\,g)\,\lambda(x) \qquad \min \sum_{S \in C} z_S \tag{5.1a}$$

$$\sum_{k \in D^S} \mu_S(k) = 1 \qquad z_S \in \mathbb{R} \qquad \forall S \in C \tag{5.1b}$$

$$\sum_{\substack{x \in D^V \\ x|_S = k}} \lambda(x) = \mu_S(k) \qquad f_S(k) \in \mathbb{R} \qquad \forall (S,k) \in T \tag{5.1c}$$

$$\lambda(x) \geq 0 \qquad F(x\,|\,f) \geq F(x\,|\,g) \qquad \forall x \in D^V \tag{5.1d}$$

$$\mu_S(k) \geq 0 \qquad z_S \geq f_S(k) \qquad \forall (S,k) \in T \tag{5.1e}$$

where we also wrote the dual on the left-hand side. Note that $g \in \mathbb{R}^T$ is a given WCSP.

In accordance with the pair (1.1), we will further on refer to the left-hand problem (5.1) as primal and the right-hand problem (5.1) as dual. In the primal (5.1), there is a non-negative variable $\lambda(x)$ for each assignment $x \in D^V$ and a non-negative variable $\mu_S(k)$ for each tuple $(S,k) \in T$. The dual (5.1) has a real-valued variable z_S for each scope $S \in C$ and a real-valued variable $f_S(k)$ for each tuple $(S,k) \in T$. The primal constraints and dual variables are thus indexed by $T \cup C$ whereas primal variables and dual constraints are indexed by $T \cup D^V$. Note that these are disjoint unions, i.e., $T \cap C = T \cap D^V = \emptyset$. The index sets here are different from (1.1) where they were $[m]$ and $[n]$.

In this section, we partition any $J \subseteq T \cup D^V$ into $\{J_A, J_X\}$ where $J_A \subseteq T$ and $J_X \subseteq D^V$, i.e., $J_A = J \cap T$ and $J_X = J \cap D^V$.[81]

[81] This is analogous to the previously used notation. In §4, subsets of primal variables and dual constraints were denoted by J. In §3.3, subsets of T (i.e., CSPs) were frequently denoted by A and subsets of D^V by X. We also emphasise that, unlike other sections with linear programs, the symbol x is no longer a variable of the optimization problem, but is used only for indexing the variables and constraints.

Remark 5.1. *Recall from Theorem 1.16 that the common optimal value of the primal-dual pair* (5.1) *coincides with the optimal value of WCSP* g*. The primal* (5.1) *is therefore also an LP formulation of WCSP* g *in some sense.*

To see this more clearly, realize that any solution feasible for the primal satisfies

$$\sum_{x\in D^V} \lambda(x) = \sum_{k\in D^S} \sum_{\substack{x\in D^V \\ x|_S=k}} \lambda(x) = \sum_{k\in D^S} \mu_S(k) = 1 \tag{5.2}$$

where we chose an arbitrary $S \in C$ *and used primal constraints* (5.1b) *and* (5.1c)*. Moreover, for any non-negative values of variables* λ *satisfying* $\sum_{x\in D^V} \lambda(x) = 1$*, variables* μ *can be defined using* (5.1c) *so that* (λ, μ) *is feasible for the primal* (5.1)*. By non-negativity of* λ *and* $\sum_{x\in D^V} \lambda(x) = 1$*, the primal objective* (5.1a) *can be interpreted as a convex combination of numbers* $F(x|g)$*,* $x \in D^V$*. Thus, primal-feasible solutions can attain any objective value from the interval* $[\min_x F(x|g), \max_x F(x|g)]$ *and no value outside of it.*

A solution (z, f) feasible for the dual (5.1) is optimal if and only if there exists a feasible solution for the primal (5.1) satisfying complementary slackness. These conditions read

$$\sum_{k\in D^S} \mu_S(k) = 1 \qquad \forall S \in C \tag{5.3a}$$

$$\sum_{\substack{x\in D^V \\ x|_S=k}} \lambda(x) = \mu_S(k) \qquad \forall (S,k) \in T \tag{5.3b}$$

$$\lambda(x) \geq 0 \qquad \forall x \in J_X \tag{5.3c}$$

$$\lambda(x) = 0 \qquad \forall x \in D^V - J_X \tag{5.3d}$$

$$\mu_S(k) \geq 0 \qquad \forall (S,k) \in J_A \tag{5.3e}$$

$$\mu_S(k) = 0 \qquad \forall (S,k) \in T - J_A \tag{5.3f}$$

where

$$J = \tau(z,f) = \underbrace{\{\, (S,k) \in T \mid z_S = f_S(k) \,\}}_{J_A \subseteq T} \cup \underbrace{\{\, x \in D^V \mid F(x|f) = F(x|g) \,\}}_{J_X \subseteq D^V} \tag{5.4}$$

is the set of dual constraints that are active for the current dual-feasible point (z, f).[82]

Note, system (5.3) is a special case of (2.2) because already the primal-dual pair (5.1) is in the form (1.1). A possible interpretation of conditions (5.3) is given by the following proposition.

Proposition 5.1. *Let* $J_A \subseteq T$ *and* $J_X \subseteq D^V$*. System* (5.3) *is feasible if and only if* $\mathrm{SOL}(J_A) \cap J_X \neq \emptyset$*.*[83]

[82] The set J_A in (5.4) is similar to the set $A^*(f)$ defined in §1.5.2. To be precise, we have that $J_A = A^*(f)$ if $z_S = \max_{k\in D^S} f_S(k)$ for all $S \in C$. However, the variables z_S may also attain other (i.e., greater) values in general.

[83] Condition $\mathrm{SOL}(J_A) \cap J_X \neq \emptyset$ corresponds to statement (c) in Theorem 3.1. Also, feasibility of (5.3) is (by complementary slackness) the condition for (z, f) to be optimal for the dual (5.1). Putting this together, Proposition 5.1 could be expected.

Proof. For the 'only if' direction, let (λ, μ) be feasible for (5.3). By Remark 5.1, there is $x^* \in D^V$ such that $\lambda(x^*) > 0$. By conditions (5.3b) and non-negativity of (λ, μ), we have $\mu_S\big(x^*|_S\big) > 0$ for all $S \in C$. Due to conditions (5.3d) and (5.3f), we necessarily have $x^* \in J_X$ and $\big(S, x^*|_S\big) \in J_A$ for all $S \in C$, hence $x^* \in \mathrm{SOL}(J_A)$.

For the 'if' direction, let $x^* \in \mathrm{SOL}(J_A) \cap J_X$. By $x^* \in \mathrm{SOL}(J_A)$, we have $\big(S, x^*|_S\big) \in J_A$ for all $S \in C$. Now, values $\lambda(x) = [\![x = x^*]\!]$ for all $x \in D^V$ and $\mu_S(k) = [\![x^*|_S = k]\!]$ for all $(S, k) \in T$ are feasible for (5.3). In detail, for any $S \in C$, (5.3a) is satisfied since $k = x^*|_S$ holds for exactly one $k \in D^S$. Condition (5.3b) is immediate. Next, we have $x^* \in J_X$, so $\lambda(x^*)$ can be non-zero by (5.3c). Analogously, we have $\big(S, x^*|_S\big) \in J_A$ for all $S \in C$, so the variables $\mu_S\big(x^*|_S\big)$ can be non-zero by (5.3e). □

Proposition 5.1 formalizes the fact that (5.3) is an LP *formulation* of CSP J_A with an additional global constraint $x \in J_X$. This can be compared to the case of the (dual of) basic LP relaxation where complementary slackness conditions (2.8) can be interpreted as an LP *relaxation* of the active-tuple CSP [146], as discussed in §2.3.

5.2.1 Optimality of BCD

Let us now focus on coordinate-wise minimization of the dual (5.1). By Fact 1.1, optimizing the dual (5.1) coordinate-wise is weaker than optimizing (1.45) coordinate-wise because the z variables are updated separately in the LP formulation (5.1). Nevertheless, we are still able to show that pre-ILMs of the dual (5.1) w.r.t. individual variables are global minima of (5.1). Following Fact 1.1, this implies that pre-ILMs of (1.45) w.r.t. individual variables are also global minima.

Because the primal-dual pair (5.1) is precisely in the form (1.1) that we studied in §4.1, we can proceed directly by carrying over the corresponding notion of a B-consistent set and analyzing its meaning here for this particular linear program. In (5.1), the dual variables and primal constraints are indexed by elements of $T \cup C$, so the set of blocks $\mathcal{B}$ contains all singleton subsets of $T \cup C$, i.e.,

$$\mathcal{B} = \{\, \{(S, k)\} \mid (S, k) \in T \,\} \cup \{\, \{S\} \mid S \in C \,\}. \tag{5.5}$$

Recalling the previously defined notation, instead of talking about $\mathcal{B}$-consistency of J, we will talk about $\mathcal{B}$-consistency of $J_A \cup J_X$. To simplify formulations, we will assume that $\mathcal{B}$ is always defined by (5.5) whenever it is mentioned in §5.2.

To prove our results formally, recall from Definition 4.1 that a set is $\mathcal{B}$-consistent (where $\mathcal{B}$ is (5.5)) if and only if it is $\{(S, k)\}$-consistent for each $(S, k) \in T$ and $\{S\}$-consistent for each $S \in C$. First, we state two lemmas that characterize B-consistent sets for $B \in \mathcal{B}$, i.e., when B is a singleton set.

Lemma 5.1. *Let $J_A \subseteq T$, $J_X \subseteq D^V$, and $B = \{S\}$ where $S \in C$. The set $J_A \cup J_X$ is B-consistent if and only if $T_S \cap J_A \neq \emptyset$ (recall T_S from (1.24)), i.e., there exists $k \in D^S$ with $(S, k) \in J_A$.*

Proof. By definition of B-consistency (Definition 4.1), the system formed by a single equality (5.3a) (for the given S) and all conditions (5.3c)-(5.3f) must be feasible. This happens if and only if at least one variable $\mu_S(k)$ for some $(S, k) \in T_S$ is allowed to be non-zero due to (5.3a), i.e., $T_S \cap J_A \neq \emptyset$.

Clearly, the system formed by a single equality (5.3a) (for the given S) and all conditions (5.3c)-(5.3f) never implies that any of the inequalities should hold with equality, i.e., it never contains an always-active inequality (assuming that this system is feasible). □

Lemma 5.2. *Let $J_A \subseteq T$, $J_X \subseteq D^V$, and $B = \{(S,k)\}$ where $(S,k) \in T$. The set $J_A \cup J_X$ is B-consistent if and only if*

$$\big(\exists x \in J_X \colon x|_S = k\big) \iff (S,k) \in J_A. \tag{5.6}$$

Proof. Again, by definition of B-consistency, we analyze the system formed by a single equality (5.3b) (for the given (S,k)) and all conditions (5.3c)-(5.3f). This system is always feasible by setting all variables to zero.

If $(S,k) \in T - J_A$ (i.e., $\mu_S(k) = 0$), then the system implies that all variables $\lambda(x)$ on the left-hand side of the equality (5.3b) need to be zero too since they are non-negative, so we must have $x \in D^V - J_X$ for all x with $x|_S = k$, i.e., $\nexists x \in J_X \colon x|_S = k$.

Similarly, if $\nexists x \in J_X \colon x|_S = k$ (i.e., $\lambda(x) = 0$ for all $x \in D^V$ with $x|_S = k$), then the left-hand side of the equality (5.3b) is zero and it thus implies $\mu_S(k) = 0$, so we must have $(S,k) \in T - J_A$. □

Theorem 5.1. *Let $J_A \subseteq T$ and $J_X \subseteq D^V$. If $J_A \cup J_X$ is $\mathcal{B}$-consistent,* (5.3) *is feasible.*

Proof. Let $S^* \in C$. By Lemma 5.1 applied to S^*, we have $(S^*,k^*) \in T_{S^*} \cap J_A \subseteq J_A$ for some $k^* \in D^{S^*}$. Therefore, by Lemma 5.2 applied to (S^*,k^*), there exists $x \in J_X$ with $x|_{S^*} = k^*$. Applying Lemma 5.2 again to $B = \big\{\big(S, x|_S\big)\big\}$ for each $S \in C$ yields that $\big(S, x|_S\big) \in J_A$ due to $x \in J_X$. Consequently, $x \in \mathrm{SOL}(J_A)$ and system (5.3) is feasible by Proposition 5.1. □

Theorem 5.1 states that the propagation rule (4.1) with blocks being singleton subsets is refutation complete for system (5.3), i.e., it is always able to detect infeasibility of (5.3) whenever (5.3) is infeasible. Combining this with Corollary 5.1, we obtain one of the main results of §5.2:

Corollary 5.2. *Any (pre-)ILM of the dual* (5.1) *w.r.t. individual variables (i.e., w.r.t. $\mathcal{B}$) is a global minimum of the dual* (5.1)*. Therefore, any (pre-)ILM of* (1.45) *w.r.t. individual variables is a global minimum of* (1.45)*.*

5.2.2 Enforcing Positive Consistency

Now, we will argue that $\mathcal{B}$-consistency in a precise sense corresponds to positive consistency of a certain CSP. To this end, let us slightly deviate from our assumption on the structure of all CSPs to be the same in this subsection. We will show that the propagator $p_{\mathcal{B}}$ is in fact enforcing positive consistency in the CSP J_A with an additional global constraint $x \in J_X$. The structure of this CSP is $(V, D, C \cup \{V\})$ and its set of allowed tuples is

$$J_A \cup \{(V,x) \mid x \in J_X\}. \tag{5.7}$$

Clearly, the solution set of this CSP is $\mathrm{SOL}(J_A) \cap J_X$. To connect positive consistency of CSP (5.7) with $\mathcal{B}$-consistency of $J_A \cup J_X$, we need the following lemma.

Lemma 5.3. *Let $J_A \subseteq T$ and $J_X \subseteq D^V$. CSP* (5.7) *is positively consistent if and only if* (5.6) *holds for all $(S,k) \in T$.*

Proof. For the 'if' direction, we proceed to show that any tuple from (5.7) is used in some solution of this CSP. For (V, x) with $x \in J_X$, applying (5.6) for all $\big(S, x|_S\big)$, $S \in C$ yields $\big(S, x|_S\big) \in J_A$, consequently $x \in \mathrm{SOL}(J_A)$ and $x \in \mathrm{SOL}(J_A) \cap J_X$. For $(S, k) \in J_A$, (5.6) implies that there is $x \in J_X$ with $x|_S = k$. As in the previous case, $x \in J_X$ implies that x is a solution of the CSP and x uses tuple (S, k).

For the 'only if' direction, let $(S, k) \in T$. By positive consistency, if $(S, k) \in J_A$, then there exists $x \in \mathrm{SOL}(J_A) \cap J_X$ such that $x|_S = k$, so $x \in J_X$ which yields the $\Longleftarrow$ direction in (5.6). To prove the other direction, if $\exists x \in J_X \colon x|_S = k$, then tuple (V, x) is allowed in CSP (5.7) and we must have $x \in \mathrm{SOL}(J_A) \cap J_X$, so $(S, k) = \big(S, x|_S\big) \in J_A$. □

Theorem 5.2. *Let $J_A \subseteq T$ and $J_X \subseteq D^V$. The following are equivalent:*

(a) CSP (5.7) *is positively consistent and non-empty,*

(b) $J_A \cup J_X$ is $\mathcal{B}$-consistent.

Proof. (a) $\Longrightarrow$ (b): A non-empty positively consistent CSP is satisfiable (recall Example 1.12), so $\mathrm{SOL}(J_A) \cap J_X \neq \emptyset$ which implies satisfiability of CSP J_A and consequently $T_S \cap J_A \neq \emptyset$ for all $S \in C$. So, $J_A \cup J_X$ is $\{S\}$-consistent for all $S \in C$ by Lemma 5.1. By Lemma 5.3, positive consistency implies that (5.6) holds for all $(S, k) \in T$. Using Lemma 5.2, this establishes $\{(S, k)\}$-consistency of $J_A \cup J_X$ for all $(S, k) \in T$.

(b) $\Longrightarrow$ (a): Theorem 5.1 together with Proposition 5.1 imply $\mathrm{SOL}(J_A) \cap J_X \neq \emptyset$, so the CSP is satisfiable and therefore non-empty. Due to $\mathcal{B}$-consistency, set $J_A \cup J_X$ is $\{(S, k)\}$-consistent for each $(S, k) \in T$, so (5.6) holds for each $(S, k) \in T$ by Lemma 5.2. By Lemma 5.3, this implies that the CSP is positively consistent. □

These results allow us to characterize ILMs and pre-ILMs of the dual (5.1) w.r.t. $\mathcal{B}$ by positive consistency of CSP (5.7):

Corollary 5.3. *Let (z, f) be feasible for the dual* (5.1) *and let J_A and J_X be as in* (5.4).

(a) (z, f) is an ILM of the dual (5.1) *w.r.t. individual variables (i.e., w.r.t. $\mathcal{B}$) if and only if CSP* (5.7) *is positively consistent and non-empty.*

(b) (z, f) is a pre-ILM of the dual (5.1) *w.r.t. individual variables (i.e., w.r.t. $\mathcal{B}$) if and only if CSP* (5.7) *has non-empty positive consistency closure.*

Proof. (a): By Theorem 4.2b, (z, f) is an ILM if and only if $J_A \cup J_X$ is $\mathcal{B}$-consistent which is equivalent to CSP (5.7) being positively consistent and non-empty by Theorem 5.2.

(b): The CSP has non-empty positive consistency closure if and only if it is satisfiable, i.e., $\mathrm{SOL}(J_A) \cap J_X \neq \emptyset$. By Proposition 5.1, this is equivalent to feasibility of (5.3). By definition of J_A and J_X, (5.3) is feasible precisely when (z, f) is optimal for (5.1). Using Corollaries 5.2 and 1.3, the set of pre-ILMs coincides with global minima. □

Compare the result given by Corollary 5.3 to the results stated in §4.4.1: for (1.43) (i.e., the dual of the basic LP relaxation of WCSP), ILMs and pre-ILMs w.r.t. individual variables are defined in a completely analogous way except that one uses arc consistency (of CSP $A^*(g^\varphi)$) instead of positive consistency (of CSP (5.7)).

5.2.3 Coordinate-Wise Updates: Convergence and Hardness

We derive coordinate-wise updates for the dual (5.1) satisfying the relative-interior rule and show that if these updates are performed in a cyclic order, then the current solution converges to the set of optimizers of the dual. However, we will also show that performing these updates is likely to be intractable.

The coordinate-wise update of a single variable z_S, $S \in C$ is unique because z_S is to be minimized in the dual (5.1) and

$$z_S := \max_{k \in D^S} f_S(k) \tag{5.8}$$

is the least value of z_S allowed by dual constraints (5.1e) for the fixed values of variables f. An update for a single variable $f_S(k)$, $(S,k) \in T$ is given in the following proposition.

Proposition 5.2. *Let (z,f) be feasible for the dual (5.1) and $(S,k) \in T$. Consider the coordinate-wise update*

$$f_S(k) := \tfrac{1}{2} z_S + \tfrac{1}{2} \delta(g,f,S,k) \tag{5.9}$$

where

$$\delta(g,f,S,k) = \max_{\substack{x \in D^V \\ x|_S = k}} \Big(F(x\,|\,g) - \sum_{\substack{S' \in C \\ S' \neq S}} f_{S'}\big(x|_{S'}\big) \Big). \tag{5.10}$$

This update satisfies the relative-interior rule.

Proof. Variable $f_S(k)$ appears in the constraints (5.1d) for all $x \in D^V$ with $x|_S = k$. Written explicitly, this is

$$\overbrace{f_S(k) + \sum_{\substack{S' \in C \\ S' \neq S}} f_{S'}\big(x|_{S'}\big)}^{F(x\,|\,f)} \geq F(x\,|\,g) \qquad \forall x \in D^V : x|_S = k \tag{5.11}$$

which yields the lower bound $f_S(k) \geq \delta(g,f,S,k)$. Next, $f_S(k)$ also appears in one of the constraints (5.1e) which results in the upper bound $z_S \geq f_S(k)$.

Thus, to stay within the feasible set, we need $f_S(k) \in [\delta(g,f,S,k), z_S]$. Because the objective is independent of $f_S(k)$, it suffices to choose any point from this interval. To satisfy the relative-interior rule, the update (5.9) chooses the midpoint of this interval. □

To ensure convergence, we proceed to prove that the sequence of points (z,f) generated by the updates (5.8) and (5.9) is bounded.

Proposition 5.3. *Let (z,f) be feasible for the dual (5.1). Any sequence of points obtained by performing updates (5.8) and (5.9) is bounded.*

Proof. Variables z are bounded from above because they never increase. Moreover, they are also bounded from below because, for any feasible (z,f), $\sum_{S' \in C} z_{S'}$ is an upper bound on the optimal value of WCSP g, so no z_S can get below $\max_x F(x\,|\,g) - \sum_{S' \in C - \{S\}} z_{S'}$ which is non-decreasing because g is constant and z are non-increasing. Furthermore, the variables f are bounded from above by the corresponding z variables and from below because z are non-increasing and

$$f_S(k) \geq \max_{\substack{x \in D^V \\ x|_S = k}} \Big(F(x\,|\,g) - \sum_{\substack{S' \in C \\ S' \neq S}} f_{S'}\big(x|_{S'}\big) \Big) \geq \max_{\substack{x \in D^V \\ x|_S = k}} \Big(F(x\,|\,g) - \sum_{\substack{S' \in C \\ S' \neq S}} z_{S'} \Big) \tag{5.12}$$

which follows from $f_S(k) \geq \delta(g, f, S, k)$ and $z_{S'} \geq f_{S'}(k)$ for each $(S', k) \in T$. □

Since both updates (5.8) and (5.9) are clearly continuous in (z, f) and the resulting sequence is bounded (Proposition 5.3), it follows from Theorem 1.10 that if updates (5.8) and (5.9) are performed for all variables in a cyclic order, then the current point converges to the set of optimizers of the dual (5.1). Note that, by combining Corollary 5.2 with Corollary 1.3, the set of optimizers coincides with the set of pre-ILMs.

However, by Theorem 1.16, the optimal value of (5.1) coincides with the optimal value of WCSP g. It is therefore not surprising that performing BCD with the relative-interior rule is likely to be intractable.

Theorem 5.3. *The following problem is NP-complete: Given $f, g \in \mathbb{Z}^T$, $z \in \mathbb{Z}^C$, and $(S^*, k^*) \in T$ such that (z, f) is feasible for the dual* (5.1) *and $z_{S^*} = f_{S^*}(k^*)$, decide whether $f_{S^*}(k^*) \in \operatorname{ri}[\delta(g, f, S^*, k^*), z_{S^*}]$ (i.e., whether $f_{S^*}(k^*)$ is in the relative interior of optimizers of the dual* (5.1) *while the remaining variables are fixed).*

Proof. In this proof, we abbreviate $\delta(g, f, S^*, k^*)$ to δ. First, see that $z_{S^*} \in \operatorname{ri}[\delta, z_{S^*}]$ if and only if $\delta = z_{S^*}$ (recall Example 1.1 for $n = 1$). So, due to $f_{S^*}(k^*) = z_{S^*}$, the problem boils down to deciding whether $f_{S^*}(k^*) = \delta$.

Membership in NP follows from the fact that one can generate $x \in D^V$ with $x|_{S^*} = k^*$ and verify whether the term in the maximum in (5.10) equals $f_{S^*}(k^*)$. Such an x exists if and only if $f_{S^*}(k^*) = \delta$.

To show NP-hardness, we proceed similarly as in Theorem 3.13 by reduction from 3-coloring [116, §8.6.1, 84], i.e., given a graph $G = (V, C)$, decide whether it is 3-colorable. For the purpose of our reduction, we consider the structure (V, D, C) where $|D| = 3$ and formulate the 3-coloring problem as a WCSP $g \in \{0, 1\}^T$ defined by

$$g_{\{i,j\}}(k_i, k_j) = [\![k_i \neq k_j]\!] \qquad \forall \{i, j\} \in C,\ k \in D^{\{i,j\}}. \tag{5.13}$$

Each $x \in D^V$ corresponds to an assignment of the 3 colors from D to the vertices and $F(x\,|\,g) = |\{\, \{i, j\} \in C \mid x_i \neq x_j \,\}|$ is the number of edges in G whose adjacent vertices have different colors in assignment $x \in D^V$. Thus, $\max_x F(x\,|\,g) = |C|$ if and only if G is 3-colorable (and $\max_x F(x\,|\,g) \leq |C| - 1$ otherwise).

Now, we pick an arbitrary edge $S^* = \{i^*, j^*\} \in C$ and define (z, f) by

$$z_S = |C| \cdot [\![S = S^*]\!] \qquad \forall S \in C \tag{5.14a}$$

$$f_S(k) = |C| \cdot [\![S = S^*]\!] \qquad \forall (S, k) \in T. \tag{5.14b}$$

Clearly, $F(x\,|\,f) = |C|$ for all $x \in D^V$ and (z, f) is feasible for the dual (5.1). Since the colors can be arbitrarily permuted in a solution, we can choose any $k^* \in D^{S^*}$ such that $k^*_{i^*} \neq k^*_{j^*}$ and see that

$$\max_{x \in D^V} F(x\,|\,g) = \max_{\substack{x \in D^V \\ x|_{S^*} = k^*}} F(x\,|\,g) = \max_{\substack{x \in D^V \\ x|_{S^*} = k^*}} \Big(F(x\,|\,g) - \underbrace{\sum_{\substack{S \in C \\ S \neq S^*}} f_S\big(x|_S\big)}_{0} \Big) = \delta \tag{5.15}$$

where the second equality follows from the definition of f in (5.14b).

For the above-defined g, f, z, S^*, and k^*, we have that $f_{S^*}(k^*) \in \operatorname{ri}[\delta, z_{S^*}]$ if and only if G is 3-colorable. □

Following Theorem 5.3, performing even a single coordinate-wise update satisfying the relative-interior rule is likely intractable. Let us also note that, for (z, f) feasible for the dual (5.1), deciding whether (z, f) is a pre-ILM of the dual (5.1) w.r.t. individual variables is NP-complete. This is an immediate consequence of Corollary 3.3 because pre-ILMs coincide with global minima (recall Corollary 1.3).

In contrast, performing arbitrary exact BCD updates (1.13) (without the relative-interior rule) is tractable and trivial. Indeed, for any (z, f) feasible for the dual (5.1), we can simply keep each f_t, $t \in T$ constant and update each z_S as in (5.8). Eventually, when all z variables satisfy $z_S = \max_{k \in D^S} f_S(k)$, the current point is an LM of the dual (5.1) w.r.t. individual variables. Such local minima can be however arbitrarily bad.

5.3 Two More Classes of Linear Programs Solvable by BCD

In this section, we identify two classes of linear programs that are solvable to optimality by BCD with the relative-interior rule. We begin by defining the precise form of the considered linear program and stating the main result of this section in Theorem 5.4. Then, we prove this theorem in §5.3.1. Finally, in §5.3.2, we list some practical LP problems to which the theorem applies. This section follows our presentation given in [53a] with some parts of the text reused.

We consider the pair of mutually dual linear programs

$$\begin{aligned}
&\max\ \underline{y}^\top p + \overline{y}^\top q + c^\top x && \min\ b^\top y + 1^\top z && && (5.16a)\\
&x_j \geq 0 && z_j + A_j^\top y \geq c_j && \forall j \in [n] && (5.16b)\\
&x_j \leq 1 && z_j \geq 0 && \forall j \in [n] && (5.16c)\\
&p_i \geq 0 && y_i \geq \underline{y}_i && \forall i \in [m] && (5.16d)\\
&q_i \leq 0 && y_i \leq \overline{y}_i && \forall i \in [m] && (5.16e)\\
&A^i x + p_i + q_i = b_i && y_i \in \mathbb{R} && \forall i \in [m] && (5.16f)
\end{aligned}$$

where we will call the left-hand problem primal and the right-hand problem dual. The entries $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, $\underline{y} \in \mathbb{R}^m_{-\infty}$, and $\overline{y} \in \mathbb{R}^m_{+\infty}$ (assuming $\underline{y} \leq \overline{y}$) are given constants where $\mathbb{R}_{+\infty} = \mathbb{R} \cup \{+\infty\}$ and $\mathbb{R}_{-\infty} = \mathbb{R} \cup \{-\infty\}$. We optimize over variables (p, q, x) in the primal and (y, z) in the dual.

We allow infinite values in the vectors $\underline{y}$ and $\overline{y}$ that constitute lower and upper bounds on the dual variables y, respectively. For this purpose, in products $\underline{y}^\top p$ and $\overline{y}^\top q$, we adopt that $+\infty \cdot 0 = -\infty \cdot 0 = 0$. This formalism allows us to introduce also inequality constraints into the primal and to set arbitrary bounds on the dual variables y.

Example 5.1. *To illustrate, let $i \in [m]$, $\overline{y}_i = +\infty$, and $\underline{y}_i = 0$. In this setting, it is undesirable to have $q_i < 0$ in the primal because then the primal objective is $-\infty$, so we need $q_i = 0$ to guarantee a finite objective. On the other hand, due to $\underline{y}_i = 0$, variable p_i does not have any influence on the objective and appears only in a single primal constraint, $A^i x + p_i + q_i = b_i$. Therefore, we can eliminate both $p_i \geq 0$ and $q_i = 0$ from this constraint to obtain $A^i x \leq b_i$. Note that this is precisely the primal constraint that corresponds to a non-negative dual variable $y_i \geq 0$.*

If we want to make a dual variable y_i unbounded, we can set $\overline{y}_i = +\infty$ and $\underline{y}_i = -\infty$. Then, by analogous reasoning, we need to have $q_i = p_i = 0$ in the primal. Consequently,

these two variables can be omitted from the primal due to being zero and the primal constraint (5.16f) *corresponding to* $y_i \in \mathbb{R}$ *becomes* $A^i x = b_i$*, as expected.* △

In analogy to §1.1.2 or §2.4, at any minimum (z, y) of the dual (5.16), we have

$$z_j = \max\{c_j - A_j^\top y,\, 0\} \qquad \forall j \in [n] \tag{5.17}$$

which allows us to express the dual as a box-constrained minimization of a convex piecewise-affine function, namely

$$\min\ b^\top y + \sum_{j\in[n]} \max\{c_j - A_j^\top y,\, 0\} \tag{5.18a}$$

$$\underline{y}_i \le y_i \le \overline{y}_i \qquad \forall i \in [m]. \tag{5.18b}$$

Before we state the conditions that guarantee that any (pre-)ILM of (5.18) w.r.t. individual variables is a global minimum, we need a number of definitions.

Definition 5.1 ([26, §1.1])**.** *The* bipartite graph associated with matrix $A \in \mathbb{R}^{m\times n}$ *is a graph with* $m+n$ *vertices whose partitions are* $\{\mathsf{r}_1, \dots, \mathsf{r}_m\}$ *and* $\{\mathsf{c}_1, \dots, \mathsf{c}_n\}$ *and contains an edge* $\{\mathsf{r}_i, \mathsf{c}_j\}$ *if* $A_{ij} \neq 0$*.*

Definition 5.2. *A matrix* $A \in \mathbb{R}^{m\times n}$ *is* bipartite-acyclic *if the bipartite graph associated with* A *is acyclic.*

Definition 5.3. *Matrix* $A \in \mathbb{R}^{m\times n}$ *is* 2-in-row *if for each* $i \in [m]$ *one of the following conditions is satisfied:*

(a) row i *of matrix* A *has at most 2 non-zero elements, i.e.,* $|\{\, j \in [n] \mid A_{ij} \neq 0 \,\}| \le 2$*,*

(b) row i *of matrix* A *has 3 non-zero elements and at least one of them is the only non-zero element in its column, i.e.,* $|\{\, j \in [n] \mid A_{ij} \neq 0 \,\}| = 3$ *and there exists* $j^* \in [n]$ *such that* $A_{ij^*} \neq 0$ *and* $\forall i^* \in [m] - \{i\}\colon A_{i^*j^*} = 0$*.*

Remark 5.2. *One can also characterize 2-in-row matrices* $A \in \mathbb{R}^{m\times n}$ *by the associated bipartite graph,* $(\{\mathsf{r}_1, \dots, \mathsf{r}_m\}, \{\mathsf{c}_1, \dots, \mathsf{c}_n\}, E)$*. A is 2-in-row if and only if each* r_i*,* $i \in [m]$ *has degree at most 3 and, after removing all vertices* c_j *with degree 1, each* r_i *has degree at most 2.*

Now, we are able to state the main result of this section in Theorem 5.4.

Theorem 5.4. *Let* $A \in \{-1, 0, 1\}^{m\times n}$*,* $b \in \mathbb{Z}^m$*,* $c \in \mathbb{R}^n$*,* $\underline{y} \in \mathbb{R}^m_{-\infty}$*, and* $\overline{y} \in \mathbb{R}^m_{+\infty}$*. If* A *is 2-in-row or bipartite-acyclic, then any (pre-)ILM of* (5.18) *w.r.t. individual variables is a global minimum of* (5.18)*.*

As the proof of this theorem is lengthy, we present it in the next subsection. [84]

[84] Our results from §4 allowed us to significantly simplify our previous proof given in [51a]. It is also possible to prove Theorem 5.4 (at least for the case of 2-in-row matrices) even without relying on the results from §4, as we did in [51a] when results from §4 were yet unknown. Such an approach is more involved and requires a precise analysis of block-optimality conditions with the relative-interior rule and case analysis.

5.3.1 Proof of Theorem 5.4

Following Fact 1.1, applying BCD to (5.18) along individual variables (i.e., optimizing (5.18) coordinate-wise) is equivalent to applying BCD to the dual (5.16) along m blocks of variables where each block contains all z variables and a single variable y_i. Being slightly informal, we will refer to a single such block by B_i. The set of all such blocks is $\mathcal{B} = \{B_i \mid i \in [m]\}$. Whenever we refer to B_i or $\mathcal{B}$ in §5.3.1, we assume that these sets have the values defined here.

In this setting, Theorem 5.4 is equivalent to the fact that (under the specified conditions on the matrix A and vector b), any (pre-)ILM of the dual (5.16) w.r.t. $\mathcal{B}$ is a global minimum. Since dual (5.16) is a linear program, we will use our results from §4 together with Corollary 5.1. Although the primal-dual pair (5.16) is not in the form (1.1), our results can be extended to more general forms, as discussed in §4.3.

It is clear that in any ILM (or pre-ILM) (z, y) of (5.16) w.r.t. $\mathcal{B}$, (5.17) holds – if not, it would not be block-optimal for variables z. Under this assumption, (z, y) is optimal for the dual (5.16) if and only if there exist $x \in \mathbb{R}^n$ and $p, q \in \mathbb{R}^m$ such that

$$
\begin{aligned}
0 \le x_j \le 1 \qquad & \forall j \in X_U(y) = \{j \in [n] \mid A_j^\top y = c_j\} && \text{(5.19a)}\\
x_j = 0 \qquad & \forall j \in X_0(y) \ = \{j \in [n] \mid A_j^\top y > c_j\} && \text{(5.19b)}\\
x_j = 1 \qquad & \forall j \in X_1(y) \ = \{j \in [n] \mid A_j^\top y < c_j\} && \text{(5.19c)}\\
p_i = 0 \qquad & \forall i \in P_0(y) \ = \{i \in [m] \mid y_i > \underline{y}_i\} && \text{(5.19d)}\\
p_i \ge 0 \qquad & \forall i \in P_+(y) \ = \{i \in [m] \mid y_i = \underline{y}_i\} && \text{(5.19e)}\\
q_i = 0 \qquad & \forall i \in Q_0(y) \ = \{i \in [m] \mid y_i < \overline{y}_i\} && \text{(5.19f)}\\
q_i \le 0 \qquad & \forall i \in Q_-(y) = \{i \in [m] \mid y_i = \overline{y}_i\} && \text{(5.19g)}\\
A^i x + p_i + q_i = b_i \qquad & \forall i \in [m]. && \text{(5.19h)}
\end{aligned}
$$

These conditions follow from the complementary slackness theorem applied to the primal-dual pair (5.16) while noting the substitution (5.17). Notice that $\{X_U(y), X_0(y), X_1(y)\}$ is a partition of $[n]$. Also, $\{P_0(y), P_+(y)\}$ and $\{Q_0(y), Q_-(y)\}$ are partitions of $[m]$.

Lemma 5.4. *Let (z, y) be an ILM of (5.16) w.r.t. $\mathcal{B}$ and $i^* \in [m]$. The system*

$$
\begin{aligned}
0 \le x_j \le 1 \qquad & \forall j \in X_U(y) && \text{(5.20a)}\\
x_j = 0 \qquad & \forall j \in X_0(y) && \text{(5.20b)}\\
x_j = 1 \qquad & \forall j \in X_1(y) && \text{(5.20c)}\\
\begin{cases} p_{i^*} = 0 & \text{if } i^* \in P_0(y) \\ p_{i^*} \ge 0 & \text{if } i^* \in P_+(y) \end{cases} & && \text{(5.20d)}\\
\begin{cases} q_{i^*} = 0 & \text{if } i^* \in Q_0(y) \\ q_{i^*} \le 0 & \text{if } i^* \in Q_-(y) \end{cases} & && \text{(5.20e)}\\
A^{i^*} x + p_{i^*} + q_{i^*} = b_{i^*} & && \text{(5.20f)}
\end{aligned}
$$

is feasible and does not contain any always-active inequality.

In other words, system (5.20) is feasible and implies neither $x_j = 0$ nor $x_j = 1$ for any $j \in X_U(y)$. Also, if $i^* \in P_+(y)$ (or $i^* \in Q_-(y)$), it does not imply $p_{i^*} = 0$ (or $q_{i^*} = 0$), respectively.

Proof. Here, we apply our results from §4. The set of constraints from (5.19) corresponding to an aforementioned block $B_{i^*} \in \mathcal{B}$ is in fact (5.20). This is because such a system (as in (4.2)) always contains all the primal variables. Also, it contains conditions $x_j \leq 1$ or $x_j = 1$ for each $j \in [n]$ because all z variables are in the block B_{i^*} (see (5.16c): z_j variables are the dual variables corresponding to the primal constraints $x_j \leq 1$). Finally, it contains the equality (5.19h) for $i = i^*$ because y_{i^*} is in the block B_{i^*} (see (5.16f): variables y correspond to constraints (5.20f)). Clearly, variables p_i and q_i for $i \neq i^*$ do not appear in any constraint of the subsystem and can be removed from the set of equalities and inequalities without changing any of its properties.

The claim now follows from Definition 4.1 (recall Footnote 71) applied to block $B_{i^*} \in \mathcal{B}$ combined with Lemma 4.1b. □

System (5.20) can be further simplified by eliminating variables p_{i^*} and q_{i^*}, as we state in the following corollary of Lemma 5.4.

Corollary 5.4. *Let (z, y) be an ILM of (5.16) w.r.t. $\mathcal{B}$ and $i^* \in [m]$. The system*

$$0 \leq x_j \leq 1 \qquad \forall j \in X_U(y) \tag{5.21a}$$

$$x_j = 0 \qquad \forall j \in X_0(y) \tag{5.21b}$$

$$x_j = 1 \qquad \forall j \in X_1(y) \tag{5.21c}$$

$$\begin{cases} A^{i^*}x = b_{i^*} & \text{if } i^* \in P_0(y) \cap Q_0(y) \\ A^{i^*}x \geq b_{i^*} & \text{if } i^* \in P_0(y) \cap Q_-(y) \\ A^{i^*}x \leq b_{i^*} & \text{if } i^* \in P_+(y) \cap Q_0(y) \end{cases} \tag{5.21d}$$

is feasible and implies neither $x_j = 0$ nor $x_j = 1$ for any $j \in X_U(y)$.

Proof. The corollary follows from the fact that the set of x feasible for (5.21) is the projection of the feasible set of (5.20) onto the x variables (cf. Example 5.1).

Note that the case with $i^* \in P_+(y) \cap Q_-(y)$ is not stated in (5.21d). This is due to the fact that for such i^*, constraint (5.20f) vanishes after projecting out p_{i^*} and q_{i^*}, i.e., corresponds to $A^{i^*}x \in \mathbb{R}$ and is always satisfied. This case happens if and only if $\underline{y}_{i^*} = \overline{y}_{i^*}$. So, if $\underline{y} < \overline{y}$, we have that $P_+(y) \cap Q_-(y) = \emptyset$ for any $y \in \mathbb{R}^m$. □

As in the previous corollary, one can also eliminate all p and q variables from the whole system (5.19) to obtain

$$0 \leq x_j \leq 1 \qquad \forall j \in X_U(y) \tag{5.22a}$$

$$x_j = 0 \qquad \forall j \in X_0(y) \tag{5.22b}$$

$$x_j = 1 \qquad \forall j \in X_1(y) \tag{5.22c}$$

$$\begin{cases} A^i x = b_i & \text{if } i \in P_0(y) \cap Q_0(y) \\ A^i x \geq b_i & \text{if } i \in P_0(y) \cap Q_-(y) \\ A^i x \leq b_i & \text{if } i \in P_+(y) \cap Q_0(y) \end{cases} \qquad \forall i \in [m] - (P_+(y) \cap Q_-(y)). \tag{5.22d}$$

Clearly, (5.22) and (5.19) are equisatisfiable, i.e., (5.22) is feasible if and only if (5.19) is feasible.

Before we prove Theorem 5.4 for bipartite-acyclic matrices A, we need two more auxiliary lemmas given below.

Lemma 5.5 (cf. [76, Theorem 45]). *Let $d \in \{-1,0,1\}^n$ and $\delta \in \mathbb{Z}$. The polyhedra*

$$\{\, x \in [0,1]^n \mid d^\top x \leq \delta \,\} \tag{5.23a}$$

$$\{\, x \in [0,1]^n \mid d^\top x = \delta \,\} \tag{5.23b}$$

are integral and their projection onto each x_j, $j \in [n]$ is either $\emptyset$, $\{0\}$, $\{1\}$, or $[0,1]$.

Proof. Recall that a polyhedron is integral if its extremal points (i.e., vertices [110, Theorem 2.4, 103, §4.4]) have integral coordinates. For a polyhedron P, a point $v \in P$ is its extremal point if $v', v'' \in P$, $0 < \alpha < 1$, and $v = \alpha v' + (1-\alpha)v''$ implies $v = v' = v''$. So, an extremal point cannot be a strict convex combination of two different points of the polyhedron.

We prove the lemma for polyhedron (5.23a) as the proof would change only slightly for (5.23b). We proceed by contradiction (to some extent similarly to [138, Lemma 14.4]): let v be an extremal point of (5.23a) and $j^* \in [n]$ such that $v_{j^*} \notin \{0,1\}$.

If $d_{j^*} = 0$ or $d^\top v < \delta$, then we can both increase or decrease v_{j^*} by some $\epsilon > 0$ so that point v stays in the polyhedron (5.23a). Consequently, it is not an extremal point.

If $d_{j^*} \neq 0$ and $d^\top v = \delta$, there needs to exist $j' \in [n] - \{j^*\}$ such that $v_{j'} \notin \{0,1\}$ and $d_{j'} \neq 0$ due to $\delta \in \mathbb{Z}$ and $d \in \{-1,0,1\}^n$. For some suitable $\epsilon > 0$ which is sufficiently small, points $w^+, w^- \in \mathbb{R}^n$ defined by

$$w_j^{\pm} = \begin{cases} v_j & \text{if } j \in [n] - \{j^*, j'\} \\ v_{j^*} \mp d_{j^*}\epsilon & \text{if } j = j^* \\ v_{j'} \pm d_{j'}\epsilon & \text{if } j = j' \end{cases} \qquad \forall j \in [n] \tag{5.24}$$

satisfy $d^\top w^+ = d^\top w^- = \delta$ and $w^+, w^- \in [0,1]^n$, so they belong to (5.23a). The fact that there exists such a value for ϵ follows from $0 < v_{j^*}, v_{j'} < 1$. Point $v = \frac{1}{2}w^+ + \frac{1}{2}w^-$ is therefore not an extremal point.

The property of projection is directly implied by integrality. □

Lemma 5.6. *Let (z,y) be an ILM of (5.16) w.r.t. $\mathcal{B}$, $A \in \{-1,0,1\}^{m\times n}$, and $b \in \mathbb{Z}^m$. The CSP with discrete variables $x_j \in \{0,1\}$, $j \in [n]$ (where some domains of the variables are reduced by (5.22b) and (5.22c)) and constraints (5.22d) is AC (in the sense of (1.30)). Moreover, if A is bipartite-acyclic, this CSP is satisfiable.*

Proof. We first show that the CSP is AC in the sense of (1.30). For contradiction, suppose that there is $j^* \in [n]$,

$$k^* \in \begin{cases} \{0,1\} & \text{if } j^* \in X_U(y) \\ \{0\} & \text{if } j^* \in X_0(y) \\ \{1\} & \text{if } j^* \in X_1(y) \end{cases}, \tag{5.25}$$

and that for some $i^* \in [m] - (P_+(y) \cap Q_-(y))$, there is no $x \in \{0,1\}^n$ with $x_{j^*} = k^*$ satisfying (5.21). In other words, there is no integral solution x of (5.21) with $x_{j^*} = k^*$. However, the polyhedron defined by (5.21) is integral by Lemma 5.5, so (5.21) is infeasible or, if $j^* \in X_U(y)$, (5.21) implies $x_{j^*} = 1 - k^*$. None of these options is allowed by Corollary 5.4.

If A is bipartite-acyclic, it follows directly from Definition 5.2 that this CSP has acyclic factor graph. For such a structure, AC is refutation complete and the CSP is satisfiable (recall Example 1.11). □

Proof (**Theorem 5.4 for bipartite-acyclic matrices**). Let (z, y) be an ILM of (5.16) w.r.t. $\mathcal{B}$ (i.e., y is ILM of (5.18) w.r.t. individual variables). If A is bipartite-acyclic, Lemma 5.6 implies that there is $x \in \{0,1\}^n$ satisfying (5.22). However, (5.22) is a projection of (5.19), so (5.19) is feasible too and (z, y) is optimal for the dual (5.16) (and y is optimal for (5.18)). □

We now focus on the case of 2-in-row matrices in Theorem 5.4. For this part, we rely on a slightly different proof technique that requires the following lemma.

Lemma 5.7. *Let $n \in \{1,2,3\}$ and $P \subseteq [0,1]^n$ be an integral polyhedron such that the projection of P onto each coordinate x_j, $j \in [n]$ is the interval $[0,1]$.*

(a) If $n \in \{1,2\}$, then x defined by $x_j = \frac{1}{2}$ for all $j \in [n]$ belongs to P.

(b) If $n = 3$, then there is $x_3 \in [0,1]$ such that $\left(\frac{1}{2}, \frac{1}{2}, x_3\right) \in P$.

Proof. We begin by proving (a). If $n = 1$, we have $P = [0,1]$ and $x = \frac{1}{2} \in [0,1]$. If $n = 2$, the set of extremal points of P is a subset of $\{(0,0),(1,0),(0,1),(1,1)\} = \{0,1\}^2$ because P is integral. Since P is bounded, it equals the convex hull of its extremal points [110, §2.3]. This leaves 16 options for the choice of the extremal points based on exhaustive enumeration, from which only 7 options satisfy the assumptions on projections. For each of these 7 options, $x = \left(\frac{1}{2}, \frac{1}{2}\right) \in P$.

We continue with (b). Let $P' = \{\, (x_1, x_2) \mid \exists x_3 \in [0,1] \colon (x_1, x_2, x_3) \in P \,\} \subseteq [0,1]^2$ be the projection of P onto the coordinates x_1 and x_2. Clearly, P' also satisfies the assumptions of this lemma: P' is integral and the projection of P' onto any x_j, $j \in \{1,2\}$ is $[0,1]$. So, by (a), we have $\left(\frac{1}{2}, \frac{1}{2}\right) \in P'$ which implies $\left(\frac{1}{2}, \frac{1}{2}, x_3\right) \in P$ for some $x_3 \in [0,1]$ by definition of P'. □

Next, let us simplify (5.22) by substituting the values for the decided variables x_j, $j \in X_1(y) \cup X_0(y)$. To this end, define $b'_i = b_i - \sum_{j \in X_1(y)} A_{ij}$ for each $i \in [m]$ and let $A' \in \mathbb{R}^{[m] \times X_U(y)}$ be the matrix obtained from A by removing columns $X_0(y) \cup X_1(y)$. This yields

$$0 \le x_j \le 1 \qquad \forall j \in X_U(y) \tag{5.26a}$$

$$\begin{cases} A'^i x = b'_i & \text{if } i \in P_0(y) \cap Q_0(y) \\ A'^i x \ge b'_i & \text{if } i \in P_0(y) \cap Q_-(y) \\ A'^i x \le b'_i & \text{if } i \in P_+(y) \cap Q_0(y) \end{cases} \qquad \forall i \in [m] - (P_+(y) \cap Q_-(y)). \tag{5.26b}$$

The set of $x \in \mathbb{R}^{X_U(y)}$ feasible for (5.26) is a projection of the feasible set of (5.22) (or (5.19)) onto variables $x_j, j \in X_U(y)$.

Proof (**Theorem 5.4 for 2-in-row matrices**). Let (z, y) be an ILM of (5.16) w.r.t. $\mathcal{B}$ (i.e., y is ILM of (5.18) w.r.t. individual variables). Also recall that $A \in \{-1,0,1\}^{m \times n}$ and $b \in \mathbb{Z}^m$, so $A' \in \{-1,0,1\}^{[m] \times X_U(y)}$ and $b' \in \mathbb{Z}^m$.

Let $i^* \in [m] - (P_+(y) \cap Q_-(y))$ be arbitrary. By Lemma 5.5, the polyhedron defined by (5.26a) and a single constraint (5.26b) for $i = i^*$ is integral and its projection onto any variable x_j, $j \in X_U(y)$ is either $\emptyset$, $\{0\}$, $\{1\}$, or $[0,1]$. However, by Corollary 5.4, the projection can be neither $\emptyset$ (because the system defining the polyhedron is feasible), nor $\{0\}$, nor $\{1\}$ (because the system implies neither $x_j = 0$ nor $x_j = 1$ for any $j \in X_U(y)$).

So, the projection of the polyhedron onto any x_j, $j \in X_U(y)$ must be $[0,1]$ which allows us to apply Lemma 5.7.

If each constraint in (5.26b) contains at most 2 variables (i.e., $\sum_{j \in X_U(y)} |A'_{ij}| \leq 2$ holds for each $i \in [m] - (P_+(y) \cap Q_-(y))$), then Lemma 5.7a implies that setting $x_j = \frac{1}{2}$ for all $j \in X_U(y)$ satisfies each constraint in (5.26). So, (5.26) is feasible.

However, since some rows of A may satisfy statement (b) in Definition 5.3, some constraints in (5.26b) may contain 3 variables. By Definition 5.3, for each such a constraint, at least one of the variables in this constraint is not present in any other constraint. Let us denote the set of such variables by X', i.e.,

$$X' = \{\, j \in X_U(y) \mid \exists i^* : \overbrace{\{i \mid A'_{ij} \neq 0\} = \{i^*\}}^{\text{column } j \text{ of } A' \text{ contains exactly one non-zero element, } A'_{i^*j}}, \underbrace{|\{j^* \in X_U(y) \mid A'_{i^*j^*} \neq 0\}| = 3}_{\text{row } i^* \text{ of } A' \text{ contains exactly 3 non-zero elements}} \,\}. \quad (5.27)$$

Now, one can set $x_j = \frac{1}{2}$ for all $j \in X_U(y) - X'$ and, by Lemma 5.7b, there exist values also for the remaining variables x_j, $j \in X'$ such that x satisfies (5.26). In detail, for any constraint i^* from (5.26b) that contains some variables from X', there exists a value for these variables that satisfies this constraint (while the variables x_j, $j \in X_U(y) - X'$ are already set to $\frac{1}{2}$) – since this is the only occurrence of these variables, they can be set independently in each such a constraint. All in all, there exists a vector $x \in [0,1]^{X_U(y)}$ satisfying (5.26).

In both cases above, we proved feasibility of (5.26). Feasibility of this system implies feasibility of (5.22) which in turn implies that (5.19) is feasible too, so (z, y) is optimal for the dual (5.16) and y is optimal for (5.18). □

5.3.2 Applications

Let us now discuss some problems to which Theorem 5.4 is applicable.

First, note that the LP relaxation of weighted Max-SAT (2.11) is subsumed by the formulation (5.16) where the matrix A contains only elements of $\{-1, 0, 1\}$ and vector b contains only integers (both of these conditions are required in Theorem 5.4). Furthermore, if this is an LP relaxation of Max-2SAT, then A is 2-in-row because each primal constraint (2.11b) contains at most 2 variables x and the variable z_c appears only in this constraint (except for the box constraints $0 \leq z_c \leq 1$, which is allowed). Alternatively, if the clause-variable incidence graph is acyclic, then the corresponding matrix A is bipartite-acyclic. Therefore, (pre-)ILMs of (2.14) w.r.t. individual variables are global minima of the LP relaxation in case of Max-2SAT or when the clause-variable incidence graph is acyclic.

Remark 5.3. *These results for LP relaxation of weighted Max-SAT already follow from our previous discussions, although we did not say it explicitly. In detail, we stated in §4.4.3 that the propagation rules for weighted Max-SAT from §2.4.1 are a specific example (up to technical details) of rule* (4.1)*. By Corollary 5.1, whenever these propagation rules are refutation complete (discussed in Remark 2.6 and §2.4.5), (pre-)ILMs of* (2.14) *w.r.t. individual variables are global minima. Theorem 5.4 can be thus seen as a generalization of these results.*

A class of problems that are related to (5.16) with 2-in-row constraint matrix A are IP2 optimization problems [75, §1.2]. These include [85] integer linear programs in the form

$$\min \sum_{i \in V} c_i x_i + \sum_{(i,j) \in E} c'_{ij} z_{ij} \tag{5.28a}$$

$$a_{ij} x_i + a'_{ij} x_j \leq b_{ij} + d_{ij} z_{ij} \qquad \forall (i,j) \in E \tag{5.28b}$$

$$x_i \in \{0,1\} \qquad \forall i \in V \tag{5.28c}$$

$$z_{ij} \in \{0,1\} \qquad \forall (i,j) \in E \tag{5.28d}$$

where (V, E) is a directed graph and the following is satisfied:

- for all $(i,j) \in E$, we have $a_{ij}, a'_{ij} \in \mathbb{Z}$, $b_{ij}, c'_{ij} \in \mathbb{R}$, $d_{ij} \in \{0,1\}$,
- for all $i \in V$, we have $c_i \in \mathbb{R}$.

Moreover, for $(i,j) \in E$, constraint (5.28b) is

- *monotone* [75, Definition 1] if $a_{ij} > 0$ and $a'_{ij} < 0$,
- *binarized* [75, Definition 2] if $a_{ij}, a'_{ij} \in \{-1, 0, 1\}$.

Problem (5.28) with monotone constraints can be solved by a reduction to minimum *st*-cut or minimum cost flow on an associated graph [75, Theorem 1.1]. Problem (5.28) with some constraints non-monotone can be transformed to the monotone case by introducing additional variables and constraints, but the inverse transformation does not preserve integrality. Consequently, for (5.28) with some constraints non-monotone, the reduction yields only an upper bound on the optimal value [75, Theorem 1.1], which is however not worse than the bound provided by the LP relaxation of (5.28) [75, §1.1.2].

It is immediate from the structure of constraints (5.28b) that each contains at most two x variables and each z variable appears only in a single inequality. Consequently, the constraint matrix of this problem is 2-in-row and, if the problem is binarized and $b_{ij} \in \mathbb{Z}$ for all $(i,j) \in E$, its LP relaxation (where we replace (5.28c) and (5.28d) by $0 \leq x_i \leq 1$ and $0 \leq z_{ij} \leq 1$, respectively) satisfies the assumptions of Theorem 5.4 and thus can be also solved by BCD.

In [75, §5-§8], there are listed many problems that can be expressed in the form (5.28) and their LP relaxations are subsumed by the primal (5.16) and also satisfy the conditions stated in Theorem 5.4. These include, e.g., generalized independent set or Min-SAT.

Except for the previously mentioned problems, we noticed in [53a, §4.5] that Theorem 5.4 applies to a suitable formulation of the *roof dual* optimization problem [23, §5.1.2, 76, §7.2] which is the dual LP relaxation of the problem of maximizing a (quadratic) pseudoboolean function [23, 76, §7.1]. We note that, again, there exist specialized algorithms for optimizing the roof dual based on network flows [23, §5.1.5, 76, §7.2].

Moreover, there exist suitable formulations of the maximum flow problem, LP relaxation of weighted vertex cover, and LP relaxation of Boolean WCSP with Potts interactions that satisfy the assumptions of Theorem 5.4. We comment on these optimization problems in detail in §5.4.

[85] In [75], the objective function (5.28a) was allowed to be more general, the domains (5.28c) could be any finite subsets of $\mathbb{Z}$ and (5.28d) even infinite subsets of $\mathbb{Z}$. Also, some inequalities from (5.28b) can be in the $\geq$ direction.

5.4 Reformulations and Optimality of BCD

We argued in §4.3 that the relation between BCD and constraint propagation holds for linear programs in any form. So, suppose that we have a primal-dual pair of linear programs and we apply BCD to the dual. The form of the complementary slackness conditions (expressed in terms of the primal variables) depends on the formulation of the dual. Unsurprisingly, the propagation rule (4.1) may be weaker, stronger, or even refutation complete, based on the precise form of the conditions. Consequently, the fixed points of BCD may be worse, better, or even global optima. This explains the fact that applicability of BCD highly depends on the precise form of the optimization problem (as exemplified in §1.2.3).

In this section, we illustrate this phenomenon in detail on three different problems. First, we show that coordinate-wise optimization applied to the dual of the usual LP relaxation of minimum weight vertex cover problem [138, §14.3] (§5.4.1) and LP formulation of maximum flow problem (§5.4.2) need not attain the optimal value. However, after a slight reformulation, these linear programs become exactly solvable by BCD which is naturally explained by the corresponding propagation rule (4.1) becoming refutation complete.

Next, we consider a special dual LP relaxation of WCSP with Potts interactions. In this case, optimality of BCD is guaranteed if the WCSP is Boolean. Interestingly, if the WCSP has acyclic factor graph and is not necessarily Boolean, optimality of BCD is no longer guaranteed, which is in contrast to the usual dual LP relaxation (1.43) where (pre-)ILMs w.r.t. individual variables are global minima under such assumptions.

For each of these problems, we also show coordinate-wise updates satisfying the relative-interior rule and discuss convergence. This section is mainly based on our exposition in [53a, §4, 54a, §5.3] with some parts from [151a, §6.3 and §7] and a few new insights.

5.4.1 Example: Vertex Cover

Recall the minimum weight vertex cover problem [103, §3.3] on an undirected graph (V, E) with non-negative vertex weights $w \in \mathbb{R}^V_+$. A *vertex cover* is a subset of vertices $S \subseteq V$ satisfying $\{i, j\} \cap S \neq \emptyset$ for each edge $\{i, j\} \in E$. The task is to find a vertex cover S such that $\sum_{i \in S} w_i$ is minimal. This problem is NP-hard and its decision version was among the first problems shown to be NP-complete [84, §4, 122, Corollary 64.1a, 110, §15.6]. The LP relaxation of this problem [138, §14.3] is the left-hand problem of the primal-dual pair

$$\min w^\top x \qquad \max \sum_{\{i,j\} \in E} y_{ij} \tag{5.29a}$$

$$x_i + x_j \geq 1 \qquad y_{ij} \geq 0 \qquad \forall \{i, j\} \in E \tag{5.29b}$$

$$x_i \geq 0 \qquad \sum_{j \in N_i} y_{ij} \leq w_i \qquad \forall i \in V. \tag{5.29c}$$

In the dual (on the right), $N_i = \{j \in V \mid \{i, j\} \in E\}$ denotes the set of neighbors of vertex $i \in V$ in the graph and $y_{\{i,j\}}$ was abbreviated to y_{ij} (so that $y_{ij} = y_{ji}$). Obviously, this is a relaxation of the aforementioned problem as there is a bijection between vertex covers of (V, E) and vectors $x \in \{0, 1\}^V$ feasible for the primal (5.29).

For now, suppose that we optimize the dual (5.29) coordinate-wise (i.e., we apply BCD with blocks of size 1). This results in a greedy procedure that gradually makes each y_{ij}

as large as possible subject to the dual constraints (5.29c). (Pre-)interior local maxima of the dual (5.29) need not be its global maxima, as the following example shows.

Example 5.2. *Let* $V = \{1, 2, 3\}$, $E = \{\{1, 2\}, \{2, 3\}, \{1, 3\}\}$, *and* $(w_1, w_2, w_3) = (1, 2, 3)$, *i.e.,* (V, E) *is the complete graph with 3 vertices. Dual-feasible solution* $y = (y_{12}, y_{23}, y_{13}) = (1, 1, 0)$ *is an interior local maximum of the dual* (5.29) *w.r.t. individual variables since none of the variables can be increased while maintaining feasibility and the coordinate-wise optimizers are unique. However, the optimal solution* $y^* = (y^*_{12}, y^*_{23}, y^*_{13}) = (0, 2, 1)$ *has better objective value.* △

The primal linear program (5.29) is also not amenable to BCD. We found [53a, Example 4] that there may be even (pre-)ILMs of the primal (5.29) w.r.t. $\mathcal{B} = \{B \mid B \subsetneq V\}$ that are not global minima.

Let us add constraints $x_i \leq 1$, $i \in V$ to the primal (as in [103, §3.3]), obtaining the primal-dual pair

$$\begin{aligned}
&\min w^\top x && \max \sum_{\{i,j\}\in E} y_{ij} + \sum_{i\in V} z_i && && (5.30a)\\
&x_i + x_j \geq 1 && y_{ij} \geq 0 && \forall \{i, j\} \in E && (5.30b)\\
&x_i \geq 0 && z_i + \sum_{j\in N_i} y_{ij} \leq w_i && \forall i \in V && (5.30c)\\
&x_i \leq 1 && z_i \leq 0 && \forall i \in V. && (5.30d)
\end{aligned}$$

Constraints (5.30d) are redundant for the primal in the sense that they do not change the optimal value. Indeed, if there was an optimal solution x of the primal (5.29) with $x_i > 1$ for some $i \in V$, then we could decrease x_i to 1 and maintain feasibility of this solution while improving or preserving the objective due to non-negativity of w_i. So, (5.29) and (5.30) have the same optimal value.

Similarly to §2.4 or §5.3, variables z can be eliminated from the dual (5.30) to formulate it as a maximization of a concave piecewise-affine function over non-negative variables, i.e.,

$$\begin{aligned}
&\max \sum_{\{i,j\}\in E} y_{ij} + \sum_{i\in V} \min\Big\{w_i - \sum_{j\in N_i} y_{ij},\, 0\Big\} && && (5.31a)\\
&y_{ij} \geq 0 && \forall \{i, j\} \in E. && (5.31b)
\end{aligned}$$

The following corollary is immediate from Theorem 5.4.

Corollary 5.5. *Any (pre-)interior local maximum w.r.t. individual variables of* (5.31) *is a global maximum of* (5.31).

Proof. The primal (5.30) has box-constrained variables $x \in [0, 1]^V$ and each primal constraint (5.30b) contains exactly two variables, so the constraint matrix is 2-in-row. All other conditions imposed by Theorem 5.4 are clearly satisfied. The optimization problem (5.31) is related to the dual (5.30) in the same way as the optimization problem (5.18) is related to the dual (5.16).

The fact that we minimize in the primal and maximize in the dual (as opposed to §5.3) does not influence refutation-completeness of the propagator when applied to the complementary slackness conditions. □

Let us now explain the different behavior of BCD by referring to the propagation rule (4.1). The difference stems from the fact that in case of (5.29), we can only propagate equality in constraints (5.29b) and infer zero values of x_i. However, in (5.30), we can additionally infer $x_i = 1$ due to the added constraint $x_i \leq 1$. This results in a stronger propagation algorithm that is even refutation complete for this particular case.

Remark 5.4. *In more detail, the ability to propagate $x_i = 1$ follows from the fact that, in analogy to Fact 1.1, optimizing* (5.31) *coordinate-wise is in correspondence to optimizing the dual* (5.30) *by BCD along blocks of variables where each block contains all the z variables and a single variable y_{ij}. Consequently, each system that is used in propagation (analogous to* (5.21)*) consists of $x_i + x_j \geq 1$ and $0 \leq x_k \leq 1$, $k \in V$ where some of these inequalities may be equalities instead.*

We derived in [151a, §7] an update for variable y_{ij} in (5.31) satisfying the relative-interior rule, namely

$$y_{ij} := \tfrac{1}{2} \max \Big\{ w_i - \sum_{k \in N_i - \{j\}} y_{ik},\, 0 \Big\} + \tfrac{1}{2} \max \Big\{ w_j - \sum_{k \in N_j - \{i\}} y_{jk},\, 0 \Big\}. \tag{5.32}$$

It is clear that this update is continuous in the other variables y. Furthermore, the right-hand side of (5.32) is always at most $(w_i + w_j)/2$, so any sequence of points y obtained by such updates (possibly except for the initial point) is bounded as $0 \leq y_{ij} \leq (w_i + w_j)/2$, $\{i, j\} \in E$. Applying Theorem 1.10 and Corollaries 1.3 and 5.5, this implies that, while performing updates (5.32) in a cyclic order for each $\{i, j\} \in E$, the current point y will converge to the set of optimizers of (5.31).

Remark 5.5. *We performed experiments with updates* (5.32) *in [151a, §7]. On each tested instance, the optimum of the LP relaxation was attained.*

Let us also note that, for a given interior local maximum of the dual (5.31)*, there exists a closed-form expression for an optimal solution of the primal* (5.30)*. This expression is a special case of the general formula that we stated in [51a, Theorem 2].*

Remark 5.6. *Although the difference between* (5.29) *and* (5.30) *might seem simple, it is not obvious at first sight (at least without knowing our results from §4.2). To support this claim, LP relaxation of the related maximum weight independent set problem was studied in [118]. The corresponding primal-dual pair has a form similar to* (5.29) *except that we swap* min/max *and the directions of the inequalities in the primal constraints* (5.29b) *and the dual constraints* (5.29c) *change to $\leq$ and $\geq$, respectively. It was noticed in [118, §VI] that optimizing the dual of such a formulation coordinate-wise does not typically solve the problem to optimality. Of course, if the dual LP relaxation is reformulated analogously to* (5.31)*, it becomes optimally solvable by BCD, which follows from Theorem 5.4. However, instead of such a reformulation, a barrier method was proposed in [118, §VI].*

5.4.2 Example: Maximum Flow

One of the earliest problems of linear programming is the maximum flow problem [59, §I] where one is given a directed graph (V, E), two distinguished nodes $s \in V$ (source) and $t \in V$ (sink), and non-negative edge capacities $c \in \mathbb{R}^E_+$. The usual LP formulation of

the maximum flow problem [59, §I, 110, §6.1, 102, §6.1] is the right-hand linear program of the primal-dual pair

$$\begin{aligned}
\min c^\top q \qquad & \max \sum_{(s,j)\in E} f_{sj} && && \text{(5.33a)}\\
q_{sj} + x_j \geq 1 \qquad & f_{sj} \geq 0 && \forall (s,j) \in E && \text{(5.33b)}\\
q_{it} - x_i \geq 0 \qquad & f_{it} \geq 0 && \forall (i,t) \in E && \text{(5.33c)}\\
q_{ij} - x_i + x_j \geq 0 \qquad & f_{ij} \geq 0 && \forall (i,j) \in E,\ i \neq s,\ j \neq t && \text{(5.33d)}\\
q_{ij} \geq 0 \qquad & f_{ij} \leq c_{ij} && \forall (i,j) \in E && \text{(5.33e)}\\
x_k \in \mathbb{R} \qquad & \sum_{(i,k)\in E} f_{ik} = \sum_{(k,j)\in E} f_{kj} && \forall k \in V - \{s,t\} && \text{(5.33f)}
\end{aligned}$$

where the problem on the left is well known to be the LP formulation of the minimum st-cut problem [59, §I12, 110, §6.1]. As usual, we will refer to the left-hand problem (5.33) as the primal and to the right-hand problem (5.33) as the dual.

For simplicity, we assumed that there are no incoming edges to s and no outgoing edges from t. Furthermore, we expect that $(s,t) \notin E$ – such an edge would be always saturated (i.e., $f_{st} = c_{st}$) by a maximum flow. Finally (in contrast to the notation that was used in the previous section, §5.4.1), we emphasise that f_{ij} and f_{ji} are in general different variables here because the graph is directed and $(i,j) \neq (j,i)$.

Suppose that we optimize the dual (5.33) coordinate-wise (i.e., apply block-coordinate ascent with blocks of size 1). Clearly, the flow-conservation constraints (5.33f) make any update impossible, hence any f feasible for the dual (5.33) is an interior local maximum w.r.t. individual variables (cf. Example 1.6).

Remark 5.7. *This corresponds to the fact that the propagation rule* (4.1) *is very weak when applied to complementary slackness conditions expressed in terms of the primal variables* (q,x) *for some fixed dual-feasible* f*. In detail, some* q *variables are set to 0 and some of the primal constraints* (5.33b)-(5.33d) *are equalities. However, since the* x *variables are unbounded and each variable* q *appears only in a single constraint* (5.33b)-(5.33d) *(without any other* q *variables), no inequalities are implied to hold with equality, i.e., no propagation is done.*

We identified a more suitable formulation that uses the fact that one can impose box-constraints on the x variables in the primal (5.33), namely replace constraints $x_k \in \mathbb{R}$ with constraints $0 \leq x_k \leq 1$. Such a change does not influence the optimal value of the primal (5.33) due to its integrality (for details, we refer to [59, §I12, 110, §6.1]). This results in the primal-dual pair

$$\begin{aligned}
\min c^\top q \qquad & \max \sum_{(s,j)\in E} f_{sj} + \sum_{k\in V-\{s,t\}} z_k && && \text{(5.34a)}\\
q_{sj} + x_j \geq 1 \qquad & f_{sj} \geq 0 && \forall (s,j) \in E && \text{(5.34b)}\\
q_{it} - x_i \geq 0 \qquad & f_{it} \geq 0 && \forall (i,t) \in E && \text{(5.34c)}\\
q_{ij} - x_i + x_j \geq 0 \qquad & f_{ij} \geq 0 && \forall (i,j) \in E,\ i \neq s,\ j \neq t && \text{(5.34d)}\\
q_{ij} \geq 0 \qquad & f_{ij} \leq c_{ij} && \forall (i,j) \in E && \text{(5.34e)}\\
x_k \geq 0 \qquad & z_k \leq R_k(f) && \forall k \in V - \{s,t\} && \text{(5.34f)}\\
x_k \leq 1 \qquad & z_k \leq 0 && \forall k \in V - \{s,t\} && \text{(5.34g)}
\end{aligned}$$

where we abbreviated

$$R_k(f) = \sum_{(k,j)\in E} f_{kj} - \sum_{(i,k)\in E} f_{ik} \tag{5.35}$$

for $k \in V - \{s,t\}$.

Note that the only difference between the duals (5.33) and (5.34) is in the objective (5.34a) and constraints (5.34f) and (5.34g).

The additional z variables in (5.34) can be eliminated similarly as in §2.4 or §5.4.1. In detail, in any optimal solution (z,f) of dual (5.34), it holds that $z_k = \min\{R_k(f),\, 0\}$ for all $k \in V - \{s,t\}$. Making this substitution yields the optimization problem

$$\max \sum_{(s,j)\in E} f_{sj} + \sum_{k\in V-\{s,t\}} \min\{R_k(f),\, 0\} \tag{5.36a}$$

$$0 \le f_{ij} \le c_{ij} \qquad \forall (i,j) \in E. \tag{5.36b}$$

By Theorem 5.4, we have the following optimality result.

Corollary 5.6. *Any (pre-)interior local maximum w.r.t. individual variables of* (5.36) *is a global maximum of* (5.36)*.*

Proof. The primal (5.34) is in the form (5.16). Also, each constraint contains at most two x variables, so the constraint matrix is 2-in-row. All other conditions of Theorem 5.4 are clearly satisfied. The result now follows from Theorem 5.4 by analogous reasoning as in the proof of Corollary 5.5. □

Remark 5.8. *The reason for the success of block-coordinate ascent for this formulation can be again linked to the ability of the corresponding propagation rule to set some x variables to 0 or 1 and consequently also set q variables to 0 using the primal constraints* (5.34b)-(5.34d)*. Since any interior local maximum of* (5.36) *w.r.t. individual variables is a global maximum of* (5.36)*, the propagation rule is refutation complete by Corollary 5.1. This is in sharp contrast to the formulation* (5.33) *and Remark 5.7.*

An update satisfying the relative-interior rule for a variable f_{ij}, $(i,j) \in E$ in (5.36) depends on whether $i = s$ or $j = t$. We derived this update in [53a, §4.3.1] and it reads

$$f_{ij} := \begin{cases} \frac{1}{2}\big(c_{sj} + \pi_{sj}(R_j(f) + f_{sj})\big) & \text{if } i = s \\ \frac{1}{2}\big(\pi_{it}(f_{it} - R_i(f)) + c_{it}\big) & \text{if } j = t \\ \frac{1}{2}\big(\pi_{ij}(f_{ij} - R_i(f)) + \pi_{ij}(R_j(f) + f_{ij})\big) & \text{if } i \ne s \text{ and } j \ne t \end{cases} \tag{5.37}$$

where $\pi_{ij} \colon \mathbb{R} \to [0, c_{ij}]$ is the projection onto $[0, c_{ij}]$, i.e., $\pi_{ij}(a) = \max\{0, \min\{c_{ij}, a\}\}$. These updates are clearly continuous in the f variables and any sequence of points f feasible for (5.36) is bounded due to the capacity constraints (5.36b). Therefore, by Theorem 1.10 and Corollaries 1.3 and 5.6, while performing the updates (5.37) to individual variables f_{ij} of (5.36) in a cyclic order, the current point f will converge to the set of optimizers of (5.36).

Remark 5.9. *The updates* (5.37) *have an informal interpretation that was explained in [53a, §4.3.1]. Briefly, performing updates* (5.37) *for f_{sj}, $(s,j) \in E$ can be interpreted as trying to push flow into the network from the source (and symmetrically for f_{it}, $(i,t) \in E$), while the updates of f_{ij} for the 'intermediate' edges $(i,j) \in E$ with $i \ne s$ and $j \ne t$ propagate the flow throughout the network and, in a precise sense, try to make $|R_k(f)|$ small.*

Remark 5.10. *We practically evaluated coordinate-wise optimization of a slightly different formulation [51a, Equation (14)] where the objective was to be minimized. Using publicly available instances of the maximum flow problem from computer vision, we verified that coordinate-wise optimization is able to attain the optimal value (up to machine precision) [51a, §4.3].*

In analogy to Remark 5.5, there is also a formula for an optimal solution of the primal (5.34) *based on an interior local maximum of* (5.36)*.*

5.4.3 Example: WCSP with Potts Interactions

To show a slightly different example, let us consider the particular case of pairwise WCSP with Potts interactions, i.e., we have $|S| \leq 2$ for all $S \in C$, the unary weight functions $g_{\{i\}}$, $i \in V$ can be arbitrary whereas the weight functions of arity 2 are given by [86]

$$g_{\{i,j\}}(k_i, k_j) = -[\![k_i \neq k_j]\!] \qquad \forall \{i,j\} \in C_{\geq 2},\ k \in D^{\{i,j\}}. \tag{5.38}$$

With such weight functions, one can add the constraints

$$\varphi_{\{i,j\},i}(k) + \varphi_{\{i,j\},j}(k) = 0 \qquad \forall \{i,j\} \in C_{\geq 2},\ k \in D \tag{5.39a}$$

$$\varphi_{\{i,j\},i}(k), \varphi_{\{i,j\},j}(k) \in \left[-\tfrac{1}{2}, \tfrac{1}{2}\right] \qquad \forall \{i,j\} \in C_{\geq 2},\ k \in D \tag{5.39b}$$

to the LP relaxation (1.43) without changing its optimal value [151a, §6.3, 53a, §4.4]. This can be derived by transforming the dual of the LP relaxation that was stated in [86, Equation (ULP), 113, Equation (6)].

We will now show how the LP relaxation (1.43) with constraints (5.39) can be simplified. First, one can notice that if φ satisfies (5.39), then we have for each $\{i,j\} \in C_{\geq 2}$ and $k \in D^{\{i,j\}}$ that

$$g^{\varphi}_{\{i,j\}}(k_i, k_j) = \begin{cases} \overbrace{g_{\{i,j\}}(k_i, k_i)}^{0 \text{ (due to (5.38))}} + \overbrace{\varphi_{\{i,j\},i}(k_i) + \varphi_{\{i,j\},j}(k_i)}^{0 \text{ (due to (5.39a))}} = 0 & \text{if } k_i = k_j \\ \underbrace{g_{\{i,j\}}(k_i, k_j)}_{-1 \text{ (due to (5.38))}} + \underbrace{\varphi_{\{i,j\},i}(k_i) + \varphi_{\{i,j\},j}(k_j)}_{\leq 1 \text{ (due to (5.39b))}} \leq 0 & \text{if } k_i \neq k_j \end{cases} \tag{5.40}$$

and therefore

$$\max_{k \in D^{\{i,j\}}} g^{\varphi}_{\{i,j\}}(k_i, k_j) = 0 \qquad \forall \{i,j\} \in C_{\geq 2}. \tag{5.41}$$

Second, we assign an arbitrary direction to each edge in the undirected graph $(V, C_{\geq 2})$ to obtain a directed graph (V, E) so that for each undirected edge $\{i,j\} \in C_{\geq 2}$, there is exactly one directed edge $(i,j) \in E$. In other words, (V, E) is an orientation [116, §8.3.3] of $(V, C_{\geq 2})$. Consequently, constraint (5.39a) can be eliminated by substituting the φ variables by new λ variables,

$$\lambda_{ij}(k) = -\varphi_{\{i,j\},i}(k) = \varphi_{\{i,j\},j}(k) \qquad \forall (i,j) \in E,\ k \in D, \tag{5.42}$$

[86] More generally, there can be a non-negative scalar $w_{ij} \geq 0$ for each $\{i,j\} \in C_{\geq 2}$ and the weight functions are then defined by $g_{\{i,j\}}(k_i, k_j) = -w_{ij}[\![k_i \neq k_j]\!]$, as in [25, §7, 86, §3, 113, §2.1]. Our results from this section would also apply to this generalization. For clarity, we note that the sign is inverted in our case here because we maximize the WCSP objective whereas the papers [25, 86, 113] consider minimization. WCSP with Potts interactions is also known as the uniform labeling problem [86, §3] which is a special case of the metric labeling problem [86, §1.2].

which reduces the total number of variables by half.

All in all, using the variable substitution (5.42) together with (5.41), we can transform the optimization problem (1.43) with constraints (5.39) into the simpler form

$$\min \sum_{i\in V} \max_{k\in D} \Big(\overbrace{g_{\{i\}}(k) + \sum_{(i,j)\in E} \lambda_{ij}(k) - \sum_{(j,i)\in E} \lambda_{ji}(k)}^{g^{\uparrow\lambda}_{\{i\}}(k)} \Big) \tag{5.43a}$$

$$-\tfrac{1}{2} \leq \lambda_{ij}(k) \leq \tfrac{1}{2} \qquad \forall (i,j) \in E,\ k \in D \tag{5.43b}$$

that uses the new variables λ. Note that we defined new notation $g^{\uparrow\lambda}_{\{i\}}(k)$ for the term in the maximum in (5.43a) – we use the arrow symbol to distinguish it from the transformation that was defined earlier in (1.41).

Based on the previous discussion, it follows that the optimal value of (5.43) coincides with the optimal value of (1.43). Moreover, any optimal solution λ of (5.43) defines an optimal solution φ of (1.43) via (5.42).

Interestingly, (5.42) need not map ILMs of (5.43) to pre-ILMs of (1.43) if $|D| \geq 3$. More precisely, for an ILM λ of (5.43) w.r.t. individual variables, φ defined by (5.42) need not be an ILM (or even a pre-ILM) of (1.43) w.r.t. individual variables. We gave a detailed example of this phenomenon in [53a, Example 5]. Let us note that the example given in [53a] used a chain graph $(V, C_{\geq 2})$ and $|D| = 3$. Thus, (pre-)ILMs of (5.43) need not be global minima even if the underlying graph is acyclic. This is in contrast with the formulation (1.43) where any (pre-)ILM is a global minimum if the factor graph of (V, C) is acyclic (or, in particular, if the WCSP is pairwise and the graph $(V, C_{\geq 2})$ is acyclic) – this follows from Fact 1.2 combined with the results in §4.4.1.

However, as given by the following corollary, if the WCSP is in addition Boolean, then any ILM of (5.43) is a global minimum. In this particular case, the problem becomes supermodular [119, §11.2.1] and thus the optimal value of the LP relaxation (1.43) (and hence also (5.43)) coincides with the optimal value of the WCSP by Fact 1.2. [87]

Corollary 5.7. *If $|D| = 2$, then any (pre-)ILM w.r.t. individual variables of* (5.43) *is a global minimum of* (5.43)*.*

Proof. Let the set of labels be $D = \{\mathsf{a}, \mathsf{b}\}$. It is easy to see that

$$\sum_{i\in V} \Big(\sum_{(i,j)\in E} \lambda_{ij}(\mathsf{b}) - \sum_{(j,i)\in E} \lambda_{ji}(\mathsf{b}) \Big) = 0, \tag{5.44}$$

so $\sum_{i\in V} g^{\uparrow\lambda}_{\{i\}}(\mathsf{b}) = \sum_{i\in V} g_{\{i\}}(\mathsf{b})$. In addition, $\max\{x, y\} = \max\{x - y, 0\} + y$ holds for any $x, y \in \mathbb{R}$, hence the objective (5.43a) can be rewritten as

$$\sum_{i\in V} \max\big\{g^{\uparrow\lambda}_{\{i\}}(\mathsf{a}), g^{\uparrow\lambda}_{\{i\}}(\mathsf{b})\big\} = \sum_{i\in V} \max\big\{g^{\uparrow\lambda}_{\{i\}}(\mathsf{a}) - g^{\uparrow\lambda}_{\{i\}}(\mathsf{b}), 0\big\} + \sum_{i\in V} g_{\{i\}}(\mathsf{b}). \tag{5.45}$$

[87] In case of supermodular (or general Boolean pairwise) problems, the fixed points φ of the BCD algorithms from §1.5.4.1 are optimal for (1.43) (recall Facts 1.2 and 1.3). However, we emphasise that this fact does not (at least directly) imply Corollary 5.7 because the optimization problem (1.43) is different from (5.43). As mentioned earlier, the 'acyclic' part of Fact 1.2 does not carry over to the optimization problem here because (5.43) may have non-optimal (pre-)ILMs even if the underlying graph $(V, C_{\geq 2}) = (V, \{\{i, j\} \mid (i, j) \in E\})$ is acyclic (which is in contrast to (1.43)).

After omitting the constant term $\sum_{i\in V} g_{\{i\}}(\mathsf{b})$ (which does not influence solvability by BCD), the rewritten objective is in the form (5.18a) with a 2-in-row matrix A whose elements are from the set $\{-1, 0, 1\}$ and the box constraints (5.43b) are subsumed by the constraints (5.18b). The corollary now follows from Theorem 5.4. □

We remark that, if $|D| \geq 3$, then solving this relaxation is as hard as solving any linear program [113].

In [151a, §6.3], we derived an update satisfying the relative-interior rule for the individual λ variables in the optimization problem (5.43). For $(i, j) \in E$ and $k \in D$, the update reads

$$\lambda_{ij}(k) := \tfrac{1}{2}\pi\Big(\max_{\substack{\ell\in D\\ \ell\neq k}} g^{\uparrow\lambda}_{\{i\}}(\ell) - g^{\uparrow\lambda}_{\{i\}}(k) + \lambda_{ij}(k)\Big) + \tfrac{1}{2}\pi\Big(\max_{\substack{\ell\in D\\ \ell\neq k}} g^{\uparrow\lambda}_{\{j\}}(\ell) - g^{\uparrow\lambda}_{\{j\}}(k) - \lambda_{ij}(k)\Big) \quad (5.46)$$

where $\pi\colon \mathbb{R} \to \big[-\frac{1}{2}, \frac{1}{2}\big]$ is the projection onto $\big[-\frac{1}{2}, \frac{1}{2}\big]$, i.e., $\pi(a) = \max\big\{-\frac{1}{2}, \min\big\{\frac{1}{2}, a\big\}\big\}$.

Clearly, the values of λ are bounded due to the box constraints (5.43b) and the updates (5.46) are continuous in λ. Therefore, Theorem 1.10 yields that if the individual coordinates of λ are updated by (5.46) in a cyclic order, the point will converge to the set of pre-ILMs w.r.t. individual variables of (5.43). Moreover, by Corollary 5.7 (combined with Corollary 1.3), if the WCSP instance is in addition Boolean, it will converge to the set of global minima of (5.43).

Remark 5.11. *We experimentally evaluated coordinate-wise minimization of* (5.43) *by updates* (5.46) *on synthetic image segmentation tasks in [151a]. In all evaluated criteria (i.e., attained objective of the relaxation, runtime, and an obtained segmentation), the updates* (5.46) *were competitive to max-sum diffusion which optimizes* (1.43) *by relative-interior updates. For more details, we refer to the supplementary material of [151a].*

5.5 Discussion

The starting point of this chapter is Corollary 5.1 that follows from our results in §4 and links solvability of linear programs by BCD to refutation-completeness of an associated propagator. Thanks to this result, we identified three new classes of linear programs where (pre-)ILMs are global optima: LP formulation of WCSP (the dual (5.1)) and the problem (5.16) with 2-in-row or bipartite-acyclic matrix (and additional constraints on A and b). Note that the constraint matrix in (5.1) is neither 2-in-row nor bipartite-acyclic. The practical impact of these results is limited because performing BCD with the relative-interior rule in the dual (5.1) is likely intractable and all of the listed linear programs with 2-in-row constraint matrix can be reduced to network flow problems and solved by combinatorial algorithms. Although these classes are relatively narrow, we believe that this is not an exhaustive list of linear programs solvable by BCD and that our proof technique may have the potential to identify other such classes in the future.

The presented results are of theoretical interest. For example, we proved that the propagator defined in §4.1 in the case of linear program (5.1) enforces positive consistency. This can be compared to the case of the LP relaxation (1.43) where the associated propagator (see §4.4.1) enforces arc consistency.

Furthermore, we also explained and exemplified the differences in applicability of BCD caused by reformulations of problems in §5.4. This may provide theoretical background

to identify what formulations of linear programs are more amenable to BCD or determine whether the fixed points of BCD are global optima in other cases.

As an example, in [53a, §5], we considered the class of problems (5.16) where matrix $A \in \{-1, 0, 1\}^{m \times n}$ contains at most two non-zero entries in each *column* and $b \in \mathbb{Z}^m$. This class includes, e.g., the LP formulations of the assignment problem and shortest path problem, or the LP relaxation of maximum weight matching. For these problems, we optimized the corresponding formulation (5.18) coordinate-wise with the relative-interior rule to find that the objective of fixed points is typically close to the optimal value of the linear program on randomly generated instances. This is not guaranteed in theory as we were able to find a small instance with a non-optimal ILM for each of these problems. Based on our experience, finding such counter-examples is simpler when working directly with the associated propagator via our results from §4 (as opposed to trying to find a non-optimal ILM directly).

More generally, one could ask for which linear programs there is a formulation where any (pre-)ILM w.r.t. some small subsets of variables is a global optimum. Without further restrictions (such as linear-time reduction), this question is trivial: Let $Y \subseteq \mathbb{R}^m$ be a polytope, $\mathcal{V} \subseteq Y$ be the set of its vertices, and $b \in \mathbb{R}^m$. The optimum of the linear program $\min\{\, b^\top y \mid y \in Y \,\}$ is attained in at least one vertex $v \in \mathcal{V}$ [119, §3.3]. It is easy to show that any LM (and thus any (pre-)ILM) of

$$\min \sum_{y \in \mathcal{V}} (b^\top y) \lambda(y) \tag{5.47a}$$

$$\lambda(y) \geq 0 \qquad \forall y \in \mathcal{V} \tag{5.47b}$$

$$\sum_{y \in \mathcal{V}} \lambda(y) = 1 \tag{5.47c}$$

w.r.t. all blocks of variables of size 2 is a global minimum. An optimizer of the original problem is obtained by $y^* = \sum_{y \in \mathcal{V}} \lambda(y) y$. There exist even more trivial examples of such transformations that require non-linear time.

Remark 5.12. *On a different but related note, it can be shown [49a, Theorem 1] that for any polyhedron $Y \subseteq \mathbb{R}^m$, there exists a finite set of directions so that for any convex differentiable function $f \colon Y \to \mathbb{R}$ (in particular, any linear function), any LM*[88] *of f on Y w.r.t. this set of directions (recall Remark 1.3) is a global minimum.*

Without going into too much details, one can find the tangent cone of each face of Y [44, Definition 6.2.2]. Since Y is a polyhedron, this tangent cone is polyhedral and thus finitely generated [44, Theorem 1.3.12, 123, Corollary 7.1a]. The set of directions is obtained as the union of the generators of tangent cones for all faces of Y. Since each polyhedron has only a finite number of faces, the resulting set of directions is finite. Also, it is possible to transform the optimization of any problem along a set of directions into a form where it corresponds to coordinate-wise minimization via introducing additional variables (whose number is not greater than the number of directions).

[88] In [49a], we assumed the relative-interior rule, but this is in fact not necessary.

Conclusion

In this thesis, our practical interests laid in designing algorithms for approximately solving (more precisely, bounding the optimal value of) large-scale linear programs that nowadays emerge and cannot be tackled by off-the-shelf LP solvers in practice. From the theoretical viewpoint, we provided a connection between local consistencies and BCD for LP problems which resulted in a characterization of linear programs optimally solvable by BCD in terms of constraint propagation. In turn, this characterization helped us identify new classes of problems solvable by BCD. We now conclude with an overview of our contributions that were presented in the previous chapters and also discuss their implications for scientific development in the future.

Contributions

The basis for our theoretical results is the framework for approximate optimization of large-scale linear programs which was introduced in §2.2. In detail, one applies (generally problem-dependent) constraint propagation rules to the complementary slackness conditions in order to try to detect their infeasibility. If infeasibility is detected, any certificate of infeasibility turns out to be a direction that can be used to improve a current dual solution. We argued that this scheme subsumes the VAC / Augmenting DAG algorithm [33, 95, 146] and used it as an illustrative example (in §2.3). Next, we designed an algorithm based on constraint propagation for approximately optimizing the dual LP relaxation of weighted Max-SAT, including experiments on a publicly available benchmark (in §2.4).

The more complex algorithm that we outlined in §3.2 can be also interpreted as an application of the aforementioned framework. This algorithm computes a bound on the WCSP optimal value by approximately optimizing the problem (1.45) which has an exponential number of constraints. Although the optimization problem (1.45) was already proposed in [92], we have newly shown that one can use any method to detect unsatisfiability of the active-tuple CSP to improve the bound. [89] In other words, one can enforce any local consistency in the active-tuple CSP. This is in contrast to previous approaches for obtaining bounds using constraint propagation (that we overviewed in §1.5.4.2) which needed to be tailored to a single chosen (soft) local consistency. We experimentally verified that our method (implemented with singleton arc consistency) is able to provide superior or at least frequently competitive bounds when compared to other (soft) local consistencies (in §3.2.5). The cost for this is that our method does not produce a reparametrization of the input WCSP but only a super-reparametrization which may not preserve the objective value of the individual assignments or even the set of optimal assignments. Since the properties of super-reparametrizations or of the optimization problem (1.45) were not theoretically studied before, we filled in this gap in §3 and §5.2.

Our crucial result from §4 is the identification of the special (yet natural) constraint propagation rule (4.1) which is applicable to any system of linear inequalities and equalities. The stopping points of the method from §2.2 with this propagation rule coincide

[89] On a high level, our method can be interpreted as improving an upper bound on the WCSP optimal value by sequentially detecting unsatisfiability of certain CSPs.

with pre-ILMs (i.e., points from which no BCD updates can improve the objective). Consequently, although these methods are not guaranteed to reach a global optimum, none of the methods can improve the stopping points of the other. Moreover, we precisely characterized the types of local (and global) minima in BCD using the associated propagator in Corollary 4.1.

Next, we discovered that some known local consistencies can be enforced by this propagation rule if it is applied to a suitable system of linear inequalities (consequently, they can be also enforced by BCD with the relative-interior rule). For example, our propagator (or BCD) enforces/performs

- arc consistency, when applied to the dual of basic LP relaxation of WCSP (§4.4.1),
- pairwise consistency, if a different coupling scheme is used (§4.4.1),
- positive consistency, when applied to the linear program (1.45) (§5.2.2),
- unit propagation, when applied to the dual of LP relaxation of SAT (§4.4.2).

We also classified some popular BCD methods designed for approximate optimization of the dual LP relaxation of WCSP in terms of whether they satisfy the relative-interior rule or if their fixed points are at least pre-ILMs (in §4.4.1).

Furthermore, we precisely characterized linear programs solvable by BCD by refutation-completeness of the propagation rule (4.1) in §5.1. This provides a new technique for proving optimality of BCD which was exemplified in §5.2.1 by showing that the optimization problem (1.45) can be solved to optimality by BCD, which is however likely intractable (following §5.2.3). In addition to that, we identified in §5.3 two new classes of linear programs solvable by BCD that subsume several optimization problems, including, e.g., a suitable formulation of the maximum flow problem or LP relaxations of some combinatorial problems.

Finally, in §5.4, we analyzed three optimization problems where applicability of BCD highly depends on the precise formulation of the optimization problem – this was an LP formulation of the maximum flow problem, LP relaxation of weighted vertex cover, and a special LP relaxation of WCSP with Potts interactions.

Throughout our presentation, we referred to our experimental results from [151a, 53a, 51a] where we applied BCD with the relative-interior rule to various linear programs, namely LP formulations of maximum flow, shortest paths, or assignment problem and LP relaxations of weighted vertex cover, weighted partial Max-SAT, maximum weight matching, or WCSP with Potts interactions. For each of these problems, we designed a closed-form coordinate-wise update satisfying the relative-interior rule and evaluated the quality of the resulting stopping points that were typically at least close to global optima of the linear programs (sometimes even guaranteed to be optimal, as discussed above).

Further Development

We believe that the constraint-propagation-based approach from §2 has the potential to be useful when approximately solving other large-scale linear programs, which is indicated, e.g., by our experimental results in §2.4.4 and §3.2.5. This approach seems to provide more flexibility when compared to BCD because BCD updates may not be easily applicable to some problems. An example is the case of optimizing an upper bound on WCSP over its super-reparametrizations (1.45): applying BCD with the relative-interior rule is likely intractable, whereas BCD without the relative-interior rule may become trivial (see §5.2.3).

By placing this optimization problem into our approach with constraint propagation, we were able to design a method that is weaker than Algorithm 4.2 but is capable of providing useful bounds and can be adapted to any level of local consistency, resulting in different trade-offs between bound quality and runtime.

Let us now focus on our algorithm from §3.2 in more detail. Because its subroutines are involved and the algorithm offers significant flexibility, there may be multiple areas that could be refined in order to even further improve the provided bounds or achieve speed-ups. First, there is the choice of the local consistency that is enforced in the active-tuple CSP. Second, one can also choose the specific propagation algorithm to enforce the chosen local consistency – e.g., the computational effort for enforcing SAC varies, depending on the precise choice of the propagation algorithm [18]. Third, one may invent better heuristics or theoretical background for computing deactivating directions (or improving directions in general) that would be more suitable in the sense of not increasing the objective values for the individual assignments in the super-reparametrized WCSP so much. Lastly, there are also important considerations for implementation details, including, e.g., better choice of hyper-parameters in capacity scaling or introduction of tailored stopping conditions. Improving any of these areas may have a considerable impact on the performance of this method.

In our work, we did not experiment with using the computed bounds in branch-and-bound search. Further experiments are necessary to evaluate whether our techniques will make it possible to design better algorithms to compute exact or approximate solutions for some problems of combinatorial optimization.

Aside from possible future algorithms, we believe that our results open a number of questions for subsequent theoretical research. For example, for which WCSP instances does it hold that their set of optimal solutions is equal to the set of solutions of some CSP instance with the same structure (§3.3.2).

Next, the relation between BCD and local consistencies that we presented in §4 provides useful background for theoretical evaluation of BCD methods. As an example, one can compare different formulations of a single optimization problem to find out which is more amenable to BCD by comparing the strength of the corresponding propagators (as we did in §5.4). Other interesting tasks include, e.g., to identify other problems where the propagator from §4.1 corresponds to some well-known propagation rule or local consistency (aside from the cases that we listed in the section above). For instance, one could analyze the meaning of the propagator in case of the maximum flow formulation (5.34) and see if Algorithm 4.2 in this case corresponds to some already known maximum-flow algorithm. Another research direction may be to study which reformulations preserve (pre-)ILMs or LMs and which do not. In general, the identified connection may provide theoretical basis for analysis of BCD in terms of constraint propagation.

Our results from §5 may help discover new classes of linear programs where the fixed points of BCD are global optima or may lead to better design for choices of blocks of variables so that the propagation is more effective and BCD thus reaches at least better fixed points. Moreover, it is natural to ask for which problems is the constraint propagation rule (4.1) refutation complete – this kind of problems is studied for classical local consistencies in CSPs (recall §1.4.1.3) and could perhaps lead to a different characterization of the class of linear programs solvable by BCD.

A broader question is whether the obtained results can be extended beyond linear programs, e.g., to general convex optimization problems. To generalize the constraint-

propagation-based approach from §2, a possible starting point could be stating the optimality conditions (or conditions that are at least necessary for optimality) in a suitable way that, if proved infeasible, would allow us to obtain an improving direction or a feasible point with better objective. Regarding a generalization of the results from §4 or §5, for this one needs to identify the precise form of constraint propagation that corresponds to performing BCD. Of course, such a constraint propagation rule is not guaranteed to exist in more general cases.

Appendix

List of Publications

We enclose the list of publications of the author of this thesis. All publications are related to the topic of the dissertation and are indexed in Scopus. The list of citations is based on Google Scholar with self-citations omitted. All stated information is as of May 2nd, 2022.

Publications in Impacted Journals and CORE A*/A Conferences

Since there is a delay in indexing by Web of Science (WoS), the conference papers listed in this section are not currently indexed in WoS, but the proceedings of at least 7 previous years of the conferences are indexed.

- Dlask, T. and Werner, T. Classes of linear programs solvable by coordinate-wise minimization. In: *Annals of Mathematics and Artificial Intelligence* (2021). Cited as [53a].
  - Authorship shares: Dlask: 75%, Werner: 25%.

- Dlask, T. and Werner, T. Bounding linear programs by constraint propagation: application to Max-SAT. In: *International Conference on Principles and Practice of Constraint Programming* (2020). Cited as [52a].
  - Authorship shares: Dlask: 55%, Werner: 45%.
  - Rank A conference according to the CORE Conference Rankings
  - Cited by:
    - Marino, R. Learning from survey propagation: a neural network for MAX-E-3-SAT. In: *Machine Learning: Science and Technology* (2021).
    - Montalbano, P., de Givry, S., and Katsirelos, G. Contrainte de sac-à-dos à choix multiples dans les réseaux de fonctions de coûts. In: *Proceedings of JFPC* (2021).

- Dlask, T. and Werner, T. On relation between constraint propagation and block-coordinate descent in linear programs. In: *International Conference on Principles and Practice of Constraint Programming* (2020). Cited as [54a].
  - Authorship shares: Dlask: 60%, Werner: 40%.
  - Rank A conference according to the CORE Conference Rankings
  - Cited by:
    - Montalbano, P., de Givry, S., and Katsirelos, G. Contrainte de sac-à-dos à choix multiples dans les réseaux de fonctions de coûts. In: *Proceedings of JFPC* (2021).

- Dlask, T., Werner, T., and de Givry, S. Bounds on Weighted CSPs Using Constraint Propagation and Super-Reparametrizations. In: *International Conference on Principles and Practice of Constraint Programming* (2021). Cited as [55a].
  - Authorship shares: Dlask: 45%, Werner: 35%, de Givry: 20%.
  - Rank A conference according to the CORE Conference Rankings

- Werner, T., Průša, D., and Dlask, T. Relative Interior Rule in Block-Coordinate Descent. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020). Cited as [151a].
  - Authorship shares: Werner: 40%, Průša: 40%, Dlask: 20%.
  - Rank A* conference according to the CORE Conference Rankings
  - Cited by:
    - Lange, J.H. and Swoboda, P. Efficient message passing for 0–1 ILPs with binary decision diagrams. In: *International Conference on Machine Learning* (2021).
    - Abbas, A. and Swoboda, P. FastDOG: fast discrete optimization on GPU. *arXiv preprint arXiv:2111.10270* (2021).
    - Savchynskyy, B. Discrete graphical models -— an optimization perspective. In: *Foundations and Trends in Computer Graphics and Vision* (2019).

Other Publications

- Dlask, T. and Werner, T. A class of linear programs solvable by coordinate-wise minimization. In: *Learning and Intelligent Optimization* (2020). Cited as [51a].
  - Authorship shares: Dlask: 65%, Werner: 35%.
  - Note: the proceedings of this conference from 5 previous years were indexed by WoS; we do not know whether the proceedings from 2020 will be indexed.
  - Cited by:
    - Lange, J.H. and Swoboda, P. Efficient message passing for 0–1 ILPs with binary decision diagrams. In: *International Conference on Machine Learning* (2021).
- Dlask, T. Unit propagation by means of coordinate-wise minimization. In: *International Conference on Machine Learning, Optimization, and Data Science* (2020). Cited as [50a].
  - Note: the proceedings of this conference from years 2017, 2019, and 2021 were indexed by WoS; we do not know whether the proceedings from 2020 will be indexed.
- Dlask, T. On coordinate-wise minimization applied to general convex optimization problems. In: *Procedia Computer Science* (2020). Cited as [49a].

Submissions Under Review

- Dlask, T., Werner, T., and de Givry, S. Super-Reparametrizations of Weighted CSPs: Properties and Optimization Perspective. Cited as [56a].
  - Authorship shares: Dlask: 45%, Werner: 35%, de Givry: 20%.
  - Submitted to Artificial Intelligence in 2021, no reviews yet.
  - Preprint is available online: `https://arxiv.org/abs/2201.02018`.

List of Abbreviations

AC	Arc Consistency
a.k.a.	also known as
BCD	Block-Coordinate Descent
CC	Cycle Consistency
cf.	compare
CSP	Constraint Satisfaction Problem
DAC	Directional Arc Consistency
DAG	Directed Acyclic Graph
EDAC	Existential Directional Arc Consistency
EDmaxRPC	Existential Directional max Restricted Path Consistency
EDPIC	Existential Directional Path Inverse Consistency
e.g.	for example
etc.	and so forth
FDAC	Full Directional Arc Consistency
i.e.	that is
ILM	Interior Local Minimum
LM	Local Minimum
LP	Linear Programming
Max-SAT	Maximum Satisfiability
maxRPC	max-Restricted Path Consistency
MPLP	Max Product Linear Programming
NP	complexity class non-deterministic polynomial time (as in NP-hard or NP-complete)
OSAC	Optimal Soft Arc Consistency
PIC	Path Inverse Consistency
pre-ILM	pre-Interior Local Minimum
RPC	Restricted Path Consistency
SAT	boolean satisfiability problem of a formula in conjunctive normal form
SPAM	Shortest Path Adaptive Minorant
SRMP	Sequential Reweighted Message Passing
TRW-S	Sequential Tree-Reweighted message passing
VAC	Virtual Arc Consistency
VCC-SR	Virtual Cycle Consistency via Super-Reparametrizations
VSAC-SR	Virtual Singleton Arc Consistency via Super-Reparametrizations
WCSP	Weighted Constraint Satisfaction Problem
WoS	Web of Science
w.r.t.	with respect to

Overview of Notation

Even though unusual notation is always defined in text on its first occurrence, we give an overview of frequently appearing notation here. Some specialized notation that does not occur frequently is defined in places where it is needed.

Sets

$\mathbb{R}$	set of real numbers
$\mathbb{R}_+$	set of non-negative real numbers
$\mathbb{R}_{++}$	set of positive real numbers
$\mathbb{R}_{+\infty}$	set of real numbers extended by $+\infty$, i.e., $\mathbb{R}_{+\infty} = \mathbb{R} \cup \{+\infty\}$
$\mathbb{R}_{-\infty}$	set of real numbers extended by $-\infty$, i.e., $\mathbb{R}_{-\infty} = \mathbb{R} \cup \{-\infty\}$
$\mathbb{Z}$	set of integers
$\mathbb{N}$	set of positive integers
$\mathbb{N}_0$	set of non-negative integers
$[n]$	$\{1, \ldots, n\}$, i.e., set of positive integers lower than or equal to $n \in \mathbb{N}$
2^S	power set of set S, i.e., set of all subsets of S (including S and $\emptyset$)
S^n	n-ary Cartesian power of set S (for $n \in \mathbb{N}$)
$S^{m \times n}$	set of all matrices with m rows and n columns whose elements are taken from set S (e.g., $\mathbb{R}^{m \times n}$ is the set of all real matrices with m rows and n columns)
$\operatorname{ri} S$	relative interior of a convex set S
$\operatorname{span} S$	linear hull of set S
$\operatorname{aff} S$	affine hull of set S
$\operatorname{conv} S$	convex hull of set S
$\operatorname{cone} S$	conic hull of set S
A^B	set of all mappings $B \to A$ (for $a \in A^B$ and $b \in B$, we also denote $a(b)$ by a_b)
$A - B$	set difference, $A - B = \{a \in A \mid a \notin B\}$
$A \subseteq B$	set inclusion, i.e., $A \subseteq B$ if $A \cap B = A$
$A \subsetneq B$	strict (a.k.a. proper) set inclusion, i.e., $A \subsetneq B$ if $A \subseteq B$ and $A \neq B$

The notation for n-ary Cartesian power of a set S (denoted by S^n) is a simplification of $S^{[n]}$ (where $n \in \mathbb{N}$).

For $x, y \in \mathbb{R}^n$, $[x, y]$ denotes the line segment $\{\alpha x + (1-\alpha)y \mid 0 \leq \alpha \leq 1\}$. In particular, for $n = 1$, this simplifies to the real interval $[x, y] = \{\alpha \in \mathbb{R} \mid x \leq \alpha \leq y\}$

Vectors, Matrices, and Matrix/Dot Product

A^i	i-th row of matrix A
A_j	j-th column of matrix A
A_{ij}	element of matrix A in i-th row and j-th column
$A^\top$	transpose of matrix A (also applicable to vectors)
$x^\top y$	dot product of two column vectors $x, y \in \mathbb{R}^n$, i.e., $x^\top y = \sum_{i=1}^n x_i y_i$ (or, more generally, $x^\top y = \sum_{i \in S} x_i y_i$ for $x, y \in \mathbb{R}^S$)

All vectors are considered to be column vectors, so $x^\top y$ is in fact the matrix product of a row vector $x^\top$ and a column vector y. This is in analogy to matrix products $A^i x$, $A_j^\top y$, or Ax (for $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$, $i \in [m]$, and $j \in [n]$). As we never require matrix power, there is no ambiguity whenever writing A^i.

Let A and B be sets. For a vector $x \in A^B$ and $B' \subseteq B$, $x|_{B'}$ denotes the subvector of x containing only indices from B'. Analogously, if $x \in A^B$ is seen as a mapping $x\colon B \to A$, then $x|_{B'}$ is its restriction to set B'. In case that B' is a singleton set, i.e., $B' = \{b\}$ for some $b \in B$, we simplify $x|_{\{b\}}$ to x_b. As an example, for $x \in \mathbb{R}^4$, we have $x = (x_1, x_2, x_3, x_4)$ and $x|_{\{2,3\}} = (x_2, x_3)$. This notation also extends to (outputs of) functions, i.e., for $f\colon D \to A^B$, $d \in D$, $B' \subseteq B$, and $b \in B$, we have $f(d)|_{B'} \in A^{B'}$ and $f(d)_b \in A$.

Sequences

A sequence of sets is indexed by a subscript (e.g., $S_1, S_2, S_3, \ldots$) to avoid ambiguity with the n-ary Cartesian power of a set (which is denoted by S^n). On the other hand, sequences of vectors (or tuples) are indexed by a superscript (e.g., $b^1, b^2, b^3, \ldots$) because their components are obtained by a subscript (e.g., $b^1 = (b^1_1, b^1_2, b^1_3) \in \mathbb{R}^3$). Sequences of matrices are not required.

Functions

$f \circ g$	composition of function f with g as $(f \circ g)\colon x \mapsto f(g(x))$
$[\![\psi]\!]$	Iverson bracket: $[\![\psi]\!] = 1$ if ψ is true and $[\![\psi]\!] = 0$ if ψ is false
$\vee$	join operation in §1.3 and Example 1.11, logical disjunction in other sections
$\wedge$	meet operation in §1.3, logical conjunction in other sections

Throughout the thesis, $\operatorname{argmin}_{x \in X} f(x) = \{\, x \in X \mid f(x) = \min_{x' \in X} f(x') \,\}$ denotes the set of minimizers of function f over set X. The case with argmax is analogous.

Partially Ordered Sets

$\preceq$	a partial order
$Q^\uparrow_S$	set of all upper bounds on Q in S
$Q^\downarrow_S$	set of all lower bounds on Q in S
$\bigwedge_S Q$	greatest lower bound on Q in S (we simplify $\bigwedge_S \{q_1, q_2\}$ to $q_1 \wedge_S q_2$)
$\bigvee_S Q$	least upper bound on Q in S (we simplify $\bigvee_S \{q_1, q_2\}$ to $q_1 \vee_S q_2$)
$\top$	top element
$\bot$	bottom element
$\operatorname{im} f$	image of mapping f (see (1.22))

CSP and Weighted CSP

V	finite set of variables
D	finite domain of each variable
C	non-empty set of non-empty scopes of constraints, $C \subseteq 2^V$, $\emptyset \notin C$
$C_{\geq 2}$	set of non-unary scopes, i.e., $C_{\geq 2} = \{S \in C \mid \lvert S\rvert \geq 2\}$
T	set of all tuples, partitioned into T_S (see (1.23) and (1.24))
$A, A', \ldots$	CSP instances, i.e., sets of allowed tuples

$\mathrm{SOL}(A)$	solution set of CSP A
$\mathscr{C}_\Phi$	(dual) closure operator associated with Φ-consistency (see (1.27))
$d, f, g, \ldots$	WCSP instances, i.e., vectors with their weights
$F(x \mid f)$	objective value of WCSP f for assignment x
$B(f)$	upper bound on the optimal value of WCSP f (see (1.39))
$A^*(f)$	set of active tuples for WCSP f (also see (2.10) for $A^*_\epsilon(f)$)
M^*	set of WCSPs $f \in \mathbb{R}^T$ such that $F(x \mid f) \geq 0$ for all assignments $x \in D^V$
$M^\perp$	set of WCSPs $f \in \mathbb{R}^T$ such that $F(x \mid f) = 0$ for all assignments $x \in D^V$
$\mathrm{OPT}(f)$	set of all optimal assignments for WCSP f (see (3.23))

We also note that we write $y_S(k)$ and y_t interchangeably for any $y \in \mathbb{R}^T$ and $t = (S, k) \in T$.

Linear Programs

$\sigma(x)$	set of indices of primal constraints active at x (see (1.2a))
$\tau(y)$	set of indices of dual constraints active at y (see (1.2b), also see (2.6) for $\tau_\epsilon(y)$)

SAT and Weighted Max-SAT

V	set of logical variables
C	set of clauses
V_c^+	set of variables that occur in clause c non-negated
V_c^-	set of variables that occur in clause c negated
V_c	set of all variables that occur in clause c
C_i^+	set of clauses where variable i occurs non-negated
C_i^-	set of clauses where variable i occurs negated
x_i^c	value of variable i in clause c (see (2.16))

Let A and B be sets with $B \subseteq A$. For a vector $x \in \mathbb{R}^A$, $x(B)$ denotes $\sum_{i \in B} x_i$. This special notation is used only in the context of (Max-)SAT to simplify formulations.

Graphs

An *undirected graph* is a pair (V, E) where V is a finite set of nodes (a.k.a. vertices) and E contains (a subset of) 2-element subsets of V, i.e., for edge $\{i, j\} \in E$, we have $i, j \in V$ and $i \neq j$, which forbids loops. In an undirected graph, $N_i = \{j \in V \mid \{i, j\} \in E\}$ is the set of neighbors of node $i \in V$.

A *directed graph* is a pair (V, E) where V is a finite set of nodes and $E \subseteq V \times V$, i.e., edge $(i, j) \in E$ is oriented from $i \in V$ to $j \in V$. We generally forbid loops in directed graphs too, i.e., $(i, i) \notin E$ for all $i \in V$. The set of successors of node $i \in V$ is $N_i^+ = \{j \in V \mid (i, j) \in E\}$.

An (undirected) *hypergraph* is a pair (V, E) where V is a finite set of nodes and $E \subseteq 2^V$ is a set of hyperedges with $\emptyset \notin E$. We do not use directed hypergraphs.

Text Flow and References

End of example is indicated by $\triangle$, QED symbol is the standard $\square$. Papers/works where the author of this thesis contributed are marked with 'a', e.g., as in [55a]. Whenever referring to, e.g., '§1' (or '§1.1'), we mean 'Chapter 1' (or 'Section 1.1') etc. A group citation, e.g., '[123, §7.4, 103, §6.1, 24]' should be interpreted as '§7.4 in [123], §6.1 in [103], and [24]'.

Bibliography

[1] `https://miat.inrae.fr/toulbar2`.

[2] `https://forgemia.inra.fr/thomas.schiex/cost-function-library`, commit `356bbb85`.

[3] `https://software.cs.uni-koeln.de/spinglass`.

[4] Adler, I. and Monteiro, R. D. "A geometric view of parametric linear programming". In: *Algorithmica* 8.1 (1992), pp. 161–176.

[5] Apt, K. R. "From chaotic iteration to constraint propagation". In: *International Colloquium on Automata, Languages, and Programming*. Springer. 1997, pp. 36–55.

[6] Apt, K. R. "The Rough Guide to Constraint Propagation". In: *Conference on Principles and Practice of Constraint Programming*. Springer, 1999, pp. 1–23.

[7] Astesana, J., Cosserat, L., and Fargier, H. "Constraint-based Vehicle Configuration: A Case Study". In: *2010 22nd IEEE International Conference on Tools with Artificial Intelligence*. Vol. 1. 2010, pp. 68–75.

[8] Bacchus, F., Chen, X., Van Beek, P., and Walsh, T. "Binary vs. non-binary constraints". In: *Artificial Intelligence* 140.1-2 (2002), pp. 1–37.

[9] Bacchus, F., Järvisalo, M., and Martins, R. "MaxSAT Evaluation 2018: New Developments and Detailed Results". In: *Journal on Satisfiability, Boolean Modeling and Computation* 11.1 (2019). Instances available at `https://maxsat-evaluations.github.io/`., pp. 99–131.

[10] Bachem, A. and Grötschel, M. *New aspects of polyhedral theory*. Ed. by B.Korte. Inst. für Ökonometrie und Operations Research, North-Holland Publishing Company, 1982.

[11] Batra, D., Nowozin, S., and Kohli, P. "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*. 2011, pp. 146–154.

[12] Beck, A. "The 2-coordinate descent method for solving double-sided simplex constrained minimization problems". In: *Journal of Optimization Theory and Applications* 162.3 (2014), pp. 892–919.

[13] Benhamou, F. and Granvilliers, L. "Continuous and Interval Constraints". In: *Handbook of Constraint Programming*. Elsevier, 2006. Chap. 16.

[14] Berge, C. *Graphs and hypergraphs*. North-Holland Pub. Co., 1973.

[15] Bertsekas, D. P. "Nonlinear Programming. Athena Scientific". In: *Belmont, MA* (1999).

[16] Bertsimas, D. and Tsitsiklis, J. N. *Introduction to linear optimization*. Vol. 6. Athena Scientific Belmont, MA, 1997.

[17] Bessiere, C. "Constraint Propagation". In: *Handbook of Constraint Programming*. Elsevier, 2006. Chap. 3.

[18] Bessiere, C., Cardon, S., Debruyne, R., and Lecoutre, C. "Efficient algorithms for singleton arc consistency". In: *Constraints* 16.1 (2011), pp. 25–53.

[19] Bessiere, C. and Debruyne, R. "Theoretical analysis of singleton arc consistency". In: *Workshop on Modelling and Solving Problems with Constraints*. 2004, pp. 20–29.

[20] Bessiere, C., Fargier, H., and Lecoutre, C. "Global Inverse Consistency for Interactive Constraint Satisfaction". In: *Principles and Practice of Constraint Programming*. Ed. by Schulte, C. Springer Berlin Heidelberg, 2013, pp. 159–174.

[21] Biere, A., Heule, M., and Maaren, H. van. *Handbook of satisfiability*. Vol. 185. IOS press, 2009.

[22] Blyth, T. *Lattices and Ordered Algebraic Structures*. Universitext. Springer London, 2005. ISBN: 9781852339050.

[23] Boros, E. and Hammer, P. L. "Pseudo-boolean optimization". In: *Discrete applied mathematics* 123.1-3 (2002), pp. 155–225.

[24] Boyd, S. and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

[25] Boykov, Y., Veksler, O., and Zabih, R. "Fast approximate energy minimization via graph cuts". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.11 (2001), pp. 1222–1239. DOI: 10.1109/34.969114.

[26] Brualdi, R. A. *Combinatorial matrix classes*. Vol. 13. Cambridge University Press, 2006.

[27] Bulatov, A. A. "A dichotomy theorem for nonuniform CSPs". In: *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2017, pp. 319–330.

[28] Chekuri, C., Khanna, S., Naor, J. S., and Zosin, L. "Approximation algorithms for the metric labeling problem via a new linear programming formulation". In: *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics. 2001, pp. 109–118.

[29] Cohen, D., Cooper, M., Jeavons, P., and Krokhin, A. "A maximal tractable class of soft constraints". In: *Journal of Artificial Intelligence Research* 22 (2004), pp. 1–22.

[30] Cohen, D. and Jeavons, P. "The complexity of constraint languages". In: *Foundations of Artificial Intelligence*. Vol. 2. Elsevier, 2006, pp. 245–280.

[31] Cohen, D. A. and Jeavons, P. G. "The power of propagation: when GAC is enough". In: *Constraints* 22.1 (2017), pp. 3–23.

[32] Cook, S. A. "The complexity of theorem-proving procedures". In: *Proceedings of the third annual ACM symposium on Theory of computing*. 1971, pp. 151–158.

[33] Cooper, M. C., de Givry, S., Sanchez, M., Schiex, T., Zytnicki, M., and Werner, T. "Soft arc consistency revisited". In: *Artificial Intelligence* 174.7-8 (2010), pp. 449–478.

[34] Cooper, M. C. "Cyclic consistency: a local reduction operation for binary valued constraints". In: *Artificial Intelligence* 155.1-2 (2004), pp. 69–92.

[35] Cooper, M. C. "High-order consistency in valued constraint satisfaction". In: *Constraints* 10.3 (2005), pp. 283–305.

[36] Cooper, M. C. "Minimization of locally defined submodular functions by optimal soft arc consistency". In: *Constraints* 13.4 (2008), pp. 437–458.

[37] Cooper, M. C. "Reduction operations in fuzzy or valued constraint satisfaction". In: *Fuzzy Sets and Systems* 134.3 (2003), pp. 311–342.

[38] Cooper, M. C., de Givry, S., and Schiex, T. "Optimal Soft Arc Consistency." In: *Proceedings of the 20th International Joint Conference on Artifical Intelligence*. Vol. 7. 2007, pp. 68–73.

[39] Cooper, M. C., de Givry, S., and Schiex, T. "Valued Constraint Satisfaction Problems". In: *A Guided Tour of Artificial Intelligence Research*. Springer, 2020, pp. 185–207.

[40] Cooper, M. C., Roquemaurel, M. de, and Régnier, P. "A weighted CSP approach to cost-optimal planning". In: *AI Communications* 24.1 (2011), pp. 1–29.

[41] Creignou, N. "A dichotomy theorem for maximum generalized satisfiability problems". In: *Journal of Computer and System Sciences* 51.3 (1995), pp. 511–522.

[42] Davey, B. A. and Priestley, H. A. *Introduction to lattices and order*. Cambridge university press, 2002.

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

[44] De Loera, J. A., Hemmecke, R., and Köppe, M. *Algebraic and geometric ideas in the theory of discrete optimization*. SIAM, 2012.

[45] Debruyne, R. and Bessiere, C. "Some Practicable Filtering Techniques for the Constraint Satisfaction Problem". In: *Proceedings of IJCAI'97*. 1997, pp. 412–417.

[46] Dechter, R. *Constraint processing*. Morgan Kaufmann, 2003.

[47] Devriendt, J., Gleixner, A., and Nordström, J. "Learn to relax: Integrating 0-1 integer linear programming with pseudo-Boolean conflict-driven search". In: *Constraints* (2021), pp. 1–30.

[48a] Dlask, T. "Minimizing Convex Piecewise-Affine Functions by Local Consistency Techniques". Master's thesis. Czech Technical University in Prague, Faculty of Electrical Engineering, 2018.

[49a] Dlask, T. "On Coordinate-Wise Minimization Applied to General Convex Optimization Problems". In: *Procedia Computer Science* 176 (2020), pp. 1328–1337.

[50a] Dlask, T. "Unit Propagation by Means of Coordinate-Wise Minimization". In: *International Conference on Machine Learning, Optimization, and Data Science*. Springer. 2020.

[51a] Dlask, T. and Werner, T. “A Class of Linear Programs Solvable by Coordinate-Wise Minimization”. In: *Learning and Intelligent Optimization*. Ed. by Kotsireas, I. S. and Pardalos, P. M. Springer. 2020, pp. 52–67.

[52a] Dlask, T. and Werner, T. “Bounding linear programs by constraint propagation: application to Max-SAT”. In: *International Conference on Principles and Practice of Constraint Programming*. Springer. 2020, pp. 177–193.

[53a] Dlask, T. and Werner, T. “Classes of linear programs solvable by coordinate-wise minimization”. In: *Annals of Mathematics and Artificial Intelligence* (2021).

[54a] Dlask, T. and Werner, T. “On relation between constraint propagation and block-coordinate descent in linear programs”. In: *International Conference on Principles and Practice of Constraint Programming*. Springer. 2020, pp. 194–210.

[55a] Dlask, T., Werner, T., and de Givry, S. “Bounds on Weighted CSPs Using Constraint Propagation and Super-Reparametrizations”. In: *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*. Vol. 210. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, 23:1–23:18.

[56a] Dlask, T., Werner, T., and de Givry, S. “Super-Reparametrizations of Weighted CSPs: Properties and Optimization Perspective”. In: *Submitted to Artificial Intelligence* (2021). Preprint available online: arXiv: `2201.02018 [math.OC]`.

[57] Escamocher, G. and O’Sullivan, B. “Pushing the frontier of minimality”. In: *Theoretical Computer Science* 745 (2018), pp. 172–201.

[58] Feder, T. and Vardi, M. Y. “The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory”. In: *SIAM Journal on Computing* 28.1 (1998), pp. 57–104.

[59] Ford, L. R. and Fulkerson, D. R. *Flows in networks*. Vol. 43. Princeton University Press, 1962.

[60] Franc, V., Hlaváč, V., and Navara, M. “Sequential coordinate-wise algorithm for the non-negative least squares problem”. In: *International Conference on Computer Analysis of Images and Patterns*. Springer. 2005, pp. 407–414.

[61] Freuder, E. C. “A sufficient condition for backtrack-free search”. In: *Journal of the ACM (JACM)* 29.1 (1982), pp. 24–32.

[62] Freund, R. M., Roundy, R., and Todd, M. J. “Identifying the set of always-active constraints in a system of linear inequalities by a single linear program”. 1985.

[63] Friedman, J., Hastie, T., Höfling, H., and Tibshirani, R. “Pathwise coordinate optimization”. In: *The annals of applied statistics* 1.2 (2007), pp. 302–332.

[64] Furini, F., Traversi, E., Belotti, P., Frangioni, A., Gleixner, A., Gould, N., Liberti, L., Lodi, A., Misener, R., Mittelmann, H., et al. “QPLIB: a library of quadratic programming instances”. In: *Mathematical Programming Computation* 11.2 (2019), pp. 237–265.

[65] Globerson, A. and Jaakkola, T. S. “Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations”. In: *Advances in Neural Information Processing Systems*. 2008, pp. 553–560.

[66] Goldman, A. J. and Tucker, A. W. "Theory of Linear Programming". In: *Linear Inequalities and Related Systems. (AM-38), Volume 38*. Princeton University Press, 1956, pp. 53–97.

[67] Gottlob, G. "On minimal constraint networks". In: *Artificial Intelligence* 191 (2012), pp. 42–60.

[68] Greenberg, H. J. "Consistency, redundancy, and implied equalities in linear systems". In: *Annals of Mathematics and Artificial Intelligence* 17.1 (1996), pp. 37–83.

[69] Greenberg, H. J. "The use of the optimal partition in a linear programming solution for postoptimal analysis". In: *Operations Research Letters* 15.4 (1994), pp. 179–185.

[70] Grégoire, É., Mazure, B., and Piette, C. "MUST: Provide a finer-grained explanation of unsatisfiability". In: *International Conference on Principles and Practice of Constraint Programming*. Springer. 2007, pp. 317–331.

[71] Grégoire, E., Mazure, B., and Piette, C. "On finding minimally unsatisfiable cores of CSPs". In: *International Journal on Artificial Intelligence Tools* 17.04 (2008), pp. 745–763.

[72] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2020. URL: `https://www.gurobi.com`.

[73] Haller, S., Prakash, M., Hutschenreiter, L., Pietzsch, T., Rother, C., Jug, F., Swoboda, P., and Savchynskyy, B. "A Primal-Dual Solver for Large-Scale Tracking-by-Assignment". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 2539–2549.

[74] Haller, S., Swoboda, P., and Savchynskyy, B. "Exact map-inference by confining combinatorial search with LP relaxation". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[75] Hochbaum, D. S. "Solving integer programs over monotone inequalities in three variables: A framework for half integrality and good approximations". In: *European Journal of Operational Research* 140.2 (2002), pp. 291–321.

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

[77] Hsieh, C.-J., Chang, K.-W., Lin, C.-J., Keerthi, S. S., and Sundararajan, S. "A dual coordinate descent method for large-scale linear SVM". In: *Proceedings of the 25th International Conference on Machine learning*. 2008, pp. 408–415.

[78] Hsieh, C.-J. and Dhillon, I. S. "Fast coordinate descent methods with variable selection for non-negative matrix factorization". In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2011, pp. 1064–1072.

[79] Jahn, J. and Ha, T. X. D. "New order relations in set optimization". In: *Journal of Optimization Theory and Applications* 148.2 (2011), pp. 209–236.

[80] Jansen, B, Roos, C., Terlaky, T, and Vial, J.-P. "Interior-point methodology for linear programming: duality, sensitivity analysis and computational aspects". In: *Optimization in Planning and Operation of Electric Power Systems*. Springer, 1993, pp. 57–123.

[81] Jeavons, P. G. and Cooper, M. C. "Tractable constraints on ordered domains". In: *Artificial Intelligence* 79.2 (1995), pp. 327–339.

[82] Jégou, P. "Decomposition of domains based on the micro-structure of finite constraint-satisfaction problems". In: *AAAI*. Vol. 93. 1993, pp. 731–736.

[83] Kappes, J. H. et al. "A Comparative Study of Modern Inference Techniques for Structured Discrete Energy Minimization Problems". In: *International Journal of Computer Vision* 115.2 (2015), pp. 155–184.

[84] Karp, R. M. "Reducibility among combinatorial problems". In: *Complexity of computer computations*. Springer, 1972, pp. 85–103.

[85] Khanna, S. and Sudan, M. "The optimization complexity of constraint satisfaction problems". In: *Electonic Colloquium on Computational Complexity*. Citeseer. 1996.

[86] Kleinberg, J. and Tardos, E. "Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields". In: *Journal of the ACM (JACM)* 49.5 (2002), pp. 616–639.

[87] Kolmogorov, V. "A new look at reweighted message passing". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.5 (2014), pp. 919–930.

[88] Kolmogorov, V. "Convergent tree-reweighted message passing for energy minimization". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.10 (2006), pp. 1568–1583.

[89] Kolmogorov, V., Krokhin, A., and Rolínek, M. "The complexity of general-valued CSPs". In: *SIAM Journal on Computing* 46.3 (2017), pp. 1087–1110.

[90] Kolmogorov, V., Thapper, J., and Živný, S. "The power of linear programming for general-valued CSPs". In: *SIAM Journal on Computing* 44.1 (2015), pp. 1–36.

[91] Kolmogorov, V. and Wainwright, M. J. "On the Optimality of Tree-Reweighted Max-Product Message-Passing". In: UAI'05. Edinburgh, Scotland: AUAI Press, 2005, 316–323. ISBN: 0974903914.

[92] Komodakis, N. and Paragios, N. "Beyond loose LP-relaxations: Optimizing MRFs by repairing cycles". In: *European Conference on Computer Vision*. Springer. 2008, pp. 806–820.

[93] Komodakis, N., Paragios, N., and Tziritas, G. "MRF energy minimization and beyond via dual decomposition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.3 (2010), pp. 531–552.

[94] Koster, A. M., Hoesel, S. P. van, and Kolen, A. W. "The partial constraint satisfaction problem: Facets and lifting theorems". In: *Operations research letters* 23.3-5 (1998), pp. 89–97.

[95] Koval, V. K. and Schlesinger, M. I. "Dvumernoe programmirovanie v zadachakh analiza izobrazheniy (Two-dimensional Programming in Image Analysis Problems)". In: *Automatics and Telemechanics* 8 (1976). In Russian, pp. 149–168.

[96] Kovalevsky, V. and Koval, V. "A diffusion algorithm for decreasing energy of max-sum labeling problem". In: *Glushkov Institute of Cybernetics, Kiev, USSR* (1975). Unpublished.

[97] Lange, J.-H. and Swoboda, P. "Efficient Message Passing for 0–1 ILPs with Binary Decision Diagrams". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 6000–6010.

[98] Larrosa, J. and Schiex, T. "In the quest of the best form of local consistency for weighted CSP". In: *International Joint Conference on Artificial Intelligence*. Vol. 3. 2003, pp. 239–244.

[99] Lecoutre, C. *Constraint Networks: Techniques and algorithms*. ISTE, 2009. ISBN: 978-1-84821-106-3.

[100] Lemaréchal, C. and Hiriart-Urruty, J.-B. *Fundamentals of Convex Analysis*. Springer Grundlehren Text Editions, Springer Verlag, New York, 2004.

[101] Mackworth, A. K. "Consistency in networks of relations". In: *Artificial Intelligence* 8.1 (1977), pp. 99–118.

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

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

[104] Mehrotra, S. and Ye, Y. "Finding an interior point in the optimal face of linear programs". In: *Mathematical Programming* 62.1 (1993), pp. 497–515.

[105] Montanari, U. "Networks of constraints: Fundamental properties and applications to picture processing". In: *Information sciences* 7 (1974), pp. 95–132.

[106] Nation, J. B. *Notes on lattice theory*. 1998.

[107] Nguyen, H., Bessiere, C., de Givry, S., and Schiex, T. "Triangle-based consistencies for cost function networks". In: *Constraints* 22.2 (2017), pp. 230–264.

[108] Nguyen, H., Schiex, T., and Bessiere, C. "Dynamic virtual arc consistency". In: *The 28th Annual ACM Symposium on Applied Computing*. 2013, pp. 98–103.

[109] Niedermeier, R. and Rossmanith, P. "New upper bounds for maximum satisfiability". In: *Journal of Algorithms* 36.1 (2000), pp. 63–88.

[110] Papadimitriou, C. H. and Steiglitz, K. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[111] Peng, J., Hazan, T., McAllester, D., and Urtasun, R. "Convex max-product algorithms for continuous MRFs with applications to protein folding". In: *Proceedings of the 28th International Conference on Machine Learning*. 2011.

[112] Platt, J. *Sequential minimal optimization: A fast algorithm for training support vector machines*. Tech. rep. MSR-TR-98-14. 1998.

[113] Průša, D. and Werner, T. "LP relaxation of the Potts labeling problem is as hard as any linear program". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.7 (2016), pp. 1469–1475.

[114] Průša, D. and Werner, T. "Solving LP Relaxations of Some NP-Hard Problems Is As Hard As Solving Any Linear Program". In: *SIAM Journal on Optimization* 29.3 (2019), pp. 1745–1771.

[115] Průša, D. and Werner, T. "Universality of the Local Marginal Polytope". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.4 (Apr. 2015), pp. 898–904.

[116] Rosen, K. H. and Michaels, J. G. *Handbook of Discrete and Combinatorial Mathematics*. Boca Raton, FL: CRC Press, 1232 p., 2000.

[117] Rother, C., Kolmogorov, V., Lempitsky, V., and Szummer, M. "Optimizing binary MRFs via extended roof duality". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2007, pp. 1–8.

[118] Sanghavi, S., Shah, D., and Willsky, A. S. "Message passing for maximum weight independent set". In: *IEEE Transactions on Information Theory* 55.11 (2009), pp. 4822–4834.

[119] Savchynskyy, B. "Discrete Graphical Models – An Optimization Perspective". In: *Foundations and Trends in Computer Graphics and Vision* 11.3-4 (2019), pp. 160–429. ISSN: 1572-2740.

[120] Schlesinger, D. and Flach, B. *Transforming an arbitrary minsum problem into a binary one*. TU, Fak. Informatik, 2006.

[121] Schlesinger, M. I. "Sintaksicheskiy analiz dvumernykh zritelnikh signalov v usloviyakh pomekh (Syntactic analysis of two-dimensional visual signals in noisy conditions)". In: *Kibernetika* 4.113-130 (1976).

[122] Schrijver, A. *Combinatorial optimization: polyhedra and efficiency*. Springer Science & Business Media, 2004. ISBN: 3-540-20456-3.

[123] Schrijver, A. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[124] Sherali, H. D. and Adams, W. P. "A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems". In: *SIAM Journal on Discrete Mathematics* 3.3 (1990), pp. 411–430.

[125] Sontag, D. and Jaakkola, T. "Tree block coordinate descent for MAP in graphical models". In: *Artificial Intelligence and Statistics*. 2009, pp. 544–551.

[126] Sontag, D., Li, Y., et al. "Efficiently searching for frustrated cycles in MAP inference". In: *28th Conference on Uncertainty in Artificial Intelligence, UAI 2012*. 2012, pp. 795–804.

[127] Sontag, D., Meltzer, T., Globerson, A., Jaakkola, T., and Weiss, Y. *Tightening LP Relaxations for MAP using Message Passing*. 2008.

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

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

[130] Swoboda, P., Rother, C., Abu Alhaija, H., Kainmuller, D., and Savchynskyy, B. "A study of lagrangean decompositions and dual ascent solvers for graph matching". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1607–1616.

[131] Telgen, J. "Identifying redundant constraints and implicit equalities in systems of linear constraints". In: *Management Science* 29.10 (1983), pp. 1209–1222.

[132] Thapper, J. and Živný, S. "The complexity of finite-valued CSPs". In: *Journal of the ACM (JACM)* 63.4 (2016), pp. 1–33.

[133] Thapper, J. and Živný, S. "The power of linear programming for valued CSPs". In: *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*. IEEE. 2012, pp. 669–678.

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

[135] Tourani, S., Shekhovtsov, A., Rother, C., and Savchynskyy, B. "Taxonomy of Dual Block-Coordinate Ascent Methods for Discrete Energy Minimization". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 2775–2785.

[136] Trösser, F., de Givry, S., and Katsirelos, G. "Relaxation-Aware Heuristics for Exact Optimization in Graphical Models". In: *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Ed. by Hebrard, E. and Musliu, N. Cham: Springer International Publishing, 2020, pp. 475–491.

[137] Tseng, P. "Convergence of a block coordinate descent method for nondifferentiable minimization". In: *Journal of Optimization Theory and Applications* 109.3 (2001), pp. 475–494.

[138] Vazirani, V. V. *Approximation Algorithms*. Springer-Verlag New York, 2001. ISBN: 3-540-65367-8.

[139] Wainwright, M., Jaakkola, T., and Willsky, A. "MAP estimation via agreement on (hyper) trees: Message-passing and linear programming approaches". In: *Proceedings of the annual Allerton conference on communication control and computing*. 2002, pp. 1565–1575.

[140] Wainwright, M. J. and Jordan, M. I. "Graphical Models, Exponential Families, and Variational Inference". In: *Foundations and Trends in Machine Learning* 1.1-2 (2008), pp. 1–305.

[141] Walsh, T. "SAT v CSP". In: *International Conference on Principles and Practice of Constraint Programming*. Springer. 2000, pp. 441–456.

[142] Wang, H. and Daphne, K. "Subproblem-tree calibration: A unified approach to max-product message passing". In: *International Conference on Machine Learning*. PMLR. 2013, pp. 190–198.

[143] Wang, P.-W., Chang, W.-C., and Kolter, J. Z. "The Mixing method: low-rank coordinate descent for semidefinite programming with diagonal constraints". In: *ArXiv.org* (2017). arXiv: 1706.00476.

[144] Wang, P.-W. and Kolter, J. Z. "Low-rank semidefinite programming for the MAX2SAT problem". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 1641–1649.

[145] Werner, T. *A Linear Programming Approach to Max-sum Problem: A Review*. Tech. rep. CTU-CMP-2005-25. Center for Machine Perception, Czech Technical University, Dec. 2005.

[146] Werner, T. "A Linear Programming Approach to Max-sum Problem: A Review". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.7 (July 2007), pp. 1165–1179.

[147] Werner, T. "Marginal Consistency: Upper-Bounding Partition Functions over Commutative Semirings". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.7 (July 2015), pp. 1455–1468.

[148] Werner, T. *On Coordinate Minimization of Piecewise-Affine Functions*. Tech. rep. CTU-CMP-2017-05. Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Sept. 2017.

[149] Werner, T. "Revisiting the Linear Programming Relaxation Approach to Gibbs Energy Minimization and Weighted Constraint Satisfaction". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.8 (Aug. 2010), pp. 1474–1488.

[150] Werner, T. and Průša, D. "Relative Interior Rule in Block-Coordinate Minimization". In: *ArXiv.org* (2019). arXiv: `1910.09488 [math.OC]`.

[151a] Werner, T., Průša, D., and Dlask, T. "Relative Interior Rule in Block-Coordinate Descent". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 7559–7567.

[152] Wu, T., Lange, K., et al. "Coordinate descent algorithms for lasso penalized regression". In: *The Annals of Applied Statistics* 2.1 (2008), pp. 224–244.

[153] Xing, Z. and Zhang, W. "MaxSolver: An efficient exact algorithm for (weighted) maximum satisfiability". In: *Artificial Intelligence* 164.1-2 (2005), pp. 47–80.

[154] Yanover, C., Meltzer, T., Weiss, Y., Bennett, K. P., and Parrado-Hernández, E. "Linear Programming Relaxations and Belief Propagation–An Empirical Study." In: *Journal of Machine Learning Research* 7.9 (2006).

[155] Zadeh, N. "A Note on the Cyclic Coordinate Ascent Method". In: *Management Science* 16.9 (1970), pp. 642–644.

[156] Zalinescu, C. *Convex Analysis in General Vector Spaces*. World Scientific, 2002.

[157] Zhang, H. and Stickely, M. E. "An Efficient Algorithm for Unit Propagation". In: *Proc. of AI-MATH* 96 (1996).

[158] Zhang, S. "On the strictly complementary slackness relation in linear programming". In: *Advances in Optimization and Approximation*. Springer, 1994, pp. 347–361.

[159] Zhuk, D. "A proof of the CSP dichotomy conjecture". In: *Journal of the ACM (JACM)* 67.5 (2020), pp. 1–78.

[160] Ziegler, G. M. *Lectures on Polytopes.* Springer-Verlag, New York, 1994. ISBN: 038794365X.

[161] Živný, S. and Cooper, M. C. "The power of arc consistency for CSPs defined by partially-ordered forbidden patterns". In: *Logical Methods in Computer Science* 13 (2017).