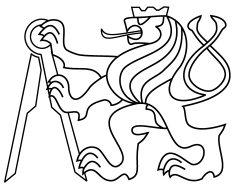
CENTER FOR
MACHINE PERCEPTION



CZECH TECHNICAL
UNIVERSITY IN PRAGUE

RESEARCH REPORT

ISSN 1213-2365

A Linear Programming Approach to Max-sum Problem: A Review

Tomáš Werner

werner@cmp.felk.cvut.cz

CTU–CMP–2005–25

December 2005

This work was supported by the the European Union, grant IST-2004-71567 COSPAL. However, this paper does not necessarily represent the opinion of the European Community, and the European Community is not responsible for any use which may be made of its contents.

Research Reports of CMP, Czech Technical University in Prague, No. 25, 2005

Published by

Center for Machine Perception, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University
Technická 2, 166 27 Prague 6, Czech Republic
fax +420 2 2435 7385, phone +420 2 2435 7637, www: http://cmp.felk.cvut.cz

A Linear Programming Approach to Max-sum Problem: A Review

Tomáš Werner

December 2005

Abstract

The max-sum labeling problem, defined as maximizing a sum of functions of pairs of discrete variables, is a general optimization problem with numerous applications, e.g., computing MAP assignments of a Markov random field. We review a not widely known approach to the problem based on linear programming relaxation, developed by Schlesinger et al. in 1976. We also show how this old approach contributes to more recent results, most importantly by Wainwright et al. In particular, we review Schlesinger's upper bound on the max-sum criterion, its minimization by equivalent transformations, its relation to constraint satisfaction problem, how it can be understood as a linear programming relaxation, and three kinds of consistency necessary for optimality of the upper bound. As special cases, we revisit problems with two labels and supermodular problems. We describe two algorithms for decreasing the upper bound. We present an example application to structural image analysis.

Keywords: structural pattern recognition, Markov random fields, linear programming, computer vision, constraint satisfaction, belief propagation, max-sum, max-product, min-sum, min-product, supermodular optimization.

Contents

1 Introduction

The max-sum (labeling) problem of the second order is defined as maximizing a sum of bivariate functions of discrete variables. It is a general optimization problem, with applications e.g. in computer vision, pattern recognition, machine learning, artificial intelligence, and statistical physics. It has been studied in several contexts using different terminologies. One interpretation is finding a configuration of a Gibbs distribution with maximal probability, which is equivalent to a maximum posterior (MAP) configuration of a Markov random field (MRF) with discrete variables. The problem has also been addressed as a generalization of the constraint satisfaction problem (CSP). For binary variables, it is part of boolean quadratic and pseudo-boolean optimization. The variables have been alternatively called sites, locations or objects, and their values states or labels. The max-sum problem is NP-hard, well-known algorithms for some tractable subclasses being dynamic programming on trees and network max-flow/min-cut.

In this report, we review a not widely known approach by Schlesinger *et al.* [Sch76b,KS76,Sch89, SF00, Sch76a, KK75, Fla98, SK78, Sch05b] to the max-sum problem in a unified and self-contained framework and show how it contributes to recent knowledge.

1.1 Approach by Schlesinger et al.

In 1976, Schlesinger [Sch76b] generalized locally conjunctive predicates considered by Minsky and Papert [MP88] to *two-dimensional (2D) grammars*. Two tasks were suggested. The first task considers analysis of ideal, noise-free images: test whether an input image belongs to the language generated by a given grammar. It leads to what is today known as the constraint satisfaction problem (CSP). Finding the largest *arc consistent* subproblem provides some necessary but not sufficient conditions for satisfiability and unsatisfiability of the problem. The second task is meant for analysis of real, noisy images: find an image belonging to the language generated by a grammar that is 'nearest' to a given image. It leads to the max-sum problem.

The paper [Sch76b] further formulates an LP relaxation of the max-sum problem and its Lagrangian dual. The dual can be interpreted as minimizing an upper bound to the max-sum problem by *equivalent transformations*, which are re-definitions of the the problem that leave the objective function unchanged. Physical models of the LP dual pair were proposed by Schlesinger and Kovalevsky [SK78]. The optimality of the upper bound is equal to *triviality* of the problem. Testing for triviality leads to a CSP. An algorithm to decrease the upper bound is suggested in [Sch76b] and presented in more detail by Koval and Schlesinger [KS76] and further in [KSK77]. Schlesinger notices [Sch76a] that the termination criterion of the algorithm, arc consistency, is necessary but not sufficient for minimality of the upper bound.

Another algorithm to decrease the upper bound is the *max-sum diffusion*, discovered by Kovalevsky and Koval [KK75] and later independently by Flach [Fla98]. It faces the same problem with spurious minima as the algorithm in [KS76].

The material in [Sch76b,KS76] is presented in detail as part of the book [Sch89]. The name '2D grammars' was later assigned a different meaning in the book [SH02] by Schlesinger and Hlaváč. In their original meaning, they largely coincide with MRF.

Schlesinger and Flach [SF00] consider the *labeling problem on a commutative semiring*. Here, the variables are called *objects* and the their values *labels*. Particular problems are obtained by choosing different commutative semirings, yielding the *or-and* (CSP), *max-sum*, *min-max*, and *sum-prod* problems. This algebraic generalization was given also by others [BMR97, AM00]. The paper [SF00] shows that the or-and and min-max problems are tractable if the bivariate functions satisfy the *interval condition*. Further, it shows that if the bivariate functions satisfy the properties of *supermodularity* (called monotonicity in [SF00]), then the problem is tractable and its LP relaxation given in [Sch76b] is tight. Interval or-and and min-max and supermodular max-sum problems are further discussed in [FS00, Fla02].

1.2 Constraint Satisfaction Problem

The CSP [Mon74, Kum92] is ubiquitous in AI and operations research. It is known also under different, less frequent names, e.g., the *consistent labeling problem* [RHZ76, HS79] or [Wal72].

The max-sum problem can be viewed as a natural generalization of CSP. Koster *et al.* [KvHK98, Kos99] consider such a generalization, the *partial CSP*. They formulate the max-sum problem as a 0-1 linear programming and consider its relaxation. They give two classes of non-trivial facets of the resulting *partial CSP polytope*, i.e., linear constraints missing in the LP relaxation.

1.3 Approaches Inspired by Belief Propagation

The max-sum problem has been intensively studied in pattern recognition as computing MAP assignments of Markov random fields (MRF), i.e., maxima of a Gibbs distribution. Unlike in CSP and operations research, where the typical task is to solve small or middle size instances to optimality, here the emphasis is rather on large sparse instances, for which even suboptimal solutions are useful in applications due to noise and data redundancy.

For graphs without cycles (trees), the max-sum problem, as well as the related sum-prod problem equivalent to computing marginals of a Gibbs distribution, can be efficiently solved by *message passing* [Pea88], also known as belief propagation. When applied to cyclic graphs, these algorithms were empirically found sometimes to converge (with the fixed points being useful approximations) and sometimes not to. There is a large literature on belief propagation and a lot of work has been done to understand the fixed points and convergence. See, e.g., the introduction to [Yed04].

Recently, Wainwright *et al.* [WJW02, WJW05] show that a convex combination of max-sum problems provides an upper bound on the original problem. These problems can be conveniently chosen as (tractable) tree problems. Then, the upper bound is tight in case of *tree agreement* (analogous to Schlesinger's triviality), i.e., if the optima on individual trees share a common configuration. Minimizing the upper bound is a convex task, which turns out to be a Lagrangian dual to an LP relaxation of the original problem. Besides directly solving this dual, *tree-reweighted message passing* (TRW) algorithm is suggested to minimize the upper bound. Importantly, it is noted [WJW03a, WJW04] that message passing can be alternatively viewed as *reparameterizations* (synonymous to equivalent transformations) of the problem. TRW and tree combination were considered in a broader view including other inference problems on graphical models [WJW03b, WJ03a, WJ03b].

TRW need neither converge nor decrease the max-sum upper bound monotonically. Kolmogorov [Kol04, Kol05a, Kol05b] suggests its sequential modification (TRW-S) and conjectures that it always converges to a fixed point characterized by a *weak tree agreement* (analogous to arc consistency). He further shows [Kol04, Kol05a] that for variables with more than two states, this fixed point might differ from a global minimum of the upper bound.

1.4 Supermodular Max-sum Problems and Max-flow

(Super-) submodularity is well-known to simplify many optimization tasks and can be considered a discrete counterpart of convexity [Lov83]. For bivariate functions it is also called the *(inverse) Monge property* [BKR96]. Topkis [Top78] explores minimizing a submodular function on a general lattice, giving many useful theorems. The invention of the ellipsoid algorithm allowed for minimization of a set submodular (i.e., with binary variables) function in polynomial time [GLS81, GLS88] and even strongly polynomial algorithms have been recently discovered for submodular functions on distributive lattices by Schrijver [Sch00] and Iwata *et al.* [IFF01].

The objective function of a supermodular max-sum problem is a special case of a supermodular function on a product of chains, which is a special case of a supermodular function on a distributive lattice. Thus, more efficient algorithms can be designed for supermodular max-sum problems than for functions on distributive lattices. It is long known that set supermodular max-sum problems can be translated to max-flow [Ham65, BH02]. Some authors suggested this independently, e.g.

Kolmogorov and Zabih [KZ02]. Others showed translation to max-flow for other subclasses of the supermodular max-sum problem: Greig *et al.* [GPS89]; Ishikawa and Geiger [IG98, Ish03] for the bivariate functions being convex univariate functions of differences of variable pairs; Cohen *et al.* [CCJK04] for Max-CSP problems. D. Schlesinger and Flach [Fla02, Sch05a] gave the translation for the full class. TRW was shown optimal for set supermodular problems by Kolmogorov and Wainwright [KW05a, KW05b]. In many works, especially in computer vision, connection with well-known supermodularity was not noticed and the property was given *ad hoc* names.

Kovtun [Kov03, Kov04] uses supermodularity to find a *partially optimal solution*, i.e., the optimal values of a subset of variables, to the (NP-hard) Potts model. Partial optimality corresponds to *strong persistency*, observed by Hammer *et al.* in quadratic 0-1 optimization (see [BH02]).

1.5 Contribution of the Reviewed Work

Schlesinger's LP relaxation of the max-sum problem is the same as that by Koster *et al.* [KvHK98], and as that by Chekuri *et al.* [CKNZ01] and Wainwright *et al.* [WJW02]. The reviewed theory is neither a subset nor a superset of more recent results, and is closest to the works by Wainwright *et al.* and Kolmogorov. In fact, if the trees are chosen to be individual edges, Schlesinger's upper bound can be obtained from a convex combination of trees and CSP corresponds to weak tree agreement. This convenient choice is w.l.o.g. because Wainwright's max-sum bound is independent on the choice of trees, once they cover all edges. However, the translation between the two theories is not straightforward and thus the old theory is an alternative, possibly opening new ways of research. The contributions of work by Schlesinger *et al.* to recent results are as follows.

Duality of minimizing Schlesinger's upper bound and maximizing the max-sum criterion over relaxed labelings is proved more straightforwardly by putting both problems to matrix forms [Sch76b], as is common in LP duality.

By complementary slackness, the max-sum problem is intimately related with CSP, because the test whether a given upper bound is tight leads to a CSP. This makes a link to large CSP literature and reveals that this test is NP-complete, which has not been noticed by others. It also naturally leads to a relaxation of CSP, which provides a simple way [Sch76a] to characterize spurious minima of the upper bound.

It is known [KW05a, KW05b] that the spurious minima do not occur for problems with binary variables. This is proved in an alternative way, additionally showing that in this case there exists a half-integral optimal relaxed labeling [Sch05b].

The max-sum diffusion [KK75, Fla98], an algorithms to decrease the upper bound, is similar to belief propagation, however, it is convergent and monotonic. With its combinatorial flavor, the Koval-Schlesinger algorithm [KS76] is dissimilar to any recent algorithm.

For supermodular max-sum problems, Schlesinger's upper bound is tight and finding an optimal labeling is tractable [SF00]. Extending works [Sch05a, Fla02, Kov03, Kov04, KW05a], we formulate this result and its relation to CSP in lattice-theoretical terms. This has not been done before in this extent.

1.6 Organization of the Report

The sequel is organized as follows. Section 2 defines the labeling problem on semirings. Section 3 introduces the or-and problem and arc consistency. Section 4, central to the article, presents the upper bound to the max-sum problem, its minimization by equivalent transformations, and its relation with the or-and problem. The next section 5 shows that the approach can be understood as an LP relaxation. A hydraulic model of this linear program is presented in appendix D. Section 6 characterizes optimality of max-sum problem using three kinds of consistency. As special cases, section 6.5 discusses problems with two labels and section 9 supermodular problems. Examples of two algorithms for decreasing the upper bound are described in sections 7 and 8. Application of the theory to structural image analysis is presented in section 10. Finally, the approach is summarized and some open problems are outlined.

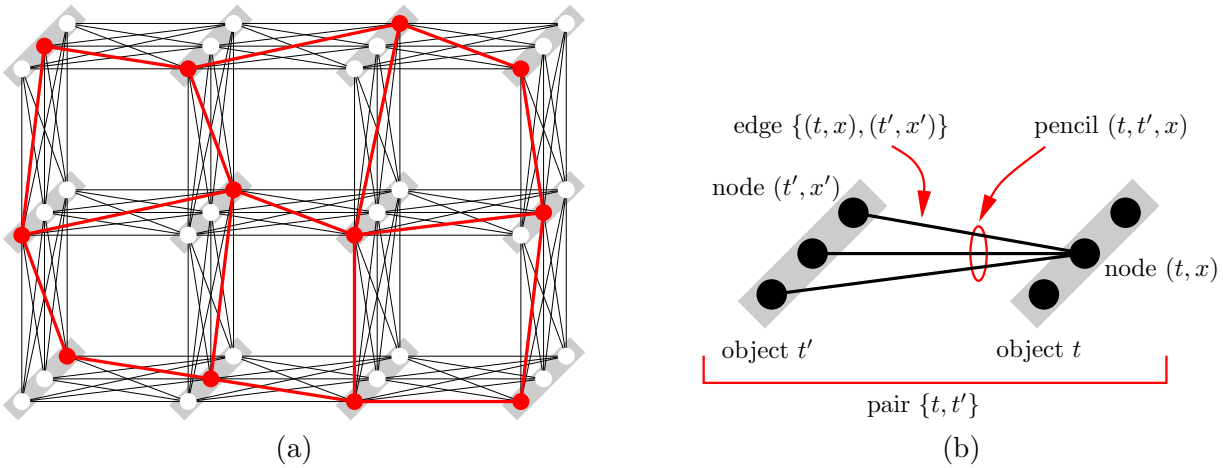


Figure 1: (a) The 3×4 grid graph G, graph G_X for $|X| = 3$ labels, and a labeling $\mathbf{x}$ (emphasized). (b) Parts of G and G_X.

1.7 Mathematical Symbols

In the sequel, $\{x, y\}$ denotes the set with elements x and y, and $\{x \mid \psi(x)\}$ the set of elements x with property $\psi(x)$. The set of all subsets or all n-element subsets of a set X is 2^X or $\binom{X}{n}$, respectively. The set of real numbers is $\mathbb{R}$. An ordered pair is (x, y), while $[x, y]$ denotes a closed interval. Vectors are denoted by boldface letters. Matrix transpose is denoted by $\mathbf{A}^\top$ and scalar product by $\langle \mathbf{x}, \mathbf{y} \rangle$. Logical conjunction (disjunction) is denoted by $\wedge$ ($\vee$). Function δ_ψ returns 1 if logical expression ψ is true and 0 if it is false. Symbol $\operatorname{argmax}_x f(x)$ denotes the set of *all* maximizers of $f(x)$. In algorithms, $x := y$ denotes assignment and $x \mathrel{+}= y$ means $x := x + y$, like in the C programming language.

In expressions like $\max_{x|\psi(x)} f(x)$, notation $x \mid \psi(x)$ is sometimes used as a shorthand for $x \in \{x \mid \psi(x)\}$. Symbols $\sum_t$, $\sum_{\{t,t'\}}$, and $\sum_x$ will abbreviate respectively $\sum_{t \in T}$, $\sum_{\{t,t'\} \in E}$, and $\sum_{x \in X}$, unless specified otherwise. Similarly in max, $\bigwedge$, $\bigotimes$, etc.

2 Labeling Problem on a Commutative Semiring

In the sequel, we will use the 'labeling' terminology from [SF00]. Let $G = (T, E)$ be an undirected graph, where T is a discrete set of **objects** and $E \subseteq \binom{T}{2}$ is a set of (object) **pairs**. The set of neighbors of an object t is $N_t = \{\, t' \mid \{t, t'\} \in E \,\}$. Each object $t \in T$ is assigned a **label** $x_t \in X$, where X is a discrete set. A **labeling** is a mapping that assigns a single label to each object, represented by a $|T|$-tuple $\mathbf{x} \in X^{|T|}$ with components x_t. When not viewed as components of $\mathbf{x}$, elements of X will be denoted by x, x' without any subscript.

Let $G_X = (T \times X, E_X)$ be another undirected graph with the edge set $E_X = \{\, \{(t, x), (t', x')\} \mid \{t, t'\} \in E,\ x, x' \in X \,\}$. To avoid confusion between G and G_X, the nodes and edges of G will be called respectively objects and pairs, whereas the terms **nodes** and **edges** will refer to G_X. The number of nodes and edges of G_X is denoted by $|G_X| = |T||X| + |E||X|^2$. The set of edges leading from a node (t, x) to all nodes of a neighboring object $t' \in N_t$ is called a **pencil** and denoted by (t, t', x). Figure 1 shows how G and G_X, their parts, and how labelings $\mathbf{x}$ on them will be illustrated.

Let an element of a set S be assigned to each node and edge. The element assigned to node (t, x) is denoted by $g_t(x)$. The element assigned to edge $\{(t, x), (t', x')\}$ is denoted by $g_{tt'}(x, x')$, where we adopt that $g_{tt'}(x, x') = g_{t't}(x', x)$. The vector obtained by concatenating all these elements (in some arbitrary but fixed order) is denoted by $\mathbf{g} \in S^{|G_X|}$.

Let the set S endowed with two binary operations $\oplus$ and $\otimes$ form a commutative semiring $(S, \oplus, \otimes)$. The semiring formulation of the **labeling problem** [SF00,AM00] is defined as computing

the expression

$$\bigoplus_{\mathbf{x}\in X^{|T|}} \Big[\bigotimes_{t} g_t(x_t) \otimes \bigotimes_{\{t,t'\}} g_{tt'}(x_t, x'_t) \Big]. \tag{1}$$

More exactly, this is a labeling problem of the *second order (pairwise)*, according to the highest arity of the functions in the brackets. We will not consider problems of higher order; any higher order problem can be translated to a second order one by introducing auxilliary variables. Note that the first term in the brackets can be omitted without loss of generality since any univariate function can be also seen as bivariate, thus the terms $g_t(\bullet)$ can be absorbed in $g_{tt'}(\bullet,\bullet)$. However, mostly it is convenient to keep both terms explicitly.

Interesting and useful labeling problems are provided (modulo isomorphisms) by the following choices of the semiring [AM00, SF00, Gau97]:

$(S, \oplus, \otimes)$	task
$(\{0,1\}, \vee, \wedge)$	CSP
$(\mathbb{R} \cup \{-\infty\}, \min, \max)$	min-max problem
$(\mathbb{R} \cup \{-\infty\}, \max, +)$	max-sum problem
$(\mathbb{R}_{0+}, +, *)$	sum-prod problem

All these problems are NP-hard. The max-sum and sum-prod problems can be stated also as finding the mode and the log-partition function of a Gibbs distribution, respectively. The topic of our report is primarily the max-sum problem, however, we will need also CSP.

3 Constraint Satisfaction Problem

The **constraint satisfaction problem** (CSP) is defined as finding a labeling that satisfies given unary and binary logical constraints, i.e., that passes through some or all of given nodes and edges. A CSP instance is denoted by $(G, X, \bar{\mathbf{g}})$, where the binary indicators $\bar{g}_t(x)$ ($\bar{g}_{tt'}(x,x')$) say whether the corresponding node (edge) is present or absent. The task is to test whether the set

$$\bar{L}_{G,X}(\bar{\mathbf{g}}) = \Big\{ \mathbf{x} \in X^{|T|} \;\Big|\; \bigwedge_t \bar{g}_t(x_t) \wedge \bigwedge_{\{t,t'\}} \bar{g}_{tt'}(x_t, x_{t'}) = 1 \Big\} \tag{2}$$

is non-empty, and possibly to find one, several, or all of its elements. An instance is **satisfiable** if $\bar{L}_{G,X}(\bar{\mathbf{g}}) \neq \emptyset$. CSP $(G, X, \bar{\mathbf{g}}')$ is a **subproblem** of $(G, X, \bar{\mathbf{g}})$ if $\bar{\mathbf{g}}' \leq \bar{\mathbf{g}}$. The **union** of CSPs $(G, X, \bar{\mathbf{g}})$ and $(G, X, \bar{\mathbf{g}}')$ is $(G, X, \bar{\mathbf{g}} \vee \bar{\mathbf{g}}')$. Here, the operations $\leq$ and $\vee$ are meant componentwise.

3.1 Arc Consistency and Kernel

CSP $(G, X, \bar{\mathbf{g}})$ is **arc consistent** if

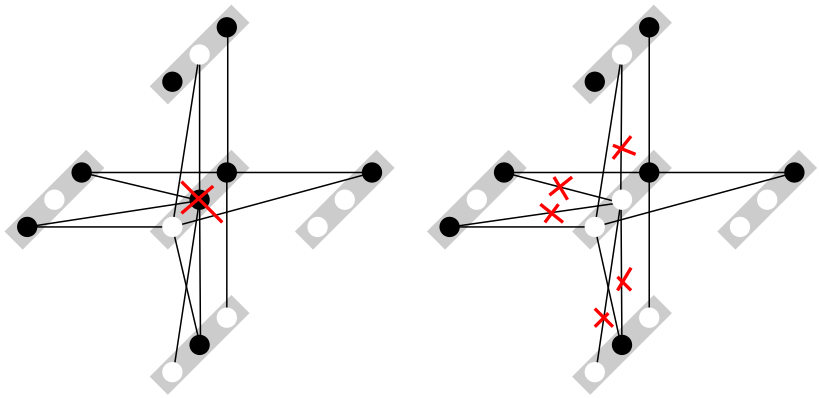
$$\bigvee_{x'} \bar{g}_{tt'}(x, x') = \bar{g}_t(x), \quad \{t, t'\} \in E,\ x \in X. \tag{3}$$

The union of arc consistent problems is arc consistent. To see this, write the disjunction of (3) for arc consistent $\bar{\mathbf{g}}$ and $\bar{\mathbf{g}}'$ as $[\bigvee_{x'} \bar{g}_{tt'}(x,x')] \vee [\bigvee_{x'} \bar{g}'_{tt'}(x,x')] = \bigvee_{x'} [\, \bar{g}_{tt'}(x,x') \vee \bar{g}'_{tt'}(x,x')\,] = \bar{g}_t(x) \vee \bar{g}'_t(x)$, which shows that $\bar{\mathbf{g}} \vee \bar{\mathbf{g}}'$ satisfies (3). Following [Sch89], by the **kernel** of a CSP we call the union of all its arc consistent subproblems. Arc consistent subproblems of a problem form a join semilattice w.r.t. the partial ordering by inclusion $\leq$, whose greatest element is the kernel.

The kernel can be found by an **arc consistency algorithm** [Wal72, Sch76b, HDT92]. We will use its following version. Starting with the original values of $\bar{\mathbf{g}}$, variables $\bar{g}_t(x)$ and $\bar{g}_{tt'}(x,x')$ violating (3) are repeatedly set to zero by applying the rules (see figure 2)

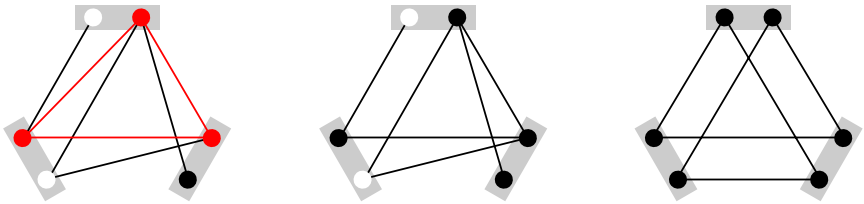
$$\bar{g}_t(x) := \bar{g}_t(x) \wedge \bigvee_{x'} \bar{g}_{tt'}(x, x'), \tag{4a}$$

$$\bar{g}_{tt'}(x, x') := \bar{g}_{tt'}(x, x') \wedge \bar{g}_t(x) \wedge \bar{g}_{t'}(x'). \tag{4b}$$



(a) (b)

Figure 2: The arc consistency algorithm deletes (a) nodes not linked with some neighbor by any edge, and (b) edges lacking an end node.



(a) (b) (c)

Figure 3: Examples of CSPs: (a) satisfiable problem (hence with a non-empty kernel); (b) problem with an empty kernel (hence insatisfiable); (c) arc consistent but insatisfiable problem. The present nodes are in black, the absent nodes in white, the absent edges are not shown.

The algorithm halts when no further change is possible. It is well-known that the result does not depend on the order of the operations.

Let $(G, X, \bar{\mathbf{g}}^*)$ be the kernel of a CSP $(G, X, \bar{\mathbf{g}})$. The crucial property of the kernel is that $\bar{L}_{G,X}(\bar{\mathbf{g}}) = \bar{L}_{G,X}(\bar{\mathbf{g}}^*)$. This is proved later as a special case of theorem 4 but it is easily seen true by the following argument. If a pencil (t, t', x) contains no edge, the node (t, x) clearly cannot belong to any labeling. Therefore, the node (t, x) can be deleted without changing $\bar{L}_{G,X}(\bar{\mathbf{g}})$. Similarly, if a node (t, x) is absent then no labeling can pass through the pencils $\{\, (t, t', x) \mid t' \in N_t \,\}$.

Thus, the local condition of arc consistency allows to give simple sufficient (but not necessary) conditions for unsatisfiability and satisfiability of a CSP (figure 3 shows examples):

- If the kernel is empty (i.e., $\bar{\mathbf{g}}^* = \mathbf{0}$) then the problem $(G, X, \bar{\mathbf{g}})$ is not satisfiable.
- If there is a unique label in each object of the kernel (i.e., $\sum_x \bar{g}^*_t(x) = 1$ for all $t \in T$) then the problem $(G, X, \bar{\mathbf{g}})$ is satisfiable.

4 Max-sum Problem

An instance of the **max-sum problem** is denoted by the triplet $(G, X, \mathbf{g})$, where the elements $g_t(x)$ and $g_{tt'}(x, x')$ of $\mathbf{g}$ are called **qualities**. The **quality of a labeling x** is the number

$$F(\mathbf{x} \,|\, \mathbf{g}) = \sum_t g_t(x_t) + \sum_{\{t,t'\}} g_{tt'}(x_t, x_{t'}). \tag{5}$$

Solving the problem means finding (one, several or all elements of) the set of optimal labelings

$$L_{G,X}(\mathbf{g}) = \operatorname*{argmax}_{\mathbf{x} \in X^{|T|}} F(\mathbf{x} \,|\, \mathbf{g}). \tag{6}$$

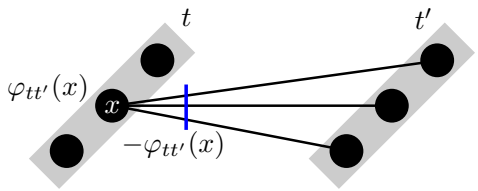


Figure 4: The elementary equivalent transformation: the quality of the node (t, x) increases by a number $\varphi_{tt'}(x)$, the qualities of all edges in the pencil (t, t', x) decrease by the same number $\varphi_{tt'}(x)$.

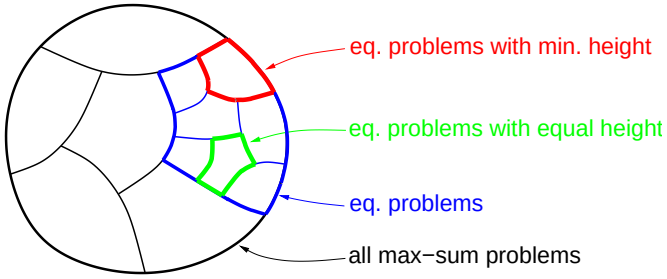


Figure 5: Classes of max-sum problems.

4.1 Equivalent Max-sum Problems

The parameterization of the max-sum problem by vectors $\mathbf{g}$ is not minimal. Problems $(G, X, \mathbf{g})$ and $(G, X, \mathbf{g}')$ are **equivalent** (denoted by $\mathbf{g} \sim \mathbf{g}'$) if the functions $F(\bullet \,|\, \mathbf{g})$ and $F(\bullet \,|\, \mathbf{g}')$ are identical. A change of $\mathbf{g}$ taking a max-sum problem to its equivalent is called an **equivalent transformation** [Sch76b, WJW03a, Kol05a]. An example ER is shown in figure 4: choose a pencil (t, t', x), add any number $\varphi_{tt'}(x)$ to $g_t(x)$ and subtract the same number from the edges $\{\, g_{tt'}(x, x') \mid x' \in X \,\}$.

A particular equivalence class is formed by **zero problems** for which $F(\bullet \,|\, \mathbf{g})$ is a zero function. By (5), the zero class $\{\, \mathbf{g} \mid \mathbf{g} \sim \mathbf{0} \,\}$ is a linear subspace of $\mathbb{R}^{|G_X|}$ and any problems $\mathbf{g}$ and $\mathbf{g}'$ are equivalent if and only if $\mathbf{g} - \mathbf{g}'$ is a zero problem.

We will parameterize equivalence classes by a vector $\boldsymbol{\varphi} \in \mathbb{R}^{2|E||X|}$ with components $\varphi_{tt'}(x)$, assigned to all pencils (t, t', x). Variables $\varphi_{tt'}(x)$ are called *potentials* in [Sch76b, KS76, Sch89] and they correspond to *messages* in belief propagation literature. The equivalent of a problem $\mathbf{g}$ given by $\boldsymbol{\varphi}$ is denoted by $\mathbf{g}^{\boldsymbol{\varphi}} = \mathbf{g} + \mathbf{0}^{\boldsymbol{\varphi}}$. It is obtained by composing the 'elementary' transformations shown in figure 4, which yields

$$g_t^{\boldsymbol{\varphi}}(x) = g_t(x) + \sum_{t' \in N_t} \varphi_{tt'}(x), \tag{7a}$$

$$g_{tt'}^{\boldsymbol{\varphi}}(x, x') = g_{tt'}(x, x') - \varphi_{tt'}(x) - \varphi_{t't}(x'). \tag{7b}$$

It is easy to verify by plugging (7) to (5) that $F(\mathbf{x} \,|\, \mathbf{g}^{\boldsymbol{\varphi}})$ identically equals $F(\mathbf{x} \,|\, \mathbf{g})$. It is more tricky to show (proved in appendix C, see [Sch76b, Kol05a]) that if G is a connected graph, any class is completely covered by the parameterization (7).

Remark. ERs are called *equivalent transformations* in [Sch76b, KS76, Sch89, SF00]. We use *ER* to comply with terminology used by Wainwright *et al.* [WJ03b]. More generally, 'equivalent transformations' can be understood as transformations of *any* labeling problem, possibly leading to a 'trivial' problem that is easy to solve [SF00, Sch05b]. The variables $\varphi_{tt'}(x)$ are analogous to messages from the belief propagation literature. In [Sch76b, KS76], these variables were called *potentials*, in analogy with electrical circuits.

4.2 Upper Bound and Its Minimization

Let the **height of object** t and the **height of pair** $\{t,t'\}$ be respectively

$$u_t = \max_x g_t(x), \quad u_{tt'} = \max_{x,x'} g_{tt'}(x,x'). \tag{8}$$

The **height of a max-sum problem** $(G,X,\mathbf{g})$ is

$$U(\mathbf{g}) = \sum_t u_t + \sum_{\{t,t'\}} u_{tt'}. \tag{9}$$

By comparing corresponding terms in (5) and (9), the problem height is an upper bound of quality, i.e., any problem $(G,X,\mathbf{g})$ and any labeling $\mathbf{x}$ satisfy $F(\mathbf{x}\,|\,\mathbf{g}) \le U(\mathbf{g})$.

Unlike the quality function, the problem height is not invariant to ETs. This naturally leads to minimizing this upper bound by ETs. It leads to the convex non-smooth minimization task

$$U^*(\mathbf{g}) = \min_{\mathbf{g}'\sim\mathbf{g}} U(\mathbf{g}') \tag{10a}$$

$$= \min_{\boldsymbol{\varphi}} \Big[\sum_t \max_x g_t^{\boldsymbol{\varphi}}(x) + \sum_{\{t,t'\}} \max_{x,x'} g_{tt'}^{\boldsymbol{\varphi}}(x,x') \Big]. \tag{10b}$$

Some ETs preserve the problem height, e.g., adding a number to all nodes of an object and subtracting the same number from all nodes of another object. Thus, there are in general many problems with the same height within every equivalence class (see figure 5). This gives an option to impose up to $|T|+|E|-1$ constraints on the numbers u_t and $u_{tt'}$ in the minimization and reformulate (10) in a number of alternative ways. This freedom of formulation is convenient for designing algorithms. Thus,

$$U^*(\mathbf{g}) = \min_{\boldsymbol{\varphi}\,|\,g_{tt'}^{\boldsymbol{\varphi}}(x,x')\le 0} \sum_t \max_x g_t^{\boldsymbol{\varphi}}(x) \tag{11a}$$

$$= \min_{\boldsymbol{\varphi}\,|\,g_t^{\boldsymbol{\varphi}}(x)=0} \sum_{\{t,t'\}} \max_{x,x'} g_{tt'}^{\boldsymbol{\varphi}}(x,x') \tag{11b}$$

$$= \min_{\boldsymbol{\varphi}} \Big[|T| \max_t \max_x g_t^{\boldsymbol{\varphi}}(x) + |E| \max_{\{t,t'\}} \max_{x,x'} g_{tt'}^{\boldsymbol{\varphi}}(x,x') \Big] \tag{11c}$$

$$= |T| \min_{\boldsymbol{\varphi}\,|\,g_{tt'}^{\boldsymbol{\varphi}}(x,x')\le 0} \max_t \max_x g_t^{\boldsymbol{\varphi}}(x) \tag{11d}$$

$$= |E| \min_{\boldsymbol{\varphi}\,|\,g_t^{\boldsymbol{\varphi}}(x)=0} \max_{\{t,t'\}} \max_{x,x'} g_{tt'}^{\boldsymbol{\varphi}}(x,x') \tag{11e}$$

$$= (|T|+|E|) \min_{\boldsymbol{\varphi}} \max \Big\{ \max_t \max_x g_t^{\boldsymbol{\varphi}}(x),\ \max_{\{t,t'\}} \max_{x,x'} g_{tt'}^{\boldsymbol{\varphi}}(x,x') \Big\}. \tag{11f}$$

E.g., form (11a) corresponds to imposing $u_{tt'} \le 0$ and (11d) to $u_{tt'} \le 0$ and $u_t = u_{t'} = u$. Other natural constraints are $u_t = 0$, or $u_t = u_{t'} = u_{tt'} = u$, or fixing all but one of u_t and $u_{tt'}$.

4.3 Trivial Problems

Node (t,x) is a **maximal node** if $g_t(x) = u_t$. Edge $\{(t,x),(t',x')\}$ is a **maximal edge** if $g_{tt'}(x,x') = u_{tt'}$. Let

$$\bar{g}_t(x) = \delta_{g_t(x)=u_t}, \quad \bar{g}_{tt'}(x,x') = \delta_{g_{tt'}(x,x')=u_{tt'}}\ . \tag{12}$$

A max-sum problem is **trivial** if the CSP $(G,X,\bar{\mathbf{g}})$ is satisfiable. It is easy to see that the upper bound is tight, $F(\mathbf{x}\,|\,\mathbf{g}) = U(\mathbf{g})$, for and only for trivial problems, i.e., for $\mathbf{x}$ composed of (some or all) maximal nodes and edges, $\mathbf{x} \in \bar{L}_{G,X}(\bar{\mathbf{g}})$. The following theorem is central to the approach.

Theorem 1 *Let P be a class of all max-sum problems equivalent with a given problem. Let P contain at least one trivial problem. Then a problem in P is trivial if and only if its height is minimal in P.*

Proof. Let $(G, X, \mathbf{g})$ be a trivial problem in P. Let a labeling $\mathbf{x}$ be composed of the maximal nodes and edges of $(G, X, \mathbf{g})$. Any $\mathbf{g}' \sim \mathbf{g}$ satisfies $U(\mathbf{g}') \geq F(\mathbf{x} \,|\, \mathbf{g}') = F(\mathbf{x} \,|\, \mathbf{g}) = U(\mathbf{g})$. Thus $(G, X, \mathbf{g})$ has minimal height.

Let $(G, X, \mathbf{g})$ be a non-trivial problem with minimal height in P. Any $\mathbf{g}' \sim \mathbf{g}$ and any optimal $\mathbf{x}$ satisfy $U(\mathbf{g}') \geq U(\mathbf{g}) > F(\mathbf{x} \,|\, \mathbf{g}) = F(\mathbf{x} \,|\, \mathbf{g}')$. Thus P contains no trivial problem. ■

In other words, theorem 1 says that (i) if a problem in P is trivial then it has minimal height in P; (ii) if P contains a trivial problem then any problem with minimal height in P is trivial; and (iii) if any problem has minimal height in P and is not trivial then P contains no trivial problem. This allows to divide a max-sum problem into two steps:

1. minimize the problem height by ETs,
2. test the resulting problem for triviality.

If $(G, X, \bar{\mathbf{g}})$ is satisfiable then $L_{G,X}(\mathbf{g}) = \bar{L}_{G,X}(\bar{\mathbf{g}})$. If not, the max-sum problem has no trivial equivalent and remains unsolved. Note, however, that even if the max-sum problem has a trivial equivalent, we might fail to recognize it in polynomial time because testing whether a given upper bound on a max-sum problem is tight is NP-complete. Indeed, this task can be easily translated to (NP-complete) CSP and *vice versa*.

Note that if there are two equivalent problems $\mathbf{g}$ and $\mathbf{g}'$ with minimal height then $L_{G,X}(\mathbf{g}) = L_{G,X}(\mathbf{g}')$. In other words, if a node (edge) is maximal in $\mathbf{g}$ and non-maximal in $\mathbf{g}'$, no labeling can pass through this node (edge).

Dividing a max-sum problem into the two steps is convenient also because it allows to encode all optimal labelings in the maximal nodes and edges of any equivalent with minimal height.

Figure 3a shows a trivial problem (thus having minimal height), 3b a problem with a non-minimal height (hence non-trivial), 3c a non-trivial problem with minimal height. The maximal nodes are black, the non-maximal nodes white, the non-maximal edges are not shown.

5 Linear Programming Formulation

This section shows that theorem 1 can be viewed to follow from a linear programming relaxation of the max-sum problem and LP duality. Appendix A surveys what we will need from LP duality.

5.1 Relaxed Labeling

So far, labelings have been represented by tuples $\mathbf{x} \in X^{|T|}$. Each object t had exactly one label, represented by variable $x_t \in X$. In this section, an alternative representation is introduced which allows each object to be 'undecided', i.e., to be assigned multiple labels with different weights.

A **relaxed labeling** is a vector $\boldsymbol{\alpha}$ with the components $\alpha_t(x)$ and $\alpha_{tt'}(x, x')$ (ordered the same way as components $g_t(x)$ and $g_{tt'}(x, x')$ in $\mathbf{g}$) satisfying the constraints

$$\sum_{x'} \alpha_{tt'}(x, x') = \alpha_t(x), \quad \{t, t'\} \in E,\ x \in X \tag{13a}$$

$$\sum_{x} \alpha_t(x) = 1, \qquad t \in T \tag{13b}$$

$$\alpha_{tt'}(x, x') \geq 0, \qquad \{t, t'\} \in E,\ x, x' \in X \tag{13c}$$

where $\alpha_{tt'}(x, x') = \alpha_{t't}(x', x)$. Number $\alpha_t(x)$ is assigned to node (t, x), number $\alpha_{tt'}(x, x')$ to edge $\{(t, x), (t', x')\}$. The set of all $\boldsymbol{\alpha}$ satisfying (13) is a polytope, denoted by $\Lambda_{G,X}$.

A binary $\boldsymbol{\alpha}$ represents a 'decided' labeling, it is just an alternative representation to $\mathbf{x} \in X^{|T|}$. There is a bijection between the sets $X^{|T|}$ and $\Lambda_{G,X} \cap \{0, 1\}^{|G_X|}$, given by $\alpha_t(x) = \delta_{x_t = x}$ and $\alpha_{tt'}(x, x') = \delta_{x_t = x} \delta_{x_{t'} = x'}$. A non-integer $\boldsymbol{\alpha}$ represents an 'undecided' labeling.

The constraint set (13) can be modified in several ways without affecting $\Lambda_{G,X}$. Clearly, one can add constraints $\alpha_t(x) \geq 0$ and $\sum_{x,x'} \alpha_{tt'}(x, x') = 1$. Further, all but one constraint (13b) can

be omitted. To see this, denote $\alpha_t = \sum_x \alpha_x(t)$ and $\alpha_{tt'} = \sum_{x,x'} \alpha_{tt'}(x,x')$ and sum (13a) over x, which gives $\alpha_t = \alpha_{tt'}$. Since G is connected, (13a) alone implies that all α_t and $\alpha_{tt'}$ are equal. Alternatively, (13b) can or replaced with, e.g., $\sum_t \alpha_t = |T|$ or $\sum_{\{t,t'\}} \alpha_{tt'} = |E|$.

Equalities (13a) and (13c) can be viewed as a continuous generalization of the logical arc consistency (3) in the following sense: for any $\boldsymbol{\alpha}$ satisfying them, the CSP $\bar{\mathbf{g}}$ given by $\bar{g}_t(x) = \delta_{\alpha_t(x)>0}$ and $\bar{g}_{tt'}(x,x') = \delta_{\alpha_{tt'}(x,x')>0}$ satisfies (3). Similarly, (13b) is a continuous counterpart of non-emptiness of the kernel.

5.2 LP Relaxation of Max-sum Problem

For the max-sum problem, the concepts of quality and equivalence extend from labelings to relaxed labelings. The quality of a relaxed labeling $\boldsymbol{\alpha}$ is the scalar product

$$\langle \mathbf{g}, \boldsymbol{\alpha} \rangle = \sum_t \sum_x \alpha_t(x)\, g_t(x) + \sum_{\{t,t'\}} \sum_{x,x'} \alpha_{tt'}(x,x')\, g_{tt'}(x,x'). \tag{14}$$

Like $F(\bullet \,|\, \mathbf{g})$, the function $\langle \mathbf{g}, \bullet \rangle$ is invariant to equivalent transformations. Substituting (7) and (13a) verifies that $\langle \mathbf{0}^{\boldsymbol{\varphi}}, \boldsymbol{\alpha} \rangle$ identically vanishes.

The **relaxed max-sum problem** is the linear program

$$\Lambda_{G,X}(\mathbf{g}) = \operatorname*{argmax}_{\alpha \in \Lambda_{G,X}} \langle \mathbf{g}, \boldsymbol{\alpha} \rangle. \tag{15}$$

The set $\Lambda_{G,X}(\mathbf{g})$ is a polytope, being the convex hull of the optimal vertices of $\Lambda_{G,X}$. If $\Lambda_{G,X}(\mathbf{g})$ has integer elements, they coincide with solutions $L_{G,X}(\mathbf{g})$ of the non-relaxed max-sum problem. If not, the problem has no trivial equivalent.

By making u_t and $u_{tt'}$ free variables, the minimization problem (10) can be posed as a linear programming. This program turns out to be dual to (15). To show this, we will write this pair of dual programs together in the same form as the LP pair (25), putting a constraint and its Lagrange multiplier always on the same line.

$$\langle \mathbf{g}, \boldsymbol{\alpha} \rangle \to \max_{\boldsymbol{\alpha}} \qquad \sum_t u_t + \sum_{\{t,t'\}} u_{tt'} \to \min_{\boldsymbol{\varphi}, \mathbf{u}} \tag{16a}$$

$$\sum_{x'} \alpha_{tt'}(x,x') = \alpha_t(x) \qquad \varphi_{tt'}(x) \in \mathbb{R}, \qquad \{t,t'\} \in E,\ x \in X \tag{16b}$$

$$\sum_x \alpha_t(x) = 1 \qquad u_t \in \mathbb{R}, \qquad t \in T \tag{16c}$$

$$\sum_{x,x'} \alpha_{tt'}(x,x') = 1 \qquad u_{tt'} \in \mathbb{R}, \qquad \{t,t'\} \in E \tag{16d}$$

$$\alpha_t(x) \geq 0 \qquad u_t - \sum_{t' \in N_t} \varphi_{tt'}(x) \geq g_t(x), \qquad t \in T,\ x \in X \tag{16e}$$

$$\alpha_{tt'}(x,x') \geq 0 \qquad u_{tt'} + \varphi_{tt'}(x) + \varphi_{t't}(x') \geq g_{tt'}(x,x'), \quad \{t,t'\} \in E,\ x,x' \in X \tag{16f}$$

This LP pair (16) can be formulated in a number of different ways, corresponding to modifications of the primal constraints (13), discusses in section 5.1, and imposing constraints on dual variables $\mathbf{u}$, discussed in section 4.2.

Independently on Schlesinger, the LP relaxation (15) was used by Koster [KvHK98], Chekuri *et al.* [CKNZ01], and Wainwright *et al.* [WJW02]. The pair (16) was given in [Sch76b, theorem 2]. In appendix D, we describe its physical model [SK78].

5.3 Optimal Relaxed Labelings as Subgradients

Assume we have an algorithm able to compute the optimum value of linear program (16), i.e., to evaluate the function

$$U^*(\mathbf{g}) = \max_{\boldsymbol{\alpha} \in \Lambda_{G,X}} \langle \mathbf{g}, \boldsymbol{\alpha} \rangle = \min_{\mathbf{g}' \sim \mathbf{g}} U(\mathbf{g}'),$$

but cannot obtain any optimal argument. Theorem 11 says that optimal $\boldsymbol{\alpha}$ coincide with subgradients of f at $\mathbf{g}$. The whole solution set $\Lambda_{G,X}(\mathbf{g})$ is the subdifferential (see also [WJ03b, section 10.1]),

$$\Lambda_{G,X}(\mathbf{g}) = \partial U^*(\mathbf{g}). \tag{17}$$

This characterization allows to obtain some properties of the polytope $\Lambda_{G,X}(\mathbf{g})$.

As described in appendix A, the components of every $\boldsymbol{\alpha} \in \Lambda_{G,X}(\mathbf{g})$ are delimited to intervals,

$$\begin{aligned} \alpha_t^-(x) \leq \quad \alpha_t(x) \quad \leq \alpha_t^+(x), \\ \alpha_{tt'}^-(x,x') \leq \alpha_{tt'}(x,x') \leq \alpha_{tt'}^+(x,x'), \end{aligned}$$

where $\alpha_t^\pm(x)$ and $\alpha_{tt'}^\pm(x,x')$ are the left and right partial derivatives of $U^*(\mathbf{g})$. These derivatives can be computed by finite differences,

$$\alpha_t^\pm(x) = \frac{U^*(\mathbf{g} + \Delta\mathbf{g}) - U^*(\mathbf{g})}{\Delta g_t(x)}$$

where components of $\Delta\mathbf{g}$ are all zero except $\Delta g_t(x)$, which is a small positive or negative number (which can be ± 1 if the non-maximal nodes and edges are set to $-\infty$ without loss of generality).

If there is an interval which contains neither 0 nor 1, then $\Lambda_{G,X}(\mathbf{g})$ contains no integer elements and the max-sum problem has no trivial equivalent.

Even if mostly integer labelings are of interest, we will show how to compute a single element $\boldsymbol{\alpha}$ of $\Lambda_{G,X}(\mathbf{g})$. Set $V := \emptyset$. Pick a node $(t,x) \notin V$. Compute $\alpha_t^\pm(x)$ with the constraint that $\{\, \alpha_t(x) \mid (t,x) \in V \,\}$ are fixed. Set $\alpha_t(x)$ to some number from interval $[\alpha_t^-(x), \alpha_t^+(x)]$. Add (t,x) to V. Repeat until $V = T \times X$. Do the same for all edges. Unfortunately, a practical algorithm to solve the relaxed max-sum problem constrained by fixing some components of $\boldsymbol{\alpha}$ seems to be unknown.

Finally, we will give a sufficient condition for $\Lambda_{G,X}(\mathbf{g})$ to contain a single element, i.e., the left derivative to equal the right derivative for every node and edge. By (15), $\Lambda_{G,X}(\mathbf{g})$ is the convex hull of the vertices $\boldsymbol{\alpha}$ of $\Lambda_{G,X}$ that maximize $\langle \mathbf{g}, \boldsymbol{\alpha} \rangle$. If $\mathbf{g}$ is a real vector in a 'general position' with respect to the vertices of $\Lambda_{G,X}$ then there is a single optimal vertex.

5.4 Remark on the Max-sum Polytope

The original non-relaxed problem (6) can be formulated as the linear program

$$\max_{\boldsymbol{\alpha} \in \Lambda_{G,X}^*} \langle \mathbf{g}, \boldsymbol{\alpha} \rangle \tag{18}$$

where $\Lambda_{G,X}^*$ is the integral hull of $\Lambda_{G,X}$, i.e., the convex hull of $\Lambda_{G,X} \cap \{0,1\}^{|G_X|}$. In [WJ03b], polytopes $\Lambda_{G,X}$ and $\Lambda_{G,X}^*$ are derived by statistical considerations and called LOCAL(G) and MARG(G), respectively. Koster *et al.* [KvHK98, Kos99] call $\Lambda_{G,X}^*$ the *Partial-CSP polytope*.

The vertices of $\Lambda_{G,X}$ are those of $\Lambda_{G,X}^*$ plus additional fractional vertices. If G is a tree then $\Lambda_{G,X} = \Lambda_{G,X}^*$ [WJ03b]. While the number of facets of $\Lambda_{G,X}$ is polynomial in $|T|$, $|E|$, and $|X|$, the number of facets of $\Lambda_{G,X}^*$ is not, in general. So is not the number of vertices of both polytopes.

Linear constraints defining all facets of $\Lambda_{G,X}^*$ are of course unknown. Koster *et al.* [KvHK98, Kos99] give some properties of $\Lambda_{G,X}^*$. In particular, they give two classes of its facets that are not facets of $\Lambda_{G,X}$, i.e., which cut off some fractional vertices of $\Lambda_{G,X}$. An example of such a facet is given by the constraint

$$\sum_{\{(t,x),(t',x')\} \in \Gamma} \alpha_{tt'}(x,x') \leq 2 \tag{19}$$

where $\Gamma \subset E_X$ is the set of edges depicted in figure 3c. It can be verified that the single element $\boldsymbol{\alpha}$ of $\Lambda_{G,X}(\mathbf{g})$, which is $\frac{1}{2}$ on all nodes and edges depicted in figure 3c and 0 on the non-depicted edges, violates (19). It is reported that adding these constraints for triangles to the linear program (15) significantly reduced the integrality gap. However, automatic generation of violated constraints is left largely unsolved.

6 Characterizing LP Optimality

This sections discusses properties of a max-sum problem that are necessary for or equivalent to optimality of the LP (16), i.e., to minimality of the problem height.

6.1 Complementary Slackness and Relaxed Satisfiability

By weak duality, any max-sum problem $(G, X, \mathbf{g})$ and any $\boldsymbol{\alpha} \in \Lambda_{G,X}$ satisfy $\langle \mathbf{g}, \boldsymbol{\alpha} \rangle \leq U(\mathbf{g})$. Strong duality and complementary slackness characterize optimality of the LP pair (16) as follows.

Theorem 2 *For a max-sum problem $(G, X, \mathbf{g})$ and an $\boldsymbol{\alpha} \in \Lambda_{G,X}$, the following statements are equivalent:*

(a) $(G, X, \mathbf{g})$ has the minimal height of all its equivalents and $\boldsymbol{\alpha}$ has the highest quality.

(b) $\langle \mathbf{g}, \boldsymbol{\alpha} \rangle = U(\mathbf{g})$.

(c) $\boldsymbol{\alpha}$ is zero on non-maximal nodes and edges.

Let us define the **relaxed CSP** $(G, X, \bar{\mathbf{g}})$ as finding relaxed labelings on given nodes and edges, i.e., finding the set

$$\bar{\Lambda}_{G,X}(\bar{\mathbf{g}}) = \{\, \boldsymbol{\alpha} \in \Lambda_{G,X} \mid \langle \mathbf{1} - \bar{\mathbf{g}}, \boldsymbol{\alpha} \rangle = 0 \,\}. \tag{20}$$

We define that a CSP $(G, X, \bar{\mathbf{g}})$ is **relaxed-satisfiable** if $\bar{\Lambda}_{G,X}(\bar{\mathbf{g}}) \neq \emptyset$.

Further in section 6, we will consider two coupled problems, the CSP $(G, K, \bar{\mathbf{g}})$ formed by the maximal nodes and edges of a max-sum problem $(G, K, \mathbf{g})$. This coupling can be imagined by considering $\bar{\mathbf{g}}$ to be a function of $\mathbf{g}$ given by (12), rather than a free binary vector. Using these coupled problems, complementary slackness manifests itself as follows.

Theorem 3 *The height of $(G, X, \mathbf{g})$ is minimal of all its equivalents if and only if $(G, X, \bar{\mathbf{g}})$ is relaxed-satisfiable. If it is so then $\Lambda_{G,X}(\mathbf{g}) = \bar{\Lambda}_{G,X}(\bar{\mathbf{g}})$.*

6.2 Arc Consistency Is Necessary for LP Optimality

In section 3, arc consistency and kernel were shown useful for characterizing satisfiability of a CSP. We will show they are useful also for characterizing optimality of the LP pair (16). First, we generalize the crucial property of the kernel to the relaxed CSP.

Theorem 4 *Let $(G, X, \bar{\mathbf{g}}^*)$ be the kernel of a CSP $(G, X, \bar{\mathbf{g}})$. Then $\bar{\Lambda}_{G,X}(\bar{\mathbf{g}}) = \bar{\Lambda}_{G,X}(\bar{\mathbf{g}}^*)$.*

Proof. We will need the following two implications. By (20), if $\bar{\mathbf{g}}' \leq \bar{\mathbf{g}}$ then $\bar{\Lambda}_{G,X}(\bar{\mathbf{g}}') \subseteq \bar{\Lambda}_{G,X}(\bar{\mathbf{g}})$. By the definition of the kernel, if a problem $\bar{\mathbf{g}}'$ is arc consistent and $\bar{\mathbf{g}}' \leq \bar{\mathbf{g}}$ then $\bar{\mathbf{g}}' \leq \bar{\mathbf{g}}^*$.

Since $\bar{\mathbf{g}}^* \leq \bar{\mathbf{g}}$, it is $\bar{\Lambda}_{G,X}(\bar{\mathbf{g}}^*) \subseteq \bar{\Lambda}_{G,X}(\bar{\mathbf{g}})$. Let $\boldsymbol{\alpha} \in \bar{\Lambda}_{G,X}(\bar{\mathbf{g}})$. Let $\bar{g}'_t(x) = \delta_{\alpha_t(x)>0}$ and $\bar{g}'_{tt'}(x, x') = \delta_{\alpha_{tt'}(x,x')>0}$. Since (13a) and (13c) imply (3), $\bar{\mathbf{g}}'$ is arc consistent. Since $\bar{\mathbf{g}}'$ is arc consistent and $\bar{\mathbf{g}}' \leq \bar{\mathbf{g}}$, it is $\bar{\mathbf{g}}' \leq \bar{\mathbf{g}}^*$. Therefore $\boldsymbol{\alpha} \in \bar{\Lambda}_{G,X}(\bar{\mathbf{g}}^*)$. ■

A corollary is that a non-empty kernel of $(G, X, \bar{\mathbf{g}})$ is necessary for its relaxed satisfiability. Dually, this can be proved as follows. Further on, we will denote the **height of pencil** (t, t', x) by $u_{tt'}(x) = \max_{x'} g_{tt'}(x, x')$ and call (t, t', x) a **maximal pencil** if it contains a maximal edge. Let us modify the arc consistency algorithm such that rather than by explicitly zeroing variables $\bar{\mathbf{g}}$ like in (4), nodes and edges of $(G, X, \bar{\mathbf{g}})$ are deleted by repeating the following ETs on $(G, X, \mathbf{g})$:

- Find a triplet (t, t', x) such that pencil (t, t', x) is non-maximal and node (t, x) is maximal, i.e. $u_{tt'}(x) < u_{tt'}$ and $g_t(x) = u_t$. Decrease node (t, x) by $\varphi_{tt'}(x) = \frac{1}{2}[u_{tt'} - u_{tt'}(x)]$ and increase all edges in pencil (t, t', x) by the same amount.
- Find a triplet (t, t', x) such that pencil (t, t', x) is maximal and node (t, x) is non-maximal, i.e. $u_{tt'}(x) = u_{tt'}$ and $g_t(x) < u_t$. Increase node (t, x) by $\varphi_{tt'}(x) = \frac{1}{2}[u_t - g_t(x)]$ and decrease all edges in pencil (t, t', x) by the same amount.

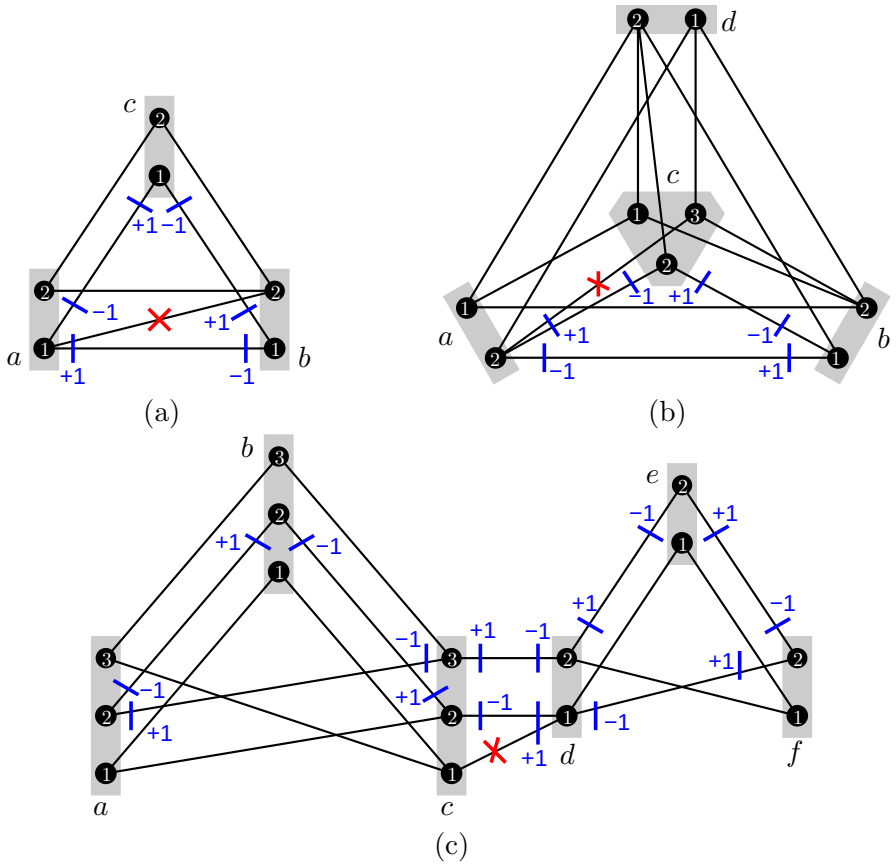


Figure 6: Examples of kernels not invariant to equivalent transformations. The transformations are depicted by numbers $\varphi_{tt'}(x)$ written next to the line segments crossing the edge pencils (t, t', x); it is assumed without loss of generality that the non-maximal edges are smaller than -1. Problem (a) has minimal height, problems (b, c) do not.

When no such triplets exist, the algorithm halts.

If the kernel of $(G, X, \bar{\mathbf{g}})$ is initially non-empty, the algorithm halts after the maximal nodes and edges not in the kernel are made non-maximal. If the kernel is initially empty, the algorithm (4) would sooner or later delete the last node (edge) in some object (pair). This cannot happen here, because each object (pair) always contains at least one maximal node (edge). Instead, the algorithm here decreases the height of some node or edge, hence the problem height.

6.3 Arc Consistency Is Insufficient for LP Optimality

One could hope that non-emptiness of the kernel is not only necessary but also sufficient for LP optimality. We will show it is not. This was observed by Schlesinger [Sch76a] during the work on the papers [Sch76b, KS76].

By theorem 4, if $\boldsymbol{\alpha} \in \bar{\Lambda}_{G,X}(\bar{\mathbf{g}})$ and a node (edge) does not belong to the kernel of $(G, X, \bar{\mathbf{g}})$ then $\boldsymbol{\alpha}$ is zero on this node (edge). Less obviously, there can be a node (edge) in the kernel such that every $\boldsymbol{\alpha} \in \bar{\Lambda}_{G,X}(\bar{\mathbf{g}})$ is zero on it. Figure 6a shows an example: it can be verified that any $\boldsymbol{\alpha}$ that is zero on the absent nodes and edges and satisfies (13a) has $\alpha_{ab}(1,2) = 0$. In figures 6b and 6c, the only element $\boldsymbol{\alpha}$ of $\bar{\Lambda}_{G,X}(\bar{\mathbf{g}})$ is zero even on all nodes and edges[1], therefore $(G, X, \bar{\mathbf{g}})$ is relaxed-insatisfiable.

This can be interpreted dually, by showing that the kernel of $(G, X, \bar{\mathbf{g}})$ is not invariant to equivalent transformations of $(G, X, \mathbf{g})$. The transformation depicted in figure 6a makes edge

[1]For figure 6c, one can reason as follows. If the edges in pair $\{c, d\}$ are not considered, it is inevitably $\alpha_t(x) = \frac{1}{3}$ for $t \in \{a, b, c\}$ and $x \in X$, and $\alpha_t(x) = \frac{1}{2}$ for $t \in \{d, e, f\}$ and $x \in X$. But by (13a), it should be $\alpha_{cd}(1,1) = \alpha_{cd}(2,1) = \frac{1}{3}$ and $\alpha_{cd}(1,1) + \alpha_{cd}(2,1) = \frac{1}{2}$, which is impossible.

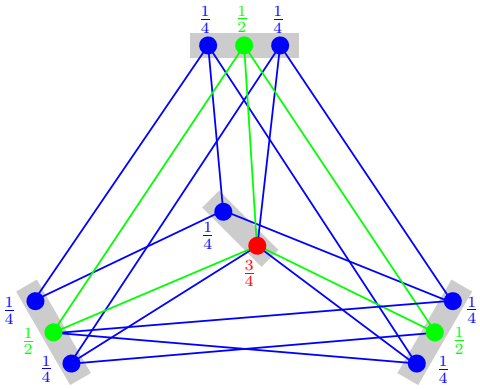


Figure 7: The CSP for which $\bar{\Lambda}_{G,X}(\bar{\mathbf{g}})$ has a single element $\boldsymbol{\alpha}$ that is not an integer multiple of $|X|^{-1}$.

$\{(a,1),(b,2)\}$ non-maximal and thus deletes it from the kernel. The transformations in figures 6b and 6c make respectively edges $\{(a,2),(c,3)\}$ and $\{(c,1),(d,1)\}$ non-maximal, which makes the kernel empty. Thus, a non-empty kernel does not suffice for minimality of the height of $(G,X,\mathbf{g})$.

6.4 Summary: Three Kinds of Consistency

To summarize, we have met three kinds of 'consistency' of the two coupled problems, related by implications as follows

$$\begin{array}{c}(G,X,\bar{\mathbf{g}}) \text{ satisfiable}\\ (G,X,\mathbf{g}) \text{ trivial}\end{array} \implies \begin{array}{c}(G,X,\bar{\mathbf{g}}) \text{ relaxed-satisfiable}\\ \text{height of } (G,X,\mathbf{g}) \text{ minimal}\end{array} \implies \text{kernel of } (G,X,\bar{\mathbf{g}}) \text{ non-empty}$$

Testing for the first condition is NP-complete. Testing for the last condition is polynomial and easy, based on the local property of arc consistency. Testing for the middle condition is polynomial but an efficient algorithm that would detect an arc consistent but relaxed-unsatisfiable state and would escape from it by a height-decreasing ET is, to our knowledge, unknown. Exceptions are problems with two labels, for which non-empty kernel equals relaxed satisfiability, and supermodular max-sum problems (lattice CSPs) and problems on trees for which non-empty kernel equals satisfiability.

As far as we know, all efficient algorithms for decreasing height of large max-sum problems use arc consistency as the termination criterion. The algorithm from section 6.2 is not practical for its slow convergence. Better examples are reviewed in sections 7 and 8. Another example is TRW-S [Kol04]. Existence of arc consistent but relaxed-unsatisfiable configurations is unpleasant here because these algorithms need not find the minimal upper bound. This corresponds to the spurious minima of TRW-S observed in [Kol04, Kol05a].

6.5 Problems with Two Labels

For problems with $|X| = 2$ labels, a non-empty kernel turns out to be both necessary and sufficient for LP optimality. This is given by the following result due to Schlesinger [Sch05b] (also given by [Kol05c]), which additionally shows that at least one relaxed labeling is an (integer) multiple of $\frac{1}{2}$. For $|X| > 2$, a relaxed labeling that is a multiple of $|X|^{-1}$ may not exist even if $\bar{\Lambda}_{G,X}(\bar{\mathbf{g}}) \neq \emptyset$, as shown in figure 7. Note, the theorem implies that for $|X| = 2$, the co-ordinates of all vertices of $\Lambda_{G,X}$ are multiples of $\frac{1}{2}$, since any vertex can be made optimal to $(G,X,\mathbf{g})$ by some choice of $\mathbf{g}$.

Theorem 5 *Let a CSP $(G,X,\bar{\mathbf{g}})$ with $|X| = 2$ labels have a non-empty kernel. Then $\bar{\Lambda}_{G,X}(\bar{\mathbf{g}}) \cap \{0,\frac{1}{2},1\}^{|G_X|} \neq \emptyset$.*

Proof. We will prove the theorem by constructing an $\boldsymbol{\alpha} \in \bar{\Lambda}_{G,X}(\bar{\mathbf{g}}) \cap \{0,\frac{1}{2},1\}^{|G_X|}$.

Delete all nodes and edges not in the kernel. Denote the number of nodes in object t and the number of edges in pair $\{t, t'\}$ respectively by

$$n_t = \sum_x \bar{g}_t(x), \quad n_{tt'} = \sum_{x,x'} \bar{g}_{tt'}(x, x').$$

All object pairs can be partitioned into five classes (up to swapping labels), indexed by triplets $(n_t, n_{t'}, n_{tt'})$:

$(n_t, n_{t'}, n_{tt'})$	$(1,1,1)$	$(1,2,2)$	$(2,2,2)$	$(2,2,3)$	$(2,2,4)$
					

Remove one edge in each pair of class $(2,2,3)$ and two edges in each pair of class $(2,2,4)$ such that they become $(2,2,2)$. Now there are only pairs of classes $(1,1,1)$, $(1,2,2)$ and $(2,2,2)$. Let

$$\alpha_t(x) = \frac{\bar{g}_t(x)}{n_t}, \quad \alpha_{tt'}(x, x') = \frac{\bar{g}_{tt'}(x, x')}{n_{tt'}}.$$

Clearly, this $\boldsymbol{\alpha}$ belongs to $\bar{\Lambda}_{G,X}(\bar{\mathbf{g}})$. ■

7 Max-sum Diffusion

This section describes the max-sum diffusion algorithm for decreasing the height of a max-sum problem [KK75,Fla98,Sch05b], which can be viewed as a modification of the algorithm in section 6.2.

7.1 The Algorithm

The **node-pencil averaging** on pencil (t, t', x) is the equivalent transformation which makes $g_t(x)$ and $u_{tt'}(x)$ equal, i.e., which adds number $\varphi_{tt'}(x) = \frac{1}{2}[u_{tt'}(x) - g_t(x)]$ to $g_t(x)$ and subtracts the same number from qualities $\{\, g_{tt'}(x, x') \mid x' \in X \,\}$ of all edges in (t, t', x). In its simplest form, the max-sum diffusion algorithm is as follows: repeat node-pencil averaging until convergence, on all pencils in any order such that each pencil is visited 'sufficiently often'.

This algorithm can be slightly reformulated. If a node (t, x) is chosen and node-pencil averaging is iterated on the pencils $\{\, (t, t', x) \mid t' \in N_t \,\}$ till convergence, the heights of all these pencils and $g_t(x)$ become equal. This can be done faster by a single equivalent transformation on the node (t, x), called the **node averaging**.

The following code repeats node averaging for all nodes in a pre-defined order until convergence. Recall that $\mathbf{g}^{\boldsymbol{\varphi}}$ is given by (7).

repeat
  for $(t, x) \in T \times X$ **do**
    for $t' \in N_t$ **do**
      $u_{tt'}(x) := \max_{x'} g^{\boldsymbol{\varphi}}_{tt'}(x, x')$;
    end for
    $\Delta u := \dfrac{g^{\boldsymbol{\varphi}}_t(x) + \sum_{t' \in N_t} u_{tt'}(x)}{|N_t| + 1}$;
    for $t' \in N_t$ **do**
      $\varphi_{tt'}(x) := u_{tt'}(x) - \Delta u$;
    end for
  end for
until convergence
$\mathbf{g} := \mathbf{g}^{\boldsymbol{\varphi}}$;

Let us remark that for $|X| = 1$, the algorithm just iteratively averages neighboring nodes and edges, converging to the state when they all become equal to their mean value $\langle \mathbf{1}, \mathbf{g} \rangle / |G_X|$.

7.2 Monotonicity

When node-pencil averaging is done on a single pencil, the problem height can decrease, remain unchanged, or increase. For an example when the height increases, consider a max-sum problem with $X = \{1, 2\}$ such that $g_t(1) = g_t(2) = 1$ and $u_{tt'}(1) = u_{tt'}(2) = -1$ for some pair $\{t, t'\}$. After making node-pencil averaging on $(t, t', 1)$, the height increases by 1.

However, monotonic height decrease can be ensured by choosing a proper order of pencils as follows (a similar theorem is true for node averagings).

Theorem 6 *After the equivalent transformation consisting of $|X|$ node-pencil averagings on the pencils $\{ (t, t', x) \mid x \in X \}$, the height does not increase.*

Proof. Before the transformation, the contribution of object t and pair $\{t, t'\}$ to the total height is $\max_x g_t(x) + \max_x u_{tt'}(x)$. After the transformation, this contribution is $\max_x[g_t(x) + u_{tt'}(x)]$. The first expression is not smaller than the second one because any two functions $f_1(x)$ and $f_2(x)$ satisfy $\max_x f_1(x) + \max_x f_2(x) \geq \max_x[f_1(x) + f_2(x)]$. ■

7.3 Properties of the Fixed Point

The fixed point of the max-sum diffusion algorithm is characterized by the following conjecture, which was formulated based on numerous experiments.

Conjecture 1 *The algorithm converges to a solution of the system*

$$\max_{x'} g_{tt'}(x, x') = g_t(x), \quad \{t, t'\} \in E, \ x \in X. \tag{21}$$

The property (21) implies arc consistency of the maximal nodes and edges. However, the converse is false: not every max-sum problem with arc consistent maximal nodes and edges satisfies (21). This is because arc consistency constrains only maximal nodes and edges, while (21) constrains also non-maximal nodes and some non-maximal edges.

Theorem 7 *If a max-sum problem satisfies (21) then its maximal nodes and edges are arc consistent.*

Proof. Assume that (21) holds. Following (3), we are to prove that a pencil (t, t', x) is maximal if and only if node (t, x) is maximal.

Assume that (t, x) is maximal. Then $g_t(x) \geq g_t(x')$ for each $x' \in X$. By (21), it is $u_{tt'}(x) \geq u_{tt'}(x')$ for each $x' \in X$. Hence (t, t', x) is maximal.

Assume that (t, x) is not maximal. Then $g_t(x) < g_t(x')$ for some $x' \in X$. By (21), it is $u_{tt'}(x) < u_{tt'}(x')$ for some $x' \in X$. Hence (t, t', x) is not maximal. ■

Any solution to (21) has the following layered structure (see figure 8). A **layer** is a maximal connected subgraph of G_X such that its each edge $\{(t, x), (t', x')\}$ satisfies $g_t(x) = g_{tt'}(x, x') = g_{t'}(x')$. It follows from (21) that all nodes and edges of a layer have the same quality, the **height of the layer**. The highest layer is formed by the maximal nodes and edges.

Remark. We observed that the layers form a poset, however, we see no use of this so far. Let a relation $\leq$ on the system $\Omega = \{\omega_i\}$ of layers be defined as follows: $\omega_1 \leq \omega_2$ if and only if there is a node (t, x) of ω_1 and a node (t', x') of ω_2 such that $g_t(x) = g_{tt'}(x, x') \leq g_{t'}(x')$. The transitive closure $\leq^*$ of $\leq$ is a partial order. The greatest element of the poset $(\Omega, \leq^*)$ is the highest layer. However, $(\Omega, \leq^*)$ is neither a lattice nor a semilattice.

Further observation is that the differences between the heights of neighboring layers provide some explicit knowledge about stability of the layers to perturbations of $\mathbf{g}$.

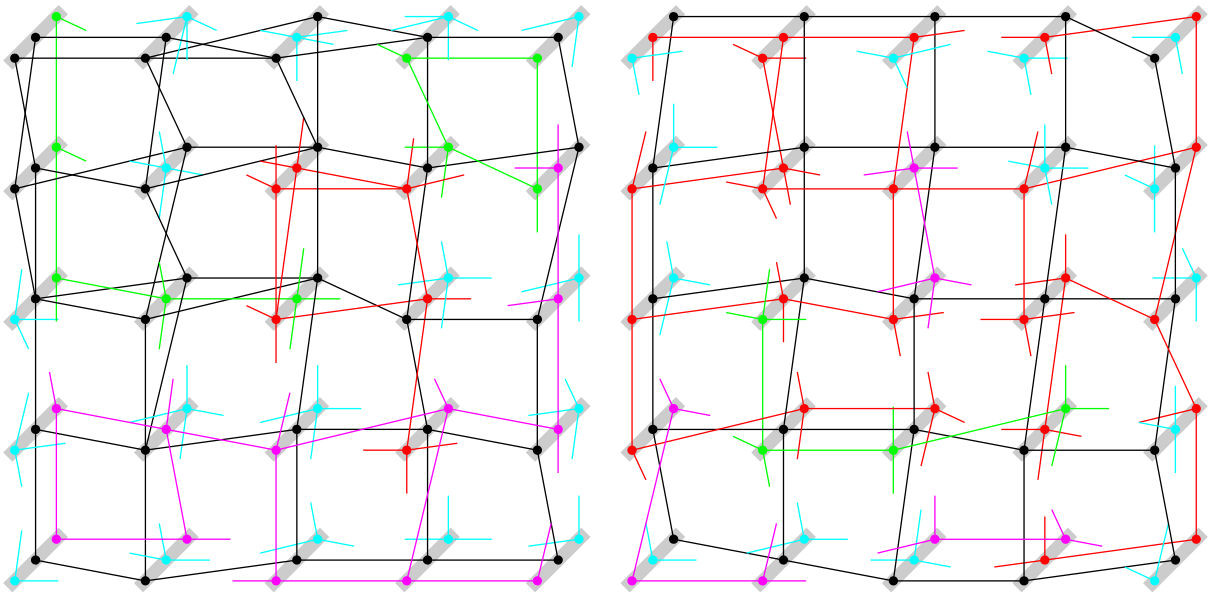


Figure 8: Two examples of max-sum problems satisfying (21). A line segment starting from node (t, x) and aiming to but not reaching (t', x') denotes an edge satisfying $g_t(x) = g_{tt'}(x, x') < g_{t'}(x')$. If $g_t(x) = g_{tt'}(x, x') = g_{t'}(x')$, the line segment joins the nodes (t, x) and (t', x'). The colors help distinguish different layers; the highest layer is drawn in black.

8 Augmenting DAG Algorithm

This section describes the algorithm for decreasing the problem height given in [KS76, Sch89]. Its main idea is to run the arc consistency algorithm (4) on the maximal nodes and edges, storing the pointers to the causes of deletions. When all nodes in a single object are deleted, it is clear that the kernel is empty. Backtracking the pointers provides a directed acyclic graph (DAG), called the augmenting DAG, along which a height-decreasing ER is done.

The iteration of the algorithm proceeds in three phases, described in subsequent sections. We use formulation (11a), i.e., we look for $\boldsymbol{\varphi}$ that minimizes $U(\mathbf{g}^{\boldsymbol{\varphi}}) = \sum_t \max_x g_t^{\boldsymbol{\varphi}}(x)$ subject to constraint that all edges are non-positive (i.e., all pairs have zero height). Initially, all edges are assumed non-positive.

8.1 Phase 1: Arc Consistency Algorithm

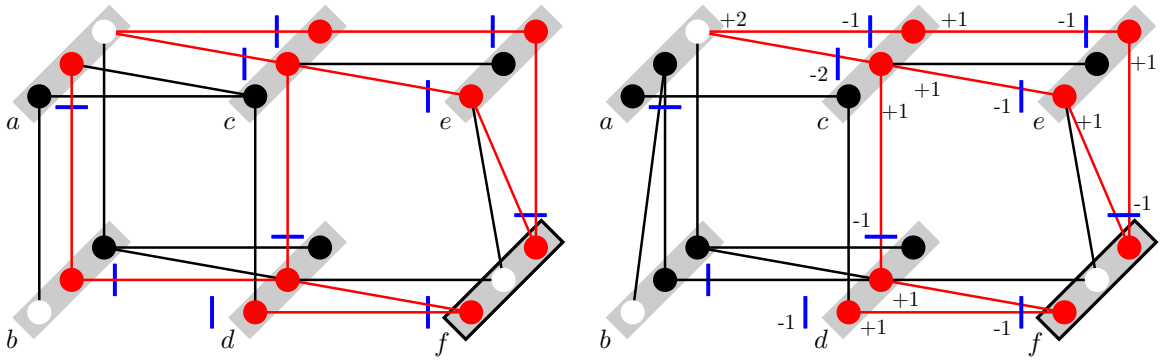
In the first phase, the arc consistency algorithm is run on the maximal nodes and edges. It is not done exactly as described by rules (4) but in a slightly modified way as follows.

1. An auxiliary variable $p_t(x) \in \{\texttt{ALIVE}, \texttt{NONMAX}\} \cup T$ is assigned to each node (t, x). Initially, we set $p_t(x) := \texttt{ALIVE}$ if (t, x) is maximal and $p_t(x) := \texttt{NONMAX}$ if (t, x) is non-maximal.
2. If a pencil (t, t', x) is found satisfying $p_t(x) = \texttt{ALIVE}$ and violating condition

$$(\exists x')[\ \{(t, x), (t', x')\} \text{ maximal},\ p_{t'}(x') = \texttt{ALIVE}\ ], \tag{22}$$

   node (t, x) is deleted by setting $p_t(x) := t'$. The object t' is called the **deletion cause** of node (t, x). This is repeated until either no such pencil exists, or an object t^* is found with $p_{t^*}(x) \neq \texttt{ALIVE}$ for all $x \in X$. In the former case, the augmenting DAG algorithm halts. In the latter case, we proceed to the next phase.

After every iteration of this algorithm, the maximal edges and the variables $p_t(x)$ define a directed acyclic subgraph D of G_X, as follows: the nodes of D are the end nodes of its edges; edge $((t, x), (t', x'))$ belongs to D if and only if it is maximal and $p_t(x) = t'$. Once t^* has been found, the



(a) (b)

Figure 9: (a) The augmenting DAG algorithm after the arc consistency algorithm (Phase 1) and (b) after finding the search direction (Phase 2).

augmenting DAG $D(t^*)$ is a subgraph of D reachable by a directed path in D from the maximal nodes of t^*.

Example. The example max-sum problem in figure 9 has $T = \{a, \dots, f\}$ and the labels in each object are $1, 2, 3$, numbered from bottom to top. Figure 9a shows the maximal edges and the values of $p_t(x)$ after the first phase, when 10 nodes have been deleted by applying rule (22) succesively on pencils $(c, a, 2)$, $(c, a, 3)$, $(e, c, 1)$, $(e, c, 3)$, $(f, e, 3)$, $(d, c, 2)$, $(b, d, 2)$, $(a, b, 2)$, $(d, b, 1)$, $(f, d, 1)$. The non-maximal edges are not shown. The nodes with $p_t(x) = \texttt{ALIVE}$ are drawn in black, with $p_t(x) = \texttt{NONMAX}$ in white, and with $p_t(x) \in T$ in red. For the deleted nodes, the causes $p_t(x)$ are denoted by short blue segments accross pencils $(t, p_t(x), x)$. The object $t^* = f$ has a black outline.

Figure 9a shows D after t^* has been found and figure 9b shows $D(t^*)$. The edges of D and $D(t^*)$ are depicted in red and the nodes, except $(a, 3)$, too. ■

8.2 Phase 2: Finding the Search Direction

In the second phase, the direction of height decrease is found in the space $\mathbb{R}^{2|E||X|}$, i.e., a vector $\Delta\boldsymbol{\varphi}$ is found such that $U(\mathbf{g}^{\boldsymbol{\varphi}+\lambda\Delta\boldsymbol{\varphi}}) < U(\mathbf{g}^{\boldsymbol{\varphi}})$ for a small positive λ.

Using the abbreviation $\Delta\varphi_t(x) = \sum_{t' \in N_t} \Delta\varphi_{tt'}(x)$, the vector $\Delta\boldsymbol{\varphi}$ has to satisfy

$$\Delta\varphi_{t^*}(x) = -1, \quad x \in X,\ p_{t^*}(x) \neq \texttt{NONMAX}, \tag{23a}$$

$$\Delta\varphi_t(x) \leq 0, \quad t \in T,\ x \in X,\ p_t(x) \neq \texttt{NONMAX}, \tag{23b}$$

$$\Delta\varphi_{tt'}(x) + \Delta\varphi_{t't}(x') \geq 0, \quad \{t, t'\} \in E,\ x, x' \in X,\ \{(t, x), (t', x')\} \text{ maximal}. \tag{23c}$$

We find the smallest vector $\Delta\boldsymbol{\varphi}$ satisfying (23). This is done by traversing $D(t^*)$ from roots to leaves, successively enforcing constraints (23). The traversal is done in a linear order on $D(t^*)$, i.e., a node is not visited before the tails of all edges entering it have been visited. In figure 9b, the non-zero numbers $\Delta\varphi_{tt'}(x)$ are written near their corresponding pencils.

8.3 Phase 3: Finding the Search Step

In the third phase, the length λ of the search step is found such that the height of no pair is increased (i.e., all edges are kept non-positive), the height of no object is increased, and the height of t^* is minimized. These read respectively

$$\begin{aligned} g_{tt'}^{\boldsymbol{\varphi}+\lambda\Delta\boldsymbol{\varphi}}(x, x') &\leq 0, && \{t, t'\} \in E,\ x, x' \in X, \\ g_t^{\boldsymbol{\varphi}+\lambda\Delta\boldsymbol{\varphi}}(x) &\leq \max_x g_t^{\boldsymbol{\varphi}}(x), && t \in T,\ x \in X, \\ g_{t^*}^{\boldsymbol{\varphi}+\lambda\Delta\boldsymbol{\varphi}}(x) &\leq \max_x g_{t^*}^{\boldsymbol{\varphi}}(x) - \lambda, && x \in X. \end{aligned}$$

To derive the last inequality, see that each node of t^* with $p_{t^*}(x) \in T$ decreases by λ and each node with $p_{t^*}(x) = \texttt{NONMAX}$ increases by $\lambda\,\Delta\varphi_{t^*}(x)$. The latter is because $D(t^*)$ can have a leaf in t^*. To minimize the height of t^*, the nodes with $p_{t^*}(x) = \texttt{NONMAX}$ must not become higher than the nodes with $p_{t^*}(x) \in T$.

Solving the above three conditions for λ yields the inequality system

$$\lambda \le \frac{g^{\boldsymbol{\varphi}}_{tt'}(x,x')}{\Delta\varphi_{tt'}(x) + \Delta\varphi_{t't}(x')}, \qquad \{t,t'\} \in E,\ x,x' \in X,\ \Delta\varphi_{tt'}(x) + \Delta\varphi_{t't}(x') < 0, \tag{24a}$$

$$\lambda \le \frac{u_t - g^{\boldsymbol{\varphi}}_t(x)}{\delta_{t=t^*} + \Delta\varphi_t(x)}, \qquad t \in T,\ x \in X,\ \delta_{t=t^*} + \Delta\varphi_t(x) > 0. \tag{24b}$$

We find the greatest λ satisfying (24).

The iteration of the augmenting DAG algorithm is completed by the equivalent transformation $\boldsymbol{\varphi} \mathrel{+}= \lambda\,\Delta\boldsymbol{\varphi}$.

8.4 Introducing Thresholds

The number of iterations depends on the lengths λ of search steps and on the difference between the initial and final height. Each iteration takes polynomial time, but this time depends on the size of the augmenting DAG. The size of the augmenting DAG and λ depend in a complicated way on the precise strategy how D is constructed during Phase 1 (note that this strategy was left unspecified in section 8.1).

Figure 10a illustrates how a 'bad' strategy of constructing D in Phase 1 can lead to an unnecessarily small λ. This is analogical to the well-known drawback of the Ford-Fulkerson max-flow algorithm. The figure shows $D(t^*)$, depicted similarly as in figure 9b. On the top figure, $u_{t^*} = 1000$, $u_t = 0$, edge A is maximal with quality 0, and edge B has quality -1. The search step is $\lambda = 1$, the bottleneck being edge B. The equivalent transformation makes A non-maximal and B maximal. The bottom figure shows $D(t^*)$ for the subsequent iteration, in which A and B swapped their rôles. The edge qualities and step lengths in the i-th iteration are as follows:

i	1	2	3	...
$g^{\boldsymbol{\varphi}}(A)$	0	-1	0	...
$g^{\boldsymbol{\varphi}}(B)$	-1	0	-1	...
λ	1	1	1	...

An even worse situation is shown in figure 10b. The step λ decreases exponentially:

i	1	2	3	4	5	6	...
$g^{\boldsymbol{\varphi}}(A)$	0	$-\frac{1}{2}$	0	$-\frac{1}{4}$	0	$-\frac{1}{8}$	...
$g^{\boldsymbol{\varphi}}(B)$	-1	0	$-\frac{1}{2}$	0	$-\frac{1}{4}$	0	...
λ	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{8}$	...

This inefficient behavior can be reduced by re-defining maximality of nodes and edges using a threshold $\varepsilon > 0$ [KS76]. A node (t,x) or an edge $\{(t,x),(t',x')\}$ is maximal if and only if respectively

$$\max_x g^{\boldsymbol{\varphi}}_t(x) - g^{\boldsymbol{\varphi}}_t(x) \le \varepsilon,$$
$$-g^{\boldsymbol{\varphi}}_{tt'}(x,x') \le \varepsilon.$$

We maintain that the edge is non-positive if and only if $g^{\boldsymbol{\varphi}}_{tt'}(x,x') \le 0$. If ε is reasonably large, 'nearly maximal' nodes and edges, like A and B in the example, are considered maximal with the hope that a larger λ will result. A possible scheme is to run the augmenting DAG algorithm several times, exponentially decreasing ε. With $\varepsilon > 0$, the algorithm terminates in a finite number of iterations [KS76]. However, polynomial complexity has not been even conjectured.

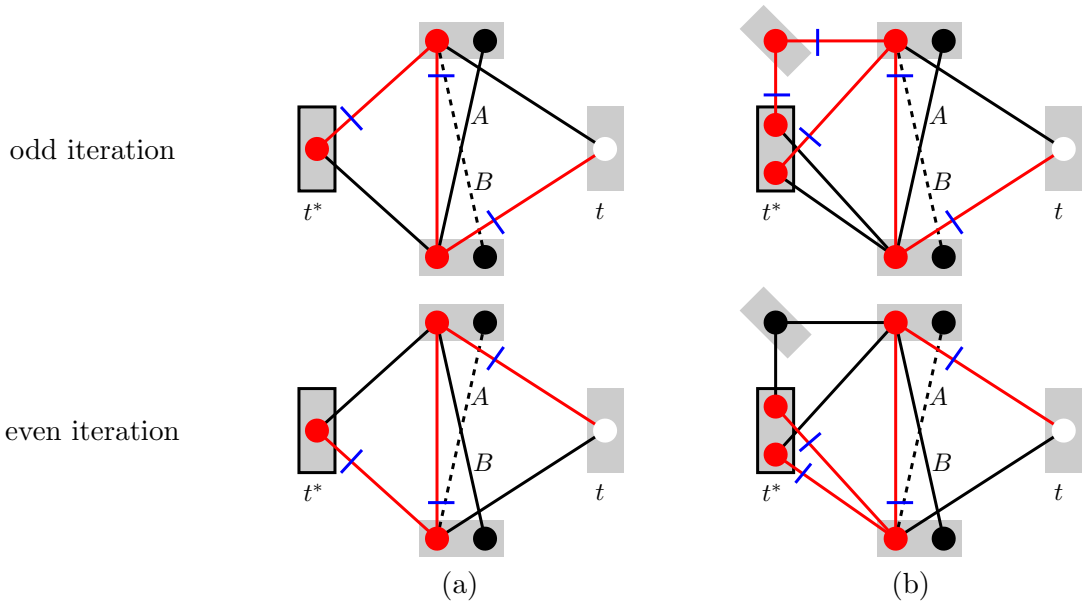


Figure 10: Examples of inefficient behavior of the augmenting DAG algorithm.

If $\varepsilon = 0$, the algorithm sometimes spends a lot of iterations to minimize the height in a subgraph of G accurately [Sch05b]. This is indeed wasteful because this accuracy is destroyed once the subproblem is left.

Remark. The partial derivatives $\alpha_t^{\pm}(x)$ and $\alpha_{tt'}^{\pm}(x, x')$ from section 5.3 can be conveniently computed by the tools of the augmenting DAG algorithm, *provided that* an arc consistent configuration with non-minimal height is not met. These derivatives are closely related to the search direction $\Delta\boldsymbol{\varphi}$. For example, increase $g_t(x)$ by 1, do Phase 1 until a dead object t^* is found, and compute $\Delta\boldsymbol{\varphi}$. Then $\alpha_t^{+}(x) = [\,\sum_{t' \in N_t} \Delta\varphi_{tt'}(x)\,]^{-1}$.

Implementing the algorithm requires solving implementation details, described only partially or not at all in the original paper [KS76]. For example, starting every iteration with empty D is inefficient, D has to be re-used rather than thrown away. Or, treating rounding errors consistently is not easy. We provide the detailed description including the code in appendix E.

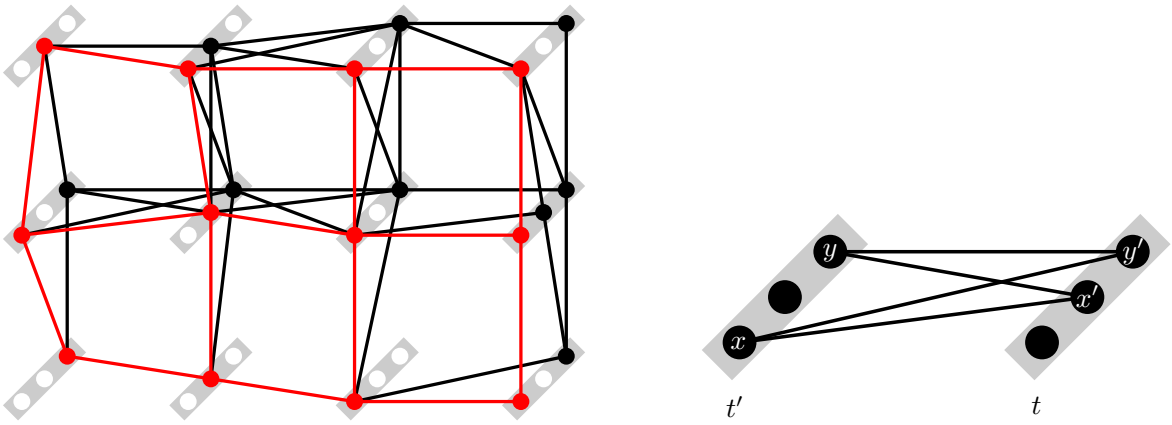
9 Supermodularity

In this section, we assume that the label set X is endowed with a known total order $\leq$, i.e., the poset $(X, \leq)$ is a chain. This total order induces the componentwise partial order on a product $(X^n, \leq)$ of these chains, where we have denoted the new order by the same symbol $\leq$. The poset $(X^n, \leq)$ is a distributive lattice, with join (meet) denoted[2] by $\vee$ ($\wedge$). We will show that any max-sum problem for which the functions $g_{tt'}(\bullet, \bullet)$ are supermodular on $(X^2, \leq)$ has a trivial equivalent and that finding an optimal labeling is tractable.

9.1 Lattice CSP

We call $(G, X, \bar{\mathbf{g}})$ a **lattice CSP** if the poset $(\bar{L}_{tt'}, \leq)$ is a lattice for every $\{t, t'\} \in E$, where we denoted $\bar{L}_{tt'} = \{\,(x, x') \mid \bar{g}_{tt'}(x, x') = 1\,\}$. It follows easily that for a lattice CSP, $\bar{L}_{G,X}(\bar{\mathbf{g}})$ is also a lattice. The following theorem shows that lattice CSP are tractable (figure 11a illustrates the argument).

[2]Elsewhere in the report, we use the symbol $\leq$ also for natural order on $\mathbb{R}$ and the symbols $\wedge$ and $\vee$ also for logical conjunction and disjunction. This overloading will not cause confusion because the meaning will be always determined by operands.



(a) (b)

Figure 11: (a) An arc consistent lattice CSP is always satisfiable. A labeling on it can be found by taking the lowest label in each object separately (in red). (b) Supermodular max-sum problems satisfy $g_{tt'}(x,x') + g_{tt'}(y,y') \geq g_{tt'}(x,y') + g_{tt'}(y,x')$ for every $x \leq y$ and $x' \leq y'$. It follows that the poset $\bar{L}_{tt'} = \{\, (x,x') \mid g_{tt'}(x,x') = u_{tt'} \,\}$ is a lattice.

Theorem 8 *Any arc consistent lattice CSP $(G, X, \bar{\mathbf{g}})$ is satisfiable. The 'lowest' labeling $\mathbf{x} = \bigwedge \bar{L}_{G,X}(\bar{\mathbf{g}})$ is given by $x_t = \min\{\, x \in X \mid \bar{g}_t(x) = 1 \,\}$.*

Proof. We will show that $\bar{g}_{tt'}(x_t, x_{t'}) = 1$ for all $\{t,t'\} \in E$. Pick a $\{t,t'\} \in E$. By (3), pencil (t,t',x_t) contains at least one edge, while pencils $\{\, (t,t',x) \mid x < x_t \,\}$ are empty. Similarly for pencils $(t',t,x_{t'})$ and $\{\, (t',t,x') \mid x' < x_{t'} \,\}$. Since $(\bar{L}_{tt'}, \leq)$ is a lattice, the meet of the edges in the pair $\{t,t'\}$ is $\{(t,x_t),(t',x_{t'})\}$. ■

9.2 Supermodular Max-sum Problems

We call $(G, X, \mathbf{g})$ a **supermodular max-sum problem** if all the functions $g_{tt'}(\bullet,\bullet)$ are supermodular. The following theorem shows that this is equivalent to supermodularity of the quality function $F(\bullet \,|\, \mathbf{g})$. We state the theorem in a slightly more general form; for that, recall that a multivariate function is separable if it is a sum of univariate functions.

Theorem 9 *In a max-sum problem, the function $F(\bullet \,|\, \mathbf{g})$ is supermodular resp. separable if and only if all the bivariate functions $g_{tt'}(\bullet,\bullet)$ are supermodular resp. separable.*

Proof. The *if* implication is true because supermodularity is closed under addition.

The *only if* part. Pick a pair $\{t,t'\} \in E$. Let two labelings $\mathbf{x}, \mathbf{y} \in X^{|T|}$ be equal in all objects except t and t' where they satisfy $x_t \leq x_{t'}$ and $y_t \geq y_{t'}$. If $F(\bullet \,|\, \mathbf{g})$ is supermodular, by (31) it is $F(\mathbf{x} \wedge \mathbf{y} \,|\, \mathbf{g}) + F(\mathbf{x} \vee \mathbf{y} \,|\, \mathbf{g}) \geq F(\mathbf{x} \,|\, \mathbf{g}) + F(\mathbf{y} \,|\, \mathbf{g})$. After substitution from (5) and some manipulations, we are left with $g_{tt'}(x_t, y_{t'}) + g_{tt'}(y_t, x_{t'}) \geq g_{tt'}(x_t, x_{t'}) + g_{tt'}(y_t, y_{t'})$.

It is easy to verify that for *any* order $\leq$, $F(\mathbf{x} \wedge \mathbf{y} \,|\, \mathbf{g}) + F(\mathbf{x} \vee \mathbf{y} \,|\, \mathbf{g}) = F(\mathbf{x} \,|\, \mathbf{g}) + F(\mathbf{y} \,|\, \mathbf{g})$ implies $g_{tt'}(x_t, y_{t'}) + g_{tt'}(y_t, x_{t'}) = g_{tt'}(x_t, x_{t'}) + g_{tt'}(y_t, y_{t'})$. This proves separability. ■

Since the function $F(\bullet \,|\, \mathbf{g})$ is invariant to equivalent transformations, theorem 9 implies that supermodularity of the functions $g_{tt'}(\bullet,\bullet)$ is so too. This is alternatively seen from the fact that an equivalent transformation means adding a zero problem, which is modular, and supermodularity is invariant to adding a modular function.

Theorem 10 [Top78, theorem 4.1] *The set A^* of maximizers of a supermodular function f on a lattice A is a sublattice of A.*

Proof. Let $a, b \in A^*$. Denote $p = f(a) = f(b)$, $q = f(a \wedge b)$, and $r = f(a \vee b)$. Maximality of p implies $p \geq q$ and $p \geq r$. The supermodularity condition $q + r \geq 2p$ yields $p = q = r$. ■

If f is strictly supermodular then A^* is even a chain [Top78, theorem 4.2].

The maximal nodes and edges of a max-sum problem with minimal height form a CSP with a non-empty kernel. By theorem 10, for a supermodular max-sum problem this kernel is a lattice problem. By theorem 8, the kernel is satisfiable. Thus, the max-sum problem has a trivial equivalent and an optimal labeling can be obtained by taking the lowest label separately in each object of the kernel.

Remark. The result of this section could be proved in a more elementary way without lattice theory; in fact, everything necessary is in figure 11. However, the abstract lattice-theoretical language we have used is useful to understand the relation with supermodular optimization [Top78, Top98, GLS81, GLS88, Sch00, IFF01] and with work by Kovtun [Kov03, Kov04]. Appendix B might help readers not familiar with lattice theory.

Rather than by LP relaxation (e.g., by max-sum diffusion or TRW-S), supermodular problems can be more efficiently solved by translation to max-flow. This is given by D. Schlesinger [Sch05a] for any number $|X|$ of labels and by Kolmogorov and Zabih [KZ02] for $|X| = 2$.

10 Experiments with Structural Image Analysis

Here, we present examples with structural image analysis, motivated by those in [Sch76b, Sch89]. They are different from non-supermodular problems of Pott's type and arising from stereo reconstruction, experimentally examined by Kolmogorov and Wainwright [KW05b, Kol05a, Kol05b, KW05a], in the fact that a lot of edge qualities are $-\infty$; in that, they are closer to CSP. In the sense of [Sch76b, Sch89], these tasks can be interpreted as finding a 'nearest' image belonging to a language generated by a 2D grammar (in full generality, 2D grammars include also hidden variables). If qualities are viewed as log-likelihoods, the task corresponds to finding a mode of a Gibbs distribution with non-negative potentials.

Let the following be given. Let G be a grid graph, representing the topology of the 4-connected image grid. Each pixel $t \in T$ has a label from $X = \{\mathsf{E}, \mathsf{I}, \mathsf{T}, \mathsf{L}, \mathsf{R}\}$. Numbers $g_{tt'}(x, x')$ are given by figure 12c, which shows three pixels forming one horizontal and one vertical pair, as follows: the black edges have quality 0, the red edges $-\frac{1}{2}$, and the edges not shown $-\infty$. The functions $g_{tt'}(\bullet, \bullet)$ for all vertical pairs are equal, as well as for all horizontal pairs.

Numbers $f(\mathsf{E}) = f(\mathsf{I}) = 1$ and $f(\mathsf{T}) = f(\mathsf{L}) = f(\mathsf{R}) = 0$ assign an intensity to each label. Thus, $\mathbf{f}(\mathbf{x}) = (\, f(x_t) \mid t \in T \,)$ is the black-and-white image corresponding to the labeling $\mathbf{x}$.

First, assume that $g_t(x) = 0$ for all t and x. The set $I = \{\, \mathbf{f}(\mathbf{x}) \mid F(\mathbf{x} \,|\, \mathbf{g}) > -\infty \,\}$ contains images feasible to the 2D grammar $(G, X, \mathbf{g})$, here images of multiple non-overlapping black 'free-form' characters 'Π' on white background. An example of such an image with labels denoted is in figure 12d. The number of characters in the image is $-F(\mathbf{x} \,|\, \mathbf{g})$.

Let an input image $(\, f_t \mid t \in T \,)$ be given. Numbers $g_t(x) = -c\,[f_t - f(x)]^2$ quantify similarity between the input image and the intensities of the labels; we set $c = \frac{1}{6}$. Now, an optimal labeling describes an image nearest to I, in the defined sense. Setting the red edges in figure 12c to a non-zero value discourages images with a large number of small characters, which is useful when highly corrupted input images are analyzed; this could be viewed as a regularization.

For the input in figure 12d, we minimized the height of the max-sum problem $(G, X, \mathbf{g})$ by the augmenting DAG algorithm and then computed the kernel of the maximal nodes and edges. To get a partial and suboptimal solution to the CSP, we used the unique label condition from section 3. The result is in figure 12e. A pixel t with a unique maximal node (t, x) has the gray level $f(x)$, while pixels with multiple maximal nodes are red. Unfortunately, there are rather many ambiguous pixels.

It turns out that if X and $\mathbf{g}$ are slightly re-defined by adding two more labels as shown in figure 12f and 12g, a unique label in each pixel is obtained. We observed this repeatedly: of several

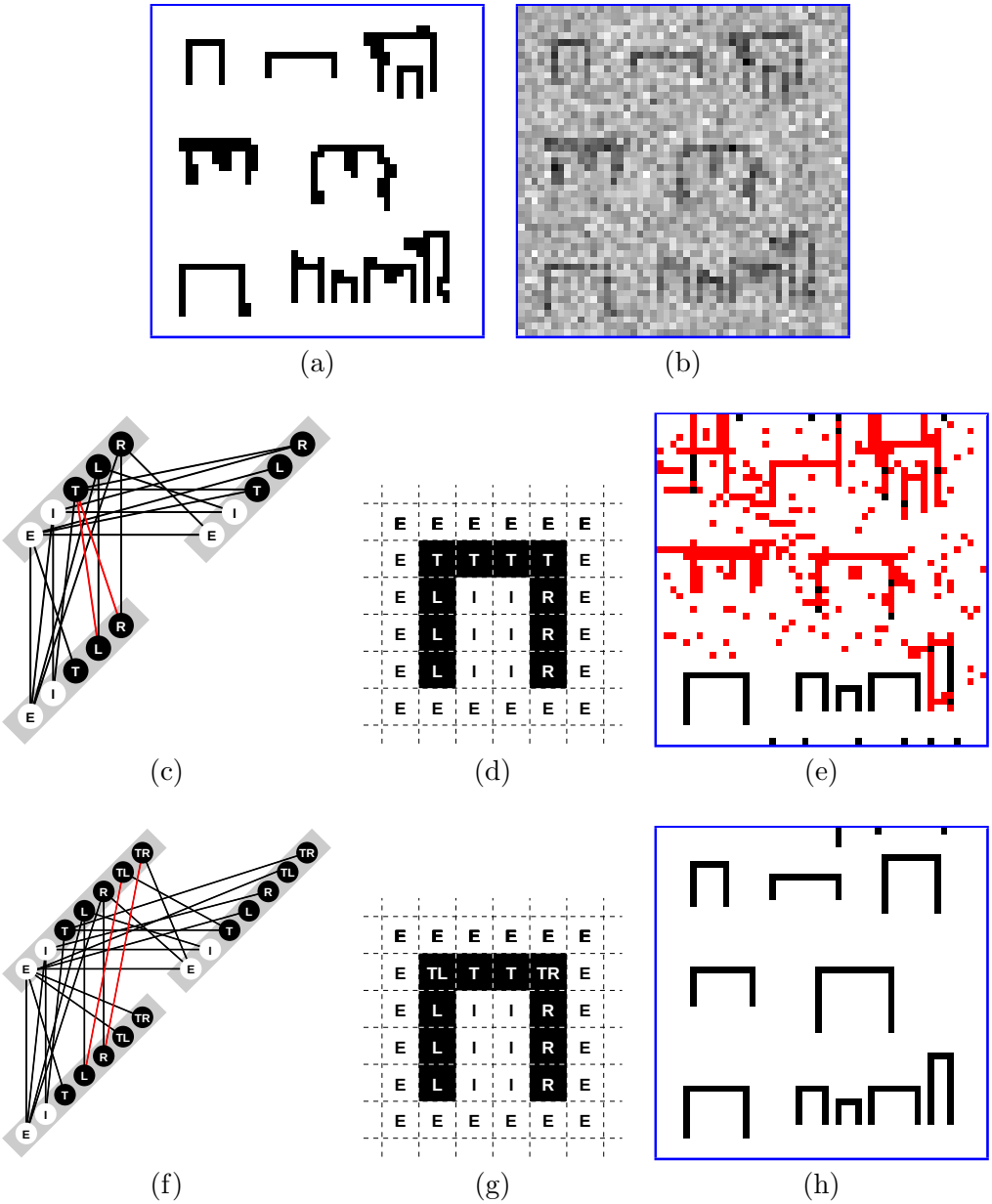
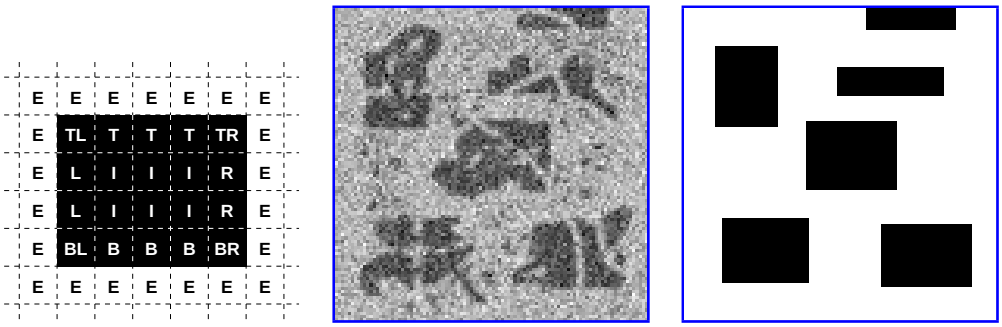
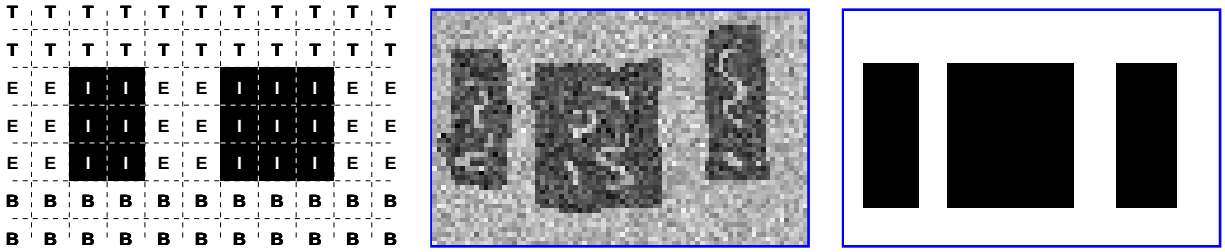


Figure 12: The 'Letters Π' example. The input image in (b) is the image in (a) plus independent normal noise. (c) The vertical and horizontal pixel pair defining the problem, (d) a labeled image feasible to this definition, and (e) the output image. (f) The alternative 'better' definition of the same problem, (g) a labeled feasible image, and (h) the output.



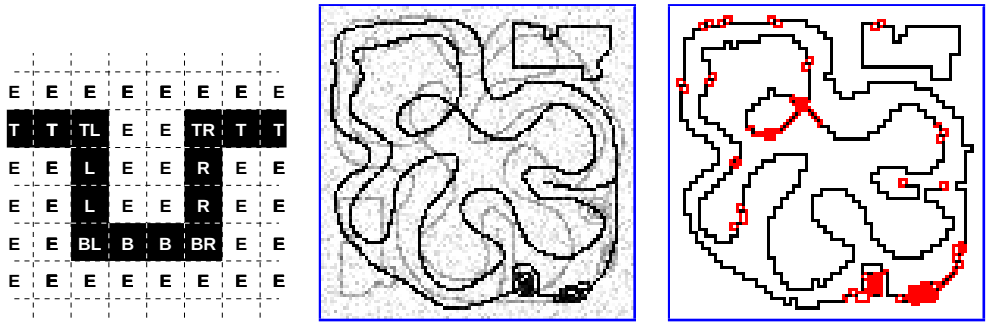
(a) (b) (c)

Figure 13: 'Rectangles'. (a) Description, (b) input, (c) output. Image size 100×100 pixels. Images with many small rectangles were discouraged by penalizing rectangle corners by quality 30. Again, a simpler definition with only 4 labels is possible, yielding ambiguous labels more often.



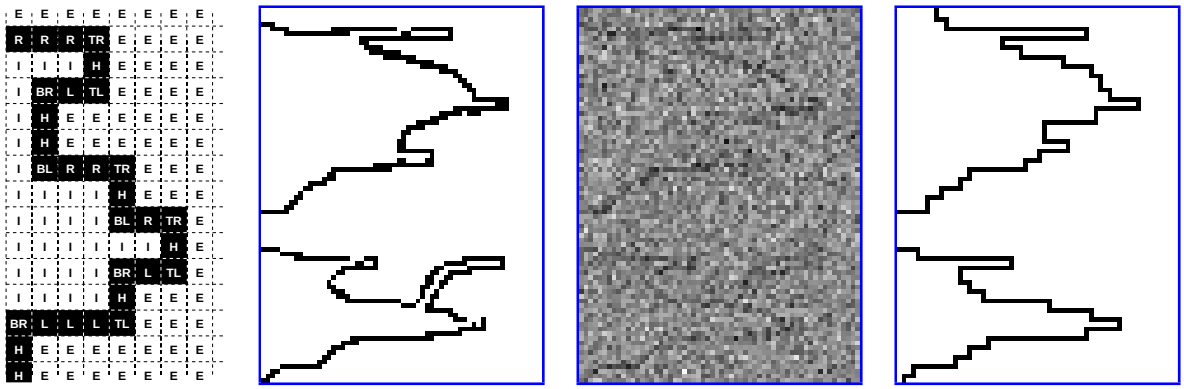
(a) (b) (c)

Figure 14: 'Aligned Rectangles'. (a) Description, (b) input, (c) output. Image size 50×80.



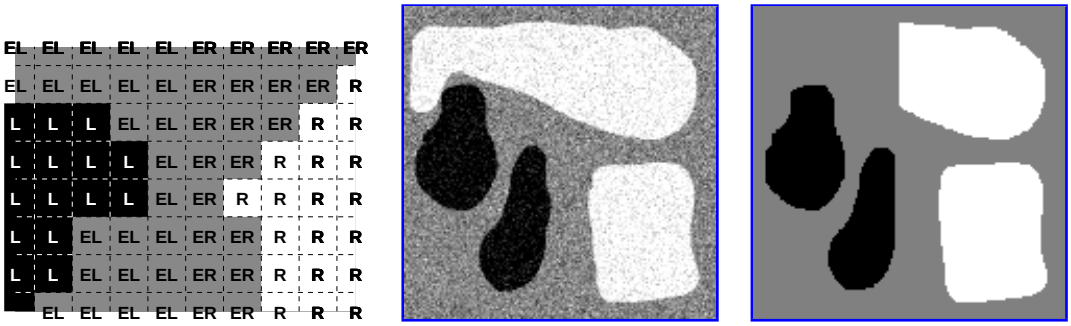
(a) (b) (c)

Figure 15: '4-connected Curve'. (a) Description, (b) input, (c) output. Image size 100×100.



(a) (b) (c) (d)

Figure 16: 'Spikes'. (a) Description. The input in (c) is the image in (b) plus independent normal noise. (c) Output. Image size 80×60.



(a) (b) (c)

Figure 17: 'Black Blobs on the Left, White Blobs on the Right'. (a) Description, (b) input, (c) output. Image size 150×150. Transitions black-gray and white-gray penalized by 0.5.

alternative formulations defining the same feasible set I, some (usually not the simplest ones) provide less ambiguous relaxed labelings more often.

The size of the input image is 50×50 pixels. For the problem in figure 12c, the runtime of the augmenting DAG algorithm was about 1.6 s on a 1.2 GHz laptop PC, and the max-sum diffusion achieved the state with arc consistent maximal nodes and edges in almost 6 minutes. For the problem in figure 12f, the augmenting DAG algorithm took 0.3 s and the diffusion 17 s.

Figures 13 through 17 present more examples of structural image analysis tasks. The models are fully defined by the figures and their captions. For each example, the pairs of neighboring labels shown in subfigure (a) have quality 0 unless stated otherwise in the caption, and the remaining pairs have quality $-\infty$. It is $g_t(x) = -[f_t - f(x)]^2$, where $f(x)$ are given by the label colors.

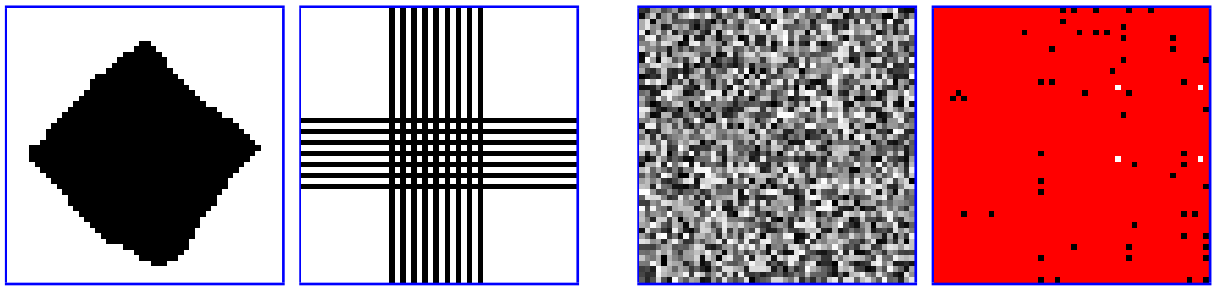
The used input images were not chosen very carefuly to achieve a unique labeling, rather, pixels with ambiguous labels were absent or rare for many inputs similar to the shown ones. An exception is the 'Curve' example, in which the shown output is typical. All images are synthetic, using hand drawn pictures and independent normal noise.

10.1 'Easy' and 'Difficult' Problems

We observed quite consistently that quantitative characteristics, like the numbers of pixels and labels or thhe initial value of $U(\mathbf{g})$, did not predict well the runtime and number of uniquely labeled pixels. Sometimes, these were low for tasks that a human would describe as 'easy' and high for 'difficult' ones. In particular, this was true for inputs 'similar' and 'dissimilar' to the feasible set I. Subfigures (a) and (c) show the input images, both far from a feasible one. For the former, the unique labeling was found but in a long runtime 4.4 s. For the latter, almost all pixels have ambiguous labels, found in runtime 8.3 s.

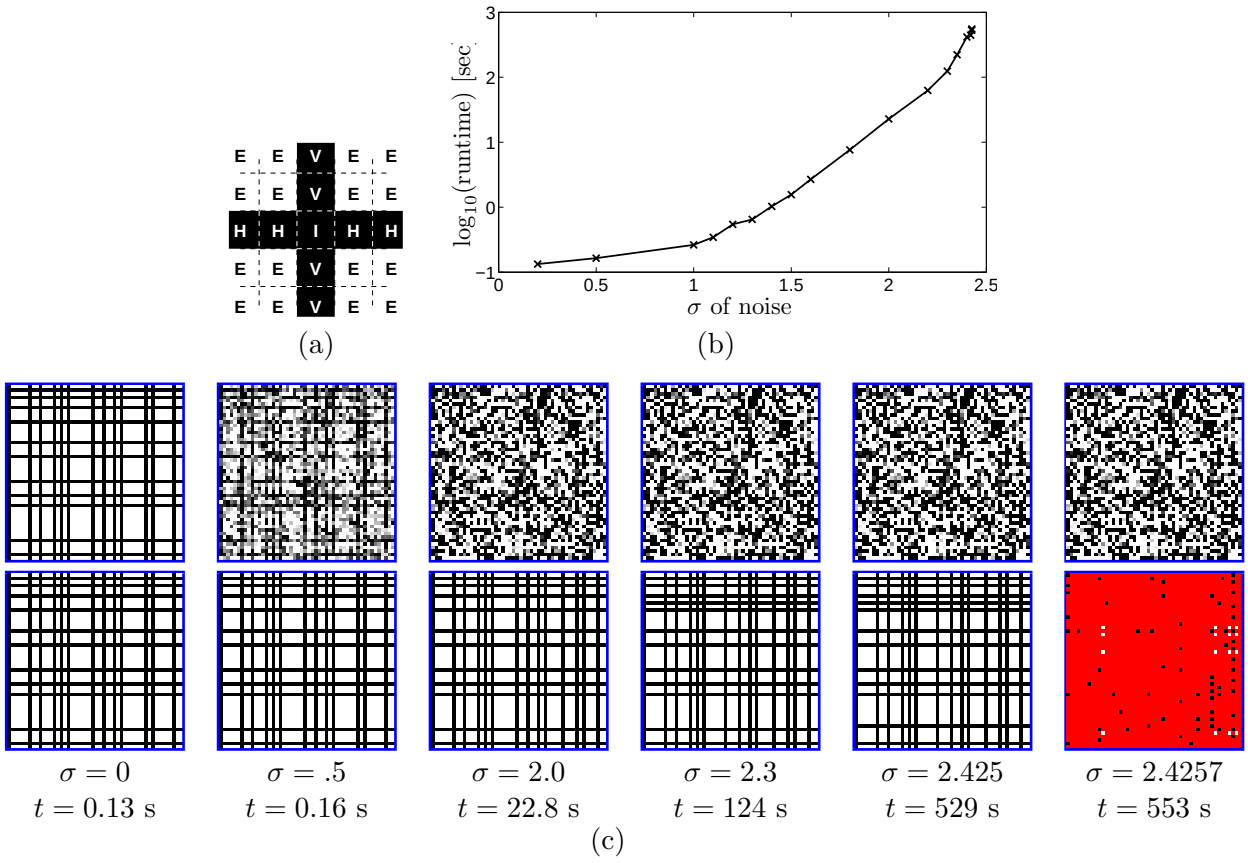
In fact, one can even observe *phase transition*, known to appear in many NP-hard problems including CSP [CKT91, Smi96]. Consider functions $g_{tt'}(x, x')$ defined by figure 19a, for which the feasible observed images $f(\mathbf{x})$ are composed of horizontal and vertical lines. Let the input be a feasible image, shown in the top left subfigure (c), with independent normal noise $N(0, \sigma)$ superimposed. For $\sigma = 0$ there is nothing to solve; the output is the bottom left subfigure (c). As σ increases[3], finding the feasible image nearest to the input becomes more and more 'difficult', which is reflected by *significantly* greater running times t of the augmenting DAG algorithm (the max-sum diffusion behaves similarly but takes longer). A solutions with unique labels in all pixels is found till $\sigma = 2.4250$. Then a tiny further increase of σ results in an abrupt change, when almost all pixels are ambiguous, as shown in the last subfigure (c). Beyond this point, the runtimes start decreasing (not shown in the figure). See [LG04], which mentions this phase transition in LP relaxation integrality for satisfactions problems.

[3]To obtain noise realizations superimposed to the input images in figure 19, we used a single realization of $N(0, 1)$ multiplied with different σ.



(a) (b) (c) (d)

Figure 18: 'Horizontal and Vertical Lines', examples of input far from the model. Image sizes 50×50. The input far from the model in (a) still resulted in an integer relaxed labeling in the output (b); runtime of the augmenting DAG algorithm was 4.4 s. Pure independent normal noise in (c) resulted in most pixels with non-integer relaxed labeling in (d); runtime 8.3 s.



(a) (b) (c)

Figure 19: 'Horizontal and Vertical Lines'. (a) Description. (b) Runtime of the augmenting DAG algorithm as a function of noise magnitude added to the top-left subfigure. (c) Several examples of input and output. In the top row of (c), the intensities were scaled to the white-black range.

11 Summary

We have reviewed the approach to max-sum problem developed by Schlesinger *et al.* in a unified framework. The approach can be summarized as follows. The original problem is formulated as a 0-1 linear programming, in which the integrality constraint is relaxed by introducing the polytope $\Lambda_{G,X}$ of relaxed labelings $\boldsymbol{\alpha}$. This leads to a LP task, which maximizes $\langle \mathbf{g}, \boldsymbol{\alpha} \rangle$ over $\Lambda_{G,X}$. This is common in optimization; however, from this alone it is not clear how to test for optimality, i.e., (dis)prove existence of an integer LP-optimal $\boldsymbol{\alpha}$, and how to design efficient algorithms.

Problem height $U(\mathbf{g})$ is an upper bound on quality $F(\mathbf{x} \,|\, \mathbf{g})$. This upper bound is tight for trivial max-sum problems, for which the CSP formed by the nodes maximal in objects and the edges maximal in pairs is satisfiable. Since CSP is NP-complete, testing whether the upper bound is tight is so too (we can easily translate any test for triviality to a CSP and *vice versa*). The upper bound is minimized by finding an equivalent problem with minimal height $U(\mathbf{g})$. After choosing a parameterization of equivalence classes by $\varphi_{tt'}(x)$, this can be formulated as a linear program, which turns out to be dual to maximizing of $\langle \mathbf{g}, \boldsymbol{\alpha} \rangle$ over $\Lambda_{G,X}$.

Thus, the max-sum problem and CSP are intimately related. For the original non-relaxed problem, this relationship is given by theorem 1. Note that this theorem is so simple that it can be proved by elementary means without referring to LP duality theorems. For the relaxed problem, the relationship is given by complementary slackness, which naturally leads to LP-relaxation of CSP. Here, the solution set of this relaxed CSP equals the subdifferential of the minimal problem height as a function of $\mathbf{g}$.

A CSP is relaxed-satisfiable is there is any (integer or non-integer) $\boldsymbol{\alpha}$ on given nodes and edges. Deleting nodes and edges violating arc consistency, i.e. taking the kernel, leaves the solutions set of a (relaxed) CSP unchanged. Thus, non-empty kernel is necessary for relaxed satisfiability, which is in turn necessary for satisfiability. Unfortunately, neither of these conditions is sufficient, the counter-examples being figures 3c and 6b,c. Exceptions are problems with two labels, for which non-empty kernel equals relaxed satisfiability, and supermodular max-sum problems (lattice CSPs), for which non-empty kernel equals satisfiability. For general problems, a unique label in each object in the kernel is sufficient (but not necessary) for satisfiability.

The fact that non-empty kernel is necessary for relaxed satisfiability, hence for minimal upper bound, can be used to design algorithms for decreasing the upper bound. We reviewed two examples of such algorithms. The first one, the max-sum diffusion, can be viewed as a simple convergent form of belief propagation. The second one, the augmenting DAG algorithm, is distantly similar to the well-known augmenting path algorithm for max-flow. We have not focused primarily on efficiency of these algorithms, in particular we have not made comparison with the max-marginal averaging by Kolmogorov [Kol05a].

In fact, one could imagine even more height minimizing algorithms based on arc consistency. Unfortunately, all such algorithms are not guaranteed to find the minimum of the upper bound because they can terminate in a relaxed-unsatisfiable state with a non-empty kernel. An efficient algorithm to escape from such spurious minima is not known. We believe that these spurious minima are equivalent to those observed by Kolmogorov [Kol04, Kol05a], but we have not been able to prove it.

We applied the presented LP-relaxation approach to the max-sum problem to several tasks of structural image analysis. To our knowledge, this kind of complex image analysis tasks has not been addressed by others.

The class of max-sum problems that can be solved to optimality by the presented approach is given by two conditions: the max-sum problem must have a trivial equivalent and testing for triviality must be tractable. Two known subclasses satisfying this are problems on trees and supermodular problems. Since this article is mainly theoretical, our experiments are qualitative rather than quantitative. However, they suggest that the class we can solve is considerably larger, containing many complex problems far from tree and supermodular ones.

It is natural to use the LP-relaxation approach to find suboptimal solutions to problems out of

the above class. For this, it would be useful to have an idea of how far we are from the optimum. We are not aware of any such results. Several quantities measuring distance from the optimum can be considered. The first is the integrality gap, $\langle \mathbf{g}, \boldsymbol{\alpha}^* \rangle - F(\mathbf{x}^* \,|\, \mathbf{g})$ where $\boldsymbol{\alpha}^*$ is an LP-optimal relaxed labeling and $\mathbf{x}^*$ is a true optimal solution to the non-relaxed problem. The second is $F(\mathbf{x}^* \,|\, \mathbf{g}) - F(\mathbf{x} \,|\, \mathbf{g})$ where $\mathbf{x}$ is a suboptimal labeling obtained from $\boldsymbol{\alpha}^*$ by a suitable approximation scheme. The latter is useless if some edge qualities $g_{tt'}(x, x')$ are $-\infty$ because finding $\mathbf{x}$ such that $F(\mathbf{x} \,|\, \mathbf{g}) > -\infty$ is equivalent to (NP-complete) CSP. To alleviate this, one could consider other ways of assessing suboptimality, as counting objects with wrong labels.

A Linear Programming Duality

This section summarizes what we need from linear programming duality.

Let $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$ and $\mathbf{c} \in \mathbb{R}^n$. Consider the pair of dual linear programs

$$\langle \mathbf{c}, \mathbf{x} \rangle \to \max \quad \langle \mathbf{y}, \mathbf{b} \rangle \to \min \tag{25a}$$

$$\mathbf{A}\mathbf{x} = \mathbf{b} \qquad \mathbf{y} \in \mathbb{R}^m \tag{25b}$$

$$\mathbf{x} \geq 0 \qquad \mathbf{y}^\top \mathbf{A} \geq \mathbf{c}^\top \tag{25c}$$

We will call the left program primal and the right program dual.

We use the following well-known LP duality theorems. *Weak duality* says that any feasible vectors $\mathbf{x}$ and $\mathbf{y}$ satisfy $\langle \mathbf{c}, \mathbf{x} \rangle \leq \langle \mathbf{y}, \mathbf{b} \rangle$. *Strong duality* says that feasible $\mathbf{x}$ and $\mathbf{y}$ are optimal if and only if they satisfy $\langle \mathbf{c}, \mathbf{x} \rangle = \langle \mathbf{y}, \mathbf{b} \rangle$. *Complementary slackness* says that feasible $\mathbf{x}$ and $\mathbf{y}$ are optimal if and only if $\langle \mathbf{y}^\top \mathbf{A} - \mathbf{c}^\top, \mathbf{x} \rangle = 0$. If one program is feasible then it is bounded if and only if the other is feasible.

The set of primal optimal vectors is a convex polyhedron, denoted by $X \subseteq \mathbb{R}^n$. It is known [BT97] that X can be characterized as follows.

Theorem 11 *Let $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$. The function $f\colon \mathbb{R}^n \to \mathbb{R}$ given by*

$$f(\mathbf{c}) = \max\{\, \langle \mathbf{c}, \mathbf{x} \rangle \mid \mathbf{x} \in \mathbb{R}^n,\, \mathbf{A}\mathbf{x} = \mathbf{b},\, \mathbf{x} \geq \mathbf{0} \,\} \tag{26}$$

is convex, and $\langle \mathbf{c}, \mathbf{x} \rangle = f(\mathbf{c})$ if and only if $\mathbf{x}$ is a subgradient of f at $\mathbf{c}$.

Recall that $\mathbf{x} \in \mathbb{R}^n$ is a subgradient of a convex function $f\colon \mathbb{R}^n \to \mathbb{R}$ at point $\mathbf{c}$ if

$$f(\mathbf{d}) \geq f(\mathbf{c}) + \langle \mathbf{d} - \mathbf{c}, \mathbf{x} \rangle \tag{27}$$

for every $\mathbf{d} \in \mathbb{R}^n$. Subgradient is a generalization of gradient for convex non-differentiable functions; if f is differentiable at $\mathbf{c}$ subgradient reduces to gradient, $\mathbf{x} = \nabla f(\mathbf{c})$. The set of all subgradients at $\mathbf{c}$ is the subdifferential $\partial f(\mathbf{c})$. The theorem says that $X = \partial f(\mathbf{c})$, where f is the optimal value of the linear program (25) as a function of $\mathbf{c}$.

Choosing $\mathbf{d} = \mathbf{c} + \varepsilon \mathbf{e}_i$, where $0 \neq \varepsilon \in \mathbb{R}$ and $\mathbf{e}_i$ is the i-th vector of the standard basis of $\mathbb{R}^n$, yields for any $\mathbf{x} \in X$

$$x_i = \langle \mathbf{e}_i, \mathbf{x} \rangle \leq \frac{f(\mathbf{c} + \varepsilon \mathbf{e}_i) - f(\mathbf{c})}{\varepsilon}. \tag{28}$$

Choosing ε first positive and then negative restricts x_i to lie in an interval,

$$x_i^- \leq x_i \leq x_i^+. \tag{29}$$

The smallest interval is obtained by taking the limits for $\varepsilon \to 0+$ and $\varepsilon \to 0-$. Then x_i^- (x_i^+) is the partial derivative of $f(\mathbf{c})$ along c_i from the left (right). In this case, the interval $[x_i^-, x_i^+]$ can be shown to be the projection of X onto $\mathbf{e}_i$,

$$x_i^- = \min_{\mathbf{x} \in X} x_i, \quad x_i^+ = \max_{\mathbf{x} \in X} x_i, \tag{30}$$

the smallest bounding box of X thus being $[x_1^-, x_1^+] \times \cdots \times [x_n^-, x_n^+]$.

In fact, this is a generalization of the shadow price interpretation of variables $\mathbf{x}$ in the LP dual pair (25), well-known in LP duality. This interpretation is usually stated for cases when the primal solution set X has a single element, i.e., $x_i^- = x_i^+$ for all i. In our case, X can have more elements, which yields an interval $[x_i^-, x_i^+]$ rather than a unique value.

B Posets, Lattices and Supermodularity

This section overviews what we need from partially ordered sets and lattice theory. For more, see [Top78, Top98] and some textbook, e.g. [DP90].

A binary relation $\leq$ on a set $A \neq \emptyset$ is a **partial order** if it is reflexive, antisymmetric and transitive. The pair $(A, \leq)$ is a **partially ordered set (poset)**. For $a, b \in A$, it can be either $a \leq b$, or $a \geq b$, or a and b can be incomparable. If every two elements are comparable, $(A, \leq)$ is **totally (or linearly) ordered** and also called a **chain**.

The (Cartesian, direct) **product of posets** $(A, \leq_A) \times (B, \leq_B)$ is the poset $(A \times B, \leq)$ where the new order $\leq$ is given componentwise, $(a, b) \leq (a', b')$ if and only if $a \leq_A a'$ and $b \leq_B b'$. The power of a poset is $(A, \leq)^n = \prod_{i=1}^n (A, \leq) = (A^n, \leq)$, where $(a_1, \ldots, a_n) \leq (b_1, \ldots, b_n)$ if and only if $a_i \leq b_i$ for all i, the new order being denoted by the same symbol $\leq$.

The **meet** $\bigwedge B$ (**join** $\bigvee B$) of a set $B \subseteq A$ is the greatest lower (least upper) bound of B with respect to $\leq$. For a two-element B we use the infix notation, $\bigwedge\{a, b\} = a \wedge b$.

A poset $(A, \leq)$ is a **lattice** if the (unique) meets and joins of all pairs of elements from A exist. If only the meets (joins) exist, it is a meet (join) **semilattice**. A lattice $(A, \leq)$ is **complete** if the meets and joins of all subsets of A exist. Every finite lattice is complete. The **least (greatest) element** of a complete lattice $(A, \leq)$ is $\bigwedge A$ ($\bigvee A$). A lattice $(A, \leq)$ is **distributive** if $a, b, c \in A$ satisfy $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$, which can be shown equivalent to $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$. A lattice $(B, \leq)$ is a **sublattice** of a lattice $(A, \leq)$ if $B \subseteq A$ and the ordering of B is the restriction of that of A.

A function $f\colon A \to \mathbb{R}$ on a lattice $(A, \leq)$ is **submodular** if all $a, b \in A$ satisfy

$$f(a \wedge b) + f(a \vee b) \leq f(a) + f(b). \tag{31}$$

It is **strictly submodular** if the inequality is strict for $a \neq b$. It is **supermodular** if $-f$ is submodular. It is **modular** if it is both sub- and supermodular. It is easy to see that a sum of (sub-, super-) modular functions is also (sub-, super-) modular.

Example. The poset $(2^U, \subseteq)$ with a non-empty set U is a complete lattice, with meet and join being the set-theoretic intersection and union. If U is finite then the function $f(X) = |X|$ defined for any $X \subseteq U$ is modular. This is the well-known formula $|X \cup Y| = |X| + |Y| - |X \cap Y|$. For a convex g, the function $f(X) = g(|X|)$ is supermodular. ■

A special case of a distributive lattice is a product of chains. Here, meet (join) is just componentwise minimum (maximum). A function $f(x_1, \ldots, x_n)$ of n variables $x_i \in A$ can be seen as a function on the product $(A^n, \leq)$ of chains.

A multivariate function is **separable** if it is a sum of univariate functions. A function on a product of chains is modular if and only if it is separable [Top78, theorem 3.3]. Since separability does not refer to any order, it follows that a function modular for some order is modular for any order.

Every univariate function is modular. A bivariate function f is submodular if and only if every $x \leq y$ and $x' \leq y'$ satisfy

$$f(x, x') + f(y, y') \leq f(x, y') + f(y, x'). \tag{32}$$

For finite A, submodularity of $f(x_1, \ldots, x_n)$ can be stated in terms of the mixed second differences, $\Delta_{x_i} \Delta_{x_j} f \leq 0$ for every $i \neq j$. (Super-) submodular functions on a product of two finite chains are also called *(inverse) Monge matrices* (see e.g. [BKR96]).

C The Parameterization of Zero Max-sum Problems

Theorem 12 *$(G, X, \mathbf{g})$ is a zero max-sum problem if and only if there are numbers $\varphi_{tt'}(x)$ and φ_t such that*

$$g_t(x) = \varphi_t + \sum_{t' \in N_t} \varphi_{tt'}(x), \qquad t \in T,\ x \in X, \tag{33a}$$

$$g_{tt'}(x, x') = -\varphi_{tt'}(x) - \varphi_{t't}(x'), \quad \{t, t'\} \in E,\ x, x' \in X, \tag{33b}$$

$$\sum_t \varphi_t = 0. \tag{33c}$$

Proof. The *if* implication is straightforward, by substituting (33) to (5) and verifying that (5) identically vanishes. We will prove the *only if* implication.

Since $(G, X, \mathbf{g})$ is a zero problem, its quality function $F(\bullet \,|\, \mathbf{g})$ is separable. Since by theorem 9 also functions $g_{tt'}(\bullet, \bullet)$ are separable, (33b) follows. We can choose the univariate functions e.g. as $\varphi_{tt'}(x) = g_{tt'}(x, y^*)$ and $\varphi_{t't}(y) = g_{tt'}(x^*, y) - g_{tt'}(x^*, y^*)$, where $x^*, y^* \in X$ are arbitrary constants.

Let $\mathbf{x}$ and $\mathbf{y}$ be two labelings that differ only in an object t where they satisfy $x_t = x$ and $y_t = y$. After substituting (5) and (33b) to the equality $F(\mathbf{x} \,|\, \mathbf{g}) = F(\mathbf{y} \,|\, \mathbf{g})$, all terms cancel out except

$$g_t(x) - \sum_{t' \in N_t} \varphi_{tt'}(x) = g_t(y) - \sum_{t' \in N_t} \varphi_{tt'}(y).$$

Since this holds for any $x, y \in X$, neither side depends on x. Denoting the right-hand side by φ_t we obtain (33a). Substituting (33a) and (33b) to the equality $F(\bullet \,|\, \mathbf{g}) = 0$ yields (33c). ■

Theorem 13 *Let G be connected. $(G, X, \mathbf{g})$ is a zero max-sum problem if and only if there are numbers $\varphi_{tt'}(x)$ such that*

$$g_t(x) = \sum_{t' \in N_t} \varphi_{tt'}(x), \qquad t \in T,\ x \in X, \tag{34a}$$

$$g_{tt'}(x, x') = -\varphi_{tt'}(x) - \varphi_{t't}(x'), \quad \{t, t'\} \in E,\ x, x' \in X. \tag{34b}$$

Proof. As before, the *if* part is easy by substitution. We will prove the *only if* part.

By theorem 12, there are $\varphi_{tt'}(x)$ and φ_t such that (33) holds. We will give an algorithm that does an equivalent transformation after which $\varphi_t = 0$ for all t.

Let G' be a spanning tree of G. It exists because G is connected. Find a pair $\{t, t'\}$ in G' such that t is a leaf. Do the following equivalent transformation of the problem $(G, X, \mathbf{g})$:

$$\begin{aligned} \varphi_{tt'}(x) &\mathrel{+}= \varphi_t, \quad x \in X, \\ \varphi_{t't}(x') &\mathrel{-}= \varphi_t, \quad x' \in X, \\ \varphi_{t'} &\mathrel{+}= \varphi_t, \\ \varphi_t &:= 0. \end{aligned}$$

Remove the object t and the pair $\{t, t'\}$ from G'. Repeat until G' is empty. ■

The latter theorem says that if G is connected, the parameterization can be simplified from (33) to (34). As a counter-example for a disconnected G, assume the family of problems $(G, X, \mathbf{g})$ given by $T = \{1, 2\}$, $E = \emptyset$, $X = \{1\}$, and $\mathbf{g}$ being arbitrary satisfying $g_1(1) = -g_2(1)$. All these are zero problems but cannot be parameterized by (34).

D Hydraulic Models

Schlesinger and Kovalevsky [SK78] suggested electrical and mechanical models of the LP pair (16), using a suitable form of (10). This section presents an example. Some readers may welcome such models as useful heuristics. We do not use them in any formal argument.

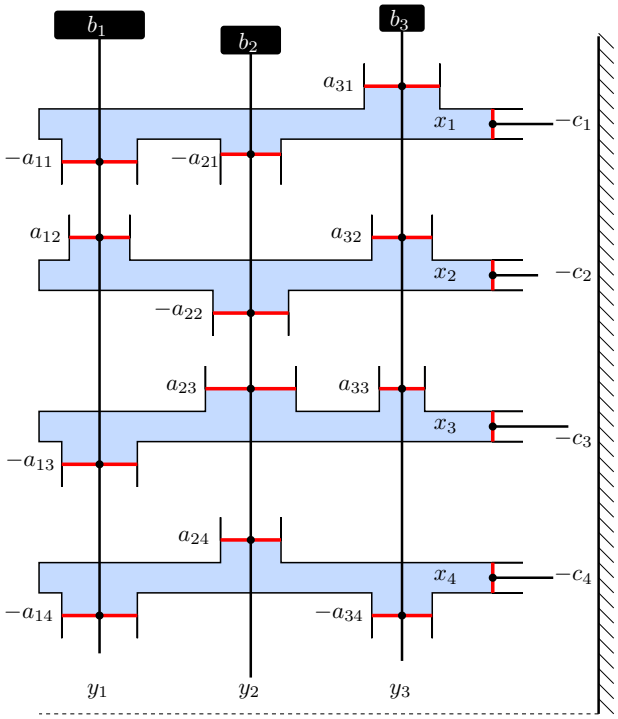


Figure 20: A hydraulic model of the linear program (25) for $m = 3$, $n = 4$ and $\mathbf{b} \geq \mathbf{0}$.

D.1 Linear Programming in General Form

Before presenting a model of the relaxed max-sum problem, to introduce the used components we will show a model of the general linear program (25) in which, without loss of generality, $\mathbf{b} \geq 0$. This model is a hydraulic modification of the electrical model in [SH02, chapter 2].

The analog computer in figure 20 consists of $n = 4$ tanks filled with incompressible liquid, each closed with $m = 3$ horizontal and one vertical pistons. The area of the vertical piston in column i and row j is $|a_{ij}|$; the pistons with $a_{ij} > 0$ aim upward and with $a_{ij} < 0$ downward. The horizontal pistons have unit area. The vertical pistons in the column i are rigidly linked by a solid rod, on the top of which a weight resides imposing force b_i downward. The distance of the lower tips of the i-th rod from a horizontal reference level is y_i. The distance of the tip of the j-th horizontal piston's rod from the vertical wall on the right is w_j where $\mathbf{w} = \mathbf{A}^\top \mathbf{y} - \mathbf{c}$. Constraints $\mathbf{w} \geq 0$ are *kinematic constraints*, saying that the horizontal rods cannot penetrate the wall. The pressure in tank j, which equals the *contact force* acting between the j-th horizontal rod and the wall, is x_j.

The solution of (25) corresponds to the minimal potential energy of the device.

D.2 Transportation Problem

Consider the following problem defined on a single pair $\{t, t'\} \in E$: given numbers $\alpha_t(x)$ and $\alpha_{t'}(x)$ satisfying $\sum_x \alpha_t(x) = \sum_x \alpha_{t'}(x) = 1$, find numbers $\alpha_{tt'}(x, x')$ maximizing the quality of the pair. This task is known as the *transportation problem*[4]. The transportation problem for the pair $\{t, t'\}$ means to compute primal variables $\alpha_{tt'}(x, x')$ and dual variables $\varphi_{tt'}(x)$ and $\varphi_{t't}(x)$ in the LP dual pair

$$\sum_{x,x'} g_{tt'}(x, x')\, \alpha_{tt'}(x, x') \to \max \qquad \sum_{x} \Big[\varphi_{tt'}(x)\, \alpha_t(x) + \alpha_{t'}(x)\, \varphi_{t't}(x)\Big] \to \min \tag{35a}$$

$$\sum_{x'} \alpha_{tt'}(x, x') = \alpha_t(x) \qquad \varphi_{tt'}(x) \in \mathbb{R}, \qquad x \in X \tag{35b}$$

$$\alpha_{tt'}(x, x') \geq 0 \qquad \varphi_{tt'}(x) + \varphi_{t't}(x') \geq g_{tt'}(x, x'), \quad x, x' \in X \tag{35c}$$

[4]The relation of max-sum problem and transportation problem was noted by Boris Flach.

For $X = \{1, 2, 3\}$, a model of this program is the mechanical device in figure 21a. It consists of six forks, three on the left and three on the right. The left (right) springs are pushed to the center by constant external forces $\alpha_t(x)$ ($\alpha_{t'}(x)$).

The forks can move horizontally, their distances from a reference vertical line being $\varphi_{tt'}(x)$ and $\varphi_{t't}(x)$. The gap between the tip of the x-th left fork and the tip of the x'-th right fork at the same vertical level is $g_{tt'}(x, x') - \varphi_{tt'}(x) - \varphi_{t't}(x')$. The two tips push each other with contact force $\alpha_{tt'}(x, x')$. Under the forces imposed by the springs, the forks move toward each other until some left tips meet some right tips, resulting in a global force equilibrium.

Primal constraints (35b) describe force equilibrium for each fork. The primal constraints (35c) say that the fork tips can only push to each other, never pull. The corresponding dual constraints (35c) say that the tips at the same level cannot penetrate each other.

Note, there is an ambiguity in determining $\varphi_{tt'}(x)$ and $\varphi_{t't}(x)$, since all forks can simultaneously move horizontally without affecting the optimum. This can be fixed e.g. by setting $\varphi_{tt'}(1) = 0$.

D.3 Relaxed Max-sum Problem

The hydraulic model of the relaxed max-sum problem is obtained by coupling transportation problems as follows. We keep all $u_{tt'} = 0$, i.e., we use (11a).

The dual constraint (16e) for the node (t, x) is modeled using a tank (its top view is in figure 21b) filled with liquid, closed with $|N_t|$ horizontal and one vertical pistons of unit area. The position of the horizontal piston leading to object $t' \in N_t$ is $\varphi_{tt'}(x)$. The height of the vertical piston is $\sum_{t' \in N_t} \varphi_{tt'}(x) + g_t(x)$, where $g_t(x)$ is the length of the vertical rod attached to the piston. The pressure in tank (t, x) is $\alpha_t(x)$.

A horizontal bar at height u_t is placed over the $|X|$ vertical pistons belonging to object t, preventing any piston go higher than u_t (side view in figure 21c). Each of the $|T|$ bars have weight 1, i.e., imposes force 1 aiming downward.

Each horizontal piston is connected to a fork from figure 21a; a single piston and fork correspond to an edge pencil (t, t', x). The whole device for the 3×3 grid graph G and $|X| = 3$ labels is shown in figure 21d. Its potential energy is $U(\mathbf{g}^{\boldsymbol{\varphi}})$ and the optimum corresponds to its minimum.

It is not difficult to design similar models for other forms of the height minimization (10). E.g. for (11d), rather than the individual weights on each object we place a single horizontal surface of the weight $|T|$ over the whole device.

Remark. It is interesting to see what non-minimality of problem height in figure 6c means in terms of the hydraulic model. Since all the tanks are interconnected via pencils containing only a single maximal edge, which are all pencils except $(d, c, 1)$, they can be replaced by a single tank and the model can be simplified to the device shown in figure 22. The tank of this device is closed by four pistons: one vertical, on which the weight resides, and three horizontal, corresponding to nodes $(c, 2)$, $(c, 1)$, and $(d, 1)$. The horizontal pistons are rigidly linked by fork $(d, c, 1)$. When the vertical piston is pressed down by the weight, the fork moves freely to the right.

E Implementation of the Augmenting DAG Algorithm

Here we give a detailed implementation of the augmenting DAG algorithm concisely described in section 8, done by the author of this text. The C++ implementation of the following code is available from `http://cmp.felk.cvut.cz/cmp/software/maxsum`.

Unike above, we allow objects to have different numbers of labels; the label set of object t is X_t. Further, the edges with quality $-\infty$ are not treated like the other edges, instead, they are never visited at all. We denote by

$$X_{tt'}(x) = \{\, x' \mid g_{tt'}(x, x') > -\infty \,\}$$

the set of finite edges in the pencil (t, t', x).

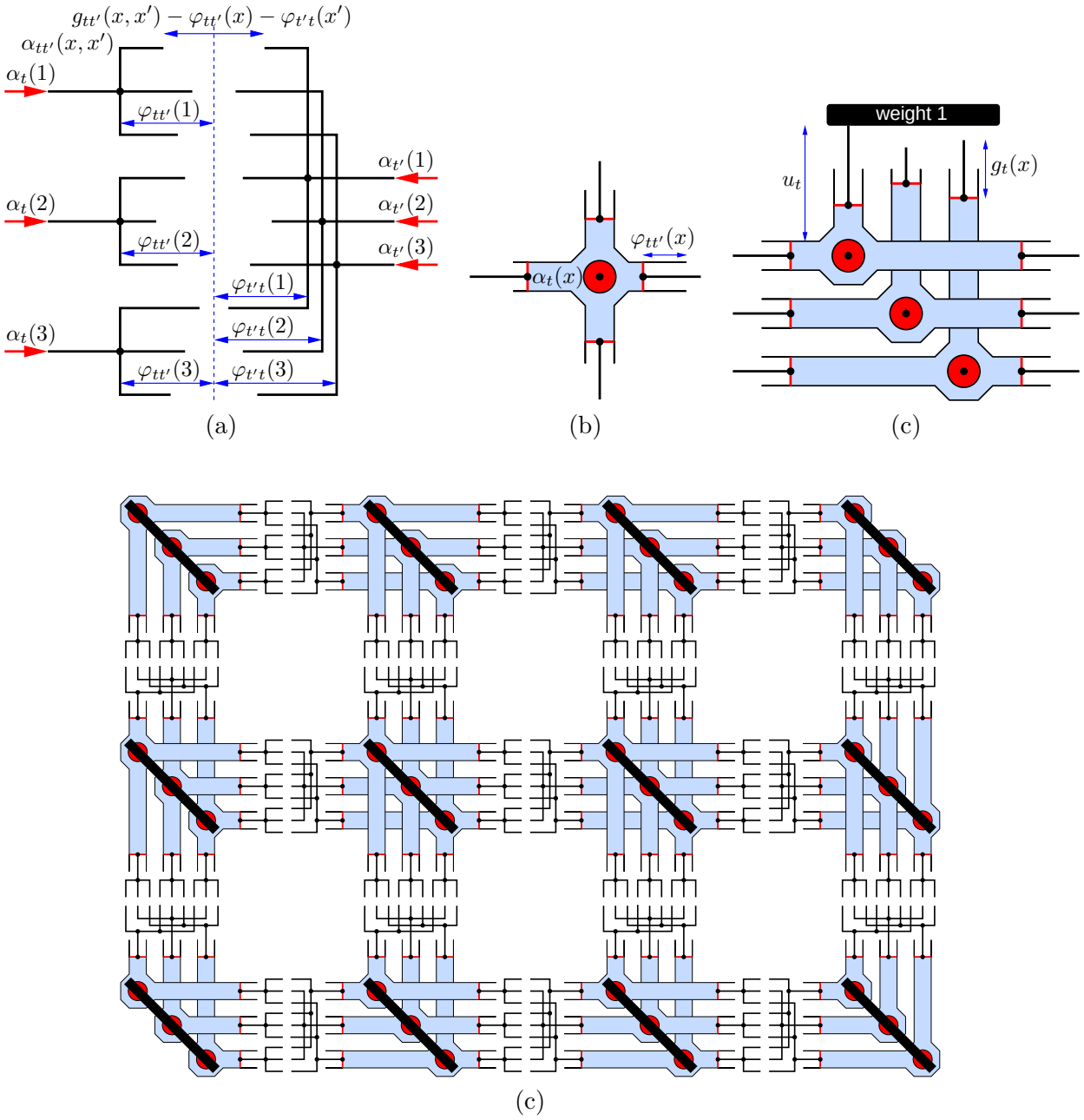


Figure 21: (a) The mechanical model of the transportation problem for $|X| = 3$ sources and destinations. (b) Top view of the tank (t, x). (c) Side view of the object t. (d) Top view of the hydraulic model of the relaxed max-sum problem, for 3×4 grid graph G and $|X| = 3$.

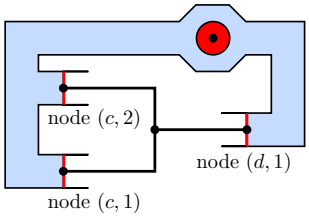


Figure 22: The 'reduced' hydraulic model of the max-sum problem in figure 6c.

Rather than floats, we use large integers to represent real-valued variables. This allows for treating rounding errors in a consistent manner.

To test for maximality of nodes and edges, thresholds $\varepsilon_t, \varepsilon_{tt'} \geq 0$ are assigned to them. The node (t, x) or the edge $\{(t, x), (t', x')\}$ is maximal if and only if respectively

$$\max_x g_t^{\boldsymbol{\varphi}}(x) - g_t^{\boldsymbol{\varphi}}(x) \leq \varepsilon_t,$$
$$-g_{tt'}^{\boldsymbol{\varphi}}(x, x') \leq \varepsilon_{tt'}.$$

The edge is feasible if and only if $g_{tt'}^{\boldsymbol{\varphi}}(x, x') \leq 0$. Initially, all ε_t and $\varepsilon_{tt'}$ have a common value.

Here is the pseudocode of the Augmenting DAG algorithm:

```
init;
while Q ≠ ∅ do
  relax;
  while n_{t*} = 0 do
    direction;
    step;
    if λ > 0 then
      repair;
    else
      threshold;
    end if
    update;
    resurrect;
  end while
end while
```

The called procedures will be specified in the following sections. In the code, all variables are global. The following notation will stand for 'loop for elements $a \in A$ satisfying property $\psi(a)$':

for $a \in A \mid \psi(a)$ **do** ... **end for**

On entering the algorithm, T, N_t, X_t, $X_{tt'}(x)$, $g_t(x)$, $g_{tt'}(x, x')$, ε_0, $\varphi_{tt'}(x)$ are given, and all edges are feasible. After the algorithm halts, the live nodes and edges form an arc consistent set.

E.1 Initialization

First, auxiliary variables are initialized. We leave their meaning to be explained later.

```
procedure init
ε_t := ε_0;  ε_{tt'} := ε_0;  Δφ_{tt'}(x) := 0;  d_t(x) := 0;  n_t := 0;  Q := ∅;
for t ∈ T do
  u_t := max_{x∈X_t} g_t^φ(x);
  for x ∈ X_t do
    if u_t − g_t^φ(x) ≤ ε_t then
      p_t(x) := ALIVE;  n_t++;
      enqueue(Q, (t, x));
    else
      p_t(x) := NONMAX;
    end if
  end for
end for
end procedure
```

E.2 Arc Consistency Algorithm

This procedure repeats the arc consistency iterations until either an object t^* is found with no node alive, or no more nodes can be deleted.

A set $Q \subseteq T \times X$ stores nodes to be tested for their possible deletion. More precisely, between the 2nd and the 3rd line of the procedure the nodes in Q satisfy

$$\begin{aligned} (t,x) \in Q \quad &\Rightarrow \quad p_t(x) = \texttt{ALIVE}, \\ (t,x) \notin Q,\ p_t(x) = \texttt{ALIVE} \quad &\Rightarrow \quad (\forall t' \in N_t)[\,(22) \text{ is satisfied}\,]. \end{aligned}$$

The access strategy to Q significantly affects the behavior of the algorithm. We chose to represent it by the queue. The operations 'enqueue' and 'dequeue' respectively add or take out an element. We will also need to test whether $(t,x) \in Q$, which is implemented using an auxiliary logical variable assigned to each node.

The variables $n_t = |\{\, x \mid p_t(x) = \texttt{ALIVE} \,\}|$ keep the number of live nodes in each object.

procedure relax
while $Q \neq \emptyset$ **do**
  $(t,x) := \text{dequeue}(Q)$;
  for $t' \in N_t$ **do**
    if $\neg[\exists x' \in X_{tt'}(x)][\,(-g^{\varphi}_{tt'}(x,x') \le \varepsilon_{tt'}) \wedge (p_{t'}(x') = \texttt{ALIVE})\,]$ **then**
      $p_t(x) := t'$;
      $n_t{-}{-}$;
      for $t' \in N_t$ **do**
        for $x' \in X_{tt'}(x) \mid (-g^{\varphi}_{tt'}(x,x') \le \varepsilon_{tt'}) \wedge (p_{t'}(x') = \texttt{ALIVE}) \wedge ((t',x') \notin Q)$ **do**
          enqueue$(Q,(t',x'))$;
        end for
      end for
      if $n_t = 0$ **then**
        $t^* := t$; **return**
      end if
      break
    end if
  end for
end while
end procedure

E.3 Finding Search Direction

Having an object t^* with no node alive, the procedure computes the search direction $\Delta\boldsymbol{\varphi}$ by traversing the augmenting DAG $D(t^*)$ in a linear order.

To traverse a DAG in a linear order, we use the obvious algorithm which repeats the following two steps: (i) visit a node to which no edge leads; (ii) remove this node and all edges leaving it. This is done until the empty graph is obtained.

The first part of the procedure stores the indegree of each node (t,x) of $D(t^*)$ to $d_t(x)$. We need an auxiliary node stack, S, accessed by the operations 'push' and 'pop'.

procedure direction
$S := \{\, (t^*,x) \mid x \in X_{t^*} \,\}$;
while $S \neq \emptyset$ **do**
  $(t,x) := \text{pop}(S)$;
  if $p_t(x) \neq \texttt{NONMAX}$ **then**
    $t' := p_t(x)$;
    for $x' \in X_{tt'}(x) \mid -g^{\varphi}_{tt'}(x,x') \le \varepsilon_{tt'}$ **do**
      if $d_{t'}(x') = 0$ **then** push$(S,(t',x'))$; **end if**
      $d_{t'}(x')$++;
    end for
  end if
end while

The second part of the procedure traverses $D(t^*)$ in a linear order. During that, $\Delta\varphi_{tt'}(x)$ are computed by propagating conditions (23), the variables $d_t(x)$ are reset to zero, and the node stack S_0 is filled with the nodes of $D(t^*)$ in the linear order. Stack S stores the nodes with $d_t(x) = 0$.

$S := \{\, (t^*, x) \mid x \in X_{t^*},\ d_{t^*}(x) = 0 \,\};\ \ S_0 := \emptyset;$
for $x \in X_{t^*} \mid p_{t^*}(x) \neq$ `NONMAX` **do** $\Delta\varphi_{t^* p_{t^*}(x)}(x) := -1;$ **end for**
while $S \neq \emptyset$ **do**
  $(t, x) := \mathrm{pop}(S);$
  $\mathrm{push}(S_0, (t, x));$
  if $p_t(x) \neq$ `NONMAX` **then**
    $t' := p_t(x);$
    $\Delta\varphi_{tt'}(x)\ {-}{=} \sum_{t'' \in N_t \setminus \{t'\}} \Delta\varphi_{tt''}(x);$
    for $x' \in X_{tt'}(x) \mid -g^{\boldsymbol{\varphi}}_{tt'}(x, x') \leq \varepsilon_{tt'}$ **do**
      $d_{t'}(x')--;$
      if $d_{t'}(x') = 0$ **then** $\mathrm{push}(S, (t', x'));$ **end if**
      $\Delta\varphi_{t't}(x') := \max\{\, \Delta\varphi_{t't}(x'),\ -\Delta\varphi_{tt'}(x) \,\};$
    end for
  end if
end while
end procedure

E.4 Finding Search Step

The search step λ is computed, by traversing $D(t^*)$ and updating λ to satisfy (24).

To fit λ to the used integer arithmetic, it is set to the nearest smaller integer $\lfloor \lambda \rfloor$ when computing (24). However, we have to recover from a possible underflow or overflow. In particular, it can happen that u_{t^*} cannot be decreased for one of the following reasons:

- $\lambda = 0$ because some of expressions (24) is smaller than 1.
- The numbers $\Delta\varphi_{tt'}(x)$, while computed in the procedure 'direction', overflow[5]. This might make condition (23c) violated and draw $\Delta\boldsymbol{\varphi}$ unusable.

In both cases, we give up on decreasing the height. Instead, we make at least one node of t^* alive by increasing a single ε_t or $\varepsilon_{tt'}$ in $D(t^*)$ by the smallest possible amount.

The following procedure computes λ and the minimal threshold ε. Returning $\lambda = 0$ indicates that u_{t^*} cannot be decreased due to one of the above two reasons. Then a node $(t_\varepsilon, x_\varepsilon)$ is returned determining the object or the object pair on which the threshold is to be increased as follows: if $(t_\varepsilon, x_\varepsilon)$ is maximal then $\varepsilon_{t_\varepsilon p_{t_\varepsilon}(x_\varepsilon)}$ is to be increased to ε; if $(t_\varepsilon, x_\varepsilon)$ is non-maximal then $\varepsilon_{t_\varepsilon}$ is to be increased to ε. If $\lambda > 0$ is returned, ε and $(t_\varepsilon, x_\varepsilon)$ are not used.

procedure step
$\lambda := +\infty;\ \ \varepsilon := +\infty;$
for $(t, x) \in S_0$ **do**
  if $p_t(x) \neq$ `NONMAX` **then**
    $t' := p_t(x);$
    for $x' \in X_t \mid \Delta\varphi_{tt'}(x) + \Delta\varphi_{t't}(x') < 0$ **do**
      $\lambda := \min\{\, \lambda,\ \lfloor g^{\boldsymbol{\varphi}}_{tt'}(x, x') / (\Delta\varphi_{tt'}(x) + \Delta\varphi_{t't}(x')) \rfloor \,\};$
      if $-g^{\boldsymbol{\varphi}}_{tt'}(x, x') > \varepsilon_{tt'}$ **then**
        if $-g^{\boldsymbol{\varphi}}_{tt'}(x, x') < \varepsilon$ **then**
          $\varepsilon := -g^{\boldsymbol{\varphi}}_{tt'}(x, x');\ \ t_\varepsilon := t;\ \ x_\varepsilon := x;$
        end if
      else
        $\lambda := 0;$

[5]Occasionally, some numbers $\Delta\varphi_{tt'}(x)$ can be very large, in theory even exponential in the depth of $D(t^*)$.

```
      end if
    end for
  else
    q := δ_{t=t*} + Σ_{t'∈N_t} Δφ_{tt'}(x);
    if q > 0 then
      λ := min{ λ, ⌊(u_t − g_t^φ(x))/q⌋ };
      if u_t − g_t^φ(x) < ε then
        ε := u_t − g_t^φ(x);  t_ε := t;  x_ε := x;
      end if
    end if
  end if
end for
end procedure
```

E.5 Updating the DAG

After the equivalent transformation $\boldsymbol{\varphi} \mathrel{+}= \lambda\,\Delta\boldsymbol{\varphi}$, some non-maximal nodes and edges will become maximal and some maximal edges non-maximal. For each of these cases, variables $p_t(x)$, Q and n_t have to be updated as follows:

- *If a maximal edge* $\{(t,x),(t',x')\}$ *becomes non-maximal:* Without loss of generality we assume that (t',x') is a node of $D(t^*)$ (note that $p_{t'}(x') \notin \{\texttt{ALIVE}, t\}$). If the node (t,x) is alive, add it to Q.
- *If a non-maximal edge* $\{(t,x),(t',x')\}$ *becomes maximal:* Without loss of generality we assume that (t,x) is a node of $D(t^*)$, and $p_t(x) = t'$. If $u_{t'} - g_{t'}^{\boldsymbol{\varphi}}(x') \le \varepsilon_{t'}$, resurrect the nodes from which there is a directed path to (t,x) and add them to Q.
- *If a non-maximal node* (t,x) *becomes maximal:* Resurrect the nodes in D from which there is a directed path to (t,x) and add them to Q.

The end nodes of the paths to be resurrected are pushed to the node stack S.

```
procedure repair
S := ∅;
for (t,x) ∈ S_0 do
  if p_t(x) = NONMAX then
    t' := p_t(x);
    for x' ∈ X_{tt'}(x) do
      if −g_{tt'}^φ(x,x') ≤ ε_{tt'} then
        for x'' ∈ X_{t't}(x') do
          if [−g_{tt'}^φ(x'',x') > ε_{tt'} ≤ g_{tt'}^{φ+λΔφ}(x'',x')] ∧ [p_t(x'') = ALIVE] ∧ [(t,x'') ∉ Q] then
            enqueue(Q,(t,x''));
          end if
        end for
      else if [−g_{tt'}^{φ+λΔφ}(x,x') ≤ ε_{tt'}] ∧ [u_{t'} − g_{t'}^{φ+λΔφ}(x') − λδ_{t'=t*} ≤ ε_{t'}] then
        push(S,(t,x));
      end if
    end for
  else if u_t − g_t^{φ+λΔφ}(x) − λδ_{t=t*} ≤ ε_t then
    push(S,(t,x));
  end if
end for
end procedure
```

If the height of t^* cannot be decreased due to an underflow or overflow, the status of some nodes and edges changes due to increasing thresholds ε_t or $\varepsilon_{tt'}$. The following procedure increases

the appropriate threshold and makes the update after non-maximal nodes in object t_ε or edges in object pair $(t_\varepsilon, p_{t_\varepsilon}(x_\varepsilon))$ possibly become maximal.

procedure threshold
$t := t_\varepsilon$; $x := x_\varepsilon$;
$S := \emptyset$;
if $p_t(x) \neq$ `NONMAX` **then**
  $t' := p_t(x)$;
  for $x \in X_t$ **do**
    for $x' \in X_{tt'}(x) \mid \varepsilon_{tt'} < -g^{\boldsymbol{\varphi}}_{tt'}(x, x') \leq \varepsilon$ **do**
      if $[p_t(x) = t'] \wedge [p_{t'}(x') \neq$ `NONMAX`$]$ **then** push$(S, (t, x))$; **end if**
      if $[p_{t'}(x') = t] \wedge [p_t(x) \neq$ `NONMAX`$]$ **then** push$(S, (t', x'))$; **end if**
    end for
  end for
  $\varepsilon_{tt'} := \varepsilon$;
else
  for $x \in X_t \mid [p_t(x) =$ `NONMAX`$] \wedge [\varepsilon_t < u_t - g^{\boldsymbol{\varphi}}_t(x) \leq \varepsilon]$ **do**
    push$(S, (t, x))$;
  end for
  $\varepsilon_t := \varepsilon$;
end if
end procedure

Finally, the nodes in S and their predecessors in D are resurrected.

procedure resurrect
while $S \neq \emptyset$ **do**
  $(t, x) :=$ pop(S);
  if $p_t(x) \neq$ `ALIVE` **then**
    $p_t(x) :=$ `ALIVE`; n_t++;
    if $(t, x) \notin Q$ **then** enqueue$(Q, (t, x))$; **end if**
    for $t' \in N_t$ **do**
      for $x' \in X_{tt'}(x) \mid [-g^{\boldsymbol{\varphi}}_{tt'}(x, x') \leq \varepsilon_{tt'}] \wedge [p_{t'}(x') = t]$ **do**
        push$(S, (t', x'))$;
      end for
    end for
  end if
end while
end procedure

E.6 Equivalent Transformation

The following procedure sets $\boldsymbol{\varphi} \mathrel{+}= \lambda\, \Delta\boldsymbol{\varphi}$ and $\Delta\varphi := \mathbf{0}$, and updates u_{t^*}.

procedure update
for $(t, x) \in S_0 \mid p_t(x) \neq$ `NONMAX` **do**
  $t' := p_t(x)$;
  $\varphi_{tt'}(x) \mathrel{+}= \lambda\, \Delta\varphi_{tt'}(x)$;
  $\Delta\varphi_{tt'}(x) := 0$;
  for $x' \in X_{tt'}(x) \mid -g^{\boldsymbol{\varphi}}_{tt'}(x, x') \leq \varepsilon_{tt'}$ **do**
    $\varphi_{t't}(x') \mathrel{+}= \lambda\, \Delta\varphi_{t't}(x')$;
    $\Delta\varphi_{t't}(x') := 0$;
  end for
end for
$u_{t^*} := \max_{x \in X_{t^*}} g^{\boldsymbol{\varphi}}_{t^*}(x)$;
end procedure

Acknowledgment

This work was supported by the the European Union, grant IST-2004-71567 COSPAL. This text would not be possible without Václav Hlaváč, who established co-operation of the Center of Machine Perception in Prague with the Kiev and Dresden groups in 1995 and has been supporting it since then. This co-operation resulted in lectures and seminars on labeling problems by Mikhail I. Schlesinger in Prague and my personal communication with him and Boris Flach. I thank Mikhail I. Schlesinger, Vladimir Kovalevsky, and Boris Flach for agreeing that I presented some of their unpublished work. Christoph Schnör and my colleagues, most notably Alexander Shekhovtsov, and further Václav Hlaváč, Mirko Navara, Tomáš Pajdla, Jiří (George) Matas, and Vojtěch Franc gave me valuable comments on the manuscript.

References

[AM00] Srinivas M. Aji and Robert J. McEliece. The generalized distributive law. *IEEE Trans. on Information Theory*, 46(2):325–343, 2000.

[BH02] Endre Boros and Peter L. Hammer. Pseudo-Boolean optimization. *Discrete Appl. Math.*, 123(1-3):155–225, 2002.

[BKR96] Rainer E. Burkard, Bettina Klinz, and Rüdiger Rudolf. Perspectives of Monge properties in optimization. *Discrete Appl. Math.*, 70(2):95–161, 1996.

[BMR97] Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. Semiring-based constraint satisfaction and optimization. *J. ACM*, 44(2):201–236, 1997.

[BT97] Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

[CCJK04] David A. Cohen, Martin C. Cooper, Peter Jeavons, and Andrei A. Krokhin. Identifying efficiently solvable cases of Max CSP. In *21st Ann. Symp. on Theor. Aspects of Comp. Sc. (STACS)*, pages 152–163. Springer, 2004.

[CKNZ01] Chandra Chekuri, Sanjeev Khanna, Joseph Naor, and Leonid Zosin. Approximation algorithms for the metric labeling problem via a new linear programming formulation. In *Symposium on Discrete Algorithms*, pages 109–118, 2001.

[CKT91] Peter Cheeseman, Bob Kanefsky, and William M. Taylor. Where the Really Hard Problems Are. In *Int. Joint Conf. on Artif. Intell., (IJCAI)*, pages 331–337, 1991.

[DP90] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, Cambridge, 1990.

[Fla98] Boris Flach. A diffusion algorithm for decreasing energy of max-sum labeling problem. unpublished, Fakultät Informatik, Technische Universität Dresden, Germany, 1998.

[Fla02] Boris Flach. Strukturelle bilderkennung. Technical report, Fakultät Informatik, Technische Universität Dresden, Germany, 2002. Habilitation thesis, in German.

[FS00] Boris Flach and Michail I. Schlesinger. A class of solvable consistent labeling problems. In *Proceedings of the Joint IAPR International Workshops on Advances in Patt. Recog.*, pages 462–471, London, UK, 2000. Springer-Verlag.

[Gau97] Stéphane Gaubert. Methods and applications of (max,+) linear algebra. Technical Report 3088, Institut national de recherche en informatique et en automatique (INRIA), 1997.

[GLS81] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[GLS88] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer Verlag, 1988. 2nd edition in 1993.

[GPS89] D.M. Greig, B.T. Porteous, and A.H. Seheult. Exact maximum a posteriori estimation for binary images. *J. R. Statist. Soc. B*, (51):271–279, 1989.

[Ham65] P. Hammer. Some network flow problems solved with pseudo-Boolean programming. *Operations Research*, 13:388–399, 1965.

[HDT92] Pascal Van Hentenryck, Yves Deville, and Choh-Man Teng. A generic arc-consistency algorithm and its specializations. *Artif. Intell.*, 57(2–3):291–321, 1992.

[HS79] R. M. Haralick and L. G. Shapiro. The consistent labeling problem. *IEEE Trans. Patt. Anal. Machine Intell.*, 1(2):173–184, 1979.

[IFF01] S. Iwata, L. Fleischer, and S. Fujishige. A combinatorial strongly polynomial-time algorithm for minimizing submodular functions. *J. Assoc. Comput. Mach.*, 48:761–777, 2001.

[IG98] H. Ishikawa and D. Geiger. Segmentation by grouping junctions. In *IEEE Conf. Comp. Vision and Patt. Recogn.*, pages 125–131, 1998.

[Ish03] Hiroshi Ishikawa. Exact optimization for Markov random fields with convex priors. *IEEE Trans. Patt. Anal. Mach. Intell.*, 25(10):1333–1336, 2003.

[KK75] V. A. Kovalevsky and V. K. Koval. A diffusion algorithm for decreasing energy of max-sum labeling problem. unpublished, Glushkov Institute of Cybernetics, Kiev, USSR., approx. 1975.

[Kol04] Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. Technical Report MSR-TR-2004-90, Microsoft Research, 2004.

[Kol05a] Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. Technical Report MSR-TR-2005-38, Microsoft Research, 2005.

[Kol05b] Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. In *Int. Workshop on Art. Intell. and Stat. (AISTATS)*, 2005.

[Kol05c] Vladimir Kolmogorov. Primal-dual algorithm for convex Markov random fields. Technical Report MSR-TR-2005-117, Microsoft Research, 2005.

[Kos99] Arie Koster. *Frequency Assignment – Models and Algorithms*. PhD thesis, Universiteit Maastricht, Maastricht, The Netherlands, 1999. ISBN 90-9013119-1.

[Kov03] Ivan Kovtun. Partial optimal labelling search for a NP-hard subclass of (max,+) problems. In *German Assoc. for Patt. Recog. Conf. (DAGM)*, pages 402–409, 2003.

[Kov04] Ivan Kovtun. *Segmentaciya zobrazhen na usnovi dostatnikh umov optimalnosti v NP-povnikh klasakh zadach strukturnoi rozmitki (Image segmentation based on sufficient conditions of optimality in NP-complete classes of structural labeling problems)*. PhD thesis, IRTC ITS Nat. Academy of Science Ukraine, Kiev, 2004. In Ukrainian.

[KS76] V. K. Koval and M. I. Schlesinger. Dvumernoe programmirovanie v zadachakh analiza izobrazheniy (Two-dimensional programming in image analysis problems). *USSR Academy of Science, Automatics and Telemechanics*, 8:149–168, 1976. In Russian.

[KSK77] V. A. Kovalevsky, M. I. Schlesinger, and V. K. Koval. Ustrojstvo dlya analiza seti. Patent Nr. 576843, USSR, priority of January 4, 1976, 1977. In Russian.

[Kum92] V. Kumar. Algorithms for constraint-satisfaction problems: A survey. *AI Magazine*, 13(1):32–44, 1992.

[KvHK98] Arie Koster, C. P. M. van Hoesel, and A. W. J. Kolen. The partial constraint satisfaction problem: Facets and lifting theorems. *Operations Research Letters*, 23(3–5):89–97, 1998.

[KW05a] V. N. Kolmogorov and M. J. Wainwright. On the optimality of tree-reweighted max-product message-passing. In *Conf. Uncert. in Artif. Intel. (UAI)*, 2005.

[KW05b] Vladimir Kolmogorov and Martin Wainwright. On the optimality of tree-reweighted max-product message-passing. Technical Report MSR-TR-2004-37, Microsoft Research, 2005.

[KZ02] Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts? In *Eur. Conf. on Comp. Vision (ECCV)*, pages 65–81. Springer-Verlag, 2002.

[LG04] Lucian Leahu and Carla P. Gomes. Quality of LP-based approximations for highly combinatorial problems. In Mark Wallace, editor, *Conf. Principles and Practice of Constraint Programming (CP), Toronto, Canada*, pages 377–392. Springer, 2004.

[Lov83] L. Lovász. Submodular functions and convexity. In A. Bachem, M. Grotschel, and B. Korte, editors, *Mathematical Programming – The State of the Art*, pages 235–257. Springer-Verlag, New York, 1983.

[Mon74] Ugo Montanari. Networks of constraints: Fundamental properties and application to picture processing. *Inf. Sci.*, 7:95–132, 1974.

[MP88] Marvin L. Minsky and Seymour A. Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA, 2 edition, 1988. First edition in 1971.

[Pea88] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[RHZ76] A. Rosenfeld, R. A. Hummel, and S. W. Zucker. Scene labeling by relaxation operations. *IEEE Trans. on Systems, Man, and Cybernetics*, 6(6):420–433, June 1976.

[Sch76a] Michail I. Schlesinger. False minima of the algorithm for minimizing energy of max-sum labeling problem. unpublished, Glushkov Institute of Cybernetics, Kiev, USSR., 1976.

[Sch76b] Michail I. Schlesinger. Sintaksicheskiy analiz dvumernykh zritelnikh signalov v usloviyakh pomekh (Syntactic analysis of two-dimensional visual signals in noisy conditions). *Kibernetika*, 4:113–130, 1976. In Russian.

[Sch89] Michail I. Schlesinger. *Matematicheskie sredstva obrabotki izobrazheniy (Mathematical Tools of Image Processing)*. Naukova Dumka, Kiev, 1989. In Russian.

[Sch00] Alexander Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *J. Comb. Theory Ser. B*, 80(2):346–355, 2000.

[Sch05a] Dmitrij Schlesinger. *Strukturelle Ansätze für die Stereorekonstruktion*. PhD thesis, Technische Universität Dresden, Fakultät Informatik, Institut für Künstliche Intelligenz, July 2005. In German.

[Sch05b] Michail I. Schlesinger. Personal communication, 2000-2005. International Research and Training Centre, Kiev, Ukraine.

[SF00] Michail I. Schlesinger and Boris Flach. Some solvable subclasses of structural recognition problems. In *Czech Patt. Recog. Workshop*, 2000.

[SH02] Michail I. Schlesinger and Václav Hlaváč. *Ten Lectures on Statistical and Structural Pattern Recognition*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.

[SK78] Michail I. Schlesinger and Vladimir Kovalevsky. A hydraulic model of a linear programming relaxation of max-sum labeling problem. unpublished, Glushkov Institute of Cybernetics, Kiev, USSR., 1978.

[Smi96] Barbara Smith. Locating the phase transition in binary constraint satisfaction problems. *Artif. Intell.*, 81:155–181, 1996.

[Top78] D. M. Topkis. Minimizing a submodular function on a lattice. *Operations Research*, 26(2):305–321, 1978.

[Top98] Donald M. Topkis. *Supermodularity and Complementarity*. Frontiers of Economic Research. Princeton University Press, Princeton, NJ, 1998.

[Wal72] David L. Waltz. Generating semantic descriptions from drawings of scenes with shadows. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1972.

[WJ03a] M. Wainwright and M. Jordan. Variational inference in graphical models: The view from the marginal polytope. In *Allerton Conf. on Communication, Control and Computing*, 2003.

[WJ03b] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. Technical Report 649, UC Berkeley, Dept. of Statistics, 2003.

[WJW02] M. Wainwright, T. Jaakkola, and A. Willsky. MAP estimation via agreement on (hyper)trees: message passing and linear programming approaches. In *Allerton Conf. on Communication, Control and Computing*, 2002.

[WJW03a] M. Wainwright, T. Jaakkola, and A. Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Trans. Inf. Theory*, 49(5):1120–1146, 2003.

[WJW03b] M. Wainwright, T. Jaakkola, and A. Willsky. Tree-reweighted belief propagation algorithms and approximate ml estimation via pseudo-moment matching. In *Int. Workshop on Art. Intell. and Stat. (AISTATS)*, 2003.

[WJW04] M. Wainwright, T. Jaakkola, and A. Willsky. Tree consistency and bounds on the performance of the max-product algorithm and its generalizations. *Stat. and Computing*, 14:143–166, 2004.

[WJW05] M. Wainwright, T. Jaakkola, and A. Willsky. MAP estimation via agreement on (hyper)trees: message passing and linear programming approaches. *IEEE Trans. Inf. Theory*, 51(11):3697–3717, 2005.

[Yed04] Jonathan Yedidia. Constructing free energy approximations and generalized belief propagation algorithms. Technical Report TR-2004-040, Mitsubishi Electric Research Lab., 2004.