A Dual Ascent Framework for Lagrangean Decomposition of Combinatorial Problems

Paul Swoboda\textsuperscript{1}, Jan Kuske\textsuperscript{2}, Bogdan Savchynskyy\textsuperscript{3}
\textsuperscript{1}IST Austria, \textsuperscript{2}Universität Heidelberg, \textsuperscript{3}TU Dresden
pswoboda@ist.ac.at

Abstract

We propose a general dual ascent framework for Lagrangean decomposition of combinatorial problems. Although methods of this type have shown their efficiency for a number of problems, so far there was no general algorithm applicable to multiple problem types. In this work, we propose such a general algorithm. It depends on several parameters, which can be used to optimize its performance in each particular setting. We demonstrate efficacy of our method on graph matching and multicut problems, where it outperforms state-of-the-art solvers including those based on subgradient optimization and off-the-shelf linear programming solvers.

1. Introduction

Computer vision and machine learning give rise to a number of powerful computational models. It is typical that inference in these models reduces to non-trivial combinatorial optimization problems. For some of the models, such as conditional random fields (CRF), powerful specialized solvers like [48, 49, 12, 52] were developed. In general, however, one has to resort to off-the-shelf integer linear program (ILP) solvers like CPLEX [2] or Gurobi [36]. Although these solvers have made a tremendous progress in the past decade, the size of the problems they can tackle still remains a limiting factor for many potential applications, as the running time scales super-linearly in the problem size. The goal of this work is to partially fill this gap between practical requirements and existing computational methods.

It is an old observation that many important optimization ILPs can be efficiently decomposed into easily solvable combinatorial sub-problems [32]. The convex relaxation, which consists of these sub-problems coupled by linear constraints is known as Lagrangean or dual decomposition [31, 50]. Although this technique can be efficiently used in various scenarios to find approximate solutions of combinatorial problems, it has a major drawback: In the most general setting only slow (sub)gradient-based techniques [51, 57, 50, 42, 61] can be used for optimization of the corresponding convex relaxation.

In the area of conditional random fields, however, it is well-known [41] that message passing or dual (block-coordinate) ascent algorithms (like e.g. TRW-S [48]) significantly outperform (sub)gradient-based methods. Similar observations were made much earlier in [59] for a constrained shortest path problem.

Although dual ascent algorithms were proposed for a number of combinatorial problems (see the related work overview below), there is no general framework, which would (i) give a generalized view on the properties of such algorithms and more importantly (ii) provide tools to easily construct such algorithms for new problems. Our work provides such a framework.

Related Work Dual ascent algorithms optimize a dual problem and guarantee monotonous improvement (non-deterioration) of the dual objective. The most famous examples in computer vision are block-coordinate ascent (known also as message passing) algorithms like TRW-S [48] or MPLP [28] for maximum a posteriori inference in conditional random fields [41].

To the best of our knowledge the first dual ascent algorithm addressing integer linear programs belongs to Bilde and Krarup [11] (the corresponding technical report in Danish appeared 1967). In that work an uncapacitated facility location problem was addressed. A similar problem (simple plant location) was addressed with an algorithm of the same class in [30]. In 1980 Fisher and Hochbaum [22] constructed a dual ascent-based algorithm for a problem of database location in computer networks, which was used to optimize the topology of Arpanet [1], predecessor of Internet. The generalized linear assignment problem was addressed by the same type of algorithms in [23]. The Authors considered a Lagrangean decomposition of this problem into multiple knapsack problems, which were solved in each iteration of the method. An improved version of this algorithm was proposed in [34]. Efficient dual ascent based solvers were also proposed for the min-cost flow in [25], for the set
covering and the set partitioning problems in [24] and the resource-constrained minimum weighted arborescence problem in [35]. The work [33] describes basic principles for constructing dual ascent algorithms. Although the authors provide several examples, they do not go beyond that and stick to the claim that these methods are structure dependent and problem specific.

The work [18] suggests to use the max-product belief propagation [73] to decomposable optimization problems. However, their algorithm is neither monotone nor even convergent in general.

In computer vision, dual block coordinate ascent algorithms for Lagrangean decomposition of combinatorial problems were proposed for multiple targets tracking [8], graph matching (quadratic assignment) problem [78] and inference in conditional random fields [48, 49, 28, 74, 75, 62, 37, 56, 72, 39]. From the latter, the TRW-S algorithm [48] is among the most efficient ones for pairwise conditional random fields according to [41]. The SRMP algorithm [49] generalizes TRW-S to conditional random fields of arbitrary order. In a certain sense, our framework can be seen as a generalization of SRMP to a broad class of combinatorial problems.

Contribution. We propose a new dual ascent based computational framework for combinatorial optimization. To this end we:
(i) Define the class of problems, called integer-relaxed pairwise-separable linear programs (IRPS-LP), our framework can be used for. Our definition captures Lagrangean decompositions of many known discrete optimization problems (Section 2).
(ii) Give a general monotonically convergent message-passing algorithm for solving IRPS-LP, which in particular subsumes several known solvers for conditional random fields (Section 4).
(iii) Give a characterization of the fixed points of our algorithm, which subsumes well-known fixed point characterizations as weak tree agreement [48] and arc-consistency [74] (Section 5).

We demonstrate efficiency of our method by outperforming state-of-the-art solvers for two famous special cases of IRPS-LP, which are widely used in computer vision: the multicut and the graph matching problems. (Section 6).

A C++ framework containing the above mentioned solvers and the datasets used in experiments can be obtained at https://github.com/pawelswoboda/LP_MP.

We give all proofs in the supplementary material.

Notation. Undirected graphs will be denoted by \( G = (V, E) \), where \( V \) is a finite node set and \( E \subseteq \binom{V}{2} \) is the edge set. The set of neighboring nodes of \( v \in V \) w.r.t. graph \( G \) is denoted by \( N_G(v) := \{ u : uv \in E \} \). The convex hull of a set \( X \subset \mathbb{R}^n \) is denoted by \( \text{conv}(X) \). Disjoint union is denoted by \( \dot{\cup} \).

2. Integer-Relaxed Pairwise-Separable Linear Programs (IRPS-LP)

Combinatorial problems of the form \( \min_{x \in X} \theta(x) \), where \( X \subseteq \{0,1\}^n \) are binary vectors, often have a decomposable representation as \( \min_{x_i \in X_i} \sum_{i=1}^k (\theta_i, x_i) \) for \( X_i \subseteq \{0,1\}^{d_i} \) being sets of binary vectors, typically corresponding to subsets of the coordinates of \( X \). This decomposed problem is equivalent to the original one under a set of linear constraints \( A_{i,j} x_i = A_{j,i} x_j \), which guarantee the mutual consistency of the considered components. By replacing \( X_i \) by its convex hull \( \text{conv}(X_i) \) we switching to real-valued vectors from binary ones and obtain a convex relaxation of the problem. It reads:

\[
\min_{\mu \in \mathbb{R}^d} \sum_{i=1}^k (\theta_i, \mu_i) , \quad \text{where } A_G \text{ is defined as }
\]

\[
A_G := \left\{ (\mu_1, \ldots, \mu_k) \mid \mu_i \in \text{conv}(X_i) \quad i \in F \right\}.
\]

(2)

Here \( F := \{ 1, \ldots, k \} \) are called factors of the decomposition and \( E \subseteq \binom{F}{2} \) correspond to the coupling constraints.

The undirected graph \( G = (F, E) \) is called factor graph. We will use variable names \( \mu \) whenever we want to emphasize \( \mu_i \in \text{conv}(X_i) \) and \( X \) whenever \( x_i \in X_i \).

Definition 1 (IRPS-LP). Assume that for each edge \( ij \in E \) the matrices of the coupling constraints \( A_{i,j} \), \( A_{j,i} \) are such that \( A_{i,j} \in \{0,1\}^{K \times d_i} \), and \( A_{i,j} x_i \in \{0,1\}^K \) for some \( K \in \mathbb{N} \), analogously for \( A_{j,i} \). The problem \( \min_{\mu \in \mathbb{R}^d} \sum_{i \in F} (\theta_i, \mu_i) \) is called an Integer-Relaxed Pairwise-Separable Linear Program, abbreviated by IRPS-LP.

In the following, we give several examples of IRPS-LP. To distinguish between notation for the factor graph of IRPS-LP, where we stick to bold letters (such as \( G, F, E \)) we will use the straight font (such as \( G, V, E \)) for the graphs occurring in the examples.

Example 1 (MAP-inference for CRF). A conditional random field is given by a graph \( G = (V, E) \), a discrete label space \( X = \prod_{u \in V} X_u \), unary \( \theta_u : X_u \to \mathbb{R} \) and pairwise costs \( \theta_{uv} : X_u \times X_v \to \mathbb{R} \) for \( u \in V, uv \in E \). We also denote \( X_{uv} := X_u \times X_v \). The associated maximum a posteriori (MAP)-inference problem reads:

\[
\min_{x \in X} \sum_{u \in V} \theta_u(x_u) + \sum_{uv \in E} \theta_{uv}(x_{uv}),
\]

(3)

where \( x_u \) and \( x_{uv} \) denote the components corresponding to node \( u \in V \) and edge \( uv \in E \) respectively. The well-known
local polytope relaxation [74] can be seen as an IRPS-LP by setting $\overline{F} = V \cup E$, that is associating to each node $v \in V$ and each edge $uv \in E$ a factor, and introducing two coupling constraints for each edge of the graphical model, i.e. $E = \{u, uv\}, \{v, uv\}$ such that $uv \in E$. For the sake of notation we will assume that each label $s \in X_u$ is associated a unit vector $(0, \ldots, 0, 1, 0, \ldots, 0)$ with dimensionality equal to $s$ the total number of labels $|X_u|$ and $1$ on the $s$-th position. Therefore, the notation $\text{conv}(X_u)$ makes sense as a convex hull of all such vectors. After denoting an $N$-dimensional simplex as $\Delta_N := \{\mu \in \mathbb{R}^N : \sum_{i=1}^N \mu_i = 1, \mu_i \geq 0\}$ the resulting relaxation reads

$$\min_{\mu \in \mathbb{R}^N} \langle \theta, \mu \rangle := \sum_{u \in V} \langle \theta_u, \mu_u \rangle + \sum_{uv \in E} \langle \theta_{uv}, \mu_{uv} \rangle$$

in the overcomplete representation [71] and $\mathbb{R}^N$ is defined as

$$\mu_u \in \text{conv}(X_u) : \mu_u \in \Delta_{|X_u|}, u \in V$$

$$\mu_{uv} \in \text{conv}(X_{uv}) : \mu_{uv} \in \Delta_{|X_{uv}|}$$

$$\Delta_{|X_{uv}|} = \Delta_{|X_u|} \cap \Delta_{|X_v|}, \forall uv \in E,$$ (5)

Here $\mu_u(x_u)$ and $\mu_{uv}(x_u, x_v)$ denote those coordinates of vectors $\mu_u$ and $\mu_{uv}$, which correspond to the label $x_u$ and the pair of labels $(x_u, x_v)$ respectively. 

**Example 2** (Graph Matching). The graph matching problem, also known as quadratic assignment [13] or feature matching, can be seen as a MAP-inference problem for CRFs (as in Example 1) equipped with additional constraints: The label set of $G$ belongs to a universe $\mathcal{L}$, i.e. $X_u \subseteq \mathcal{L} \forall u \in V$ and each label $s \in \mathcal{L}$ can be assigned at most once. The overall problem reads

$$\min_{x} \sum_{u \in V} \theta_u(x_u) + \sum_{uv \in E} \theta_{uv}(x_u, x_v) \text{ s.t. } x_u \neq x_v \forall u \neq v.$$ (6)

Graph matching is a key step in many computer vision applications, among them tracking and image registration, whose aim is to find a one-to-one correspondence between image points. For this reason, a large number of solvers have been proposed in the computer vision community [18, 76, 78, 70, 53, 69, 63, 29, 79, 38, 54, 16]. Among them two recent methods [70, 78] based on Lagrangian decomposition show superior performance and provide lower bounds for their solutions. The decomposition we describe below, however, differs from those proposed in [70, 78].

Our IRPS-LP representation for graph matching consists of two blocks: (i) the CRF itself (which further decomposes into node- and edge-subproblems with variables $(\mu_u)_{u \in V}$ and (ii) additional *label-factors* keeping track of nodes assigned the label $s$. We introduce these label-factors for each label $s \in \mathcal{L}$. The set of possible configurations of this factor $X_s := \{u \in V : x_s \in X_u\} \cup \{\#\}$ consists of those nodes $u \in V$ which can be assigned the label $s$ and an additional dummy node $\#$. The dummy node $\#$ denotes non-assignment of the label $s$ and is necessary, as not every label needs to be taken. As in Example 1, we associate a unit binary vector with each element of the set $X_s$, and $\text{conv}(X_s)$ denotes the convex hull of those vectors. The set of factors becomes $\overline{F} = V \cup E \cup \mathcal{L}$, with the set $E = \{u, uv\}, \{v, uv\}$ such that $uv \in E \cup \{u, l\}$ is defined as $\mathbb{R}^N$ the solutions. The decomposition we describe below, however, differs from those proposed in [70, 78].

Here we introduced (i) auxiliary variables $\tilde{\mu}_s(u)$ for all variables $\mu_s(u)$ and (ii) auxiliary node costs $\tilde{\theta}_s \equiv 0 \forall s \in \mathcal{L}$, which may take other values in course of optimization. Factors associated with the vectors $\mu_u$ and $\mu_{uv}$ correspond to the nodes and edges of the graph $G$ (node- and edge-factors), as in Example 1 and are coupled in the same way. Additionally, factors associated with the vectors $\tilde{\mu}_s$ ensure that the label $s$ can be taken at most once. These label-factors are coupled with node-factors (last line in (7)).

**Example 3** (Multicut). The multicut problem (also known as correlation clustering) for an undirected weighted graph $G = (G, E)$ is to find a partition $(\Pi_1, \ldots, \Pi_k)$ such that the total cost of edges connecting different components is minimized. The number $k$ of components is not fixed but is determined by the algorithm. See Fig. 1 for an illustration. Although the problem has numerous applications in computer vision [4, 5, 6, 77] and beyond [7, 60, 14, 15], there is no scalable solver, which could provide optimality bounds. Existing methods are either efficient primal heuristics [66, 58, 27, 19, 20, 9, 10] or combinatorial branch-and-bound/branch-and-cut/column generation algorithms, based on off-the-shelf LP solvers [43, 44, 47, 77]. Move-making algorithms do not provide lower bounds, hence, one cannot judge their solution quality or employ them in branch-and-bound procedures. Off-the-shelf LP solvers on the other hand scale super-linearly, limiting their application in large-scale problems.

Instead of directly optimizing over partitions (which has many symmetries making optimization difficult in a linear programming setting), we follow [17] and formulate the problem in the edge domain. Let $\theta_e, e \in E$ denote the cost of graph edges and let $C$ be the set of all cycles of the graph $G$. Each edge that belongs to different components is called a cut edge. The multicut problem reads

$$\min_{x \in \{0, 1\}^E} \sum_{e \in E} \theta_e x_e, \text{ s.t. } \forall C \forall e' \in C: \sum_{e \in C \setminus \{e'\}} x_e \geq x_{e'}.$$ (8)
Here \( x_e = 1 \) signifies a cut edge and the inequalities force each cycle to have none or at least two cut edges. The formulation (8) has exponentially many constraints. However, it is well-known that it is sufficient to consider only chordless cycles [17] in place of the set \( C \) in (8). Moreover, the graph can be triangulated by adding additional edges with zero weights and therefore the set of chordless cycles reduces to edge triples. Such triangulation is referred to as chordal completion in the literature [26]. The number of triples is cubic, which is still too large for practical efficiency and therefore violated constraints are typically added to the problem iteratively in a cutting plane manner [43, 44]. To simplify the description, we will ignore this fact below and consider all these cycles at once. Assuming a triangulated graph and redefining \( C \) as the set of all chordless cycles (triples) we consider the following IRPS-LP relaxation of the multicut problem \(^1\):

\[
\min_{\mu, \bar{\mu}} \sum_{e \in E} \theta_e \mu_e + \sum_{c \in E} \sum_{c' \in c} \bar{\theta}_{e,c,c'}, \quad \text{s.t.} \quad (9)
\]

\[
\begin{align*}
\mu_e &\in \text{conv}(\{0,1\}) = \{0, 1\}, \ e \in E \\
\forall c &\in C, \ e \in c: \\
\bar{\mu}_c &:= (\bar{\mu}_{e,c})_{e \in C} \in \text{conv}(\{0,1\}^3), \ \forall e' \in c: \sum_{e \in c \setminus \{e'\}} \bar{\mu}_{e,c} \geq \bar{\mu}_{e',c} \\
\bar{\mu}_c &\equiv \text{conv}(\{0,0,0\}, \{0,1,1\}, \{1,0,1\}, \{1,1,0\}, \{1,1,1\}) \\
\mu_e &\equiv \bar{\mu}_{e,c}
\end{align*}
\]

(10)

For the sake of notation we shortened a feasible set definition \( \mu \in \text{conv}(\mu' \in \{0,1\}^n): \text{constraints on } \mu' \) to \( \hat{\mu} \in \text{conv}(\{0,1\}^n): \text{constraints on } \hat{\mu} \). Here \( \mu_e \) is the relaxed (potentially non-integer) variable corresponding to \( x_e \). Variable \( \bar{\mu}_{e,c} \) is a copy of \( \mu_e \), which corresponds to the cycle \( c \). Therefore, each \( \mu_e \) gets as many copies \( \bar{\mu}_{e,c} \), as many chordless cycles \( c \) contain the edge \( e \). For each cycle the set of binary vectors satisfying the cycle inequality is considered. For a cycle with 3 edges this set can be written explicitly as in (10). Along with copies of \( \mu_e, e \in E \) we copy the corresponding cost \( \theta_e \), and create auxiliary costs \( \theta_{e,c} \equiv 0 \) for each cycle \( c \) containing the edge \( e \). During optimization, the cost \( \theta_e \) will be redistributed between \( \theta_e \) itself and its copies \( \theta_{e,c} \), \( c \in C \). The factors of the IRPS-LP are associated with each edge (variable \( \mu_e \)) and each chordless cycle (variable \( \bar{\mu}_c \)). Coupling constraints connect edge-factors with those cycle-factors, which contain the corresponding edge (see the last constraint in (10)). An in-depth discussion of message passing for the multicut problem with tighter relaxations can be found in [67].

3. Dual Problem and Admissible Messages

Since our technique can be seen as a dual ascent, we will not optimize the primal problem (1) directly, but instead

\(^1\)One can show that this relaxation coincides with the standard LP relaxation for the multicut problem [17]

maximize its dual lower bound.

Dual IRPS-LP The Lagrangean dual to (1) w.r.t. the coupling constraints reads

\[
\max_{\phi} \ D(\phi) := \sum_{e \in E} \min_{x_e \in \{0,1\}} \langle \theta_e^0, x_e \rangle \\
\text{s.t.} \quad \theta_e^0 := \theta_e + \sum_{j: i,j \in E} A_{i,j}(i,j) \phi_{(i,j)} \quad \forall i \in F \quad (11)
\]

Here \( \phi_{(i,j)} \in \mathbb{R}^K \) for \( A_{i,j} \in \{0,1\}^K \times d_i \) for some \( K \in \mathbb{N} \). The function \( D(\phi) \) is called lower bound and is concave in \( \phi \). The modified primal costs \( \theta^0 \) are called reparametrizations of the potentials \( \theta \). We have duplicated the dual variables by introducing \( \phi_{(i,j)} := -\phi_{(j,i)} \) to symmetrize notation. In practice, only one copy is stored and the other is computed on the fly. Note that in this doubled notation the reparametrized node and edge potentials of the CRF from Example 1 read

\[
\begin{align*}
\theta^0_{uv}(x_u) &= \theta_u(x_u) + \sum_{v: uv \in E} \phi_{u,v}(x_u) \\
\theta^0_{uv}(x_u, x_v) &= \theta_{uv}(x_u, x_v) + \phi_{uv,v}(x_v) + \phi_{uv,u}(x_u) \\
\phi_{u,v} &= -\phi_{v,u}
\end{align*}
\]

It is well-known for CRFs that cost of feasible solutions are invariant under reparametrization. We generalize this to the IRPS-LP-case.

Proposition 1. \( \sum_{i \in F} \phi_i = \sum_{i \in F} \theta_i^0, \mu_i \) whenever \( \mu_1, \ldots, \mu_k \) obey the coupling constraints.

Admissible Messages While Proposition 1 guarantees that the primal problem is invariant under reparametrization, the dual lower bound \( D(\phi) \) is not. Our goal is to find \( \phi \) such that \( D(\phi) \) is maximal. By linear programming duality, \( D(\phi) \) will then be equal to the optimal value of the primal (1).

First we will consider an elementary step of our future algorithm and show that it is non-decreasing in the dual objective. This property will ensure the monotonicity of the whole algorithm. Let \( \theta^0 \) be any reparametrization of the problem and \( D(\phi) \) be the corresponding dual value. Let us consider changing the reparametrization of a factor \( i \) by a vector \( \Delta \) with the only non-zero components \( \Delta_{(i,j)} \) and \( \Delta_{(j,i)} \). This will change reparametrization of the coupled factors \( j \) (such that \( ij \in E \)) due to \( \Delta_{(i,j)} = -\Delta_{(j,i)} \). The lemma below states properties of \( \Delta_{(i,j)} \) which are sufficient to guarantee improvement of the corresponding dual value \( D(\phi + \Delta) \):
Lemma 1 (Monotonicity Condition). Let \( ij \in \mathcal{E} \) be a pair of factors related by the coupling constraints and \( \phi_{(i,j)} \) be a corresponding dual vector. Let \( x_i^* \in \operatorname{argmin}_{x_i \in X_i} (\theta_i^\phi, x_i^*) \) and \( \Delta_{(i,j)} \) satisfy

\[
\Delta_{(i,j)}(s) \begin{cases}
  \geq 0, & \nu(s) = 1 \\
  \leq 0, & \nu(s) = 0,
\end{cases}
\]

\[
\text{where } \nu := A_{(i,j)} x_i^*.
\]

(12)

Then \( x_i^* \in \operatorname{argmin}_{x_i \in X_i} (\theta_i^\phi + \Delta, x_i) \) implies \( D(\phi) \leq D(\phi + \Delta) \).

Example 4. Let us apply Lemma 1 to Example 1. Let \( ij \) correspond to \( \{ u, uv \} \), where \( u \in V \) is some node and \( uw \in E \) is any of its incident edges. Then \( x_i^* \) corresponds to a locally optimal label \( x_u^* \in \operatorname{argmin}_{x_u \in X_u} x_u \) such that \( \nu(u) = 1 \). Therefore we may assign \( \Delta_{u,uv}(s) \) to any value from \( [0, \theta_u(x_u^*) - \theta_u(s)] \). This assures that \( (20) \) is fulfilled and \( x_u^* \) remains a locally optimal label after reparametrization even if there are multiple optima in \( X_u \).

Lemma 1 can be straightforwardly generalized to the situation when a graph node sends messages to several incident edges at once:

Definition 2. Let \( i \in \mathcal{F} \) be a factor and \( J = \{ j_1, \ldots, j_l \} \subseteq \mathcal{N}_G(i) \) be a subset of its neighbors. Let \( \theta_i^\phi := \theta_i + \sum_{j \in J} A_{(i,j)} \Delta_{(i,j)}, \Delta_{(i,j)}(= \Delta_{(j,i)}) \) satisfies \( (20) \) for all \( j \in J \) and all other coordinates of \( \Delta \) are zero. If there exists \( x_i^* \in \operatorname{argmin}_{x_i \in X_i} (\theta_i, x_i) \) such that \( x_i^* \in \operatorname{argmin}_{x_i \in X_i} (\theta_i^\Delta, x_i) \), the dual vector \( \Delta \) is called admissible. The set of admissible vectors is denoted by \( AD(\theta_i, x_i^*, J) \).

Lemma 2. Let \( \Delta \in AD(\theta_i^\phi, x_i^*, J) \) then \( D(\phi) \leq D(\phi + \Delta) \).

Message-Passing Update Step To maximize \( D(\phi) \), we will iteratively visit all factors and adjust messages \( \phi \) connected to it, monotonically increasing the lower bound (11). Such an elementary step is defined by Procedure 1.

Procedure 1: Message-Passing Update Step.

1. **Input:** Factor \( i \in \mathcal{F} \), neighboring factors \( J = \{ j_1, \ldots, j_l \} \subseteq \mathcal{N}_G(i) \), dual variables \( \phi \)

2. Compute \( x_i^* \in \operatorname{argmin}_{x_i \in X_i} (\theta_i^\phi, x_i) \)

   (13)

3. Choose \( \delta \in \mathbb{R}^{d_i} \) s.t. \( \delta(s) \begin{cases}
  > 0, & x_i^*(s) = 1 \\
  < 0, & x_i^*(s) = 0
\end{cases} \)

   (14)

4. Maximize admissible messages to \( J \):

   \[
   \Delta_{(i,J)}^* \in \operatorname{argmax}_{\Delta \in AD(\theta_i^\phi, x_i^*, J)} \langle \delta, \theta_i^\phi + \Delta \rangle
   \]

   (15)

5. **Output:** \( \Delta_{(i,J)}^* \).

Now we combine message passing updates into Algorithm 2. It visits every node of the factor graph and performs the following two operations: (i) Receive Messages, when messages are received from a subset of neighboring factors, and (ii) Send Messages, when messages to some neighboring factors are computed and reweighted by \( \omega \). Distribution of weights \( \omega \) may influence the efficiency of Algorithm 2 just like it influences the efficiency of message passing for CRFs (see [49]). We provide typical settings in Section 4.1. Usually, factors are traversed in some given a-priori order alternately in forward and backward direction, as done in TRW-S [48] and SRMP [49]. We refer to [49] for a motivation for such a schedule of computations.

We will discuss parameters of Algorithm 2 (factor partitioning \( \{ J_i \} \), weights \( w_{J_i} \)) right after the theorem stating monotonicity for any choice of parameters.

Theorem 1. Algorithm 2 monotonically increases the dual lower bound (11).
Algorithm 2: One Iteration of Message-Passing

1. for \( i \in \mathcal{F} \) in some order do
2.   Receive Messages:
3.     Choose a subset of connected factors
4.       \( J_{\text{receive}} \subseteq \mathcal{N}_G(i) \)
5.     for \( j \in J_{\text{receive}} \) do
6.       Compute \( \Delta^*_j(i, j(i)) \) with Procedure 1.
7.     Set \( \phi = \phi + \Delta^*_j(i, j(i)) \).
8.   end
9. Send Messages:
10. Choose partition \( J_1 \cup \ldots \cup J_l \subseteq \mathcal{N}_G(i) \).
11. for \( J \in \{ J_1, \ldots, J_l \} \) do
12.    Compute \( \Delta^*_J(i, J) \) with Procedure 1.
13. end
14. Choose weights \( \omega_{J_1}, \ldots, \omega_{J_l} \geq 0 \) such that \( \omega_{J_1} + \ldots + \omega_{J_l} \leq 1 \).
15. for \( J \in \{ J_1, \ldots, J_l \} \) do
16.    Set \( \phi = \phi + \omega_{J} \Delta^*_J(i, J) \).
17. end
18. end

4.1. Parameter Selection for Algorithm 2

There are the following free parameters in Algorithm 2: (i) The order of traversing factors of \( \mathcal{F} \); (ii) for each factor the neighboring factors from which to receive messages \( J_{\text{receive}} \subseteq \mathcal{N}_G(i) \); (iii) the partition \( J_1 \cup \ldots \cup J_l \subseteq \mathcal{N}_G(i) \) of factors to send messages to and (iv) the associated weights \( \omega_{J_1}, \ldots, \omega_{J_l} \) for messages.

Although for any choice of these parameters Algorithm 2 monotonically increases the dual lower bound (as stated by Theorem 1), its efficiency may significantly depend on their values. Below, we will describe the parameters for Examples 1-3, which we found the most efficient empirically.

Sending a message by some factor automatically implies receiving this message by another, coupled factor. Therefore, usually there is no need to go over all factors in Algorithm 2. It is usually sufficient to guarantee that all coupling constraints are updated by Procedure 1. Formally, we can always exclude processing some factors by setting \( J_{\text{receive}} \) and \( J_i, i = 1, \ldots, l \) to the empty set. Instead, we will explicitly specify, which factors are processed in the loop of Algorithm 2 in the examples below.

Parameters for Example 1, MAP-inference in CRFs.

Pairwise CRFs have the specific feature that node factors are coupled with edge factors only. This implies that processing only node factors in Algorithm 2 is sufficient. Below, we describe parameters, which turn Algorithm 2 into SRMP [49] (which is up to details of implementation equivalent to TRW-S [48] for pairwise CRFs). Other settings, given in the supplement, may turn it to other popular message passing techniques like MPLP [28] or min-sum diffusion [64].

We order node factors and process them according to this ordering. The ordering naturally defines the sets of incoming \( E^+_u \) and outgoing \( E^-_u \) edges for each node \( u \in \mathcal{V} \). Here \( uv \in \mathcal{E} \) is incoming for \( u \leq u \) and outgoing if \( u > u \). Each node \( u \in \mathcal{V} \) receives messages from all incoming edges, which is \( J_{\text{receive}} = \mathcal{N}_G(u) = E^+_u \). The messages are sent to all outgoing edges. Each edge \( uv \in \mathcal{E} \) in the partition in line 10 of Algorithm 2 is represented by a separate set. That is, the partition reads \( \bigcup_{e \in E^-_u} \{ e \} \). Weights are distributed uniformly and equal to \( w_e = 1 / \max(\{1, |E^-_u|\}) \), \( e \in E^-_u \).

After each outer iteration, when all nodes were processed, the ordering is reversed and the process repeats. We refer to [49] for substantiation of these parameters.

Parameters for Example 2, Graph Matching. Additionally to the node and edge factors, the corresponding IRPS-LP has also label factors (7). To this end all node factors are ordered as in Example 1. Each node factor \( u \in \mathcal{V} \) receives messages from all incoming edge factors and label factors \( J_{\text{receive}}(u) = E^+_u \cup X_u \) and sends them to all outgoing edges and label factors. The corresponding partition reads \( \bigcup_{f \in \mathcal{N}_G(u) \setminus E^-_u} \{ f \} \cup X_u \). The weights are distributed uniformly with \( w_f = 1 / \max(\{1, |E^-_u|\}) \). The label factors are processed after all node factors were visited. Each label factor receives messages from all connected node factors and sends messages back as well: \( J_{\text{receive}}(s) = \{ u \in V : s \in X_u \} \). We use the same single set for sending messages, i.e. \( J_1 = J_{\text{receive}} \). After each iteration we reverse the factor order.

Parameters for Example 3, Multicut. Similarly to Example 1, it is sufficient to go over all edge factors in the loop of Algorithm 2, since each coupling constraint contains exactly one cycle and one edge factor. Each edge factor \( e \) receives messages from all coupled cycle factors \( J_{\text{receive}}(e) = \mathcal{N}_G(\{ c \in C : e \in c \}) \) and sends them to the same factors. As in Example 1, each cycle factor forms a trivial set in the partition in line 10 of Algorithm 2, the partition reads \( \bigcup_{e \in C \in C \subseteq e} \{ e \} \). Weights are distributed uniformly with \( w_e = 1 / \max(|C|, |C|) \). After each iteration the processing order of factors is reversed.

4.2. Obtaining Integer Solution

Eventually we want to obtain a primal solution \( x \in X \) of (1), not a reparametrization \( \theta^\phi \). We are not aware of any rounding technique which would work equally well for all possible instances of IRPS-LP problem. According to our experience, the most efficient rounding is problem specific. Below, we describe our choices for the Examples 1 – 3.
Rounding for Example 1 coincides with the one suggested in [48]: Assume we have already computed a primal integer solution $x^*_v$ for all $v < u$ and we want to compute $x^*_u$. To this end, right before the message receiving step of Algorithm 2 for $i = u$ we assign

$$x^*_u \in \arg \min_{x_u} \theta_u(x_u) + \sum_{v: u \sim v \in E} \theta_{uv}(x_u, x^*_v). \quad (16)$$

Rounding for Example 2 is the same except that we select the best label $x_u$ among those, which have not been assigned yet, to satisfy uniqueness constraints:

$$x^*_u \in \arg \min_{x_u: x_u \neq x_v \forall v < u} \theta^0_u(x_u) + \sum_{v: u \sim v \in E} \theta^0_{uv}(x_u, x^*_v). \quad (17)$$

Rounding for Example 3. We use the efficient Kernighan&Lin heuristic [45] as implemented in [46]. Costs for the rounding are the reparametrized edge potentials.

5. Fixed Points and Comparison to Subgradient Method

Algorithm 2 does not necessarily converge to the optimum of (1). Instead, it may get stuck in suboptimal points, similar to those corresponding to the “weak tree agreement” [48] or ”arc consistency” [74, 3] in CRFs from Example 1. Below we characterise these fixpoints precisely.

Definition 3 (Marginal Consistency). Given a reparametrization $\theta^0$, let for each factor $i \in \mathcal{F}$ a non-empty set $\mathbb{S}_i \subseteq \arg \min_{x_i \in X_i}(\theta^0, x_i)$, $i \in \mathcal{F}$ be given. Define $\mathbb{S} = \prod_{i \in \mathcal{F}} \mathbb{S}_i$. We call reparametrization $\theta^0$ marginally consistent for $\mathbb{S}$ on $ij \in \mathcal{E}$ if

$$A_{(i,j)}(\mathbb{S}_i) = A_{(j,i)}(\mathbb{S}_j). \quad (18)$$

If $\theta^0$ is marginally consistent for $\mathbb{S}$ on all $ij \in \mathcal{E}$, we call $\theta^0$ marginally consistent for $\mathbb{S}$.

Note that marginal consistency is necessary, but not sufficient for optimality of the relaxation (1). This can be seen in the case of CRFs (Example 1), where it exactly corresponds to arc-consistency. The latter is only necessary, but not sufficient for optimality [74].

Theorem 2. If $\theta^0$ is marginally consistent, the dual lower bound $D(\phi)$ cannot be improved by Algorithm 2.

Comparison to Subgradient Method. Decomposition IRPS-LP and more general ones can be solved via the subgradient method [50]. Similar to Algorithm 2, it operates on dual variables $\phi$ and manipulates them by visiting each factor sequentially. Contrary to Algorithm 2, subgradient algorithms converge to the optimum. Moreover, on a per-iterations basis, computing subgradients is cheaper than using Algorithm 2, as only (13) needs to be computed, while Algorithm 2 needs to solve (15) additionally. However, for MAP-inference, the study [41] has shown that subgradient-based algorithms converge much slower than message passing algorithms like TRWS [48]. In Section 6 we confirm this for the graph matching problem as well.

The reason for this large empirical difference is that one iteration of the subgradient algorithm only updates those coordinates of dual variables $\phi$ that are affected by the current minimal labeling $x^*_i \in \arg \min_{x_i \in X_i}(\theta^0, x_i)$ (i.e. coordinates $k: (A^T_{(i,j)}x^*_j)_k = 1$), while in Algorithm 2 all coordinates of $\phi$ are taken into account. Also message passing implicitly chooses the stepsize so as to achieve monotonically convergence in Algorithm 1, while subgradient based algorithms must rely on some stepsize rule.

6. Experimental Evaluation

Our experiments’ goal is to illustrate applicability of the proposed technique, they are not an exhaustive evaluation. The presented algorithms are only basic variants, which can be further improved and tuned to the considered problems. Both issues are addressed in the specialized studies [68, 67]. Still, we show that the presented basic variants are already able to surpass state-of-the-art specialized solvers on challenging datasets.

6.1. Graph Matching

Solvers. We compare against two state-of-the-art algorithms: (i) the subgradient based dual decomposition solver [70] abbreviated by DD and (ii) the recent “hungarian belief propagation” message passing algorithm [78], abbreviated as HBP. While the authors of [78] have embedded their solver in a branch-and-bound routine to produce exact solutions, we have reimplemented their message passing component but did not use branch and bound to make the comparison fair. Both algorithms DD and HBP outperformed alternative solvers [18, 76, 53, 69, 63, 29, 79, 38, 54, 16] at the time of their publication, hence we have not tested against them. We call our solver AMP.

Datasets. We selected three challenging datasets. The first two are the standard benchmark datasets car and motor, both used in [55], containing 30 pairs of cars and 20 pairs of motorbikes with keypoints to be matched 1:1. The images are taken from the VOC PASCAL 2007 challenge [21]. Costs are computed from features as in [55]. Instances are densely connected graphs with 20 – 60 nodes. The third one is the novel worms datasets [40], containing 30 problem instances coming from bioimageging. The problems are made of sparsely connected graphs with up to 600 nodes and up to 1500 labels. To our knowledge, the worms dataset contains the largest
The problems come from electron microscopy of brain tissue, for which we wish to obtain a neuron segmentation. Each dataset contains 8 instances with ≤ 972, 5896 and 17074 nodes and ≤ 5656, 36221, and 107060 edges respectively.

7. Acknowledgments

The authors would like to thank Vladimir Kolmogorov for helpful discussions. The first authors was supported by the European Research Council under the European Unions Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement no 616160, the second author by DFG Grant GRK 1653 and the last author by the European Research Council under the European Unions Horizon 2020 research and innovation programme (grant agreement No 647769) and the DFG Grant ”ERBI” SA 2640/1-1.

References


Results. Our solver AMP consistently outperforms HBP and DD w.r.t. primal/dual gap and anytime performance. Most markedly on the largest worms dataset, the subgradient based algorithm DD struggles hard to decrease the primal/dual gap, while AMP gives reasonable results.

6.2. Multicuts

Solvers. We compare against state-of-the-art multicut algorithms implemented in the OpenGM [41] library, namely (i) the branch-and-cut based solver MC-ILP [44] utilizing the ILP solver CPLEX [2], (ii) the heuristic primal “fusion move” algorithm CC-Fusion [9] with random hierarchical clustering and random watershed proposal generator, denoted by the suffixes -RHC and -RWS and (iii) the heuristic primal “Cut, Glue & Cut” solver CGC [10]. Those solvers were shown to outperform other multicut algorithms [9]. Algorithm MC-ILP provides both upper and lower bounds, while CC-Fusion and CGC are purely primal algorithms. We call our message passing solver with cycle constraints added in a cutting plane fashion MP-C.

Datasets. We have selected three datasets knott-3d-{150|300|450} from OpenGM [41].


[33] M. Guignard and M. B. Rosenwein. An application-oriented guide for designing Lagrangean dual ascent algorithms. Euro-


8. Supplementary Material

Proof of Proposition 1

Proposition. \( \sum_{i \in \mathcal{E}} \langle \theta_i, \mu_i \rangle = \sum_{i \in \mathcal{E}} \langle \theta_i^\phi, \mu_i \rangle \), whenever \( \mu_1, \ldots, \mu_k \) obey the coupling constraints.

Proof. \( \sum_{i \in \mathcal{E}} \langle \theta_i, \mu_i \rangle = \sum_{i \in \mathcal{E}} \langle \theta_i^\phi, \mu_i \rangle + \sum_{i \in \mathcal{E}} \langle \phi(i,j), A(i,j) \mu_i \rangle + \langle \phi(j,i), A(j,i) \mu_j \rangle = \sum_{i \in \mathcal{E}} \langle \theta_i^\phi, \mu_i \rangle \), where \((*)\) = 0 due to \( \phi(i,j) = -\phi(j,i) \) and \( A(i,j) \mu_i = A(j,i) \mu_j \).

Proof of Proposition 2

Proposition. Let \( \text{conv}(X_i) = \{ \mu_i : A_i \mu_i \leq b_i \} \) with \( A_i \in \mathbb{R}^{n_i \times m_i} \). Let the messages in problem (15) have size \( n_1, \ldots, n_{|J|} \). Then (15) is a linear program with \( O(n_1 + \ldots + n_{|J|}) \) variables and \( O(m_1 + \ldots + n_{|J|}) \) constraints.

Proof. From LP-duality we know that \( \mu_i^* \in \text{argmin}_{\mu_i : A_i \mu_i \leq b_i} \{ \langle c_i, \mu_i \rangle : \exists y \geq 0 : A_i^T y = c_i \} \). Hence, (15) can be rewritten as

\[
\max_{y \geq 0, \Delta(i,j_1), \ldots, \Delta(i,j_{|J|})} \langle \delta, \theta^{\phi+\Delta} \rangle \\
\text{s.t.} \\
\langle b_i - A_i \mu_i^*, y \rangle = 0 \\
A_i^T y = \theta^{\phi+\Delta} \\
\Delta(i,j)(s) \begin{cases} 
\leq 0, & \nu_i(s) = 0 \\
\geq 0, & \nu_i(s) = 1 
\end{cases} \\
\text{where } \nu_i := A(i,j) \mu_i^* 
\]

\( \theta^{\phi+\Delta} \) is a linear expression and \( \mu^* \) is constant during the computation, hence (19) is a LP.

Proof of Lemma 1 and Lemma 2

Lemma. Let \( ij \in \mathcal{E} \) be a pair of factors related by the coupling constraints and \( \phi(i,j) \) be a corresponding dual vector. Let \( x_i^* \in \text{argmin}_{x_i \in X_i} \langle \theta_i^\phi, x_i \rangle \) and \( \Delta(i,j) \) satisfy

\[
\Delta(i,j)(s) \begin{cases} 
\geq 0, & \nu(s) = 1 \\
\leq 0, & \nu(s) = 0 
\end{cases} \\
\text{where } \nu := A(i,j) x_i^* .
\]

Then \( x_i^* \in \text{argmin}_{x_i \in X_i} \langle \theta_i^{\phi+\Delta}, x_i \rangle \) implies \( D(\phi) \leq D(\phi + \Delta) \).

Proof. Let \( x_i^* \in \text{argmin}_{x_i \in X_i} \langle \theta_i^\phi, x_i \rangle \) be a solution of (13) at which the dual lower bound (11) is attained before the update and \( x_j^{\phi+\Delta} \in \text{argmin}_{x_j \in X_j} \langle \theta_j^{\phi+\Delta}, x_j \rangle \) be an integral solution at which the dual lower bound is attained after \( \phi \) has been updated. Variable \( x_i^* \) as chosen in (13) is optimal for \( \theta^\phi \) and for \( \theta^{\phi+\Delta} \) by construction. We need to prove

\[
\langle \theta_i^\phi, x_i^* \rangle + \sum_{j \in J} \langle \theta_j^\phi, x_j^* \rangle \\
\leq \langle \theta_i^{\phi+\Delta}, x_i^* \rangle + \sum_{j \in J} \langle \theta_j^{\phi+\Delta}, x_j^* \rangle.
\]

We shuffle all terms with variables \( \Delta(i,j)^\star, j \in J \) to the right side and all other terms to the left side.

\[
\langle \theta_i^\phi, x_i^* - x_i^\star \rangle + \sum_{j \in J} \langle \theta_j^\phi, x_j^* - x_j^\star \rangle \\
\leq \sum_{j \in J} \langle A^T(i,j) \Delta^\star(i,j), x_i^* \rangle - \sum_{j \in J} \langle A^T(j,i) \Delta^\star(i,j), x_j^* \rangle.
\]

All terms on the left side are smaller than zero due to the choice of \( x_i^\star \) being minimizers w.r.t. \( \theta_i^\phi \). Hence, it will be enough to prove the above inequality when assuming the left side to be zero. We rewrite the scalar products by transposing \( A^T(i,j) \) and \( A^T(j,i) \).

\[
0 \leq \sum_{j \in J} \left\{ \langle \Delta(i,j), A(i,j) x_i^* - A(j,i) x_j^\star \rangle \right\}
\]

Due to \( A(i,j) x_j^\star \in \{0,1\}^{\text{dim}(\phi(j,i))} \) and \( A(i,j) x_i^* \in \{0,1\}^{\text{dim}(\phi(i,j))} \) by Definition 1 and \( \Delta^\star(i,j) \leq 0 \) whenever \( A(i,j) x_i^\star \leq 0 \), the result follows.

Lemma. Let \( \Delta \in AD(\theta^\phi, x_i^*, J) \) then \( D(\phi) \leq D(\phi + \Delta) \).

Proof. Analogous to the proof of Lemma 1.

Proof of Theorem 1

Theorem. Algorithm 2 monotonically increases the dual lower bound (11).

Proof. We prove that (i) the receiving messages and (ii) the sending messages step improve (11).

(i) Directly apply Lemma 1. (ii) The difficulty here is that we compute descent directions from the current dual variables \( \phi \) in parallel and then apply all of them simultaneously. By Lemma 2, the send message step is non-decreasing when called for each set \( J_1, \ldots, J_k \) in Algorithm 2. The dual lower bound \( L(\phi) \) is concave, hence we apply Jensen’s inequality and note that \( \omega_1 + \ldots + \omega_k \leq 1 \) to obtain the result.

Proof of Theorem 2

Theorem. If \( \theta^\phi \) is marginally consistent, the dual lower bound \( D(\phi) \) cannot be improved by Algorithm 2.

First, we need two technical lemmata.
Lemma 3. Let $X \subseteq \{0, 1\}^n$, $A \in \{0, 1\}^{K \times n}$ and $Ax \in \{0, 1\}^K \forall x \in X$. Let $x^* \in X$ be given and define $\nu^* := Ax^*$. Let $A \in \mathbb{R}^K$ be given such that $\Delta(s) \geq 0$, $\nu^*(s) = 1$, and $\nu^*(s) = 0$. Then

(i) $x^* \in \text{argmin}_{x \in X} (-\Delta, Ax)$ and (ii) for $x^{**} \in \text{argmin}_{x \in X} (-\Delta, Ax)$, $\nu^{**} = Ax^{**}$ it holds that $\Delta(s) = 0$ whenever $\nu^*(s) \neq \nu^{**}(s)$.

Proof. Let $x \in X$ and define $\nu := Ax$. Then

$$\langle -\Delta, Ax \rangle = \sum_{s: \nu^*(s) = 1} -\Delta(s) + \sum_{s: \nu^*(s) > 0} -\Delta(s) \geq \sum_{s: \nu^*(s) = 1} -\Delta(s) \quad \text{(**) \geq 0 due to } \Delta(s) \geq 0 \text{ for } \nu^*(s) = 1 \text{ and } (*) \geq 0 \text{ due to } \Delta(s) \leq 0 \text{ for } \nu^*(s) = 0. \text{ This proves (i) and (ii) is proven by observing that } (*) = 0 \text{ and } (*) = (*) \text{ must also hold.}$$

Lemma 4. Let $x_i^*, x_i^{**} \in \text{argmin}_{x_i \in X_i} (\theta^\phi, x_i)$ be two solutions to the $i$-th factor for the current reparametrization $\theta^\phi$. If $A$ is admissible w.r.t. $x_i^*$ then $x_i^{**}$ is also admissible w.r.t. $x_i^{**}$.

Proof. As both $x_i^*$ and $x_i^{**}$ are optimal to $\theta^\phi$ and $x_i^*$ is also optimal to $\theta^{\phi+\Delta}$, we have $\langle \Delta(i,j), A(j,i)x_i^* \rangle \leq \langle \Delta(i,j), A(j,i)x_i^{**} \rangle$. By Lemma 3, (i) also $\langle -\Delta(i,j), A(j,i)x_i^* \rangle \leq \langle -\Delta(i,j), A(j,i)x_i^{**} \rangle$ holds, hence equality must hold. This shows $x_i^{**} \in \text{argmin}_{x_i \in X_i} (\theta^{\phi+\Delta}, x_i)$. Second, Lemma 3, (ii) implies that $\Delta(s) = 0$ whenever $\nu^*(s) \neq \nu^{**}(s)$. This proves

$$\langle -\Delta(i,j), A(j,i)x_i^{**} \rangle \geq 0, \nu^{**}(s) = 1 \quad \text{or} \quad \langle -\Delta(i,j), A(j,i)x_i^{**} \rangle \leq 0, \nu^{**}(s) = 0, \nu^{**} := A(j,i)x_i^{**}. \quad \text{□}$$

Proof of Theorem 2. It is sufficient to show that for marginally consistent $\theta^\phi$ for $S$, the update $\Delta$ computed by Algorithm 1 on an arbitrary factor $i \in F$ and some set $J \subseteq \mathcal{N}_{G}(i)$ has the following properties: (i) $L(\phi) = L(\phi + \Delta)$, (ii) $\theta^{\phi+\Delta}$ is marginally consistent for $S$. For an easier proof, we only consider the case $J = \{j\}$. The general case can be proven analogously.

(i) Let $x_i^* \in \mathcal{S}_i$, $x_j^* \in \mathcal{S}_j$ with $A(i,j)x_i^* = A(j,i)x_j^*$. We have to show that

$$\min_{x_i \in X_i} \langle \theta^\phi, x_i \rangle + \min_{x_j \in X_j} \langle \theta^\phi, x_j \rangle = \min_{x_i \in X_i} \langle \theta^{\phi+\Delta}, x_i \rangle + \min_{x_j \in X_j} \langle \theta^{\phi+\Delta}, x_j \rangle$$

Due to $x_i^*$ optimal to $\theta^{\phi+\Delta}$, since by Lemma 4 the update $\Delta$ is admissible for $x_i^*$, it remains to show that $x_i^* \in \text{argmin}_{x_i \in X_i} (\theta^{\phi+\Delta}, x_i)$. As $x_j^* \in \text{argmin}_{x_j \in X_j} (\theta^\phi, x_j)$, it is sufficient to prove that $x_j^* \in \text{argmin}_{x_j \in X_j} (\theta^{\phi+\Delta}, x_j)$. This follows from Lemma 3 (i). We conclude by noting $\langle \theta^\phi, x_i^* \rangle + \langle \theta^\phi, x_j^* \rangle = \langle \theta^{\phi+\Delta}, x_i^* \rangle + \langle \theta^{\phi+\Delta}, x_j^* \rangle$.

(ii) The computations in (i) show that $S_i \subseteq \text{argmin}_{x_i \in X_i} (\theta^{\phi+\Delta}, x_i)$ and $S_j \subseteq \text{argmin}_{x_j \in X_j} (\theta^{\phi+\Delta}, x_j)$. The reparametrizations of all other factors stay the same: $\theta^{\phi+\Delta} = \theta^\phi$ for $k \in F \setminus \{i, j\}$. Hence, $\theta^{\phi+\Delta}$ is marginally consistent for $S$ after the update. \hfill \text{□}

9. Special Cases: Graphical Model Solvers

We will show how Algorithm 2 subsumes known message-passing algorithms MSD [74], TRWS [48], SRMP [49] and MPLP [28] for MAP-inference with common graphical models, considered in Example 1.

Solver Primitives (13) and (15). As it can be seen, all factors in (5) are of the form $X_i = \{0, 1\}^n$ and $\text{conv}(X_i) = \{\mu \geq 0 : \langle 1, \mu \rangle = 1 \}$ and $\dim(X_i)$ is a dim(n) dimensional simplex.

In all message passing algorithms [48, 49, 74, 28], there are two types of invokations of Algorithm 1 together with solutions of the accompanying optimization problem (13) and (15):

<table>
<thead>
<tr>
<th>Alg. 1 input</th>
<th>Factor Optimization (13)</th>
<th>Reparameterization adjustment (15)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i = u \in V$</td>
<td>$J = {uv}$</td>
<td>$u \in V$</td>
</tr>
<tr>
<td>$j = {u}$</td>
<td>$x_u \in X_u$</td>
<td>$\text{min} {\theta^\phi_u(x_u)}$</td>
</tr>
<tr>
<td>$x_{uv,u} \in X_{uv,u}$</td>
<td>$\Delta^\phi_{uv,u}(x_{uv,u}) = \text{min} \theta^\phi_u(x_u)$</td>
<td>$- \theta^\phi_u(x_u)$</td>
</tr>
</tbody>
</table>

MAP-inference Solvers. In Table 1 we state solvers MSD [74], TRWS [48], SRMP [49] and MPLP [28] as special cases of our framework. Factors are visited in the order they are read in.
Algorithm Current factor $J_{\text{receive}}$ $J_1 \cup \ldots \cup J_l$ $\omega$

| MSD [74] $u \in V$ $uv \in E$ $N_G(u)$ $\emptyset$ $\{uv\} \subset N_G(u)$ $\omega_1, \ldots = 1/|N_G(u)|$
| MPLP [28] $u \in V$ $uv \in E$ $\emptyset$ $\{u, v\}$ $\{u\}, \{v\}$ $\omega_1 = 1/2 = \omega_2$
| TRWS [48] $u \in V$ $uv \in E$ $\{u, v\}$ $\emptyset$ $\omega_1, \ldots = 1/\max(\{v \in N_G(u) : v > u\}, \{v \in N_G(u) : v < u\})$
| SRMP [49] $u \in V$ $uv \in E$ $\emptyset$ $\{u, v\}$ $\{u\}, \{v\}$ $\omega_1, \ldots = 1/\max(\{v \in N_G(u) : v > u\}, \{v \in N_G(u) : v < u\})$

Table 1. [74, 48, 49, 28] as special cases of Algorithm 2.

**Remark 1.** We have only treated the case of unary $\theta_u$, $u \in V$ and pairwise potentials $\theta_{uv}$, $uv \in E$ here. MPLP [28] and SRMP [49] can be applied to higher order potentials as well, which we do not treat here. SRMP [49] is a generalisation of TRWS [48] to the higher-order case.

**Remark 2.** There are convergent message-passing algorithms such that factors comprise trees [72, 65]. Their analysis is more difficult, hence we omit it here.

Note that our framework generalizes upon [48, 49, 28, 74, 65, 72] in several ways: (i) Our factors need not be simplices or trees. (ii) Our messages need not be marginalization between unary/pairwise/triplet/… factors. (iii) We can compute message updates on more than one coupling constraint simultaneously, i.e. we may choose $J_1 \cup \ldots \cup J_l$ in Algorithm 2 to be different than singleton sets. (i) and (ii) affect LP-modeling, (iii) affects computational efficiency: By considering multiple messages at once in Procedure 1, we may be able to make larger updates $\Delta^*$, resulting in faster convergence.