Exact MAP-Inference by Confining Combinatorial Search with LP Relaxation

Stefan Haller[†], **Paul Swoboda**[‡], **Bogdan Savchynskyy**[†]
[†] University of Heidelberg, [‡] IST Austria
stefan.haller@iwr.uni-heidelberg.de

Abstract

We consider the MAP-inference problem for graphical models, which is a valued constraint satisfaction problem defined on real numbers with a natural summation operation. We propose a family of relaxations (different from the famous Sherali-Adams hierarchy), which naturally define lower bounds for its optimum. This family always contains a tight relaxation and we give an algorithm able to find it and therefore, solve the initial non-relaxed NP-hard problem.

The relaxations we consider decompose the original problem into two non-overlapping parts: an easy LP-tight part and a difficult one. For the latter part a combinatorial solver must be used. As we show in our experiments, in a number of applications the second, difficult part constitutes only a small fraction of the whole problem. This property allows to significantly reduce the computational time of the combinatorial solver and therefore solve problems which were out of reach before.

Introduction

This paper focuses on *energy minimization* or *maximum a posteriori (MAP) inference* for undirected graphical models. This problem is closely related to weighted and valued constraint satisfaction. In the most common pairwise case it amounts to minimizing a partially separable function $E_{\mathcal{G}}$ taking real values on a discrete set of finite-valued vectors $\mathcal{X}_{\mathcal{V}}$[1]

$$\min_{x \in \mathcal{X}_{\mathcal{V}}} \left[E_{\mathcal{G}}(\theta, x) := \sum_{u \in \mathcal{V}} \theta_u(x_u) + \sum_{uv \in \mathcal{E}} \theta_{uv}(x_u, x_v) \right]. \quad (1)$$

The problem is known to be NP-hard (e. g. Li, Shekhovtsov, and Huber 2016), and therefore a number of approximate algorithms were proposed to this end. In contrast, our goal is an efficient method able to solve large-scale, but mostly simple problem instances *exactly*. Such instances typically arise in computer vision, machine learning and other areas of artificial intelligence. Although approximate methods often provide reasonable solutions, having an exact solver can be quite critical at the modeling stage, when one has to differentiate between modeling and optimization errors. In this case one usually resorts to either specialized combinatorial solvers (see references in Kappes et al. 2015; Hurley et al. 2016) or off-the-shelf integer linear program (ILP) solvers like CPLEX (CPLEX, IBM 2014) or Gurobi (Gurobi Optimization 2016). However, neither specialized nor off-the-shelf solvers scale well, as the problem instances get larger. Our method is able to use the fact that a linear program (LP) relaxation of the problem is "almost" tight, i. e. the obtained solution is close to the optimal one. It restricts application of an exact solver to a small fraction of the problem, where the LP relaxation is not tight and yet obtains a provably optimal solution to the whole problem. This allows to solve problems for which no efficient solving technique was available.

Related work *LP relaxations* are an important building block for a number of algorithms addressing the MAP-inference problem (1). It was probably first considered in (Shlezinger 1976; see Werner 2007 for the recent review) both in its primal and dual form. The notion of *reparametrization* (known also as *equivalent transformations* or *equivalence preserving transformations*) was introduced in the same work as well. Although the bound provided by the LP relaxation is often good, the class of problems, where it is tight, is limited (see Kolmogorov, Thapper, and Zivny 2015). Practically important problems from this class are mainly those having acyclic structure or submodular costs. Therefore a number of works were devoted to *cutting plane* techniques to tighten the relaxation (e. g. Koster, Van Hoesel, and Kolen 1998; Sontag 2007; de Givry and Katsirelos 2017). Sometimes the tightening itself may lead to an exact solution, however, in general it is accomplished with *branch-and-bound* or A^* algorithms. The most prominent representative of the first class are the DAOOPT (Marinescu and Dechter 2005; Otten and Dechter 2010) and Toulbar2 (Cooper et al. 2010) solvers. The latter has recently shown impressive results on a number of benchmarks (Hurley et al. 2016). In contrast, the A^* algorithm so far was mainly used in specific applications (e. g. Bergtholdt et al. 2010).

Recently developed LP-relaxation-based *partial optimality methods* (e. g. Shekhovtsov 2014; Shekhovtsov, Swoboda, and Savchynskyy 2015; Swoboda et al. 2016) can find optimal labels for a significant part of variables without solving the combinatorial problem (1). Afterwards, a combinatorial solver can be applied to the rest of the variables to obtain a complete solution. These methods work well if the pairwise costs θ_{uv} play the role of a "smoothing regularizer" by slightly penalizing differences in values in neighboring vari-

[1] We rigorously define notation in Section "Preliminaries".

ables u and v. However, they struggle as the pairwise costs get more weight and move towards "hard constraints", when some pairs of variable values are strongly penalized or even forbidden.

The *CombiLP* method (Savchynskyy et al. 2013) is the closest to our work. It iteratively splits the problem into a "simple" and a "difficult" part based on consistency of reparametrized unary and pairwise costs, known as (virtual) arc-consistency (see Werner 2007), and checks for agreement of their solutions. The "simple" part is addressed with a dedicated LP solver, whereas the "difficult" one is solved with an ILP method. Although CombiLP has shown promising results on the OpenGM benchmark (Kappes et al. 2015), its usage is beneficial for sparse graphical models only, when $|\mathcal{E}| \ll |\mathcal{V}|^2$.

Contribution Based on CombiLP, we propose a method, which is not restricted to sparse models. Similar to CombiLP, we split the problem into LP and ILP parts based on local consistency properties. Our new consistency criterion guarantees that the concatenation of the obtained LP and ILP solutions is optimal for the whole problem, given that the criterion is satisfied. When the criterion is not satisfied, we increase the ILP subproblem and correspondingly decrease the LP one, like it is done in CombiLP. There are several crucial differences to the CombiLP approach, however:

- Our "difficult" ILP subproblem is kept much more compact, which is critical for densely-connected graphs. This leads to substantial computational savings.
- Our optimality criterion is stronger than those of CombiLP: Satisfaction of CombiLP's criterion for a given splitting implies satisfaction of ours.

Additionally, we treat the problem of an initial reparametrization suitable for the used splitting criterion and propose a method, which allows to use *arbitrary* dual LP solvers within our algorithm, whereas the CombiLP implementation has a *fixed* dedicated LP solver. This allowed us to choose a more efficient LP solver and to significantly (up to 18 times) speed up the original CombiLP implementation.

Finally, our criterion and implementation[2] are also able to deal with higher order models, which intrinsically have a higher connectivity. We show efficacy of our method on publicly available benchmarks from computer vision, machine learning and bio-imaging.

Preliminaries

Graphical Models and MAP-inference Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph with *the set of nodes* $\mathcal{V}$ and *the set of edges* $\mathcal{E}$. The neighborhood of $u \in \mathcal{V}$ is defined as $\mathrm{Nb}(u) := \{v \in \mathcal{V} \mid uv \in \mathcal{E}\}$. Each node $v \in \mathcal{V}$ is associated with a finite *set of labels* $\mathcal{X}_v$. For any subset $\mathcal{V}' \subseteq \mathcal{V}$ of graph nodes the Cartesian product $\mathcal{X}_{\mathcal{V}'} = \prod_{v \in \mathcal{V}'} \mathcal{X}_v$ defines *the set of labelings* of the subset $\mathcal{V}' \subseteq \mathcal{V}$, when each node from $\mathcal{V}'$ is assigned a label. This includes also the special cases $\mathcal{V}' = \{u, v\} \in \mathcal{E}$ and $\mathcal{V}' = \mathcal{V}$ denoted as $\mathcal{X}_{uv}$ and $\mathcal{X}_{\mathcal{V}}$ respectively. We assume that $\arg\min$ stands for the set of minimal elements. At the same time, when used with "=" or ":=" operators, it returns some element from this set.

Let $\mathcal{I} = \{s \in \mathcal{X}_u \mid u \in \mathcal{V}\} \cup \{(s,t) \in \mathcal{X}_{uv} \mid uv \in \mathcal{E}\}$ be the set of indices enumerating all labels and label pairs in neighboring graph nodes. For each node and edge the *cost* functions $\theta_u : \mathcal{X}_u \to \mathbb{R}$, $u \in \mathcal{V}$ and $\theta_{uv} : \mathcal{X}_{uv} \to \mathbb{R}$, $uv \in \mathcal{E}$ assign a cost to a label or label pair respectively. The vector $\theta \in \mathbb{R}^{\mathcal{I}}$ contains all values of the functions θ_u and θ_{uv} as its coordinates.

ILP formulation and LP relaxation One way to address the MAP-inference problem (1) is to consider its ILP formulation (see e. g. Shlezinger 1976; Werner 2007)

$$\min_{\mu \geq 0} \langle \theta, \mu \rangle$$

$$\sum\nolimits_{s \in \mathcal{X}_u} \mu_u(s) = 1, \; u \in \mathcal{V} \tag{2}$$

$$\sum\nolimits_{s \in \mathcal{X}_u} \mu_{uv}(s,t) = \mu_v(t), \; uv \in \mathcal{E}, t \in \mathcal{X}_v$$

$$\mu \in \{0,1\}^{\mathcal{I}} \tag{3}$$

A natural LP relaxation is obtained by omitting the integrality constraints (3). The resulting LP (2) is known as a *local polytope* (Werner 2007) or simply *an LP relaxation* of (1). We will call the problem (1) LP-tight, if the optimal values of (1) and its LP relaxation (2) coincide. This also implies that there is an integer solution to the relaxed problem (2). We will say that the LP relaxation *has an integer solution* in a node u if there is $s \in \mathcal{X}_u$ such that $\mu_u(s) = 1$. Due to constraints of (2) it implies that $\mu_u(s') = 0$ for $s' \in \mathcal{X}_u \setminus \{s\}$.

Linear programs of the form (2) are as difficult as linear programs in general (Prusa and Werner 2013) and therefore obtaining exact solutions for large-scale instances may require significant time. However, there are fast specialized solvers (e. g. Kolmogorov 2006; Cooper et al. 2008) returning approximate dual solutions of (2).

Partial Optimality Observation Practical importance of the LP relaxation (2) is based on the fact that often most coordinates of its (approximate) relaxed solution are assigned integer values. The non-integer coordinates can be rounded (Ravikumar, Agarwal, and Wainwright 2010) and the resulting labeling can be used as if it was a solution of the non-relaxed problem. A number of problems have been successfully addressed with this type of methods (Kappes et al. 2015). However, apart from special cases (e. g. Boros and Hammer 2002; Rother et al. 2007) there is no guarantee that the integer coordinates keep their values in an optimal solution of the non-relaxed problem.

Even though there is no guarantee that the rounded integer solution is a sensible approximation for the optimal solution, empirical tests have shown that usually many integer coordinates coincide with the ones found in the optimal solution. This is a purely practical observation with little theoretical background. Nevertheless, this observation can be used to address the non-relaxed problem efficiently and it is a basis of our method. An alternative, the *partial optimality* approach was pursued by e. g. Shekhovtsov, Swoboda, and Savchynskyy 2015; Swoboda et al. 2016. We will provide a corresponding empirical comparison later in the paper.

[2]Code is available at `github.com/fgrsnau/combilp`.

Idea of the Algorithm

Graph partition A subgraph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ of the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is called *induced* by the set of its nodes $\mathcal{V}'$, if $\mathcal{E}' = \{uv \in \mathcal{E} \mid u, v \in \mathcal{V}'\}$, i. e. the set $\mathcal{E}'$ of its edges contains all edges from $\mathcal{E}$ connecting nodes from $\mathcal{V}'$.

The subgraphs $\mathcal{A} = (\mathcal{V}_\mathcal{A}, \mathcal{E}_\mathcal{A})$ and $\mathcal{B} = (\mathcal{V}_\mathcal{B}, \mathcal{E}_\mathcal{B})$ are called *partition* of the graph $\mathcal{G}$, if $\mathcal{V}_\mathcal{A} \cup \mathcal{V}_\mathcal{B} = \mathcal{V}$, $\mathcal{V}_\mathcal{A} \cap \mathcal{V}_\mathcal{B} = \emptyset$ and $\mathcal{A}$ and $\mathcal{B}$ are induced by $\mathcal{V}_\mathcal{A}$ and $\mathcal{V}_\mathcal{B}$ respectively. The subgraph $\mathcal{B}$ as complement to $\mathcal{A}$ will be denoted as $\mathcal{G} \setminus \mathcal{A}$. The other way around, $\mathcal{G} \setminus \mathcal{B}$ stands for $\mathcal{A}$, if $\mathcal{A}, \mathcal{B}$ is a partition of $\mathcal{G}$. Notation $\mathcal{E}_{\mathcal{AB}}$ will be used for the set of edges connecting $\mathcal{V}_\mathcal{A}$ and $\mathcal{V}_\mathcal{B}$: $\mathcal{E}_{\mathcal{AB}} = \{uv \in \mathcal{E} \mid u \in \mathcal{V}_\mathcal{A}, v \in \mathcal{V}_\mathcal{B}\}$.

In the following, we will show how to partition the problem graph $\mathcal{G}$ into (i) an easy part with subgraph $\mathcal{A}$, which can be solved exactly with approximate LP solvers and (ii) a difficult part with subgraph $\mathcal{B}$, which will require an ILP solver.

Lower bound induced by partition Till the end of this section we will assume $\mathcal{A}$, $\mathcal{B}$ are a partition of a graph $\mathcal{G}$. For the sake of notation, when considering different subgraphs of $\mathcal{G}$ we will nevertheless use a cost vector θ corresponding to the master graph $\mathcal{G}$, i. e. $E_\mathcal{A}(\theta, x)$ will stand for $\sum_{u \in \mathcal{V}_\mathcal{A}} \theta_u(x_u) + \sum_{uv \in \mathcal{E}_\mathcal{A}} \theta_{uv}(x_u, x_v)$, where $\theta \in \mathbb{R}^\mathcal{I}$.

Additionally, for $x' \in \mathcal{X}_\mathcal{A}$ and $x'' \in \mathcal{X}_\mathcal{B}$, their concatenation $x' \circ x'' \in \mathcal{X}_\mathcal{G}$ will be defined as

$$(x' \circ x'')_v = \begin{cases} x'_v\,, & v \in \mathcal{V}_\mathcal{A}\,, \\ x''_v\,, & v \in \mathcal{V}_\mathcal{B}\,. \end{cases} \tag{4}$$

Note that the energy function $E_\mathcal{G}$ can be decomposed into subproblems on $\mathcal{A}$ and $\mathcal{B}$ and it holds

$$E_\mathcal{G}(\theta, x' \circ x'') = E_\mathcal{A}(\theta, x') + E_\mathcal{B}(\theta, x'') + \sum_{uv \in \mathcal{E}_{\mathcal{AB}}} \theta_{uv}(x'_u, x''_v) \tag{5}$$

and therefore,

$$E_\mathcal{G}(\theta, x' \circ x'') \geq E_\mathcal{A}(\theta, x') + E_\mathcal{B}(\theta, x'') + \\ + \sum_{uv \in \mathcal{E}_{\mathcal{AB}}} \min_{(s,t) \in \mathcal{X}_{uv}} \theta_{uv}(s,t) \tag{6}$$

constitutes a lower bound for the energy function $E_\mathcal{G}$.

Proposition 1 (Sufficient optimality condition)**.** *The lower bound specified in* (6) *is tight if for all* $uv \in \mathcal{E}_{\mathcal{AB}}$ *it holds that* $(x'_u, x''_v) \in \arg\min_{(s,t) \in \mathcal{X}_{uv}} \theta_{uv}(s,t)$*, where* $x' \in \arg\min_{x \in \mathcal{X}_\mathcal{A}} E_\mathcal{A}(\theta, x)$*,* $x'' \in \arg\min_{x \in \mathcal{X}_\mathcal{B}} E_\mathcal{B}(\theta, x)$*.*

It is trivial to show that the labeling $x' \circ x''$ is optimal for $E_\mathcal{G}$ if the lower bound (6) is tight.

When considering the set of all possible partitions of $\mathcal{G}$ into $\mathcal{A}$ and $\mathcal{B}$ there is always at least one that leads to a tight lower bound (6). It corresponds to a trivial partition, where either the subgraph $\mathcal{A}$ or $\mathcal{B}$ is empty. The first case corresponds to solving the whole problem with an ILP method, whereas the second one corresponds to the case when the LP relaxation is tight, i. e. all coordinates of an LP solution are integer.

However, as our experimental evaluation shows, there exist often tight non-trivial partitions, with a large subgraph $\mathcal{A}$ and a small subgraph $\mathcal{B}$.

Conceptual Algorithm These partitions can be obtained for example by *a conceptual* Algorithm 1, which assigns all nodes of the graph having an integer solution to $\mathcal{A}$ and all others to $\mathcal{B}$. After solving both subproblems one checks fulfillment of the sufficient optimality condition defined by Proposition 1. Should the condition hold, the problem is solved. Otherwise one increases the subproblem $\mathcal{B}$ (and respectively decreases $\mathcal{A}$) by including those nodes $u \in \mathcal{V}_\mathcal{A}$, where the condition $(x'_u, x''_v) \in \arg\min_{(s,t) \in \mathcal{X}_{uv}} \theta_{uv}(s,t)$ does not hold for at least one $v \in \mathcal{V}_\mathcal{B}$, in terms of Proposition 1.

Algorithm 1 Conceptual Dense-CombiLP Algorithm

```
Solve LP relaxation (2)
Assign all nodes with integer solution to A
repeat
    Set B := (G \ A)
    Compute optimal labelings x' and x'' on A and B
        respectively. Use LP solver for A and ILP for B.
    if (Prop. 1 holds for x' and x'') then
        return (x' ∘ x'')
    else
        Move those u from A to B, where Prop. 1 fails
    end if
until B = G
```

Relation to CombiLP Algorithm 1 differs from CombiLP (Savchynskyy et al. 2013) in one very important aspect. Namely, the subgraphs used in CombiLP are overlapping, whereas ours are not. This substantially improves performance of the method in cases when the graph $\mathcal{G}$ has a high connectivity. In later sections of this paper we will give a detailed theoretical and empirical comparison of the methods.

In the following, we will turn the conceptual algorithm into a working one. In order to do so, we will give positive answers to a number of important questions:

- Why and when is the subproblem on $\mathcal{A}$ LP-tight? This is critical, since we assume $\mathcal{A}$ to be close to $\mathcal{G}$ in its size and therefore it must be solvable by a (polynomial) LP method.
- Can we avoid running an LP solver for $\mathcal{A}$ in each iteration?
- Can we use (fast specialized) approximate LP solvers on $\mathcal{A}$ instead of (slow off-the-shelf) exact ones?
- How to encourage conditions of Proposition 1 to be fulfilled for a possibly small $\mathcal{B}$?

Although our construction mostly follows the one given in (Savchynskyy et al. 2013), we repeat it here to keep the paper self-contained.

Theoretical Background

Reparametrization Decompositions of the energy function $E_\mathcal{G}(\theta, x)$ into unary and pairwise costs are not unique, which is, there exist other costs $\theta' \in \mathbb{R}^\mathcal{I}$ such that $E_\mathcal{G}(\theta, x) = E_\mathcal{G}(\theta', x)$ for all labelings $x \in \mathcal{X}_\mathcal{V}$. It is known (see e. g. Werner 2007) and straightforward to check that such *equivalent* costs can be obtained with an arbitrary vector $\phi = (\phi_{u,v}(s) \in \mathbb{R} \mid u \in \mathcal{V},\ v \in \mathrm{Nb}(u),\ s \in \mathcal{X}_u)$

as follows:

$$\theta'_u(s) \equiv \theta^\phi_u(s) := \theta_u(s) - \sum_{v \in \mathrm{Nb}(u)} \phi_{u,v}(s) \qquad (7)$$

$$\theta'_{uv}(s,t) \equiv \theta^\phi_{uv}(s,t) := \theta_{uv}(s,t) + \phi_{u,v}(s) + \phi_{v,u}(t)\,.$$

The costs θ^ϕ are called *reparametrized* and the vector ϕ is known as a *reparametrization*. Costs related by (7) are also called *equivalent*. In this sense, all vectors $\theta \in \mathbb{R}^{\mathcal{I}}$ can be split into equivalence classes according to (7). Other established terms for reparametrizations are *equivalence preserving transformations* (Cooper and Schiex 2004) and *equivalent transformations* (Shlezinger 1976).

Dual Problem By swapping the $\min$ and $\sum$ operations in (1) one obtains a lower bound $D(\theta) \leq E_{\mathcal{G}}(\theta, x)$ to the energy[3], which reads as

$$D(\theta) := \sum_{u \in \mathcal{V}} \min_{s \in \mathcal{X}_u} \theta_u(s) + \sum_{uv \in \mathcal{E}} \min_{(s,t) \in \mathcal{X}_{uv}} \theta_{uv}(s,t)\,. \qquad (8)$$

Although the energy $E_{\mathcal{G}}(\theta, x)$ remains the same for all cost vectors from a given equivalence class ($E_{\mathcal{G}}(\theta, x) = E_{\mathcal{G}}(\theta^\phi, x)$), the lower bound $D(\theta)$ is dependent on the reparametrization ($D(\theta) \neq D(\theta^\phi)$). Therefore, a natural maximization problem arises as maximization of the lower bound over all equivalent costs: $\max_\phi D(\theta^\phi)$. It is known (e. g. Werner 2007) that this maximization problem is equivalent to the Lagrangian dual to the LP relaxation (2). In turn, this implies that the minimum of (2) coincides with the maximum of $D(\theta^\phi)$. Therefore, one speaks about *optimal reparametrizations* as those ϕ, where the maximum is attained. Apart from its lower bound property the function $D(\theta^\phi)$ is important because (i) function $D(\theta^\phi)$ is concave w. r. t. ϕ as a sum of minima of linear functions; (ii) there exist many of scalable and efficient algorithms for its (approximate) maximization, e. g. (Kolmogorov 2006; Cooper et al. 2008).

Strict Arc-Consistency From a practical point of view it is important how an optimal reparametrization ϕ can be translated into a labeling, i. e. into an (approximate) solution of the energy minimization problem (1). The following definition plays a crucial role for this question in general and for our method in particular:

Definition 1 (Strict arc-consistency)**.** The node $u \in \mathcal{V}$ is called *strictly arc-consistent* w. r. t. the costs θ if there exists a label $x_u \in \mathcal{X}_u$ and labels $x_v \in \mathcal{X}_v$ for all $v \in \mathrm{Nb}(u)$, such that it holds (i) $\theta_u(x_u) < \theta_u(s)$ for all $s \in \mathcal{X}_u \setminus \{x_u\}$; and (ii) $\theta_{uv}(x_u, x_v) < \theta_{uv}(s,t)$ for all $(s,t) \in \mathcal{X}_{uv} \setminus \{(x_u, x_v)\}$.

The *set of strictly arc-consistent nodes* is denoted by $\mathcal{S}(\theta) := \{v \in \mathcal{V} \mid v \text{ is strictly arc-consistent}\}$.

If all nodes are strictly arc-consistent w. r. t. the reparametrized costs θ^ϕ, then it is straightforward to check that $D(\theta^\phi) = E_{\mathcal{G}}(\theta^\phi, x^*) = E_{\mathcal{G}}(\theta, x^*)$, where

$$x^*_u = \arg\min_{s \in \mathcal{X}_u} \theta^\phi_u(s)\,. \qquad (9)$$

In turn, this implies that ϕ is an optimal reparametrization and x^* is an exact solution of the energy minimization problem (1).

[3]It can be shown that this bound is in general less tight than (6).

Reconstructing labeling from reparametrization Although there is no guarantee that the strict arc-consistency property holds for all nodes even with an optimal reparametrization, the rule (9) is still used to obtain an approximate minimizer for (1) with arbitrary, also non-optimal reparametrizations ϕ (although, a number of more sophisticated rules were proposed, they are based on (9) and reduce to it if the strict arc-consistency holds for all nodes, see e. g. Ravikumar, Agarwal, and Wainwright 2010). Moreover, for an optimal reparametrization ϕ, when the strict arc-consistency holds for a node u, the complementary slackness conditions imply (e. g. Werner 2007) that strict arc-consistency of a node u guarantees an integer solution of the LP relaxation in u.

From the application point of view, an (approximate) solution (9) is typically considered as good, if most of the nodes $u \in \mathcal{V}$ satisfy the strict arc-consistency property. At the same time, unless the strict arc-consistency holds for *all* nodes, there is in general no theoretical guarantee that x^*_u obtained as (9) coincide with the corresponding coordinate y_u of an optimal solution $y = \arg\min_{x \in \mathcal{X}_\mathcal{V}} E_{\mathcal{G}}(\theta, x)$, even if the node u is strictly arc-consistent.

Algorithm 2 described in the next section, provides such guarantees by solving the non-relaxed minimization problem (1).

Detailed Algorithm Description

Let us consider Algorithm 2. It differs from Algorithm 1 provided above in several aspects: Instead of solving the relaxed problem (2) in the primal domain, it solves its dual formulation and resorts to the optimally reparametrized costs. Strict arc-consistency is used in place of integrality to form the initial set $\mathcal{A}$, which is justified by the fact that strict arc-consistency is sufficient for integrality.

The reparametrization step in line 2 plays a crucial role for the whole method. Due to this step, solving the energy minimization problem on $\mathcal{A}$ becomes trivial because of its strict arc-consistency. It can be performed by selecting the best label in each node independently, according to (9). Therefore, *there is no computational overhead* of resolving the problem on $\mathcal{A}$ in each iteration. Also, as more and more nodes from the initial subgraph $\mathcal{A}$ move over to the subgraph $\mathcal{B}$ their strict arc-consistency *encourages* solution on $\mathcal{B}$ to coincide with the locally optimal labels. Moreover, instead of an optimal dual solution ϕ *any*, also *approximate*, *non-optimal* reparametrization can be used. According to Proposition 1, this does not affect correctness of Algorithm 2. Therefore, *approximate solvers can be used* in line 1 of the algorithm. However, the better the dual solution is, the larger the set of strictly arc-consistent nodes $\mathcal{S}(\theta)$ is and therefore, the lower computational complexity of the ILP phase of the algorithm. Finally, reparametrization of the costs typically speeds up the ILP solver in line 6, as it serves as preprocessing.

Analysis of the Method

Family of Tight Partitions The proposition below that if the sufficient optimality criterion (Proposition 1) of Algorithm 2 is fulfilled for a partition $\mathcal{A}, \mathcal{B}$, then for any other partition $\mathcal{A}', \mathcal{B}'$ such that $\mathcal{B}' \supseteq \mathcal{B}$ the criterion holds as well:

Algorithm 2 Dense-CombiLP Algorithm

1: Maximize the dual (8) $\phi := \arg\max_{\psi} D(\theta^{\psi})$
2: Switch to the reparametrized costs: $\theta := \theta^{\phi}$
3: Induce $\mathcal{A}$ from $\mathcal{V}_{\mathcal{A}} := \mathcal{S}(\theta)$ and get optimum on $\mathcal{A}$:
$x'_u := \arg\min_{s \in \mathcal{X}_u} \theta_u(s), u \in \mathcal{V}_{\mathcal{A}}$
4: **repeat**
5: Set $\mathcal{B} := (\mathcal{G} \setminus \mathcal{A})$
6: Compute an optimal labeling x'' on $\mathcal{B}$ with ILP solver
7: $O_{uv} := \arg\min_{(s,t) \in \mathcal{X}_{uv}} \theta_{uv}(s,t)$, see Prop. 1
8: **if** $\forall uv \in \mathcal{E}_{\mathcal{AB}} : (x'_u, x''_v) \in O_{uv}$ **then**
9: **return** $(x' \circ x'')$
10: **else**
11: $\mathcal{V}_{\mathcal{A}} := \mathcal{V}_{\mathcal{A}} \setminus \{u \mid \exists uv \in \mathcal{E}_{\mathcal{AB}} : (x'_u, x''_v) \notin O_{uv}\}$
12: Induce $\mathcal{A}$ from $\mathcal{V}_{\mathcal{A}}$
13: **end if**
14: **until** $\mathcal{B} = \mathcal{G}$

Proposition 2. *Let $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{A}'$, $\mathcal{B}'$ be partitions of $\mathcal{G}$ s. t. $\mathcal{B} \subseteq \mathcal{B}'$. Let also $\mathcal{V}_{\mathcal{A}} \subseteq \mathcal{S}(\theta)$. Let x', x'' be the solutions of the MAP-inference problem* (1) *on $\mathcal{A}$ respectively $\mathcal{B}$ and y', y'' analogously for $\mathcal{A}'$ and $\mathcal{B}'$. If $x' \circ x''$ is the unique optimal assignment and it fulfills requirements of Proposition 1, then they are fulfilled for $y' \circ y''$ as well.*

This property shows that there are potentially many partitions, which results in a tight bound and allows to apply a greedy strategy for growing the subgraph $\mathcal{B}$ by adding all inconsistent nodes (violating Proposition 1) at once, as it is done in line 11 of Algorithm 2.

Comparison to CombiLP As mentioned above, the CombiLP-method is very similar to ours, but uses a different optimality criterion. Below we show that our criterion is in a certain sense stronger than theirs. To this end, following (Savchynskyy et al. 2013), we introduce the notion of a *boundary complement subgraph*:

Definition 2 (Savchynskyy et al. 2013)**.** Let $\mathcal{A}$ be an induced subgraph of $\mathcal{G}$. A subgraph $\mathcal{B}$ is called *boundary complement to $\mathcal{A}$ w. r. t. $\mathcal{G}$* if it is induced by the set $(\mathcal{V}_{\mathcal{G}} \setminus \mathcal{V}_{\mathcal{A}}) \cup \mathcal{V}_{\partial A}$, where $\mathcal{V}_{\partial A} = \{v \in \mathcal{V}_{\mathcal{A}} \mid \exists uv \in \mathcal{E}_{\mathcal{G}} : u \in \mathcal{V}_{\mathcal{G}} \setminus \mathcal{V}_{\mathcal{A}}\}$ is the set of nodes in $\mathcal{A}$ incident to nodes outside $\mathcal{A}$.

The optimality criterion used in CombiLP reads:

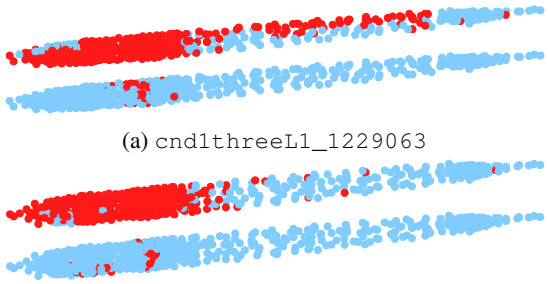
Theorem 1 (Savchynskyy et al. 2013)**.** *Let $\mathcal{A}$ be a subgraph of $\mathcal{G}$ and $\mathcal{B}$ be its boundary complement w. r. t. $\mathcal{A}$. Let $x_{\mathcal{A}}$ and $x'_{\mathcal{B}}$ be labelings minimizing $E_{\mathcal{A}}$ and $E_{\mathcal{B}}$ respectively* and *let all nodes $v \in \mathcal{V}_{\mathcal{A}}$ be strictly arc-consistent. Then from*

$$x_v = x'_v \text{ for all } v \in \mathcal{V}_{\partial A} \tag{10}$$

it follows that the labeling x^* with coordinates

$$x^*_v = \begin{cases} x_v, & v \in \mathcal{A} \\ x'_v, & v \in \mathcal{B} \setminus \mathcal{A} \end{cases}, \; v \in \mathcal{V}_{\mathcal{G}}, \text{ is optimal on } \mathcal{G}.$$

As can be seen from comparing Proposition 1 and Theorem 1, the main difference between the methods is that we use a partition of the graph $\mathcal{G}$, i. e. non-intersecting subgraphs, whereas the subgraphs in CombiLP are boundary complement and therefore intersect.



(a) `cnd1threeL1_1229063`

(b) `unc54L1_0123071`

Figure 1: Visualization of the maximal ILP subproblem for `worms`, where dots correspond to cell nuclei. Red dots are part of the ILP subproblem ($\bullet \in \mathcal{V}_{\mathcal{B}}$) and for blue dots the solution of the LP-relaxation is used ($\bullet \in \mathcal{V}_{\mathcal{A}}$). For both instances, the upper image shows the result for `clp` and the lower one corresponds to `dclp`, see the experimental evaluation section for information about the solvers.

The following proposition states that the bounds produced by our method are at least as tight as those of CombiLP:

Proposition 3. *Let $\mathcal{A}, \mathcal{B}$ be a partition of a graph $\mathcal{G}$ and $\mathcal{A}', \mathcal{B}$ be boundary complement for $\mathcal{G}$ and $\mathcal{A} \subseteq \mathcal{A}' \subseteq \mathcal{S}(\theta)$. Let also x', x'' be optimal labelings on $\mathcal{A}'$ and $\mathcal{B}$. If the condition* (10) *holds for x' and x'', i. e. $x'_v = x''_v$ for all $v \in \mathcal{V}_{\partial A'}$, then Proposition 1 holds for $x'_{\mathcal{A}}$ and x'' as well, where $x'_{\mathcal{A}}$ is the restriction of x' to the set $\mathcal{V}_{\mathcal{A}}$. In other words, for the same subgraph $\mathcal{B}$ fulfillment of Theorem 1 implies fulfillment of Proposition 1.*

Technical Details

Post-Processing of Reparametrization The maximum of the dual objective $D(\theta^{\phi})$ is typically non-unique. Since D is a concave function, the set of its maxima is convex and therefore it contains either a unique element or a continuum. Unfortunately, not all optimal (or suboptimal ones, corresponding to the same value of D) reparametrizations are equally good for our method. Moreover, different dual algorithms return different reparametrizations and the fastest algorithm may not return an appropriate one.

Therefore, we developed a post-processing algorithm to turn an arbitrary reparametrization into a suitable one without decreasing the value of D. This algorithm consists of two steps: (i) several iterations of a message passing (dual block-coordinate ascent) algorithm, which accumulates weights in unary costs and (ii) partial redistribution of unary costs between incident pairwise cost functions. This two-step procedure empirically turns most of the nodes, where the LP relaxation (2) has an integer solution, into strictly arc-consistent ones. The details of both steps are described in the supplement.

Higher Order Extensions All discussed techniques are easily extended to the higher-order MAP-inference problem

$$\min_{x \in \mathcal{X}_{\mathcal{V}}} \left[E_{\mathcal{G}}(\theta, x) := \sum_{c \in \mathcal{C}} \theta_c(x_c) \right]. \tag{11}$$

dataset (avg. $\lvert V\rvert$)		popt	clp	dclp
worms	(558)	100%	69.30%	**26.08%**
protein-folding	(37)	79.22%	100%	**71.03%**
color-seg	(79k)	12.10%	0.16%	**0.06%**
mrf-stereo	(138k)	45.19% (53.30%)	33.58% (0.45%)	**33.49%** **(0.20%)**
OnCallRostering	(948)	—	98.80%	**65.68%**

Table 1: Average size of the final ILP subproblem (smaller is better). The table shows the percentage of *labels* that reside in the ILP subproblem. The measure makes comparable clp and dclp, which work nodewise, and popt working labelwise. See supplement for details. Values in parentheses for mrf-stereo show the percentage only for the two problem instances, which were solved by clp and dclp.

where the cliques $c \in \mathcal{C} \subseteq 2^{\mathcal{V}}$ in the decomposition of the energy function $E_{\mathcal{G}}(\theta, x)$ may contain terms dependent on 3, 4 and more nodes. The bound (6) in the higher-order case reads as

$$E_{\mathcal{G}}(\theta, x' \circ x'') \geq E_{\mathcal{A}}(\theta, x') + \\ + E_{\mathcal{B}}(\theta, x'') + \sum_{c \in \mathcal{C}_{\mathcal{AB}}} \min_{y_c \in \mathcal{X}_c} \theta_c(x_c)\,, \quad (12)$$

where $\mathcal{C}_{\mathcal{AB}} := \{c \in \mathcal{C} \mid \exists u \in \mathcal{V}_{\mathcal{A}}, v \in \mathcal{V}_{\mathcal{B}} \colon u, v \in c\}$ similar to $\mathcal{E}_{\mathcal{AB}}$ in the pairwise case. Proposition 1 for the higher-order case turns into:

Proposition 4. *Let* $x' \in \arg\min_{x \in \mathcal{X}_{\mathcal{A}}} E_{\mathcal{A}}(\theta, x)$ *and* $x'' \in \arg\min_{x \in \mathcal{X}_{\mathcal{B}}} E_{\mathcal{B}}(\theta, x)$. *The lower bound* (12) *is tight if for all* $c \in \mathcal{C}_{\mathcal{AB}}$ *and* $v \in c$ *it holds that* $y^*_v = \begin{cases} x'_v\,, & v \in \mathcal{V}_{\mathcal{A}} \\ x''_v\,, & v \in \mathcal{V}_{\mathcal{B}} \end{cases}$ *where* $y^* \in \arg\min_{y \in \mathcal{X}_c} \theta_c(y)$.

The proof follows the same reasoning as the proof of Proposition 1 and is omitted here.

Experimental Evaluation

Algorithms In this section we compare our proposed algorithm with other related methods. As baselines we use CPLEX 12.6.2 (CPLEX, IBM 2014) and ToulBar2 0.9.8.0 (Cooper et al. 2010) where the first is the well-known commercial optimizer and the latter is one of the best dedicated branch-and-bound solvers for (1), see comparison in (Hurley et al. 2016). We used comparable parameters and settings like the ones used in (Hurley et al. 2016). They are denoted by cpx or tb2 respectively. The original CombiLP (Savchynskyy et al. 2013) implementation is referred as clp-orig. For a fair comparison, we modified it to make it compatible with arbitrary LP and ILP solvers, in particular, by applying the reparametrization post-processing algorithm described above. The modified method referred as clp is up to an order of magnitude *faster* than the original one clp-orig (see Table 2). For the experiments with clp and dclp we used both CPLEX and ToulBar2 as ILP-solvers. The corresponding variants of clp are denoted as clp-cpx and clp-tb2 respectively and similarly for dclp. Since the

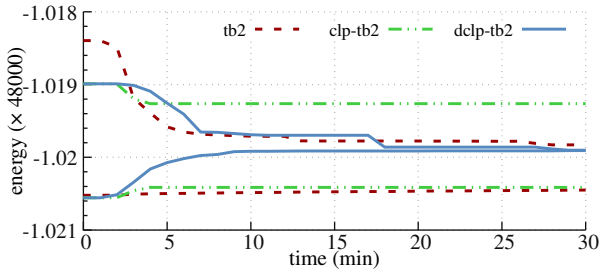


Figure 2: Primal and dual bound progress for worms. All values are averaged over instances where at least one of the solvers returned the optimal solution (25 instances). To improve clarity only the three best solvers are shown.

ToulBar2 variants (clp-tb2 and dclp-tb2) were superior to the CPLEX variants in *all* our tests, we will mainly discuss the former here (see supplement for all results). TRW-S (Kolmogorov 2006) is used as fast block-coordinate-descend LP-solver everywhere except higher-order models. We used a fast implementation of the solver from the work (Shekhovtsov, Swoboda, and Savchynskyy 2015). Only for higher-order examples we resort to SRMP (Kolmogorov 2015) using the *minimal* or *basic LP relaxation* (for details see Kolmogorov 2015). We set the maximum number of TRW-S/SRMP iterations to 2000. Furthermore we tested the performance of a recent partial optimality technique (Shekhovtsov, Swoboda, and Savchynskyy 2015) which is denoted by popt. As this approach does not solve the whole problem, we run ToulBar2 on the reduced model and measure the total running time (popt-tb2). We set the maximal running time for all methods to 1 hour.

Datasets We verify performance of the algorithms on the following publicly available datasets: worms (Kainmueller et al. 2017), color-seg (Lellmann and Schnörr 2011), mrf-stereo (Scharstein and Szeliski 2002) and OnCallRostering (Stuckey et al. 2014), protein-folding (Yanover, Schueler-Furman, and Weiss 2008). Each of these datasets is included to highlight specific strengths and weaknesses of the competing methods. The worms dataset (30 instances) serves as a prime example for our algorithm due to its relatively densely connected graph structure and a small duality gap. The mrf-stereo (3 instances) and color-seg (19 instances) datasets consist of sparsely connected grid-models and are used to compare performance to the CombiLP method clp. The protein-folding dataset can be split into easy problems (many nodes, sparsely connected) and hard problems (only around 33-40 nodes, fully connected). In the following, we only consider the hard problems (11 instances in total). Last but not least, the dataset OnCallRostering (3 instances) is included as an example of higher-order models, which include cliques of order four. Unfortunately, we were unable to convert other instances of this dataset from the benchmark (Hurley et al. 2016) because of a memory bottlenecks in the conversion process. Apart from OnCallRostering and worms, all other problem instances were taken from the OpenGM benchmark (Kappes et al. 2015).

dataset (#instances)	density	cpx		tb2		popt-tb2		clp-orig		clp-tb2		dclp-tb2	
worms (30)	10.6 %	1	54.7	13	8.3	13	8.0	15	14.2	17	6.9	**25** (17)	**5.8** (3.1)
protein-folding (11)	100 %	2	48.5	**11**	1.1	**11**	1.7	10	16.8	**11**	0.9	**11**	**0.8**
color-seg (19)	0.007 %	5	4.9	15	22.1	**18**	**0.3**	**18**	7.6	**18**	1.1	**18**	1.4
mrf-stereo (3)	0.003 %	0	—	0	—	1	0.9	**2**	46.9	**2**	3.2	**2**	**2.3**
OnCallRostering (3)	0.9 %	2	0.9	2	0.1	—	—	—	—	**3**	2.3	**3**	**1.1**

Table 2: Overview of benchmark results. For each method the left number displays the *number of solved instances* and the right one the *average runtime in minutes* for solved instances. We computed the graph density as $\frac{2|\mathcal{E}|}{|\mathcal{V}|(|\mathcal{V}|-1)}$ for pairwise models and as $\frac{|\{(u,v)\in\mathcal{V}^2 | \exists c\in\mathcal{C}\colon u\in c, v\in c, u\neq v\}|}{|\mathcal{V}|(|\mathcal{V}|-1)}$ for OnCallRostering. Values in parentheses for dclp-tb2 show the average running time on the 17 problem instances solved by the best competitor clp-tb2.

Results We compare and analyse performance of our method in the following three settings: *(i)* targeted dense problems like the worms and protein-folding datasets; *(ii)* sparse problems (mrf-stereo and color-seg), and *(iii)* exemplary higher-order problems (OnCallRostering).

(i) dense models: On the worms dataset our method dclp-tb2 clearly outperforms competitors, as Table 2 shows. dclp-tb2 solves 25 instances out of 30, the next competitor clp-tb2 – only 17. Moreover, our solver is also more than 2 times faster than clp-tb2 in average. This is due to the fact that the resulting ILP subproblem of dclp is much smaller that those of clp, see Figure 1 for visual comparison. The partial optimality method is unable to reduce the problem (see Table 1) because of infinite pairwise costs to disallow assigning the same label to different nodes. Figure 2 shows primal and dual bounds as a function of computational time for this dataset.

Although on the protein-folding dataset dclp-tb2 also outperforms all its competitors, the improvement over clp-tb2 and tb2 is not that pronouncing as for the worms dataset. This is because the final ILP subproblem of dclp covers a significant part of the whole graph (over 71% in average). To satisfy its optimality criterion, dclp performs up to 5 iterations with smaller ILP subproblems. In contrast, clp considers the whole graph as an ILP subproblem right at the very first iteration. Interestingly that even under this circumstances clp-tb2 outperforms tb2 and clp-cpx outperforms cpx (see supplement for details). The latter solves only 2 problem instances out of 11, whereas clp-cpx is able to cope with 9. We attribute it to the reparametrization, which is performed by clp prior to passing the problem to cpx or tb2 and plays a role of an efficient presolving.

(ii) sparse models: Sparse (grid-structured) datasets mrf-stereo and color-seg with about 10^5 graph nodes each are very well suitable for both clp and dclp methods and are difficult for cpx and tb2. Both clp and dclp are able to solve all the problems except the largest one (teddy from mrf-stereo dataset with over 1.6×10^5 nodes and 60 labels) in similar time. On color-seg the method clp-tb2 is somewhat faster, whereas dclp-tb2 requires less time on mrf-stereo. This is due to the fact that dclp consistently produces smaller ILP subproblems (see Table 1 for comparison), but clp may require less iterations due to the start with a larger ILP subproblem. Partial optimality popt-tb2 is the winner for the color-seg dataset: Although its ILP subproblems are larger than those of clp and dclp, it runs an ILP solver only once. However, results of popt-tb2 on mrf-stereo are useful only up to a limited extend: They are sufficient to solve only a single, the simplest problem from that dataset (tsukuba). dclp-tb2 and clp-tb2 in contrast solve two problem instances each.

(iii) higher-order models: The dataset OnCallRostering is included mainly to show applicability of our method to higher-order models. Generally, higher-order models pose additional difficulties to solvers because they are intrinsically dense and the size of an ILP formulation of the problem grows exponentially with the problem order, therefore even small problems may not fit into memory of an ILP solver. The dclp method again shows its advantage over clp as similarly as in the case of the worms dataset: Since the problems are intrinsically dense, the ILP subproblem for dclp is smaller, which results in $2\times$ speed-up compared to clp. We also found tb2 and cpx to be quite efficient on this dataset, although they were able to solve only 2 problems out of 3.

Conclusions

We presented a new method, suitable to solve efficiently large-scale MAP-inference problems. The prerequisites for efficiency is the "almost" tight LP relaxation, i. e. the non-strict-arc-consistent subset of nodes should constitute only a small portion of the problem. In this case, it isnot the size of the problem which important, but only the size of its non-strict-arc-consistent subproblem. Comparing to previous works, our method is able to further reduce this size, which is especially notable if the underlying graph structure of the model is non-sparse. In the future, we plan to extend the method to a broader class of combinatorial problems.

Acknowledgement

This work was supported by the DFG grant "Exact Relaxation-Based Inference in Graphical Models" (SA 2640/1-1). We thank the Center for Information Services and High Performance Computing (ZIH) at TU Dresden for generous allocations of computer time.

References

Bergtholdt, M.; Kappes, J.; Schmidt, S.; and Schnörr, C. 2010. A study of parts-based object class detection using complete graphs. *International journal of computer vision* 87(1):93–117.

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

Cooper, M., and Schiex, T. 2004. Arc consistency for soft constraints. *Artificial Intelligence* 154(1-2):199–227.

Cooper, M. C.; de Givry, S.; Sanchez, M.; Schiex, T.; and Zytnicki, M. 2008. Virtual arc consistency for weighted csp. In *AAAI*, volume 8, 253–258.

Cooper, M. C.; de Givry, S.; Sánchez, M.; Schiex, T.; Zytnicki, M.; and Werner, T. 2010. Soft arc consistency revisited. *Artificial Intelligence* 174(7):449–478.

CPLEX, IBM. 2014. ILOG CPLEX 12.6 Optimization Studio.

de Givry, S., and Katsirelos, G. 2017. Clique cuts in weighted constraint satisfaction. In *International Conference on Principles and Practice of Constraint Programming*, 97–113. Springer.

Gurobi Optimization, I. 2016. Gurobi optimizer reference manual.

Hurley, B.; O'Sullivan, B.; Allouche, D.; Katsirelos, G.; Schiex, T.; Zytnicki, M.; and De Givry, S. 2016. Multi-language evaluation of exact solvers in graphical model discrete optimization. *Constraints* 21(3):413–434.

Kainmueller, D.; Jug, F.; Rother, C.; and Meyers, G. 2017. Graph matching problems for annotating c. elegans. `http://dx.doi.org/10.15479/AT:ISTA:57`. Accessed: 2017-09-10.

Kappes, J. H.; Andres, B.; Hamprecht, F. A.; Schnörr, C.; Nowozin, S.; Batra, D.; Kim, S.; Kausler, B. X.; Kröger, T.; Lellmann, J.; et al. 2015. A comparative study of modern inference techniques for structured discrete energy minimization problems. *International Journal of Computer Vision* 115(2):155–184.

Kolmogorov, V.; Thapper, J.; and Zivny, S. 2015. The power of linear programming for general-valued csps. *SIAM Journal on Computing* 44(1):1–36.

Kolmogorov, V. 2006. Convergent tree-reweighted message passing for energy minimization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 28(10):1568–1583.

Kolmogorov, V. 2015. A new look at reweighted message passing. *IEEE transactions on pattern analysis and machine intelligence* 37(5):919–930.

Koster, A. M.; Van Hoesel, S. P.; and Kolen, A. W. 1998. The partial constraint satisfaction problem: Facets and lifting theorems. *Operations research letters* 23(3):89–97.

Lellmann, J., and Schnörr, C. 2011. Continuous multiclass labeling approaches and algorithms. *SIAM Journal on Imaging Sciences* 4(4):1049–1096.

Li, M.; Shekhovtsov, A.; and Huber, D. 2016. Complexity of discrete energy minimization problems. In *European Conference on Computer Vision*, 834–852. Springer.

Marinescu, R., and Dechter, R. 2005. And/or branch-and-bound for graphical models. In *IJCAI*, 224–229.

Otten, L., and Dechter, R. 2010. Toward parallel search for optimization in graphical models. In *ISAIM*.

Prusa, D., and Werner, T. 2013. Universality of the local marginal polytope. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1738–1743.

Ravikumar, P.; Agarwal, A.; and Wainwright, M. J. 2010. Message-passing for graph-structured linear programs: Proximal methods and rounding schemes. *Journal of Machine Learning Research* 11(Mar):1043–1080.

Rother, C.; Kolmogorov, V.; Lempitsky, V.; and Szummer, M. 2007. Optimizing binary mrfs via extended roof duality. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, 1–8. IEEE.

Savchynskyy, B.; Kappes, J. H.; Swoboda, P.; and Schnörr, C. 2013. Global MAP-optimality by shrinking the combinatorial search area with convex relaxation. In *Advances in Neural Information Processing Systems*, 1950–1958.

Scharstein, D., and Szeliski, R. 2002. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision* 47(1-3):7–42.

Shekhovtsov, A.; Swoboda, P.; and Savchynskyy, B. 2015. Maximum persistency via iterative relaxed inference with graphical models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 521–529.

Shekhovtsov, A. 2014. Maximum persistency in energy minimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1162–1169.

Shlezinger, M. 1976. Syntactic analysis of two-dimensional visual signals in the presence of noise. *Cybernetics and systems analysis* 12(4):612–628.

Sontag, D. A. 2007. *Cutting plane algorithms for variational inference in graphical models*. Ph.D. Dissertation, Massachusetts Institute of Technology.

Stuckey, P. J.; Feydy, T.; Schutt, A.; Tack, G.; and Fischer, J. 2014. The minizinc challenge 2008–2013. *AI Magazine* 35(2):55–60.

Swoboda, P.; Shekhovtsov, A.; Kappes, J.; Schnörr, C.; and Savchynskyy, B. 2016. Partial Optimality by Pruning for MAP-Inference with General Graphical Models. *IEEE Trans. Patt. Anal. Mach. Intell.* 38(7):1370–1382.

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

Yanover, C.; Schueler-Furman, O.; and Weiss, Y. 2008. Minimizing and learning energy functions for side-chain prediction. *Journal of Computational Biology* 15(7):899–911.

Supplementary Material for "Exact MAP-Inference by Confining Combinatorial Search with LP Relaxation"

Stefan Haller[†], Paul Swoboda[‡], Bogdan Savchynskyy[†]
[†] University of Heidelberg, [‡] IST Austria
stefan.haller@iwr.uni-heidelberg.de

Proof of Proposition 1

Proof. From $(y^*_u, y^*_v) \in \arg\min_{(s,t)\in\mathcal{X}_{uv}} \theta_{uv}(s,t)$ it follows that $\min_{(s,t)\in\mathcal{X}_{uv}} \theta_{uv}(s,t) = \theta_{uv}(y^*_u, y^*_v)$ for all $uv \in \mathcal{E}_{\mathcal{AB}}$. Hence the right-hand-sides of (5) and (6) are equal. □

Proof of Proposition 2

Lemma 1. *From requirements of Proposition 1 follows that* $x' \circ x'' = y' \circ y''$.

Proof. Since $\mathcal{A}' \subseteq \mathcal{A}$ and $\mathcal{V}_{\mathcal{A}} \subseteq \mathcal{S}(\theta)$, it holds that $\mathcal{V}_{\mathcal{A}'} \subseteq \mathcal{S}(\theta)$. From strict arc-consistency we know that x' and y' are determined by (9), hence $x'_v = y'_v$ for all $v \in \mathcal{V}_{\mathcal{A}'}$.

From decomposition (5) we known that $E_{\mathcal{B}'}(\theta, y'') \geq E_{\mathcal{B}'\cap\mathcal{A},B}(\theta, y'') + E_{\mathcal{B}}(\theta, y'') + \sum_{uv\in\mathcal{E}_{\mathcal{B}'\cap\mathcal{A},\mathcal{B}}} \theta(y''_u, y''_v) \geq E_{\mathcal{B}'\cap\mathcal{A},B}(\theta, x') + E_{\mathcal{B}}(\theta, x'') + \sum_{uv\in\mathcal{E}_{\mathcal{B}'\cap\mathcal{A},\mathcal{B}}} \min_{(s,t)\in\mathcal{X}_{uv}} \theta(s,t) = E_{\mathcal{B}'}(\theta, x' \circ x'')$ where the last equality holds due to Proposition 1. From optimality of y'' we know that $E_{\mathcal{B}'}(\theta, x' \circ x'') \geq E_{\mathcal{B}'}(\theta, y'')$, hence $(x' \circ x'')_v = y''_v$ for all $v \in \mathcal{V}_{\mathcal{B}'}$ as there is only one unique solution. □

Proof of Proposition 2:

Proof. From the requirements of the Proposition we already know that y' and y'' are the optimal assignment of $\mathcal{A}'$ and $\mathcal{B}'$ respectively. It remains to show that $\theta_{uv}(y'_u, y''_v) = \min_{(s,t)\in\mathcal{X}_{uv}} \theta_{uv}(s,t)$ for all $uv \in \mathcal{E}_{\mathcal{A}'\mathcal{B}'}$.

Applying (5) to $y' \circ y''$ results in $E_{\mathcal{G}}(\theta, y' \circ y'') = E_{\mathcal{A}'}(\theta, y') + E_{\mathcal{B}'}(\theta, y'') + \sum_{uv\in\mathcal{E}_{\mathcal{A}'\mathcal{B}'}} \theta_{uv}(y'_u, y''_v)$. Due to $\mathcal{A}' \subseteq \mathcal{A}$ if $uv \in \mathcal{E}_{\mathcal{A}'\mathcal{B}'}$ either $u, v \in \mathcal{V}_{\mathcal{A}}$ or $uv \in \mathcal{E}_{\mathcal{AB}}$. In the first case it follows from $\mathcal{V}_{\mathcal{A}} \subseteq \mathcal{S}(\theta)$, as for the optimal strictly arc-consistent label x' for $\mathcal{A}'$ it holds that $\theta(x'_u, x'_v) = \min_{(s,t)\in\mathcal{X}_{uv}} \theta_{uv}(s,t)$ and $(x'_u, x'_v) = (y'_u, y''_v)$ (Lemma 1). In the second case $(y'_u, y''_v) = (x'_u, x''_v)$ (Lemma 1) and from fulfillment of Proposition (1) for $x' \circ x''$ follows that $\theta_{uv}(y'_u, y''_v) = \min_{(s,t)\in\mathcal{X}_{uv}} \theta_{uv}(s,t)$.

Hence $\theta_{uv}(y'_u, y''_v) = \min_{(s,t)\in\mathcal{X}_{uv}} \theta_{uv}(s,t)$ for $uv \in \mathcal{E}_{\mathcal{A}'\mathcal{B}'}$ and all requirements of Proposition (1) are fulfilled for $y' \circ y''$. □

Proof of Proposition 3

Proof. The prerequisites already assure that x'' is optimal for $\mathcal{B}$, so it remains to show that $x'_{\mathcal{A}}$ is optimal for $\mathcal{A}$ and that $(x'_u, x''_v) \in \arg\min_{(s,t)\in\mathcal{X}_{uv}} \theta(s,t)$ for all $uv \in \mathcal{E}_{\mathcal{AB}}$. The optimality of $x'_{\mathcal{A}}$ for $\mathcal{A}$ follows trivially from $\mathcal{A} \subseteq \mathcal{A}' \subseteq \mathcal{S}(\theta)$ and the fact that x' is optimal for $\mathcal{A}'$, as for both $\mathcal{A}$ and $\mathcal{A}'$ the optimal labeling is determined by (9). From Definition 2 we know that $\mathcal{E}_{\mathcal{AB}} \subseteq \{uv \in \mathcal{E} \mid u \in \mathcal{V}_{\mathcal{A}'}, v \in \mathcal{V}_{\partial A}\}$. In other words, all edges $\mathcal{E}_{\mathcal{AB}}$ are covered by subgraph $\mathcal{A}'$ (note that $\mathcal{V}_{\partial A} \subseteq \mathcal{A}'$). Due to $\mathcal{V}_{\mathcal{A}'} \subseteq \mathcal{S}(\theta)$, for all $uv \in \mathcal{E}_{\mathcal{AB}}$ it is true that $\theta_{uv}(x'_u, x'_v) = \min_{(s,t)\in\mathcal{X}_{uv}} \theta(s,t)$. From the preconditions we know that $x'_v = x''_v$ for all $v \in \mathcal{V}_{\partial A}$, hence $(x'_u, x''_v) \in \arg\min_{(s,t)\in\mathcal{X}_{uv}} \theta(s,t)$. □

Reparametrization Post-Processing

The details of both steps are described below.

(i) In order to obtain a fast post-processing algorithm we modified one of the fastest dual methods, TRW-S of (Kolmogorov 2006), which also can be seen as a special case of its higher-order counterpart SRMP (Kolmogorov 2015). For the sake of brevity we refer to the latter method, because of its simpler presentation and because it works also for higher order models (see below). Our whole modification consisted in reassigning the weights $\omega^+_{\alpha,\beta}$ defined by expression (14) in (Kolmogorov 2015) with the values

$$\omega^+_{\alpha,\beta} = \begin{cases} \frac{1}{|O^+_\beta|+\max\{|I^+_\beta|,|I_\beta - I^+_\beta|\}+\lambda} & \text{if } (\alpha,\beta) \in I^+_\beta \\ 0, & \text{if } (\alpha,\beta) \in I_\beta - I^+_\beta \end{cases}. \tag{13}$$

We also performed the same reassignment of the weights $\omega^-_{\alpha,\beta}$ (defined by expression (16) in (Kolmogorov 2015)), with I^-_β in place of I^+_β. We refer to (Kolmogorov 2015) for a detailed description of the notation and the algorithm itself. The only difference between (13) and the original expression (14) from (Kolmogorov 2015) is an additional term $\lambda > 0$ added to the denominator of the expression in the upper line of (13). The non-zero λ leads to redistribution of the labeling costs between unary cost functions. Therefore, non-optimal labels get higher costs than those belonging to an optimal labeling. We empirically found the value $\lambda = 0.1$ to work well in practice.

(ii) It is a property of a (modified) SRMP method that locally optimal pairwise costs $\min_{(s,t)\in\mathcal{X}_{uv}} \theta_{uv}(s,t)$ are always 0 for any graph edge $uv \in \mathcal{E}_{\mathcal{G}}$. The mentioned above partial redistribution of unary costs between incident pairwise cost functions was done as $\theta_{uv}(s,t) \mathrel{+}= \frac{\theta_u(s)}{\mathrm{Nb}(u)+1}$ for all $u \in \mathcal{V}_{\mathcal{G}}$ and $s \in \mathcal{X}_u$.

Labelwise relative ILP size As the partial optimality techniques work on a labelwise basis, we use a labelwise measure for comparing the size of the final ILP subproblem. For `popt` we use the formula (35) of (Shekhovtsov, Swoboda, and Savchynskyy 2015) to compute the relative number of eliminated labels. Subtracting this value from 100% yields the values in Table 1. The final formula for `popt` looks like the

following

$$1 - \frac{\sum_{v \in \mathcal{V}} |\mathcal{X}_v \setminus p_v(\mathcal{X}_v)|}{\sum_{u \in \mathcal{V}} (|\mathcal{X}_v| - 1)} \,. \tag{14}$$

For `clp` and `dclp` we evaluate the following expression to compute the value with the same semantic:

$$1 - \frac{\sum_{v \in \mathcal{V}_\mathcal{A}} (|\mathcal{X}_v| - 1)}{\sum_{u \in \mathcal{V}_\mathcal{G}} (|\mathcal{X}_v| - 1)} \,. \tag{15}$$

As `popt` is a polynomial time algorithm, it will output a reduced model for all instance of the benchmark. As `clp` and `dclp` try to solve the NP-hard MAP-inference problem (1), they do not terminate for all instances. As the maximal ILP subproblem is only defined after Proposition 1 holds, we assume the worst-case and use 100% as ILP subproblem size for unsolved instances.

Complete benchmark table For lack of space we removed some solvers from the experimental evaluation. The following tables show the results for each instance and each solver separately..

instance	cpx	tb2	popt-tb2		clp-orig		clp-cpx		clp-tb2		dclp-cpx		dclp-tb2	
C18G1_2L1_1	—	—	100%	—	—	—	—	—	—	—	19.8%	58.1	19.8%	14.8
cnd1threeL1_1213061	—	1.3	100%	1.1	34.1%	8.9	34.6%	1.7	34.6%	0.6	3.9%	0.5	3.9%	0.4
cnd1threeL1_1228061	—	0.6	100%	1.2	36.7%	7.2	35.4%	1.8	35.4%	0.5	5.3%	0.4	5.3%	0.5
cnd1threeL1_1229061	—	—	100%	—	—	—	—	—	—	—	20.8%	56.4	20.8%	20.5
cnd1threeL1_1229062	—	—	100%	—	—	—	—	—	—	—	—	—	—	—
cnd1threeL1_1229063	—	1.7	100%	1.3	31.7%	15.3	26.3%	6.3	26.3%	0.5	5.3%	0.6	5.3%	0.4
eft3RW10035L1_0125071	—	—	100%	—	—	—	—	—	52.8%	51.2	16.7%	50.1	16.7%	32.4
eft3RW10035L1_0125072	—	—	100%	—	—	—	—	—	—	—	16.8%	29.0	16.8%	1.9
eft3RW10035L1_0125073	—	—	100%	—	—	—	—	—	—	—	—	—	—	—
egl5L1_0606074	—	—	100%	—	—	—	—	—	—	—	—	—	—	—
elt3L1_0503071	—	51.2	100%	55.5	61.5%	36.5	—	—	60.8%	2.3	14.6%	3.3	14.6%	0.7
elt3L1_0503072	—	9.3	100%	4.3	53.2%	15.0	49.5%	10.6	49.5%	1.2	10.0%	0.7	10.0%	0.4
elt3L1_0504073	—	—	100%	—	—	—	—	—	—	—	13.4%	1.7	13.4%	0.9
hlh1fourL1_0417071	—	—	100%	—	—	—	—	—	—	—	—	—	—	—
hlh1fourL1_0417075	—	1.8	100%	1.3	44.7%	11.5	45.8%	6.3	45.8%	0.6	5.2%	0.5	5.2%	0.5
hlh1fourL1_0417076	—	—	100%	—	—	—	—	—	68.4%	52.8	20.2%	25.9	20.2%	13.0
hlh1fourL1_0417077	—	28.8	100%	30.8	46.7%	7.1	46.7%	3.9	46.7%	0.7	5.6%	0.5	5.6%	0.5
hlh1fourL1_0417078	—	3.5	100%	2.0	54.1%	17.8	53.0%	30.1	53.0%	1.0	10.4%	1.8	10.4%	0.5
mir61L1_1228061	—	—	100%	—	—	—	—	—	—	—	13.5%	22.7	13.5%	5.1
mir61L1_1228062	—	—	100%	—	—	—	—	—	—	—	—	—	15.9%	39.2
mir61L1_1229062	—	—	100%	—	67.1%	18.0	67.5%	14.3	67.5%	1.1	8.9%	1.3	8.9%	0.6
pha4A7L1_1213061	—	—	100%	—	—	—	—	—	—	—	20.1%	17.0	20.1%	7.1
pha4A7L1_1213062	54.7	1.2	100%	1.0	8.4%	8.2	8.4%	0.4	8.4%	0.3	0.7%	0.3	0.7%	0.3
pha4A7L1_1213064	—	—	100%	—	47.4%	20.9	40.8%	16.8	40.8%	1.4	11.1%	1.2	11.1%	0.6
pha4B2L1_0125072	—	—	100%	—	—	—	—	—	—	—	20.1%	10.7	20.1%	2.3
pha4I2L_0408071	—	—	100%	—	—	—	—	—	—	—	—	—	—	—
pha4I2L_0408072	—	1.9	100%	1.3	42.5%	9.1	43.2%	4.2	43.2%	0.6	4.9%	0.5	4.9%	0.4
pha4I2L_0408073	—	3.0	100%	1.8	60.2%	11.9	60.5%	7.6	60.5%	1.0	10.7%	1.2	10.7%	0.5
unc54L1_0123071	—	1.5	100%	1.2	32.3%	10.8	32.3%	1.5	32.3%	0.6	1.9%	0.5	1.9%	0.4
unc54L1_0123072	—	1.8	100%	1.3	53.9%	12.0	53.9%	4.2	53.9%	0.7	7.9%	0.6	7.9%	0.6
average	54.7	8.3	100%	8.0	50.0%	14.0	42.7%	7.8	45.8%	6.9	11.2%	11.9	11.3%	5.8

Table 3: Complete benchmark results for `worms` dataset.

instance	cpx	tb2	popt-tb2		clp-orig		clp-cpx		clp-tb2		dclp-cpx		dclp-tb2	
1CKK	—	0.5	73.6%	1.2	100%	13.6	100%	36.0	100%	0.7	76.4%	13.2	76.4%	0.6
1CM1	—	0.6	70.1%	1.2	100%	7.9	100%	3.1	100%	0.5	42.0%	1.4	42.0%	0.3
1SY9	38.6	0.5	42.2%	0.8	100%	8.5	100%	6.4	100%	0.6	31.0%	0.7	31.0%	0.3
2BBN	—	1.2	85.9%	2.7	100%	31.1	100%	40.1	100%	1.2	62.9%	9.9	62.9%	1.0
2BCX	—	3.1	85.8%	3.6	100%	31.3	—	—	100%	1.6	—	—	94.1%	2.1
2BE6	58.4	0.5	84.9%	1.1	100%	9.3	100%	13.5	100%	0.4	96.6%	7.9	96.6%	0.7
2F3Y	—	2.7	86.3%	2.1	100%	—	—	—	100%	1.2	97.7%	50.8	97.7%	1.5
2FOT	—	1.0	89.0%	1.5	—	18.5	100%	13.1	100%	0.8	69.7%	10.6	69.7%	0.7
2HQW	—	0.6	82.0%	1.2	100%	10.5	100%	8.4	100%	0.8	68.2%	5.1	68.2%	0.5
2O60	—	0.8	84.5%	1.9	100%	24.3	100%	7.9	100%	0.8	91.0%	8.1	91.0%	0.9
3BXL	—	0.7	87.7%	1.6	100%	12.6	100%	5.2	100%	0.7	52.3%	2.4	52.3%	0.5
average	48.4	1.1	79.2%	1.7	100%	21.0	100%	14.9	100%	0.9	68.8%	11.0	71.1%	0.8

Table 4: Complete benchmark results for `protein-folding` dataset.

instance	cpx	tb2	popt-tb2		clp-orig		clp-cpx		clp-tb2		dclp-cpx		dclp-tb2	
ted-gm	—	—	29.0%	—	—	—	—	—	—	—	—	—	—	—
tsu-gm	—	—	6.7%	0.9	0.2%	35.0	0.2%	2.6	0.2%	1.5	0.1%	0.8	0.1%	0.7
ven-gm	—	—	99.9%	—	—	58.7	0.7%	4.8	0.7%	4.8	0.3%	3.9	0.3%	3.8
average	—	—	45.2%	0.9	0.2%	46.9	0.5%	3.7	0.5%	3.2	0.2%	2.4	0.2%	2.3

Table 5: Complete benchmark results for `mrf-stereo` dataset.

instance	cpx	tb2	popt-tb2		clp-orig		clp-cpx		clp-tb2		dclp-cpx		dclp-tb2	
n4/clownfish-small	—	23.7	8.9%	0.1	0.1%	5.4	0.1%	0.5	0.1%	1.0	0.1%	1.0	0.1%	1.0
n4/crops-small	—	31.1	6.8%	0.1	0%	0.7	0.1%	1.0	0.1%	1.0	0.1%	1.0	0.1%	1.0
n4/fourcolors	2.3	9.5	30.4%	0.0	0.2%	4.8	0.4%	0.7	0.4%	0.7	0.1%	0.7	0.1%	0.7
n4/lake-small	—	14.4	8.4%	0.1	0%	0.2	0.0%	1.0	0.0%	1.0	0.0%	1.0	0.0%	1.0
n4/palm-small	—	39.3	5.2%	0.1	0%	0.8	0.1%	1.0	0.1%	1.0	0.1%	1.1	0.1%	1.0
n4/penguin-small	9.2	8.3	6.3%	0.0	0%	0.3	0.0%	0.8	0.0%	0.8	0.0%	0.8	0.0%	0.4
n4/pfau-small	—	—	12.7%	2.1	0.7%	10.6	0.5%	1.6	0.5%	0.8	0.4%	1.6	0.4%	1.3
n4/snail	2.1	0.5	26.7%	0.0	0.1%	2.9	0.1%	0.9	0.1%	0.8	0.1%	0.8	0.1%	0.8
n4/strawberry-glass-2-small	—	38.9	5.8%	0.1	0%	4.1	0.0%	0.9	0.0%	0.8	0.0%	0.9	0.0%	0.9
n8/clownfish-small	—	12.4	8.9%	0.2	0.2%	11.3	0.2%	2.3	0.2%	2.3	0.1%	2.3	0.1%	2.1
n8/crops-small	—	54.0	6.8%	0.7	0.2%	11.6	0.4%	2.6	0.4%	2.6	0.2%	2.5	0.2%	2.5
n8/fourcolors	7.1	15.0	30.4%	0.1	0.5%	5.9	0.6%	1.5	0.6%	1.5	0.1%	1.6	0.1%	1.4
n8/lake-small	—	14.0	8.4%	0.1	0.1%	11.8	0.1%	2.0	0.1%	2.1	0.1%	2.0	0.1%	2.1
n8/palm-small	—	—	5.3%	0.9	0.3%	27.1	0.3%	3.2	0.3%	3.1	0.2%	3.5	0.2%	2.0
n8/penguin-small	—	14.1	6.3%	0.2	0%	3.4	0.0%	1.5	0.0%	1.8	0.0%	1.8	0.0%	1.8
n8/pfau-small	—	—	8.8%	1.1	0.3%	12.1	0.3%	2.2	0.3%	2.1	0.2%	2.1	0.2%	2.2
n8/snail	3.9	0.7	26.8%	0.0	0.1%	12.1	0.1%	1.6	0.1%	1.6	0.1%	1.6	0.1%	1.6
n8/strawberry-glass-2-small	—	55.6	5.9%	0.6	0.4%	12.5	0.4%	2.3	0.4%	2.2	0.2%	2.3	0.2%	2.2
average	4.9	22.1	12.1%	0.3	0.2%	7.6	0.2%	1.5	0.2%	1.1	0.1%	1.6	0.1%	1.4

Table 6: Complete benchmark results for `color-seg` dataset.

instance	cpx	tb2	popt-tb2		clp-orig		clp-cpx		clp-tb2		dclp-cpx		dclp-tb2	
10s-50d	—	—	—	—	—	—	—	—	99.9%	7.0	—	—	42.8%	3.4
4s-10d	0.0	0.0	—	—	—	—	97.7%	0.1	97.7%	0.0	71.9%	0.1	71.9%	0.0
4s-23d	1.7	0.0	—	—	—	—	99.0%	8.1	99.0%	0.0	82.5%	24.1	82.5%	0.1
average	0.9	0.1	—	—	—	—	98.9%	4.1	98.9%	2.3	77.2%	12.4	65.7%	1.1

Table 7: Complete benchmark results for `OnCallRostering` dataset.