Maximum Persistency via Iterative Relaxed Inference with Graphical Models

Alexander Shekhovtsov
TU Graz, Austria
shekhovtsov@icg.tugraz.at

Paul Swoboda
Heidelberg University, Germany
swoboda@math.uni-heidelberg.de

Bogdan Savchynskyy*
TU Dresden, Germany
bogdan.savchynskyy@tu-dresden.de

Abstract

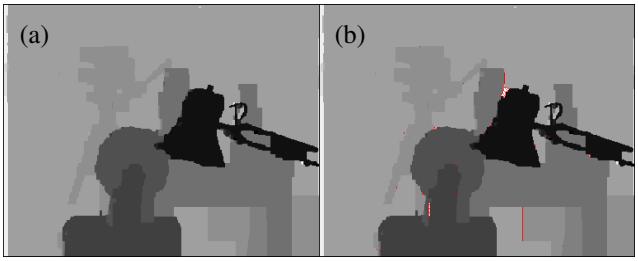
We consider the NP-hard problem of MAP-inference for graphical models. We propose a polynomial time practically efficient algorithm for finding a part of its optimal solution. Specifically, our algorithm marks each label in each node of the considered graphical model either as (i) optimal*, meaning that it belongs to all optimal solutions of the inference problem; (ii)* non-optimal *if it provably does not belong to any solution; or (iii)* undefined*, which means our algorithm can not make a decision regarding the label. Moreover, we prove optimality of our approach: it delivers in a certain sense the largest total number of labels marked as optimal or non-optimal. We demonstrate superiority of our approach on problems from machine learning and computer vision benchmarks.*

1. Introduction

We consider the energy minimization problem, known also as inference of maximum a posteriori (MAP) or maximum likelihood estimate (MLE) for graphical models. In the most common pairwise case this problem reads[1]

$$\min_{x\in\mathcal{X}} E_f(x) := f_0 + \sum_{v\in\mathcal{V}} f_v(x_v) + \sum_{uv\in\mathcal{E}} f_{uv}(x_u, x_v)\,. \quad (1)$$

The problem has numerous applications in computer vision, machine learning, communication theory, signal processing, information retrieval and statistical physics, see [10, 9, 33, 16] for an overview of applications. Problem (1) can be represented as an integer linear program and is known to be NP-hard in general. Approximative algorithms do not guarantee optimality of the found solution and moreover, apart from the roof dual relaxation [3, 17], they do not guarantee optimality of *any* part of the found solution. In this paper we show how some of these approximative methods (addressing convex relaxations of the problem (1)) can be used to identify a part of a *provably* optimal solution or to decrease the state space of variables. Such a reduction of the original problem is often sufficient to make it solvable exactly by (non-polynomial) combinatorial solvers.



TRW-S [12] solution, 670 iterations (10s).

Proofs of optimality (99.93%), 75 additional iterations (6s).

Figure 1. Given an approximate solution to a graphical model in (a), our method provides proofs of optimality and uniqueness for a part of the solution (a) shown in (b) and constructs the reminder of the optimization problem consisting of red pixels with a reduced number of labels. The result (b) approximates well the maximum persistency of [25] and is computed in a comparable time (often a small fraction) of the initial solution in (a). The graphical model is from OpenGM benchmark [10, 9]. It is more difficult than the model studied in [2, 7] for the same imagery. In particular, method of Kovtun [15] performing well there determines only 0.26% of the optimal solution for our model.

States of variables, *e.g.* $x_v = i$, are called *labels*. Labels identified as provably belonging (resp. not belonging) to some (resp. any) optimal solution are called *persistent optimal (resp. non-optimal)* or shortly *persistencies*. We show that in a certain class of methods our algorithm identifies the largest total number of persistent labels.

Related Work. All existing methods identifying persistency are based on tractable sufficient conditions in order to avoid solving the NP-hard problem (1). Dead-end elimination methods (DEE) [4] verify local sufficient conditions by inspecting a given node and its immediate neighbors at a time. The roof dual relaxation in quadratic pseudo-Boolean Optimization (QPBO, see [3, 17] and references therein) has the property that all variables that are integer in the relaxed solution are persistent. Several generalizations of roof duality to higher-order models were proposed (*e.g.*, [1, 13]). The MQPBO method [11] and generalized roof duality [35] extend roof duality to the multi-label case

*Work mostly done when author was with Heidelberg University

[1] We introduce all notations in §2.

by reducing the problem to binary variables and generalizing the concept of submodular relaxation [13], respectively. Kovtun [15] proposed a sufficient condition to identify persistencies based on specially constructed auxiliary submodular problems. A persistency approach utilizing standard scalable (approximate) MAP-inference algorithms has been developed recently by [28] for Potts models and in [29] for general MRFs. It uses the local polytope relaxation and has shown superior results on several datasets.

The recent work by Shekhovtsov [25] explains all mentioned methods addressing pairwise models in a common framework. He proposes a formulation of the problem of determining the *maximum number of persistencies* as a polynomially solvable linear program. It guarantees to find a *provably larger* persistency assignment than most of the mentioned approaches. Unfortunately, there are no efficient specialized solvers available for this linear program and therefore it is not practical in large scale applications.

Contribution. Based on works [29] and [25] we propose a novel method for computing persistency. Similarly to [29] we initialize our algorithm with an approximate solution of the MAP-inference problem (1) and iteratively increase the set of labels that can potentially belong to some optimal solution of the original problem. Upon termination, all optimal solutions are guaranteed to be covered by the built set and hence the remaining labels can be excluded from consideration as provably non-optimal. Our algorithm possesses advantages of both methods [25] and [29] and is free from their drawbacks:

- Like [29] it is efficient and well-scalable, because it requires only to solve the standard local polytope relaxation of the MAP-inference problems (1) as a subroutine. Approximate solvers for the relaxed inference problem can be used as well.
- It has a better theoretical guarantee than [29] and matches the maximum persistency of [25] in the case when the exact LP solver is used.
- We show experimentally that when using TRW-S as an approximate LP solver the persistency found is close to the maximum one.

We demonstrate efficiency of our approach on benchmark problems from machine learning and computer vision. It outperforms *all* competing methods in terms of the number of persistent labels and method [25] in terms of running time. Though we present our method for pairwise models (*c.f.*, equation (1)), it can be generalized to higher order.

Paper Organization. In §2 we give basic definitions and introduce an alternative formulation of the maximum persistency problem [25]. In §3 we propose a novel generic polynomial algorithm for this problem. In §4 we discuss practical aspects of the method, including its use with approximate solvers for the local polytope relaxation of (1). In §5 we propose several speed-ups of the method. In §6 we provide experimental evaluation and in §7 we give our conclusions and discuss future work.

For reader's convenience, the proofs of all mathematical statements are deferred to the appendix. Our implementation is available at `http://www.icg.tugraz.at/Members/shekhovtsov/persistency/`.

2. Preliminaries

Let $(\mathcal{V}, \mathcal{E})$ be a graph with the set of *nodes* $\mathcal{V}$ and the set of *edges* $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$; uv denotes an ordered pair (u, v). In equation (1) each variable x_v belong to the finite set of *labels* $\mathcal{X}_v$ for $v \in \mathcal{V}$; *potentials* $f_v \colon \mathcal{X}_v \to \mathbb{R}$, $f_{uv} \colon \mathcal{X}_u \times \mathcal{X}_v \to \mathbb{R}$ are associated with nodes and edges respectively; $f_0 \in \mathbb{R}$ is a constant term and $\mathcal{X}$ denotes the Cartesian product $\prod_{v \in \mathcal{V}} \mathcal{X}_v$.

We represent all potentials of energy (1) by a single *cost vector* $f \in \mathbb{R}^{\mathcal{I}}$, where the set $\mathcal{I}$ enumerates all components of all terms $f_\cdot(\cdot)$: $\mathcal{I} = \{0\} \cup \{(u, i) \mid u \in \mathcal{V},\ i \in \mathcal{X}_u\} \cup \{(uv, ij) \mid uv \in \mathcal{E},\ i \in \mathcal{X}_u,\ j \in \mathcal{X}_v\}$. Energy function E_f can be written as a scalar product $E_f(x) = \langle f, \delta(x) \rangle$, where $\delta(x) \in \mathbb{R}^{\mathcal{I}}$ is the suitably selected binary vector in an overcomplete representation [33]: $\delta(x)_0 = 1$, $\delta(x)_u(i) = [\![x_u{=}i]\!]$, $\delta(x)_{uv}(i, j) = [\![x_u{=}i]\!][\![x_v{=}j]\!]$, where $[\![\cdot]\!]$ is the Iverson bracket. The convex hull of vectors $\delta(x)$ for $x \in \mathcal{X}$ forms the *marginal polytope* $\mathcal{M} = \operatorname{conv} \delta(\mathcal{X}) = \operatorname{conv}\{\mu \in \{0, 1\}^{\mathcal{I}} \mid (\exists x \in \mathcal{X})\, \mu = \delta(x)\}$. The energy minimization problem (1) can be written using $\mathcal{M}$ as

$$\min_{x \in \mathcal{X}} E_f(x) = \min_{x \in \mathcal{X}} \langle f, \delta(x) \rangle = \min_{\mu \in \mathcal{M}} \langle f, \mu \rangle \ , \qquad (2)$$

i.e., it is reformulated in the vector space $\mathbb{R}^{\mathcal{I}}$.

Maximum Persistency. We formulate our persistency algorithm in the framework of improving mappings [25] briefly summarized next. A persistent subset of variable states (labels) is represented in the framework by a mapping $p \colon \mathcal{X} \to \mathcal{X}$ as follows:

Definition 2.1 ([25])**.** A mapping $p \colon \mathcal{X} \to \mathcal{X}$ is called *strictly improving* for the cost vector f if for all $x \in \mathcal{X}$ such that $p(x) \neq x$ there holds $\langle f, \delta(p(x)) \rangle < \langle f, \delta(x) \rangle$.

We restrict ourselves to *node-wise* idempotent mappings of the form $p(x)_v = p_v(x_v)$, where $p_v \colon \mathcal{X}_v \to \mathcal{X}_v$ are idempotent: $p_v(p_v(i)) = p_v(i)$ for all i. Indeed they are sufficient to explain most existing techniques for persistency [25]. A strictly improving mapping defines persistency due to the following proposition:

Proposition 2.2 ([25])**.** If p is a strictly improving mapping, then any optimal solution x^* of (1) must satisfy $(\forall v \in \mathcal{V})\ p_v(x^*_v) = x^*_v$.

In other words, Proposition 2.2 states that if $p_v(i) \neq i$, then label (v, i) is non-optimal persistent and can be ex-

cluded from consideration. Excluding all labels but one in a given node allows to find the optimal label in this node.

Verifying whether a given map is strictly improving is an NP hard decision problem [25]. A tractable sufficient condition for persistency is obtained in [25] by representing the mapping p in the space $\mathbb{R}^{\mathcal{I}}$ and applying the LP relaxation technique.

Definition 2.3 ([25])**.** A linear mapping $P\colon \mathbb{R}^{\mathcal{I}} \to \mathbb{R}^{\mathcal{I}}$ is called a *linear extension* of mapping $p\colon \mathcal{X} \to \mathcal{X}$ if it satisfies $(\forall x \in \mathcal{X})\ \ \delta(p(x)) = P\delta(x)$.

For a node-wise mapping p we construct the following linear extension, denoted $[p]$. For each $p_v\colon \mathcal{X}_v \to \mathcal{X}_v$ define the matrix $P_v \in \mathbb{R}^{\mathcal{X}_v \times \mathcal{X}_v}$ by $P_{v,ii'} = [\![p_v(i')=i]\!]$. Let $\mu \in \mathbb{R}^{\mathcal{I}}$. The linear extension $P = [p]$ is given by relations

$$(P\mu)_0 = \mu_0; \qquad (P\mu)_v = P_v\mu_v = \sum\nolimits_{i' \in \mathcal{X}_v} P_{v,ii'}\mu_v(i');$$
$$(P\mu)_{uv} = P_u\mu_{uv}P_v^{\top}. \tag{3}$$

By substitution, one can verify that $[p]$ satisfies Definition 2.3 (see [25]).

Example 1. Consider a problem with a singe variable v and 3 labels: $\mathcal{X}_v = \{1, 2, 3\}$. Two examples of mappings $p_v\colon \mathcal{X}_v \to \mathcal{X}_v$ and their corresponding matrices P_v are:

$$p_v\colon 1,2,3 \mapsto 1,1,1; \qquad p_v\colon 1,2,3 \mapsto 3,2,3;$$
$$P_v = \begin{pmatrix} 1&1&1\\0&0&0\\0&0&0 \end{pmatrix}; \qquad P_v = \begin{pmatrix} 0&0&0\\0&1&0\\1&0&1 \end{pmatrix}. \tag{4}$$

In the first case, for example, mapping P_v sends all relaxed labelings $\mu \in \mathcal{M}$ to the vector $(1\,0\,0)^{\mathsf{T}}$ which is the indicator $\delta(x)_v$ of the assignment $x_v = 1$.

Let I denote the identity matrix. From Definition 2.1 follows that p is strictly improving iff the value of

$$\min_{x\in\mathcal{X}} \langle f, \delta(x) - \delta(p(x))\rangle = \min_{x\in\mathcal{X}} \langle f, (I - [p])\delta(x)\rangle$$
$$= \min_{x\in\mathcal{X}} \big\langle (I - [p])^{\top} f, \delta(x)\big\rangle = \min_{\mu\in\mathcal{M}} \big\langle (I - [p])^{\top} f, \mu\big\rangle \tag{5}$$

is zero and $[p]\mu = \mu$ for all minimizers.

Problem (5) is of the same form as the energy minimization (2) and is therefore as difficult. A sufficient condition for persistency [25] is obtained by applying a linear programming relaxation to (5). The complicated marginal polytope $\mathcal{M}$ is replaced with a tractable outer approximation $\Lambda \supseteq \mathcal{M}$ (defined by polynomially many inequalities). Later on, Λ will denote the standard *local polytope* but for now we are going to use only that $\Lambda \supseteq \mathcal{M}$. This relaxation of (5) gives

Definition 2.4 ([25])**.** Mapping $p\colon \mathcal{X} \to \mathcal{X}$ is *strictly* Λ*-improving for cost vector* $f \in \mathbb{R}^{\mathcal{I}}$ if

$$\min_{\mu\in\Lambda} \big\langle (I - [p])^{\top} f, \mu\big\rangle = 0 \tag{6}$$

and $[p]\mu = \mu$ for all minimizers.

It is a sufficient condition for improving mapping and hence persistency:

Proposition 2.5 ([25])**.** If mapping p is strictly Λ-improving then it is strictly improving.

The set of all discrete mappings p satisfying Definition 2.4 will be denoted by $\mathbb{S}_f$. Problem (6) is called the *verification LP* and the decision problem to test $p \in \mathbb{S}_f$ is called the *verification problem*. For node-wise maps the verification problem can be simplified as follows.

Proposition 2.6. Let $\mathcal{O}^*$ denote the set of all minimizers of the verification LP (6) and let

$$\mathcal{O}_v^* = \{i \in \mathcal{X}_v \mid (\exists \mu \in \mathcal{O}^*)\ \ \mu_v(i) > 0\}, \tag{7}$$

which is the *support set* of all optimal solutions in node v. Then $p \in \mathbb{S}_f$ iff $(\forall v \in \mathcal{V}\ \ \forall i \in \mathcal{O}_v^*)\ \ p_v(i) = i$. Proof in §A.

In what follows, we will relate notation $\mathcal{O}_v^*$ to $\mathcal{O}^*$ as in (7). The maximum persistency approach [25] consists of finding a mapping that delivers *the maximal* number of persistent labels in a certain class of mappings. We consider *the subset-to-one* class of maps $\mathcal{P}^{2,y}$, for which maximum persistency was shown tractable [25]. A mapping $p\colon \mathcal{X} \to \mathcal{X}$ in this class has the following form. In a node $v \in \mathcal{V}$ a subset of labels $\mathcal{Y}_v \subseteq \mathcal{X}_v$ is mapped to a fixed label y_v and all other labels $\mathcal{X}_v \backslash \mathcal{Y}_v$ are mapped to themselves. The mapping p is thus defined by:

$$p_v(i) = \begin{cases} y_v, & \text{if } i \in \mathcal{Y}_v; \\ i, & \text{if } i \notin \mathcal{Y}_v. \end{cases} \tag{8}$$

For disambiguation, we assume that $\mathcal{Y}_v$ does not include y_v itself. It is clear that finding the best mapping in this class can be expressed as finding the defining subsets $\mathcal{Y}_v$. The *test labeling* y is fixed and can be chosen as an approximate solution to the energy minimization problem (1).

The following proposition simplifies the verification problem for mappings from $\mathcal{P}^{2,y}$:

Proposition 2.7. A mapping $p \in \mathcal{P}^{2,y}$ is strictly Λ-improving for the cost vector $f \in \mathbb{R}^{\mathcal{I}}$ iff there holds $(\forall v \in \mathcal{V})\ \ \mathcal{O}_v^* \cap \mathcal{Y}_v = \varnothing$. Proof in §A.

A strictly improving map $p \in \mathcal{P}^{2,y}$ identifies as non-optimal persistent all labels in $\mathcal{Y}_v$, or equivalently all labels not in $p_v(\mathcal{X}_v)$. It is natural to compare two maps p and q by the sets of the labels they retain.

Definition 2.8. For $p, q \in \mathcal{P}^{2,y}$, map p is *better equal* than q, denoted by $p \geq q$, if $(\forall v \in \mathcal{V})\ p_v(\mathcal{X}_v) \subseteq q_v(\mathcal{X}_v)$.

The *maximum persistency problem* can be formulated as finding the best mapping in $\mathbb{S}_f \cap \mathcal{P}^{2,y}$ in the sence of Definition 2.8. In order for this formulation to be well-defined there must exist the unique maximal element under partial ordering $\geq$ (hence, the maximum). This property will be shown as a part of the correctness proof for the proposed algorithm (proof of Theorem 3.2 in §A).

Our formulation of the maximum persistency is equivalent to that of [25] in the case of mappings $\mathcal{P}^{2,y}$. Indeed, [25] proposes to minimize the total number of non-persistent labels:

$$\min_{p \in \mathcal{P}^{2,y}} \sum_{v \in \mathcal{V}} |p_v(\mathcal{X}_v)|, \text{ s.t. } p \in \mathbb{S}_f. \qquad \text{(MAX-SI)}$$

It can be seen from Definition 2.8 that $q \geq p$ iff $|q_v(\mathcal{X}_v)| \leq |p_v(\mathcal{X}_v)|$, which proves the equivalence.

In what follows we propose *a practical* algorithm for solving the MAX-SI problem, contrary to [25], which transforms it into a general large-scale LP making its solution practically out of reach for general solvers.

3. Primal Algorithm for Maximum Persistency

We propose Algorithm 1 (its dual variant, Algorithm 2, will be considered in §4) to find the maximum strictly Λ-improving mapping in $\mathcal{P}^{2,y}$. The algorithm can be interpreted as a discrete cutting plane method (cutting plane in a general sense). It starts with a feasible set equal to $\mathcal{P}^{2,y}$. In each iteration it computes the maximum map p in the current feasible set and verifies whether $p \in \mathbb{S}_f$. If it is so, then this map is the solution. Otherwise, the feasible set is refined such that it still contains all maps in $\mathbb{S}_f \cap \mathcal{P}^{2,y}$ but does not contain the previous maximum p.

Algorithm 1 uses a test labeling y as input. It constitutes an approximate solution of the energy minimization problem (1) and can be obtained by *e.g.*, rounding of the solution of the relaxed inference problem $\min_{\mu \in \Lambda} \langle f, \mu \rangle$.

The current feasible set of improving mappings is defined by the collection of sets $(\mathcal{Y}_v \mid v \in \mathcal{V})$. The best mapping p within this set is defined in line 3 according to (8). On each iteration in lines 5-7 the algorithm verifies whether the current map p already satisfies $p \in \mathbb{S}_f$ by Proposition 2.7 and if not, it *prunes* (line 9) the sets $\mathcal{Y}_v$ by removing labels corresponding to the support set $\mathcal{O}^*_v$ of all optimal solutions of the verification LP. These sets can be determined from a strictly complementary pair of primal-dual solutions as discussed in §4. Note that set $\mathcal{O}^*$ appearing in line 5, which is the facet of all optimal solutions, need not be computed explicitly. Our final goal is a *practically* efficient method solving (MAX-SI) approximately. At the same time, Algorithm 1 is implementable as well and defines the baseline for the approximation. Let us now establish properties of Algorithm 1 formally.

Proposition 3.1. Algorithm 1 runs in polynomial time and returns a mapping $p \in \mathbb{S}_f \cap \mathcal{P}^{2,y}$. Proof in §A.

Theorem 3.2. Mapping p returned by Algorithm 1 is the maximum of $\mathbb{S}_f \cap \mathcal{P}^{2,y}$ and thus it solves (MAX-SI). Proof in §A.

Comparison to [29]. Algorithm 1 is similar to the algorithm in [29] in that it iteratively solves the LP relaxation of an auxiliary problem in order to find persistent labels. However, the method [29] (i) does not identify non-optimal labels (which corresponds to a smaller class of mappings than $\mathcal{P}^{2,y}$) and (ii) solves *different* auxiliary problem corresponding to a weaker persistency criterion (*c.f.* [25, 24])[2]. Algorithm 1 thus generalizes the method [29] and is guaranteed to find the same or a larger persistent set of labels. The generalization is not incremental, since it is based on a different formalism [25]. Its practical superiority is clearly demonstrated in §6. Similarly to [29], Algorithm 1 can use approximate dual solvers, as discussed in the next section.

Comparison to [25]. In [25] the problem (MAX-SI) is formulated as a *general* linear program [25, (ε-L1)] of size comparable to the size of the relaxed MAP-inference problem. Algorithm 1 is a new method to solve the same problem in a more combinatorial fashion w.r.t. to the variables defining the mapping. In the formulation of [25], big instances (typical for applications in machine learning or computer vision) can not be addressed by out-of-the-shelf LP solvers, as, *e.g.*, popular interior point and simplex methods have quadratic space complexity. In contrast, Algorithm 1 requires to solve repeatedly only standard relaxed MAP-inference problems (line 5), for which specialized well-scalable solvers are available (linear space complexity and faster in practice than general first-order methods). As a result, Algorithm 1 can solve *the same* problem as [25, (ε-L1)] in a more efficient way, as we demonstrate experimentally in §6.

Generality of the Algorithm. Proofs of *all* the above statements require only that $\mathcal{M} \subseteq \Lambda$. This means Algorithm 1 can be used with *any* polytope Λ satisfying this property, *i.e.*, with an arbitrary LP relaxation of problem (1). Moreover, in order to use the algorithm with higher order models one needs merely to (straightforwardly) generalize Definition 2.3 as done in [24].

4. Persistency with (Approximate) Dual Solvers

Though Algorithm 1 is quite general, to use it in practice one has to address several important issues. In line 5 the relaxed energy minimization problem has to be solved and in line 6 support sets of *all* its primal solutions have to be identified. However, finding even a single solution of the relaxed problem with standard methods like simplex or interior point can be practically infeasible and one has to switch to specialized solvers developed for this problem. The required support set of all optimal solution can be in principle found with algorithms based on smoothing technique [18, 20], but waiting until such solvers converge in

[2]The recent modification of method [29] proposed in [30] uses a persistency criterion almost equivalent to ours, but still applied to a smaller class of mappings than $\mathcal{P}^{2,y}$.

Algorithm 1: Iterative Pruning LP-Primal

Input: Potentials $f \in \mathbb{R}^{\mathcal{I}}$, test labeling $y \in \mathcal{X}$;
Output: Map p that solves (MAX-SI);

1 $(\forall v \in \mathcal{V})\ \mathcal{Y}_v := \mathcal{X}_v \backslash \{y_v\}$;
2 **repeat**
3 $\quad (\forall u \in \mathcal{V}) \quad p_v(i) := \begin{cases} y_v, & \text{if } i \in \mathcal{Y}_v \\ i, & \text{if } i \notin \mathcal{Y}_v \end{cases}$;
4 $\quad g := (I - [p])^\top f$;
5 $\quad \mathcal{O}^* = \operatorname{argmin}_{\mu \in \Lambda} \langle g, \mu \rangle$;
6 $\quad \mathcal{O}^*_v = \{i \in \mathcal{X}_v \mid (\exists \mu \in \mathcal{O}^*)\ \mu_v(i) > 0\}$;
7 $\quad$ **if** $(\forall v \in \mathcal{V})\ \mathcal{O}^*_v \cap \mathcal{Y}_v = \varnothing$ **then return** p;
8 $\quad$ **for** $v \in \mathcal{V}$ **do** /* pruning */
9 $\quad\quad \mathcal{Y}_v := \mathcal{Y}_v \backslash \mathcal{O}^*_v$;

Algorithm 2: Iterative Pruning LP-Dual

Input: Potentials $f \in \mathbb{R}^{\mathcal{I}}$, test labeling $y \in \mathcal{X}$;
Output: Map p that solves (MAX-SI);

1 $(\forall v \in \mathcal{V})\ \mathcal{Y}_v := \mathcal{X}_v \backslash \{y_v\}$;
2 **repeat**
3 $\quad (\forall u \in \mathcal{V}) \quad p_v(i) := \begin{cases} y_v, & \text{if } i \in \mathcal{Y}_v \\ i, & \text{if } i \notin \mathcal{Y}_v \end{cases}$;
4 $\quad g := (I - [p])^\top f$;
5 $\quad \varphi \in \arg\max_{\varphi} \{g^\varphi_0 \mid (\forall \omega \in \mathcal{V} \cup \mathcal{E})\ g^\varphi_\omega \geq 0,\ g^\varphi \text{ is AC}\}$;
6 $\quad \mathcal{O}_v(\varphi) := \{i \in \mathcal{X}_v \mid g^\varphi_v(i) = 0\}$;
7 $\quad$ **if** $(\forall v \in \mathcal{V})\ \mathcal{O}_v(\varphi) \cap \mathcal{Y}_v = \varnothing$ **then return** p;
8 $\quad$ **for** $v \in \mathcal{V}$ **do**
9 $\quad\quad \mathcal{Y}_v := \mathcal{Y}_v \backslash \mathcal{O}_v(\varphi)$;

each iteration of Algorithm 1 can make the whole procedure quite impractical. In general, we would like to avoid restricting ourselves to certain selected solvers to be able to chose the most efficient one for a given problem. Moreover, it is desirable to use solvers working in the dual domain (*e.g.* [12, 20, 6]) as they are the most efficient ones. In this section we propose a modification of Algorithm 1, allowing to (i) stop the solver for the LP relaxation in line 5 before it converges; (ii) use any dual solver, including those, which do not converge to the optimum of the relaxed problem in general, but satisfy certain weaker conditions (*e.g.* TRW-S [12]); (iii) formulate the stopping condition (line 7) in the dual domain and provide *an efficient way* of estimating a superset of the set of optimal solutions $\mathcal{O}^*$ without reconstructing primal solutions, as the latter can constitute a difficult problem [19].

Given the abovementioned practical improvements we will still be able *to guarantee* as an output a strictly Λ-improving mapping p, but possibly non-maximal with respect to the problem (MAX-SI) (experiments in §6 suggest that we lose only slightly in maximality but gain significantly in speed).

LP Relaxation. We consider the standard local polytope relaxation [27, 34] of the energy minimization problem (1):

$$\begin{array}{lcr} \min \langle f, \mu \rangle & = & \max f^\varphi_0 \\ \sum_j \mu_{uv}(i,j) = \mu_u(i), & & \varphi_{uv}(i) \in \mathbb{R}, \\ \sum_i \mu_{uv}(i,j) = \mu_v(j), & & \varphi_{vu}(j) \in \mathbb{R}, \\ \sum_i \mu_u(i) = \mu_0, & & \varphi_u \in \mathbb{R}, \\ \mu_u(i) \geq 0, & & f^\varphi_u(i) \geq 0, \\ \mu_{uv}(i,j) \geq 0, & & f^\varphi_{uv}(i,j) \geq 0. \\ \mu_0 = 1 & & \end{array} \qquad \text{(LP)}$$

Here the constraints of the primal (minimization) problem define the *local polytope* Λ. Given a dual vector φ, the *reparametrization* f^φ (see, *e.g.*, [34]) is defined as

$$f^\varphi_u(i) = f_u(i) + \textstyle\sum_{v \in \mathrm{nb}(u)} \varphi_{uv}(i) - \varphi_u, \qquad \text{(9a)}$$

$$f^\varphi_{uv}(i,j) = f_{uv}(i,j) - \varphi_{uv}(i) - \varphi_{vu}(j), \qquad \text{(9b)}$$

$$f^\varphi_0 = f_0 + \textstyle\sum_u \varphi_u, \qquad \text{(9c)}$$

where $\mathrm{nb}(u) = \{v \mid (u,v) \in \mathcal{E} \vee (v,u) \in \mathcal{E}\}$. There holds $\langle f^\varphi, \mu \rangle = \langle f, \mu \rangle$ for all $\mu \in \Lambda$ (as well as $E_f = E_{f^\varphi}$). Using the reparametrization, the dual problem can be briefly expressed as

$$\max_\varphi f^\varphi_0 \quad \text{s.t.} \quad (\forall \omega \in \mathcal{V} \cup \mathcal{E})\ f^\varphi_\omega \geq 0. \qquad (10)$$

Expressing $\mathcal{O}^*_v$ in the Dual Domain. Let μ and φ be a primal and a dual (non-unique) optimal solutions to (LP). From complementary slackness we know that if $\mu_v(i) > 0$ then the respective dual constraint $f^\varphi_v(i) \geq 0$ holds with equality. However, the reverse implication is only true if μ and φ are *strictly complementary* [32]. In this case the node-wise support sets $\mathcal{O}^*_v$ of all optimal primal solutions equal the sets $\mathcal{O}_v(\varphi)$ of active constraints of the dual, defined as

$$\mathcal{O}_v(\varphi) = \left\{i \in \mathcal{X}_v \mid f^\varphi_v(i) = 0\right\} = \operatorname*{argmin}_i f^\varphi_v(i) \qquad (11)$$

(the sets of local minimizers of the reparametrized problem). However, for a general optimal solution φ only the inclusion $\mathcal{O}^*_v \subseteq \mathcal{O}_v(\varphi)$ holds and the latter set can be almost arbitrary large.

If we prune maps based on $\mathcal{O}_v(\varphi)$ instead of $\mathcal{O}^*_v$, we loose maximality of the resulting improving mapping. Unfortunately, most of the popular well-scalable solvers do not guarantee strict complementarity. However, there is a property, which (i) gives $\mathcal{O}_v(\varphi) \approx \mathcal{O}^*_v$ in practical applications; (ii) is satisfied in the limit by most of the solvers (including approximative ones like TRW-S and MPLP [6]) or can be enforced by a simple post-processing algorithm.

Definition 4.1 ([34])**.** Reparametrized problem f^φ is called *arc consistent* (AC) if: (i) for all $uv \in \mathcal{E}$ from $f^\varphi_{uv}(i,j) = 0$ follows that $f^\varphi_u(i) = 0$ and $f^\varphi_v(j) = 0$; (ii) for all $u \in \mathcal{V}$ from $f^\varphi_u(i) = 0$ follows that for all $v \in \mathrm{nb}(u)$ such $j \in \mathcal{X}_v$ exists that $f^\varphi_{uv}(i,j) = 0$.

Proposition 4.2. Arc consistency is a necessary condition for strict complementarity: if $\mathcal{O}_v(\varphi) = \mathcal{O}_v^*$ for all $v \in \mathcal{V}$ then f^φ is AC. Proof in §A.

Obtaining Improving Mappings with Dual Solvers. We propose Algorithm 2 which is based on a dual solver achieving the arc consistency condition. The algorithm solves (MAX-SI) when the dual solver (in line 4) performs well, *i.e.*, provides a solution φ that satisfies strict complementarity. Otherwise it is suboptimal and we need to reestablish correctness and termination.

Proposition 4.3. Algorithm 2 terminates in a finite number of iterations and delivers a mapping $p \in \mathbb{S}_f \cap \mathcal{P}^{2,y}$. Proof in §A.

The following lemma provides a basis to prove correctness of Algorithm 2 when using an approximate dual solver achieving at least arc consistency.

Lemma 4.4. If $(\forall v \in \mathcal{V})\ \mathcal{O}_v(\varphi) \cap \mathcal{Y}_v = \varnothing$ hold for an AC dual vector φ, then φ is dual optimal. Proof in §C.

Practical Computational Strategy. Lemma 4.4 proves that virtually any algorithm converging to arc consistency either finds such a dual vector φ that $\mathcal{O}_v(\varphi) \cap \mathcal{Y}_v \neq \varnothing$ for some v or returns a dual optimal AC φ. This justifies the following practical strategy: we stop the dual inference solver (line 5 of Algorithm 2) when either 1) after a certain number of iterations there are some labels to prune, *i.e.*, $(\exists v)\mathcal{O}_v(\varphi) \cap \mathcal{Y}_v \neq \varnothing$ or 2) an AC (and hence optimal dual) solution φ is found.

Indeed, such a practical strategy (which still guarantees that the found mapping p is strictly improving) is not only much faster than the theoretically optimal Algorithm 1, but also delivers nearly maximal persistency, as we show in §6.

5. Speeding-Up Persistency Algorithms

Let us get back to Algorithm 1. Recall that it considers the current maximum map p and (implicitly) all $q \in \mathbb{S}_f \cup \mathcal{P}^{2,y}$ that have to be retained. We can replace the verification LP in step 5 by a simpler (reduced) verification LP as suggested by the following theorem.

Theorem 5.1 (Reduction)**.** Let $p, q \in \mathcal{P}^{2,y}$, $q \leq p$. Let

$$\mathcal{O}^* = \operatorname{argmin}_{\mu \in \Lambda} \langle \bar{g}, \mu \rangle, \tag{12}$$

where the *reduced* cost vector $\bar{g}$ is defined as:

$$g := (I - [p])^\top f; \qquad \bar{g}_v(i) = g_v(i),\ v \in \mathcal{V}; \tag{13a}$$

$$\bar{g}_{uv}(i,j) = \tag{13b}$$
$$\begin{cases} 0, & i \notin \mathcal{Y}_u, j \notin \mathcal{Y}_v\,, \\ \Delta_{vu}(j) := \min_{i' \notin \mathcal{Y}_u} g_{uv}(i', j), & i \notin \mathcal{Y}_u, j \in \mathcal{Y}_v\,, \\ \Delta_{uv}(i) := \min_{j' \notin \mathcal{Y}_v} g_{uv}(i, j'), & i \in \mathcal{Y}_u, j \notin \mathcal{Y}_v\,, \\ \min\{\Delta_{vu}(j) + \Delta_{uv}(i), g_{uv}(i,j)\}, & i \in \mathcal{Y}_u, j \in \mathcal{Y}_v\,. \end{cases}$$

Then $q \in \mathbb{S}_f$ if and only if $q(\mathcal{O}_v^*) = \mathcal{O}_v^*$. Proof in§B.

The *reduced verification LP*, given by (12), has in general a different set of optimal solutions, however the theorem asserts that it induces the same set of strictly improving maps, which makes it an equivalent replacement for line (5) in Algorithm 1/2. The reduction has the following advantages: (i) subsets of labels $\mathcal{X}_v \backslash \mathcal{Y}_v$ can be contracted to a single representative label y_v, because associated unary and pairwise costs are equal; (ii) costs $\bar{g}$ satisfy *partial submodularity*: $\bar{g}_{uv}(x_u, x_v) + \bar{g}_{uv}(y_u, y_v) \leq \bar{g}_{uv}(x_u, y_v) + \bar{g}_{uv}(y_u, x_v)$ for all x_u, x_v, which we will use later.

Next, we propose several sufficient conditions to quickly prune some non-optimal labels without affecting the final solution found by the algorithm. Lemma 5.2 below suggests to solve a yet simpler verification LP, $\min_{\mu \in \Lambda'} \langle \bar{g}, \mu \rangle$ over a subset Λ' of Λ. This does not guarantee to remove all non-optimal labels (which implies one has to switch to Λ afterwards), but can be much more efficient then the optimization over Λ. After the lemma we provide two examples of such efficient procedures.

Lemma 5.2. Let $q \in \mathbb{S}_f \cap \mathcal{P}^{2,y}$, $q \leq p$, $Q = [q]$. Let $\Lambda' \subseteq \Lambda$ and $Q(\Lambda') \subseteq \Lambda'$. Let $\bar{g}$ be defined by (13) (depends on p) and let $\mathcal{O}^* = \operatorname{argmin}_{\mu \in \Lambda'} \langle \bar{g}, \mu \rangle$. Then $(\forall v \in \mathcal{V})\ q_v(\mathcal{O}_v^*) = \mathcal{O}_v^*$. Proof in §B.

While Theorem 5.1 is necessary and sufficient for pruning, Lemma 5.2 is only sufficient.

Pruning of Negative Labelings. As follows from Definition 2.4, an existence of a labeling x such that $\big\langle (I - [p])^\top f, \delta(x) \big\rangle \leq 0$ and $x \neq p(x)$ is sufficient to prove that the mapping p is *not* strictly Λ-improving. Hence one could consider updating the current mapping p without waiting for an exact solution of the inference problem in line 5. Lemma 5.2 gives an answer, for which nodes v the label x_v can be pruned from the set $\mathcal{Y}_v$ without loss of optimality. We need to solve the auxiliary *cut* problem

$$\mathcal{O}^* := \operatorname{argmin}_{\mu \in \Lambda_x} \langle \bar{g}, \mu \rangle \tag{14}$$

and exclude x_v from Y_v if $x_v \in \mathcal{O}_v^*$. Here, the feasible set $\Lambda_x = \{\mu \in \Lambda \mid (\forall v \in \mathcal{V})\ \mu(y_v) + \mu(x_v) = 1\} \subseteq \Lambda$ corresponds to the binary problem with the label set $\{y_v, x_v\}$ in each node $v \in \mathcal{V}$. Due to the partial submodularity of $\bar{g}$ the problem (14) is submodular and can be solved by min-cut/max-flow algorithms [14].

Single Node Pruning. Let us consider "a single node" polytope $\Lambda_{u,i} := \{\mu \in \Lambda \mid \mu_u(y_u) + \mu_u(i) = 1; (\forall v \neq u)\ \mu_v(y_v) = 1\}$. It is a special case of Λ_x when y and x differ in a single node u only. In this case problem (14) amounts to calculating $\bar{g}_u(x_u) + \sum_{v \in \mathrm{nb}(u)} \bar{g}_{uv}(x_u, y_v)$. If the value is non-positive, x_u must be excluded from $\mathcal{Y}_u$.

Efficient Message Passing. In many practical cases message passing for f can be computed in time linear in the

Problem family	[29]-CPLEX		[29]-TRWS		ε-L1 [25]		Our-CPLEX		Our-TRWS	
10x10 Potts-3	0.18s	58.46%	0.05s	58.38%	0.05s	**72.27%**	0.18s	**72.27%**	0.04s	72.21%
10x10 full-3	0.24s	2.64%	0.09s	1.22%	0.06s	**62.90%**	0.24s	**62.90%**	0.05s	62.57%
20x20 Potts-3	3.25s	73.95%	0.21s	68.49%	0.87s	**87.38%**	2.43s	**87.38%**	0.06s	**87.38%**
20x20 full-3	2.81s	0.83%	0.37s	0.83%	0.95s	**72.66%**	3.03s	**72.66%**	0.07s	72.31%
20x20 Potts-4	12.45s	23.62%	0.39s	18.43%	19.40s	**74.28%**	8.56s	**74.28%**	0.08s	73.63%
20x20 full-4	3.96s	0.01%	0.39s	0.01%	21.08s	**6.28%**	12.41s	**6.58%**	0.08s	**6.58%**

Table 1. Performance evaluation on random instances of [25]. For each problem family (size, type of potentials and number of labels) average performance over 100 samples is given. To allow for precise comparison all methods are initialized with the same test labeling y found by LP relaxation. Our-TRWS closely approximates Our-CPLEX, which matches ε-L1 [25], and scales much better.

Problem family	#I	#L	#V	MQPBO		MQPBO-10		Kovtun		[29]-TRWS		Our-TRWS	
mrf-stereo	3	16-60	> 100000	†		†		†		2.5h	13%	117s	**73.56%**
mrf-photomontage	2	5-7	≤ 514080	93s	22%	866s	16%	†		3.7h	16%	483s	**41.98%**
color-seg	3	3-4	≤ 424720	22s	11%	87s	16%	**0.3s**	98%	1.3h	**>99%**	61.8s	**99.95%**
color-seg-n4	9	3-12	≤ 86400	22s	8%	398s	14%	**0.2s**	67%	321s	90%	4.9s	**99.26%**
ProteinFolding	21	≤ 483	≤ 1972	685s	2%	2705s	2%	†		48s	18%	9.2s	**55.70%**
object-seg	5	4-8	68160	3.2s	0.01%	†		**0.1s**	93.86%	138s	98.19%	2.2s	**100%**

Table 2. Average performance on OpenGM benchmarks. Columns #I,#L,#V denote the number of instances, labels and variables respectively. † – result is not available (memory / implementation / other reason).

Our-CPLEX	Our Algorithm 1 (Iterative Relaxed Inference) using CPLEX [8].
Our-TRWS	Our Algorithm 2 using TRW-S [12]. Initial solution uses at most 1000 iterations (or the method has converged). All speedups.
[29]-CPLEX	Method of Swoboda *et al*. [29, 30] with CLPEX.
[29]-TRWS	Method [29, 30] with TRW-S.
ε-L1 [25]	Single LP formulation of the maximum strong persistency [25] solved with CPLEX.
Kovtun	One-against-all method of Kovtun [15].
MQPBO	Multilabel QPBO [11].
MQPBO-10	MQPBO with 10 random permutations, accumulating persistency.

Table 3. List of Evaluated Methods

number of labels [5]. Is this advantage preserved if we consider the cost vector $g = (I - P)^{\mathsf{T}} f$ or even $\bar{g}$ (13)? It turns out that the answer in both cases is positive, we give details in §D.3.

Summary of Speedups. We apply the techniques described in this section in the loop of Algorithm 2 as follows.
Attempt a single node pruning for *all* nodes $u \in \mathcal{V}$ and all labels $i \in \mathcal{Y}_v$. Run the dual solver (line 4) on the reduced problem $\bar{g}$ (13) using warm start from the current reparametrization φ until either of the following:

1. it has found a primal solution x such that: $\langle \bar{g}, \delta(x) \rangle \leq 0$ and $p(x) \neq x$;
2. iteration limit was exceeded or the solver has converged.

In the first case, apply the pruning negative labeling technique to x. Otherwise, perform step 7. If the dual solver has converged, Lemma 4.4 guarantees either correct termination or that further pruning is possible. At the same time, warm start allows the solver to converge eventually despite the iteration limit. Details of implementation and a proof of finite termination with TRW-S specifically are given in §D.

6. Experimental Evaluation

In the experiments we study how well we approximate the maximum persistency [25], give a direct comparison to the most relevant scalable method [29][3], illustrate the contribution of different speedups and give an overall performance comparison to a larger set of relevant methods. As a measure of persistency we use the percentage of labels eliminated by the improving mapping p

$$\frac{\sum_{v \in \mathcal{V}} |\mathcal{X}_v \setminus p_v(\mathcal{X}_v)|}{\sum_{v \in \mathcal{V}} (|\mathcal{X}_v| - 1)} 100\%. \tag{15}$$

Random Instances. Table 1 gives comparison to [29] and [25] on random instances generated as in [25] (small problems on 4-connected grid with uniformly distributed integer potentials for "full" model and of the Potts type for "Potts" model, all not LP-tight). It can be seen that our exact Algorithm 1 performs identically to the ε-L1 formulation [25]. Although it solves a series of LPs, as opposed to a single LP solved by ε-L1, it scales better to larger instances. Instances of size 20x20 in the ε-L1 formulation are already too difficult for CPLEX: it takes excessive time and sometimes returns a computational error. The performance of the dual Algorithm 2 confirms that we loose very little in terms of persistency but gain significantly in speed.

[3]Note, [30] points out that numerical results published in [29] were incorrect due to an implementation error, the results that we report are consistent with [30].

Instance	Initialization	Extra time for persistency				
	(1000 it.)	no speedups	+reduction	+node pruning	+labeling pruning	+fast msgs
Protein folding 1CKK	8.5s	268s (26.53%)	168s (26.53%)	2.0s (26.53%)	2.0s (26.53%)	2.0s (26.53%)
colorseg-n4 pfau-small	9.3s	439s (88.59%)	230s (93.41%)	85s (93.41%)	76s (93.41%)	19s (93.41%)

Table 4. Exemplary evaluation of speedups: from left to right we add techniques described in §5. 1CKK: an example when the final time for persistency is only a fraction of the initialization time. pfau-small: an example when times for initialization and persistency are comparable; speedups also help to improve the persistency as they are based on exact criteria.

Instance	#L	#V	[29]-CPLEX		[29]-TRWS		Our-CPLEX		Our-TRWS	
1CKK	≤ 445	38	2503s	0%	46s	0%	2758s	**27%**	8.5+2s	26.53%
1CM1	≤ 350	37	2388s	0%	51s	0%	4070s	**34%**	9+3.9s	29.97%
1SY9	≤ 425	37	1067s	0%	67s	0%	2629s	51%	11+4.2s	**57.98%**
2BBN	≤ 404	37	9777s	0%	5421s	0%	9677s	9%	16+4.3s	**14.17%**
PDB1B25	≤ 81	1972	325s	22%	120s	22%	1599s	84%	4.3+7.3s	**87.84%**
PDB1D2E	≤ 81	1328	483s	59%	83s	59%	154s	**98%**	1.6+1.8s	**98.25%**

Table 5. Comparison to [29] using exact and approximate LP solvers. Examples of hard ProteinFolding instances [16, 36]. For Our-TRWS the initialization + persistency time is given. Better persistency by Our-TRWS *vs*. Our-CPLEX in some cases can be explained by selecting the test labeling y in Our-TRWS using the (sequential) rounding scheme [12] (unlike in Table 1).

Benchmark Problems. Table 2 summarizes average performance on the OpenGM MRF benchmark [10, 9]. The dataset include previous benchmark instances from computer vision [31] and protein structure prediction [16, 36] as well as other models from the literature. Details per instance are given in the supplementary §E.

Speedups. In this experiment we report how much speed improvement was achieved with each subsequent technique of §5. The evaluation in Table 4 starts with a basic implementation using a warm start (a comparison to the cold start is indeed pointless). The solver is allowed to run at most 50 iterations in the partial optimality phase until pruning is attempted. We expect that on most datasets the percentage of persistent labels improves when we apply the speedups (since they preserve maximality, unlike the general pruning based on approximate solvers).

Discussion. Tables 1 and 5 demonstrate that Our-TRWS, which is using a suboptimal dual solver, closely approximates maximum persistency [25]. The proposed method is significantly faster and scales much better. The method of Swoboda *et al*. [29] is the closest contender to our method in terms of algorithm design. Tables 1, 2 and 5 clearly show that our method determines a larger set of persistent variables. This holds true with exact (CPLEX) as well as approximate (TRWS) solvers. We believe that both the stronger persistency criterion and the possibility to eliminate individual labels contribute to this result. Although our method searches over a significantly larger space of possible eliminations (which would normally require more outer iterations), it finishes significantly faster due to speedups. The reported runtimes must be taken with some caution: all evaluated methods including ours admit some further optimization. Nevertheless, it is clear that the proposed method is much more practical than [29] and [25] and gives significantly better results than other techniques.

7. Conclusions and Outlook

We presented an approach to find persistencies for a certain class of NP-hard problems employing only a solver for a convex relaxation. Using a suboptimal solver for the relaxed problem, we still correctly identify persistencies while the whole approach becomes scalable. Our method with an exact solver matches the maximum persistency [25] and with a suboptimal solver closely approximates it, outperforming state of the art persistency techniques [29, 11, 15]. The speedups we have developed allow to achieve this at a reasonable computational cost making the method much more practical than the works [25, 29] we build on. In fact, our approach takes an approximate solver, like TRW-S, and turns it into a method with partial optimality guarantees at a reasonable computation overhead.

We believe that many of the presented results can be extended to higher order graphical models and tighter relaxations. Practical applicability with other approximate solvers can be explored. A further research direction that seems promising is mixing different optimization strategies such as persistency and cutting plane methods.

Acknowlegement

Alexander Shekhovtsov was supported by the Austrian Science Fund (FWF) under the START project BIVISION, No. Y729. Paul Swoboda and Bogdan Savchynskyy were supported by the German Research Foundation (DFG) within the program "Spatio-/Temporal Graphical Models and Applications in Image Analysis", grant GRK 1653.

References

[1] W. P. Adams, J. B. Lassiter, and H. D. Sherali. Persistency in 0-1 polynomial programming. *Mathematics of Operations Research*, 23(2):359–389, Feb. 1998. 1

[2] K. Alahari, P. Kohli, and P. H. S. Torr. Reduce, reuse & recycle: Efficiently solving multi-label MRFs. In *CVPR*, 2008. 1

[3] E. Boros and P. L. Hammer. Pseudo-Boolean optimization. *Discrete Applied Mathematics*, 2002. 1

[4] J. Desmet, M. D. Maeyer, B. Hazes, and I. Lasters. The dead-end elimination theorem and its use in protein side-chain positioning. *Nature*, 356, 1992. 1

[5] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. *International Journal of Computer Vision*, 70(1):41–54, 2006. 7, 16

[6] A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *NIPS*, 2007. 5

[7] I. Gridchyn and V. Kolmogorov. Potts model, parametric maxflow and k-submodular functions. In *ICCV*, 2013. 1

[8] ILOG, Inc. ILOG CPLEX: High-performance software for mathematical programming and optimization. See `http://www.ilog.com/products/cplex/`. 7, 22

[9] J. H. Kappes, B. Andres, F. A. Hamprecht, C. Schnörr, S. Nowozin, D. Batra, S. Kim, B. X. Kausler, T. Kröger, J. Lellmann, N. Komodakis, B. Savchynskyy, and C. Rother. A comparative study of modern inference techniques for structured discrete energy minimization problems. *International Journal of Computer Vision*, pages 1–30, 2015. 1, 8, 17

[10] J. H. Kappes, B. Andres, F. A. Hamprecht, C. Schnörr, S. Nowozin, D. Batra, S. Kim, B. X. Kausler, J. Lellmann, N. Komodakis, and C. Rother. A comparative study of modern inference techniques for discrete energy minimization problem. In *CVPR*, 2013. 1, 8, 17

[11] P. Kohli, A. Shekhovtsov, C. Rother, V. Kolmogorov, and P. Torr. On partial optimality in multi-label MRFs. In *ICML*, 2008. 1, 7, 8, 22

[12] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *PAMI*, 28(10), Oct. 2006. 1, 5, 7, 8, 15, 22

[13] V. Kolmogorov. Generalized roof duality and bisubmodular functions. *Discrete Applied Mathematics*, 160(4-5):416–426, 2012. 1, 2

[14] V. Kolmogorov and R. Zabin. What energy functions can be minimized via graph cuts? *PAMI*, 26(2):147–159, 2004. 6

[15] I. Kovtun. Sufficient condition for partial optimality for (max, +) labeling problems and its usage. *Control Systems and Computers*, 2, 2011. Special issue. 1, 2, 7, 8, 12, 13

[16] The probabilistic inference challenge (PIC2011). `http://www.cs.huji.ac.il/project/PASCAL/`. 1, 8, 17

[17] C. Rother, V. Kolmogorov, V. S. Lempitsky, and M. Szummer. Optimizing binary MRFs via extended roof duality. In *CVPR*, 2007. 1

[18] B. Savchynskyy, J. Kappes, S. Schmidt, and C. Schnörr. A study of Nesterov's scheme for Lagrangian decomposition and MAP labeling. In *CVPR*, 2011. 4

[19] B. Savchynskyy and S. Schmidt. Getting feasible variable estimates from infeasible ones: MRF local polytope study. In *Advanced Structured Prediction*. MIT Press, 2014. 5

[20] B. Savchynskyy, S. Schmidt, J. H. Kappes, and C. Schnörr. Efficient MRF energy minimization via adaptive diminishing smoothing. In *UAI*, 2012. 4, 5

[21] M. I. Schlesinger and K. V. Antoniuk. Diffusion algorithms and structural recognition optimization problems. *Cybernetics and Sys. Anal.*, 47(2):175–192, Mar. 2011. 15

[22] M. I. Schlesinger and B. Flach. Some solvable subclasses of structural recognition problems. In *Czech Pattern Recognition Workshop 2000*, February 2000. 14

[23] A. Shekhovtsov. *Exact and Partial Energy Minimization in Computer Vision*. PhD Thesis CTU–CMP–2013–24, CMP, Czech Technical University in Prague, 2013. 12

[24] A. Shekhovtsov. Higher order maximum persistency and comparison theorems. *Optimization Online e-prints*, 2014. under review in CVIU. 4

[25] A. Shekhovtsov. Maximum persistency in energy minimization. In *CVPR*, 2014. 1, 2, 3, 4, 7, 8

[26] A. Shekhovtsov. Maximum persistency in energy minimization. Technical report, Graz University of Technology, 2014. 10, 11

[27] M. Shlezinger. Syntactic analysis of two-dimensional visual signals in the presence of noise. *Cybernetics and Systems Analysis*, 4:113–130, 1976. See review [34]. 5

[28] P. Swoboda, B. Savchynskyy, J. H. Kappes, and C. Schnörr. Partial optimality via iterative pruning for the Potts model. In *SSVM*, 2013. 2

[29] P. Swoboda, B. Savchynskyy, J. H. Kappes, and C. Schnörr. Partial optimality by pruning for MAP-inference with general graphical models. In *CVPR*, 2014. 2, 4, 7, 8, 17, 19, 20, 21, 22

[30] P. Swoboda, A. Shekhovtsov, J. H. Kappes, C. Schnörr, and B. Savchynskyy. Partial optimality by pruning for MAP-inference with general graphical models. *ArXiv e-prints*, Oct 2014. 4, 7

[31] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. F. Tappen, and C. Rother. A comparative study of energy minimization methods for Markov random fields with smoothness-based priors. *PAMI*, 30(6), 2008. 8, 17

[32] R. J. Vanderbei. *Linear Programming: Foundations and Extensions*. Department of operations and research and financial engineering, Princeton university, 2001. 5, 10

[33] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Found. Trends Mach. Learn.*, 1(1-2):1–305, Jan. 2008. 1, 2

[34] T. Werner. A linear programming approach to max-sum problem: A review. *PAMI*, 29(7), 2007. 5, 9

[35] T. Windheuser, H. Ishikawa, and D. Cremers. Generalized roof duality for multi-label optimization: Optimal lower bounds and persistency. In *ECCV*, 2012. 1

[36] C. Yanover, O. Schueler-Furman, and Y. Weiss. Minimizing and learning energy functions for side-chain prediction. *Jour. of Comp. Biol.*, 15(7), 2008. 8, 17

[37] C. Zach. A principled approach for coarse-to-fine MAP inference. In *CVPR*, 2014. 17