

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

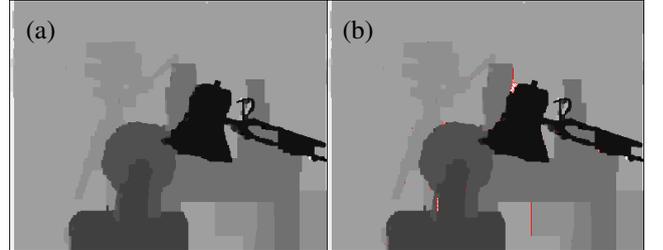
We consider the NP-hard problem of MAP-inference for graphical models. We propose a polynomial time practically efficient algorithm for finding a part of its optimal solution. Specifically, our algorithm marks each label in each node of the considered graphical model either as (i) optimal, meaning that 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 reads¹

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

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



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 remainder 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.

to make it solvable exactly by (non-polynomial) combinatorial solvers.

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

¹We 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* $f_v: \mathcal{X}_v \rightarrow \mathbb{R}$, $f_{uv}: \mathcal{X}_u \times \mathcal{X}_v \rightarrow \mathbb{R}$ 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]: $\delta(x)_0 = 1$, $\delta(x)_u(i) = \llbracket x_u = i \rrbracket$, $\delta(x)_{uv}(i, j) = \llbracket x_u = i \rrbracket \llbracket x_v = j \rrbracket$, where $\llbracket \cdot \rrbracket$ is the Iverson bracket. The convex hull of vectors $\delta(x)$ for $x \in \mathcal{X}$ forms the *marginal polytope* $\mathcal{M} = \text{conv } \delta(\mathcal{X}) = \text{conv} \{ \mu \in \{0, 1\}^{\mathcal{I}} \mid (\exists x \in \mathcal{X}) \mu = \delta(x) \}$. The energy minimization problem (1) can be written using \mathcal{M} as

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

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 $p: \mathcal{X} \rightarrow \mathcal{X}$ as follows:

Definition 2.1 ([25]). A mapping $p: \mathcal{X} \rightarrow \mathcal{X}$ 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 $p_v: \mathcal{X}_v \rightarrow \mathcal{X}_v$ 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 $P: \mathbb{R}^{\mathcal{I}} \rightarrow \mathbb{R}^{\mathcal{I}}$ is called a *linear extension* of mapping $p: \mathcal{X} \rightarrow \mathcal{X}$ 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 $p_v: \mathcal{X}_v \rightarrow \mathcal{X}_v$ define the matrix $P_v \in \mathbb{R}^{\mathcal{X}_v \times \mathcal{X}_v}$ by $P_{v,ii'} = \llbracket p_v(i')=i \rrbracket$. Let $\mu \in \mathbb{R}^{\mathcal{I}}$. The linear extension $P = [p]$ is given by relations

$$\begin{aligned} (P\mu)_0 &= \mu_0; & (P\mu)_v &= P_v\mu_v = \sum_{i' \in \mathcal{X}_v} P_{v,ii'}\mu_v(i'); \\ (P\mu)_{uv} &= P_u\mu_{uv}P_v^\top. \end{aligned} \quad (3)$$

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

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

$$\begin{aligned} p_v: 1, 2, 3 &\mapsto 1, 1, 1; & p_v: 1, 2, 3 &\mapsto 3, 2, 3; \\ P_v &= \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; & P_v &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}. \end{aligned} \quad (4)$$

In the first case, for example, mapping P_v sends all relaxed labelings $\mu \in \mathcal{M}$ to the vector $(1 \ 0 \ 0)^\top$ 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

$$\begin{aligned} \min_{x \in \mathcal{X}} \langle f, \delta(x) - \delta(p(x)) \rangle &= \min_{x \in \mathcal{X}} \langle f, (I - [p])\delta(x) \rangle \\ &= \min_{x \in \mathcal{X}} \langle (I - [p])^\top f, \delta(x) \rangle = \min_{\mu \in \mathcal{M}} \langle (I - [p])^\top f, \mu \rangle \end{aligned} \quad (5)$$

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 $p: \mathcal{X} \rightarrow \mathcal{X}$ is *strictly Λ -improving* for cost vector $f \in \mathbb{R}^{\mathcal{I}}$ if

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

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

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

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 $p: \mathcal{X} \rightarrow \mathcal{X}$ 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 $\mathcal{X}_v \setminus \mathcal{Y}_v$ are mapped to themselves. The mapping p is thus defined by:

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

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 $(\forall v \in \mathcal{V}) \mathcal{O}_v^* \cap \mathcal{Y}_v = \emptyset$. 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 sense 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:

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

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

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

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 $(\forall v \in \mathcal{V}) \mathcal{Y}_v := \mathcal{X}_v \setminus \{y_v\}$;
- 2 **repeat**
- 3 $(\forall u \in \mathcal{V}) p_v(i) := \begin{cases} y_v, & \text{if } i \in \mathcal{Y}_v \\ i, & \text{if } i \notin \mathcal{Y}_v \end{cases}$;
- 4 $g := (I - [p])^\top f$;
- 5 $\varphi \in \arg \max_{\varphi} \{g_0^\varphi \mid (\forall \omega \in \mathcal{V} \cup \mathcal{E}) g_\omega^\varphi \geq 0, g^\varphi \text{ is AC}\}$;
- 6 $\mathcal{O}_v(\varphi) := \{i \in \mathcal{X}_v \mid g_v^\varphi(i) = 0\}$;
- 7 **if** $(\forall v \in \mathcal{V}) \mathcal{O}_v(\varphi) \cap \mathcal{Y}_v = \emptyset$ **then return** p ;
- 8 **for** $v \in \mathcal{V}$ **do**
- 9 $\mathcal{Y}_v := \mathcal{Y}_v \setminus \mathcal{O}_v(\varphi)$;

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 choose 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):

$$\begin{aligned}
\min \langle f, \mu \rangle &= \max f_0^\varphi \\
\sum_j \mu_{uv}(i, j) &= \mu_u(i), & \varphi_{uv}(i) &\in \mathbb{R}, \\
\sum_i \mu_{uv}(i, j) &= \mu_v(j), & \varphi_{vu}(j) &\in \mathbb{R}, \\
\sum_i \mu_u(i) &= \mu_0, & \varphi_u &\in \mathbb{R}, \\
\mu_u(i) &\geq 0, & f_u^\varphi(i) &\geq 0, \\
\mu_{uv}(i, j) &\geq 0, & f_{uv}^\varphi(i, j) &\geq 0. \\
\mu_0 &= 1
\end{aligned} \tag{LP}$$

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

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

$$f_{uv}^\varphi(i, j) = f_{uv}(i, j) - \varphi_{uv}(i) - \varphi_{vu}(j), \tag{9b}$$

$$f_0^\varphi = f_0 + \sum_u \varphi_u, \tag{9c}$$

where $\text{nb}(u) = \{v \mid (u, v) \in \mathcal{E} \vee (v, u) \in \mathcal{E}\}$. 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

$$\max_{\varphi} f_0^\varphi \quad \text{s.t.} \quad (\forall \omega \in \mathcal{V} \cup \mathcal{E}) f_\omega^\varphi \geq 0. \tag{10}$$

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 $f_v^\varphi(i) \geq 0$ 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

$$\mathcal{O}_v(\varphi) = \{i \in \mathcal{X}_v \mid f_v^\varphi(i) = 0\} = \operatorname{argmin}_i f_v^\varphi(i) \tag{11}$$

(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 lose 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 $f_{uv}^\varphi(i, j) = 0$ follows that $f_u^\varphi(i) = 0$ and $f_v^\varphi(j) = 0$; (ii) for all $u \in \mathcal{V}$ from $f_u^\varphi(i) = 0$ follows that for all $v \in \text{nb}(u)$ such $j \in \mathcal{X}_v$ exists that $f_{uv}^\varphi(i, j) = 0$.

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 $(\forall v \in \mathcal{V}) \mathcal{O}_v(\varphi) \cap \mathcal{Y}_v = \emptyset$ 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 $\mathcal{O}_v(\varphi) \cap \mathcal{Y}_v \neq \emptyset$ 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.*, $(\exists v) \mathcal{O}_v(\varphi) \cap \mathcal{Y}_v \neq \emptyset$ 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

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

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

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

$$\bar{g}_{uv}(i, j) = \quad (13b)$$

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

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 $\mathcal{X}_v \setminus \mathcal{Y}_v$ 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 than 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 $\langle (I - [p])^\top f, \delta(x) \rangle \leq 0$ 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

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

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 $\bar{g}_u(x_u) + \sum_{v \in \text{nb}(u)} \bar{g}_{uv}(x_u, y_v)$. 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 <i>et al.</i> [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 $g = (I - P)^T f$ 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

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

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.

³Note, [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 (1000 it.)	Extra time for persistency				
		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%)
coloreg-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 signifi-

cantly 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.

Acknowledgement

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. Zabini. 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. Antoniuik. 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

Appendix

Contents

A . Proofs of the Generic Algorithms	10
B . Proofs of the Reduction	11
C . Termination with AC Solvers	13
D . Implementation with TRW-S	14
D.1 Algorithm Details	14
D.2 Properties of TRW-S	14
D.3 Efficient Message Passing	16
E . Detailed Experimental Evaluation	17

A. Proofs of the Generic Algorithms

Proposition 2.2 "Let p be a strictly improving mapping. Then any optimal solution x^* of (1) must satisfy $p_v(x_v^*) = x_v^*$, $v \in \mathcal{V}$ ".

Proof. Let p be a strictly improving mapping and $x^* \in \arg \min_{x \in \mathcal{X}} \langle f, \delta(x) \rangle$ such that $p(x) \neq x$. Then from Definition 2.1 follows $\langle f, \delta(p(x^*)) \rangle < \langle f, \delta(x^*) \rangle$, which contradicts optimality of x^* . \square

Proposition 2.5 "If mapping p is strictly Λ -improving then it is strictly improving."

Proof. From Definition 2.4 follows that for all $\mu \in \Lambda$ such that $[p]\mu \neq \mu$ there holds $\langle f, [p]\mu \rangle < \langle f, \mu \rangle$. Since $\Lambda \supseteq \mathcal{M}$ it holds also for $\mu = \delta(x)$ for all $x \in \mathcal{X}$, which proves the proposition. \square

Proposition 2.6: " $p \in \mathbb{S}_f$ iff $(\forall v \in \mathcal{V} \ \forall i \in \mathcal{O}_v^*) p_v(i) = i$ ".

Proof. Let $p \in \mathbb{S}_f$. Assume for contradiction that $(\exists v \in \mathcal{V} \ \exists i \in \mathcal{O}_v^*) p_v(i) \neq i$. Since $i \in \mathcal{O}_v^*$ there exists $\mu \in \mathcal{O}^*$ such that $\mu_v(i) > 0$. It's image $\mu' = [p]\mu$ has $\mu'_v(i) = 0$ due to $p_v(i) \neq i$ by evaluating the extension (3). This contradicts to $[p]\mu = \mu$.

Now let $(\forall v \in \mathcal{V} \ \forall i \in \mathcal{O}_v^*) p_v(i) = i$. Clearly, $[p]\mu = \mu$ holds for all μ on the support set given by $(\mathcal{O}_v^* \mid v \in \mathcal{V})$, hence for \mathcal{O}^* . It remains to show that the value of minimum in (6) is zero. For $\mu \in \mathcal{O}^*$ we have $[p]\mu = \mu$ and the objective in (6), $\langle (I - [p])^\top f, \mu \rangle = \langle f, \mu - [p]\mu \rangle$ vanishes. \square

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

Proof. Follows by Definition 2.4 and Proposition 2.6. \square

Proposition 3.1: "Algorithm 1 is polynomial and returns a strictly Λ -improving mapping $p \in \mathbb{S}_f \cap \mathcal{P}^{2,y}$ ".

Proof. Solving the verification LP in every iteration as well as finding the support sets of all optimal solutions \mathcal{O}_v^* is polynomial. These sets equal to the support set of any strict relative interior optimal solution, *i.e.*, a solution found by an interior point method, see, *e.g.*, [32].

At every iteration, if the algorithm has not terminated yet, at least one of the sets \mathcal{Y}_v strictly shrinks in line 9. Therefore the algorithm terminates in at most $\sum_v (|\mathcal{X}_v| - 1)$ iterations. On termination $p \in \mathbb{S}_f$ by Proposition (2.7). \square

Theorem 3.2: "Mapping p returned by Algorithm 1 is the maximum of $\mathbb{S}_f \cap \mathcal{P}^{2,y}$ ".

Proof. Two following lemmas form a basis for the proof.

Lemma A.1 (special case of [26], Theorem 3(b)). Let $q \in \mathbb{S}_f$; $p \geq q$ and \mathcal{O}^* is the set of optimal relaxed labelings, $\arg \min_{\mu \in \Lambda} \langle (I - [p])^\top f, \mu \rangle$, *i.e.*, as in line 5 of Algorithm 1. Then $(\forall \mu \in \mathcal{O}^*) \ q(\mu) = \mu$.

Additionally, if $q \in \mathcal{P}^{2,y}$ and $\mathcal{O}_v^* = \{i \in \mathcal{X}_v \mid (\exists \mu \in \mathcal{O}^*) \ \mu_v(i) > 0\}$ then

$$(\forall v \in \mathcal{V}, \forall i \in \mathcal{O}_v^*) \ q_v(i) = i. \quad (16)$$

Proof. We prove the additional part. It follows similarly to Proposition 2.7. Assume for contradiction that $(\exists v \in \mathcal{V} \ \exists i \in \mathcal{O}_v^*) \ q_v(i) \neq i$. Since $i \in \mathcal{O}_v^*$ there exists $\mu \in \mathcal{O}^*$ such that $\mu_v(i) > 0$. It's image $\mu' = [q]\mu$ has $\mu'_v(i) = 0$ due to $q_v(i) \neq i$ by evaluating the extension (3). This contradicts to $[q]\mu = \mu$, the direct statement of [26, Theorem 3(b)]. \square

Lemma A.2. Algorithm 1 maintains invariant that $(\forall q \in \mathbb{S}_f \cap \mathcal{P}^{2,y}) \ p \geq q$.

Proof. We prove by induction. The statement holds trivially for the first iteration. Assume it is true for the current iteration t . Let p^t denote the mapping p computed in line 3 on iteration t . Then for any $q \in \mathbb{S}_f \cap \mathcal{P}^{2,y}$ holds $p^t \geq q$ and therefore Lemma A.1 applies. We show that line 9 only prunes maps that are not in $\mathcal{P}^{2,y} \cap \mathbb{S}_f$ as follows.

Let p^{t+1} be the mapping on the next iteration, *i.e.* computed by line 3 after pruning line 9.

Assume for contradiction that $\exists q \in \mathcal{P}^{2,y} \cap \mathbb{S}_f$ such that $p^{t+1} \not\geq q$. By negating the definition and expanding,

$$(\exists v \in \mathcal{V}) \ p_v^{t+1}(\mathcal{X}_v) \neq q_v(\mathcal{X}_v), \quad (17a)$$

$$(\exists v \in \mathcal{V} \ \exists i \in \mathcal{X}_v) \ i \in p_v^{t+1}(\mathcal{X}_v) \ \wedge \ i \notin q_v(\mathcal{X}_v), \quad (17b)$$

$$(\exists v \in \mathcal{V} \ \exists i \in \mathcal{X}_v) \ p_v^{t+1}(i) = i \ \wedge \ q_v(i) \neq i. \quad (17c)$$

If i was pruned in line 9, $i \in \mathcal{O}_v^*$, then it must be that

$q_v(i) = i$, which contradicts to (17c). Therefore

$$(\exists v \in \mathcal{V} \exists i \in \mathcal{X}_v \setminus \mathcal{O}_v^*) p_v^{t+1}(i) = i \wedge q_v(i) \neq i. \quad (18)$$

However, in this case $p_v^{t+1}(i) = p_v^t(i) = i$ and $p^t \geq q$ fails to hold, which contradicts to the assumption of induction. Therefore $p^{t+1} \geq q$ holds by induction on every iteration. \square

By Proposition 3.1 the algorithm terminates and returns a map in $\mathbb{S}_f \cap \mathcal{P}^{2,y}$. By Lemma A.2 the returned map p satisfies $p \geq q$ for all $q \in \mathbb{S}_f \cap \mathcal{P}^{2,y}$. Therefore q is the element of $\mathbb{S}_f \cap \mathcal{P}^{2,y}$ which is larger or equal to any other element of this set. It is the maximum. \square

Proposition 4.2: "Let $\mathcal{O}_v(\varphi) = \mathcal{O}_v^*$ for all $v \in \mathcal{V}$. Then f^φ is arc consistent".

Proof. Condition $\mathcal{O}_v(\varphi) = \mathcal{O}_v^*$ implies that φ satisfies strict complementarity with some primal optimal solution μ . The strict complementarity implies that $(\forall i \in \mathcal{X}_v) (f_u^\varphi(i) = 0 \Rightarrow \mu_u(i) > 0)$. By feasibility of μ , there must hold $(\forall v \in \text{nb}(u)) (\exists j \in \mathcal{X}_v) \mu_{uv}(i, j) > 0$. And by using complementary slackness again, it must be that $f_{uv}^\varphi(i, j) = 0$. Similarly, the second condition of arc consistency is verified. It follows that f^φ is arc consistent. \square

Proposition 4.3: "Algorithm 2 terminates in a finite number of iterations and delivers a strictly Λ -improving mapping $p \in \mathbb{S}_f \cap \mathcal{P}^{2,y}$ ".

Proof. Since $\mathcal{O}_v^* \subseteq \mathcal{O}_v(\varphi)$, Algorithm 2 prunes a superset of maps pruned by Algorithm 1. Algorithm 2 is finite, because in case the termination condition in line 5 is not satisfied at least one of the sets \mathcal{Y}_v shrinks in line (9). Since $\mathcal{O}_v^* \subseteq \mathcal{O}_v(\varphi)$ from the stopping condition $(\forall v \in \mathcal{V}) \mathcal{O}_v(\varphi) \cap \mathcal{Y}_v = \emptyset$ follows $(\forall v \in \mathcal{V}) \mathcal{O}_v^* \cap \mathcal{Y}_v = \emptyset$, which is sufficient for p to be strictly Λ -improving according to Proposition 2.7. \square

B. Proofs of the Reduction

In this section we prove the reduction Theorem 5.1 and the correctness of heuristics (pruning of negative labelings and individual nodes) that are based on Lemma 5.2. These theorems revisit the key elements on which Algorithm 1 builds: the verification LP (2.7), Proposition 2.6 and Lemma A.1, which either certifies $p \in \mathbb{S}_f$ or allows to make pruning. A correct pruning can be done when we have a guarantee to preserve all strictly improving maps q , assuming $q \leq p$. Therefore theorems in this section are formulated for such pairs.

The proof chain considers adjustments to the cost vector that preserve the set of mappings that are strictly improving for the cost vector. These adjustments do not in general preserve optimal solutions to the associated LP relaxation. In that the reduction is different from an equivalent transformation.

Theorem B.1. Let $q \leq p$. Then $q \in \mathbb{S}_f$ iff $q \in \mathbb{S}_g$ for $g = (I - [p])^\top f$.

Proof. Let $Q = [q]$, $P = [p]$. Because $q \leq p$ there holds $PQ = P$. It implies $(I - P)(I - Q) = (I - Q)$. Therefore,

$$\langle g, (I - Q)\mu \rangle = \langle (I - P)^\top f, (I - Q)\mu \rangle \quad (19)$$

$$= \langle f, (I - P)(I - Q)\mu \rangle = \langle f, (I - Q)\mu \rangle. \quad (20)$$

Assume $\mu \in \Lambda$ is such that $Q\mu \neq \mu$. Then equality (19) ensures that $\langle g, (I - Q)\mu \rangle > 0$ iff $\langle f, (I - Q)\mu \rangle > 0$. The theorem follows from definition of $\mathbb{S}_f, \mathbb{S}_g$. \square

The reduction that we further build is based on the following result.

Theorem B.2 (Characterization [26]). Let $P = [p]$, p idempotent component-wise. Then

$$(\forall \mu \in \Lambda) \langle f, P\mu \rangle \leq \langle f, \mu \rangle \quad (21)$$

iff exists reparametrization φ such that

$$P^\top f^\varphi \leq f^\varphi. \quad (22)$$

The reduction in Theorem 5.1 is proven in several steps. The following theorem assumes arbitrary mapping q , not necessarily in $\mathcal{P}^{2,y}$ and take as input sets U_u that are subsets of *immovable* labels. In the context of Theorem 5.1, we will use $U_u = \mathcal{X}_u \setminus \mathcal{Y}_u$.

Theorem B.3 (Reduction 1). Let $q: \mathcal{X} \rightarrow \mathcal{X}$ component-wise idempotent. Let $U_u \subseteq \{i \in \mathcal{X}_u \mid q_u(i) = i\}$ for $u \in \mathcal{V}$. Let $g_{uv}(i, j) = 0$ for $(i, j) \in U_u \times U_v$. Let \bar{g} be defined by

$$\bar{g}_v = g_v, \quad v \in \mathcal{V}; \quad (23a)$$

$$\bar{g}_{uv}(i, j) = \begin{cases} \min_{i' \in U_u} g_{uv}(i', j), & i \in U_u, j \notin U_v, \\ \min_{j' \in U_v} g_{uv}(i, j'), & i \notin U_u, j \in U_v, \\ g_{uv}(i, j), & \text{otherwise.} \end{cases} \quad (23b)$$

Then $q \in \mathbb{S}_g$ iff $q \in \mathbb{S}_{\bar{g}}$.

Proof. Note, unary components of g and \bar{g} are equal. We will prove by separate pairwise components.

Direction \Leftarrow . Let us prove the following inequality:

$$\begin{aligned} (\forall i, j \in \mathcal{X}_{uv}) g_{uv}(q_u(i), q_v(j)) - \bar{g}_{uv}(q_u(i), q_v(j)) \\ \leq g_{uv}(i, j) - \bar{g}_{uv}(i, j). \end{aligned} \quad (24)$$

We need to consider only cases where $\bar{g}_{uv}(i, j) \neq g_{uv}(i, j)$. Let $i \in U_u$ and $j \notin U_v$ (the remaining case is symmetric). In this case $q_u(i) = i$. Substituting \bar{g} we have to prove

$$\begin{aligned} g_{uv}(i, q_v(j)) - \min_{i' \in U_u} g_{uv}(i', q_v(j)) \\ \leq g_{uv}(i, j) - \min_{i' \in U_u} g_{uv}(i', j). \end{aligned} \quad (25)$$

Clearly, LHS is zero by assumption about g . At the same time RHS is non-negative since $i \in U_s$. The inequality holds. Inequality (24) implies (by multiplication of pairwise inequalities and unary inequalities with respective components of μ and summing) that

$$(\forall \mu \in \Lambda) \langle g, Q\mu \rangle - \langle \bar{g}, Q\mu \rangle \leq \langle g, \mu \rangle - \langle \bar{g}, \mu \rangle. \quad (26)$$

Inequality (26) is equivalent to

$$\langle \bar{g}, (I - Q)\mu \rangle \leq \langle g, (I - Q)\mu \rangle. \quad (27)$$

Whenever the LHS of (27) is strictly positive then so is the RHS and therefore from $q \in \mathbb{S}_{\bar{g}}$ follows $q \in \mathbb{S}_g$.

Direction \Rightarrow . Assume $q \in \mathbb{S}_g$. By Theorem B.2, there exist dual multipliers φ such that for $g' := g - A^\top \varphi$ the following component-wise inequalities hold:

$$(\forall u \in \mathcal{V}, \forall i \in \mathcal{X}_u) g'_u(q_u(i)) \leq g'_u(i); \quad (28)$$

$$(\forall uv \in \mathcal{E}, \forall ij \in \mathcal{X}_{uv}) g'_{uv}(q_u(i), q_v(j)) \leq g'_{uv}(i, j).$$

Let us expand the pairwise inequality in the case $i \in U_u, j \notin U_v$. Let $q_v(j) = j^*$. Using $q_u(i) = i$ we obtain

$$\begin{aligned} g_{uv}(i, j^*) - \varphi_{uv}(i) - \varphi_{vu}(j^*) \\ \leq g_{uv}(i, j) - \varphi_{uv}(i) - \varphi_{vu}(j). \end{aligned} \quad (29)$$

Terms $\varphi_{uv}(i)$ cancel:

$$g_{uv}(i, j^*) - \varphi_{vu}(j^*) \leq g_{uv}(i, j) - \varphi_{vu}(j). \quad (30)$$

We take min over $i \in U_u$ of both sides:

$$\min_{i \in U_u} g_{uv}(i, j^*) - \varphi_{vu}(j^*) \leq \min_{i \in U_u} g_{uv}(i, j) - \varphi_{vu}(j). \quad (31)$$

Finally we can subtract $\varphi_{uv}(i)$ from both sides to obtain

$$\bar{g}_{uv}^\varphi(i, j^*) \leq \bar{g}_{uv}^\varphi(i, j). \quad (32)$$

The case when $i \notin U_u, j \in U_v$ is symmetric. In the remaining cases, $\bar{g}_{uv}^\varphi(i, j) = \bar{g}(i, j) - \varphi_{uv}(i) - \varphi_{vu}(j) = g(i, j) - \varphi_{uv}(i) - \varphi_{vu}(j) = g^\varphi(i, j)$. In total, \bar{g}^φ satisfies all component-wise inequalities as g' in (28). By Theorem B.2,

$$(\forall \mu \in \Lambda) \langle \bar{g}, Q\mu \rangle \leq \langle \bar{g}, \mu \rangle. \quad (33)$$

We have shown that $\langle \bar{g}, (I - Q)\mu \rangle \geq 0$. It remains to prove that the inequality holds strictly when $Q\mu \neq \mu$. It can be shown that for $q \in \mathbb{S}_g$ at least one of unary inequalities (28) from the support of μ holds strictly and therefore inequality (33) is also strict. \square

Theorem B.4 (Reduction 2). Let $q: \mathcal{X} \rightarrow \mathcal{X}$ component-wise idempotent. Let $\bar{g} = g - \Delta^+$ and $\Delta^+ \in \mathbb{R}^{\mathcal{X}}$ has zero unary components and pairwise components as follows:

$$\begin{aligned} \Delta_{uv}^+(i, j) = \max \{ 0, g_{uv}(i, j) + g_{uv}(q_u(i), q_v(j)) \\ - g_{uv}(i, q_v(j)) - g_{uv}(q_u(i), j) \}. \end{aligned} \quad (34)$$

Then $q \in \mathbb{S}_g$ iff $q \in \mathbb{S}_{\bar{g}}$.

Proof. Note, unary components of g and \bar{g} are equal. We will prove by separate pairwise components.

Direction \Leftarrow . Let us prove the inequality (24). It is equivalent to

$$\Delta_{uv}^+(q_u(i), q_v(j)) \leq \Delta_{uv}^+(i, j). \quad (35)$$

Consider the following cases:

- $q_u(i) = i$ or $q_v(j) = j$: In this case $\Delta_{uv}^+(i, j) = 0 = \Delta_{uv}^+(q_u(i), q_v(j))$ by substitution.
- $q_u(i) \neq i, q_v(j) \neq j$: By idempotency, it must be that $q_u(q_u(i)) = q_u(i)$ and $q_v(q_v(j)) = q_v(j)$. It follows that $\Delta_{uv}^+(q_u(i), q_v(j)) = 0$ and $\Delta_{uv}^+(i, j) \geq 0$ by definition.

Inequality (24) implies (by multiplication of pairwise inequalities and unary inequalities with respective components of μ and summing) that

$$(\forall \mu \in \Lambda) \langle g, Q\mu \rangle - \langle \bar{g}, Q\mu \rangle \leq \langle g, \mu \rangle - \langle \bar{g}, \mu \rangle. \quad (36)$$

Note, cost vector \bar{g} satisfying (36) is called *auxiliary* for g in [15, 23]. Inequality (36) is equivalent to

$$\langle \bar{g}, (I - Q)\mu \rangle \leq \langle g, (I - Q)\mu \rangle. \quad (37)$$

Whenever the LHS of (37) is strictly positive then so is the RHS and therefore from $q \in \mathbb{S}_{\bar{g}}$ follows $q \in \mathbb{S}_g$.

Direction \Rightarrow . Assume $q \in \mathbb{S}_g$. By Theorem B.2, there exist dual multipliers φ such that for $g' := g - A^\top \varphi$ the following component-wise inequalities hold:

$$(\forall u \in \mathcal{V}, \forall i \in \mathcal{X}_u) g'_u(q_u(i)) \leq g'_u(i); \quad (38)$$

$$(\forall uv \in \mathcal{E}, \forall ij \in \mathcal{X}_{uv}) g'_{uv}(q_u(i), q_v(j)) \leq g'_{uv}(i, j).$$

Let $\bar{g}' := \bar{g} - A^\top \varphi = g - \Delta^+ - A^\top \varphi = g' - \Delta^+$. Let us show that component-wise inequalities (38) hold for \bar{g}' . Clearly they hold for unary components and for pairwise components where $\Delta_{uv}^+(i, j) = 0$. Let $uv \in \mathcal{E}$ and $\Delta_{uv}^+(i, j) > 0$. Let $i' = q_u(i)$ and $j' = q_v(j)$. It must be that $i' \neq i$ and $j' \neq j$. Let us denote $a = g'_{uv}(i', j')$, $b = g'_{uv}(i', j)$, $c = g'_{uv}(i, j')$ and $d = g'_{uv}(i', j)$. By idempotency of q , there holds $0 = \Delta_{uv}^+(i', j) = \Delta_{uv}^+(i, j') = \Delta_{uv}^+(i', j')$. Let $\bar{d} := g'_{uv}(i, j) - \Delta_{uv}^+(i, j) = d - (a + d - b - c) = b + c - a$. From (38) we have that $a \leq b, c, d$. It follows that $2a \leq b + c$ or $a \leq b + c - a = \bar{d}$. We proved that $\bar{g}'_{uv}(q_u(i), q_v(j)) \leq \bar{g}'_{uv}(i, j)$. In total, \bar{g}' satisfies all component-wise inequalities, same as g' in (28). By Theorem B.2,

$$(\forall \mu \in \Lambda) \langle \bar{g}, Q\mu \rangle \leq \langle \bar{g}, \mu \rangle. \quad (39)$$

We have shown that $\langle \bar{g}, (I - Q)\mu \rangle \geq 0$. The inequality holds strictly when $Q\mu \neq \mu$. In this case for $q \in \mathbb{S}_g$ at least one of unary inequalities (38) from the support of μ holds strictly and therefore inequality (39) is also strict. \square

Remark. Note, taking into account that $g_{vu}(i', j') = 0$ for $i' \in \mathcal{X}_v \setminus \mathcal{Y}_v, j' \in \mathcal{X}_u \setminus \mathcal{Y}_u$, the reduction that we obtained in all cases can be interpreted as forcing the mixed discrete derivative

$$g_{vu}(i, j') + g_{vu}(i', j) - g_{vu}(i, j) - g_{vu}(i', j') \quad (40)$$

to be non-negative. The cost vector \bar{g} is therefore a (partial) submodular *truncation* of g . One can notice certain similarity with construction of auxiliary problems by Kovtun [15], where full submodularity is enforced. We essentially proved that a part of Kovtun's construction of constructing an auxiliary problem is optimal.

Theorem B.5. Let $q, p \in \mathcal{P}^{2,y}, q \leq p$. Let \bar{g} be defined by (13). Then $q \in \mathbb{S}_f$ iff $q \in \mathbb{S}_{\bar{g}}$.

Proof. Let $g = (I - P)^\top f$. By Theorem B.1, $q \in \mathbb{S}_f$ iff $q \in \mathbb{S}_g$. We need to consider only pairwise terms. Let $uv \in \mathcal{E}$. Since $q \leq p$, if $p_u(i) = i$ then necessarily $q_u(i) = i$. Let p be defined using sets \mathcal{Y}_u as in (8). The reduction \bar{g} in (13) will be composed of reductions by Theorem B.3 and Theorem B.4.

From $g = (I - P)^\top f$ we have that for $i \in \mathcal{X}_u \setminus \mathcal{Y}_u$ and $j \in \mathcal{X}_v \setminus \mathcal{Y}_v$ $g_{uv}(i, j) = 0$. Conditions of Theorem B.3 are satisfied with $U_s = \mathcal{X}_u \setminus \mathcal{Y}_u$. We obtain part of the reduction (13) for cases when $i \notin \mathcal{Y}_u$ or $j \notin \mathcal{Y}_v$. Let us denote the reduced vector \bar{g}' . Applying Theorem B.4 to it, we obtain \bar{g} as defined in (13). \square

We are ready to show the statements claimed in §5.

Theorem 5.1 (Reduction). The theorem formulated in §5 states that mapping $q \leq p$ is in \mathbb{S}_f iff $q(\mathcal{O}_v^*) = \mathcal{O}_v^*$.

Proof. By Theorem B.5, $q \in \mathbb{S}_f$ iff $q \in \mathbb{S}_{\bar{g}}$, where \bar{g} is defined by (13). There holds characterization $q \in \mathbb{S}_{\bar{g}}$ iff $q_u(\mathcal{O}_u^*) = \mathcal{O}_u^*$, where \mathcal{O}_u^* is defined by (12). \square

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. Note, unlike Theorem 5.1, this lemma states a necessary condition only. Let $\mu \in \mathcal{O}^*$. Assume for contradiction that $Q\mu \neq \mu$. In this case, by Theorem 5.1, we have that $\langle \bar{g}, Q\mu \rangle < \langle \bar{g}, \mu \rangle$. Since $\mu \in \Lambda'$ and $Q(\Lambda') \subseteq \Lambda$ there holds $Q\mu \in \Lambda'$. It follows that $Q\mu$ is a feasible solution of a better cost than μ which contradicts optimality of μ . It must be therefore that $Q\mu = \mu$. The claim $q(\mathcal{O}_v^*) = \mathcal{O}_v^*$ follows. \square

We have achieved the following. Suppose that we solve the verification LP with the reduced cost vector \bar{g} . If there holds $p_v(\mathcal{O}_v^*) = \mathcal{O}_v^*$ for all v , then $p \in \mathbb{S}_f$ by Theorem 5.1. In the opposite case, Theorem 5.1 asserts that for all strictly relaxed-improving mappings $q \leq p$ there must hold $q_v(\mathcal{O}_v^*) = \mathcal{O}_v^*$. Therefore the reduced verification LP is valid to be used in the step 5 of Algorithm 1.

A practical aspect of the reduction is that, e.g., TRW-S is able to find a labeling with a negative cost much faster since we have decreased many edge costs. Additionally, minimization over polytope Λ_x with reduced g is a purely submodular problem (recall that $y_v \in \mathcal{X}_v \setminus \mathcal{Y}_v$). We thus can solve it with a regular minimum cut and not QPBO minimum cut.

C. Termination with AC Solvers

For completeness, we give two theorems when the solver is applied to the verification problem with and without reduction. Both results apply when the solver has found an arc consistent solution. The theorem with reduction allows to obtain somewhat stronger guarantees. The guarantees are necessary in order to show that our Algorithm 2 terminates. In the next section we prove guarantees for TRW-S also in the case when it has not converged yet.

Theorem C.1. Consider the verification LP defined by $g = (I - P^\top)f$. Let g^φ be an arc-consistent reparametrization. Then at least one of the two conditions is satisfied:

- (a) $LB(\varphi) \stackrel{\text{def}}{=} g_0 + \sum_u \varphi_u = 0$ and φ is dual optimal;
- (b) $(\exists u \in \mathcal{V}) \mathcal{O}_u(\varphi) \cap \mathcal{Y}_u \neq \emptyset$.

Proof. Assume (b) does not hold: $(\forall u \in \mathcal{V}) \mathcal{O}_u(\varphi) \cap \mathcal{Y}_u = \emptyset$. Then for each node u there is a label $z_u \in \mathcal{O}_u(\varphi) \setminus \mathcal{Y}_u$. By arc consistency, for each edge uv there is a label $j \in \mathcal{O}_u(\varphi) \subseteq \mathcal{X}_u \setminus \mathcal{Y}_u$ such that $g_{uv}^\varphi(z_u, j) = 0$ and similarly, there exists $i \in \mathcal{O}_u(\varphi) \subseteq \mathcal{X}_u \setminus \mathcal{Y}_u$ such that $g_{uv}^\varphi(i, z_v) = 0$.

By construction, $g_{uv}(i, j) = 0$ for all $ij \in \mathcal{X}_{uv} \setminus \mathcal{Y}_{uv}$ and therefore the following modularity equality holds:

$$g_{uv}^\varphi(z_u, z_v) + g_{uv}^\varphi(i, j) = g_{uv}^\varphi(z_u, j) + g_{uv}^\varphi(i, z_v). \quad (41)$$

From local minimality of (z_u, j) we have

$$g_{uv}^\varphi(z_u, j) \leq g_{uv}^\varphi(i, j). \quad (42)$$

By adding (41) and (42) we obtain

$$g_{uv}^\varphi(z_u, z_v) \leq g_{uv}^\varphi(i, z_v) \quad (43)$$

and hence (z_u, z_v) is locally minimal too: $g_{uv}^\varphi(z_u, z_v) = 0$. Therefore $\delta(z)$ and dual point φ satisfy complementarity slackness and hence they are primal-dual optimal and $LB(\varphi) = E_g(z) = 0$. \square

We can now show the result needed to establish termination of Algorithm 2 based on a solver delivering arc consistency, but not necessarily solving the LP relaxation in all cases.

Lemma 4.4: "Let $(\forall v \in \mathcal{V}) \mathcal{O}_v(\varphi) \cap \mathcal{Y}_v = \emptyset$ hold for an arc consistent dual vector φ . Then φ is dual optimal".

Proof. Corollary from Theorem C.1. \square

For the reduced problem \bar{g} , a stronger condition is satisfied.

Theorem C.2. Consider the reduced problem \bar{g} defined in Theorem 5.1. Let \bar{g}^φ be an arc-consistent reparametrization. Then at least one of the two conditions is satisfied:

- (a) $LB(\varphi) = 0$ and φ is optimal;
- (b) $(\exists u \in \mathcal{V}) \mathcal{O}_u(\bar{g}^\varphi) \subseteq \mathcal{Y}_u$.

Proof. Since for problem \bar{g} all labels in $\mathcal{X}_u \setminus \mathcal{Y}_u$ are equivalent w.r.t. unary and pairwise costs, we can contract these sets and w.l.o.g. assume that $\mathcal{X}_u \setminus \mathcal{Y}_u = \{y_u\}$.

Assume (b) does not hold. Then for each node u label y_u is locally minimal: $\bar{g}^\varphi(y_u) = \min_i \bar{g}_u^\varphi(i)$. By arc consistency, for each edge uv there is a label $j \in \mathcal{X}_v$ such that pair (y_u, j) is locally minimal and similarly there exists $i \in \mathcal{X}_v$ such that pair (i, y_v) is locally minimal. From partial submodularity, inequality (40), follows that the pair (y_u, y_v) is locally minimal as well. Therefore, integer labeling y and dual point φ satisfy complementarity slackness and hence they are primal-dual optimal and $LB(\varphi) = E_{\bar{g}}(y) = 0$. \square

Both proofs above are analogues to how fixed points of TRW-S or max-sum diffusion are shown to correspond to exact solutions for submodular problems [22], *i.e.*, we use the same argument to construct an integer feasible solution that satisfy complementary slackness.

D. Implementation with TRW-S

When we consider specifically TRW-S there are several questions regarding correctness and efficiency:

- TRW-S may not achieve arc consistency in finite time. Can we stop it earlier? Will there be some progress possible in terms of sets \mathcal{Y}_u ?
- Can we exploit efficient distance transforms on the reduced problem?

In this section we answer all the above questions positively.

D.1. Algorithm Details

The algorithm will make use of the reduction, pruning cuts and the warm start. The warm start is motivated by that every next outer iteration of Algorithm 1 may result in just a small adjustment to the problem \bar{g} and therefore it is desirable to reuse the messages (reparametrization) in TRW-S from the previous outer iteration. We first give a full specification in Algorithm 3 and state formal properties of the algorithm. Then we prove the claims by deriving some new properties of TRW-S.

Theorem D.1 (Correctness of Algorithm 3). For any stopping condition in line 4, the algorithm terminates in at most $\sum_v (|\mathcal{X}_v| - 1)$ outer iterations and outputs $p \in \mathbb{S}_f$.

Theorem D.2. Message passing in TRW-S for reduced edge term \bar{g}_{uv} can be computed in $O(|\mathcal{Y}_u| + |\mathcal{Y}_v|)$ extra time compared to the message passing for the original term f_{uv} .

Algorithm 3: Efficient Iterative Pruning with TRW-S

Input: Problem $f \in \mathbb{R}^{\mathcal{I}}$, test labeling $y \in \mathcal{X}$;

Output: Improving map $q \in \mathcal{P}^{2,y} \cap \mathbb{S}_f$;

- 1 $(\forall s \in \mathcal{V}) \mathcal{Y}_s := \mathcal{X}_s \setminus \{y_s\}$;
 - 2 Set φ to the initial reparametrization / messages if available;
 - 3 **repeat**
 - 4 Construct reduced characterization problem \bar{g} for problem f and sets $(\mathcal{Y}_u \mid u \in \mathcal{V})$ according to Table 6;
 - 5 **repeat**
 - 6 Run 10 TRW-S iterations with problem \bar{g} and reparametrization φ . Provides LB_T, φ, O_u and best x ;
 - 7 **if** $(\forall s \in \mathcal{V}) O_u \cap \mathcal{Y}_s = \emptyset$ **then return** q ;
 - 8 **if** $E_{\bar{g}}(x) < 0$ **then**
 - 9 Apply pruning cut with x ;
 - 10 Apply single node pruning;
 - 11 **goto** step 4 to rebuild \bar{g} ;
 - 12 **until** any stopping condition (*e.g.*, iteration limit);
 - 13 Mark immovable: $(\forall u \in \mathcal{V}) \mathcal{Y}_u := \mathcal{Y}_u \setminus O_u$;
 - 14 Apply single node pruning;
-

Proof (of Theorem D.1). Using Margin Theorem D.3 below, either current mapping q is returned or whenever we decide to stop TRW-S iterations, the set of minimal labels O_u for some s contains a new label to be marked as immovable. It follows that Algorithm 3 terminates. The convergence of TRW-S does not affect correctness of our algorithm, only a possibly non-maximal map q is found due to non-optimal dual point used in step 13. The termination condition in step 7 indicates that y is an optimal primal solution and in this case it must be that $LB = 0$, therefore current φ is an optimal dual point. In this case $O_u^* \subseteq O_u$, where set O_u^* is the support set of all primal optimal solutions (see Algorithm 1). We therefore have $O_u^* \subseteq O_u \subseteq \mathcal{X}_u \setminus \mathcal{Y}_u$ for all u . It follows that stopping condition of generic Algorithm 1 is satisfied as well and hence $q \in \mathbb{S}_f$. \square

Note, whenever the algorithm found a pruning cut or used local pruning conditions, no loss of maximality occurs. In our experiments for some instances, the algorithm finished before ever reaching the step 13. In such cases the reduction $q \in \mathbb{S}_f$ is the maximum.

D.2. Properties of TRW-S

In this section we give several theoretical guarantees for TRW-S when it is applied to solve the verification problem $(I - [p]^T)f$ or to its reduced version.

TRW-S algorithm, though observed to always converge in practice to a fixed point, has very weak theoretical convergence guarantees. It is known that there is a convergent subsequence whose limit satisfies Week Tree Agreement

(WTA) [12]. A finite iteration epsilon variant of this statement exists [21]. In either case we have no guarantee to obtain an arc consistent solution required to prove Proposition 4.3. The major obstacles are: (i) WTA is achieved only in the limit and (ii) there is no guarantee that the set of labels which are in WTA (let alone its AC subset) converges as well.

Fortunately, TRW-S enjoys very useful for us properties especially when it is applied to the reduced verification problem. We first give a short specification of TRW-S. In our notation, the order of TRW-S updates is fully specified by orientations of edges in \mathcal{E} . For a serial implementation, this order can be completed to a total order on \mathcal{V} by defining $u < v$ iff $uv \in \mathcal{E}$. We specify only the forward pass of TRW-S as Algorithm 4 and note that the backward pass is obtained by reversing all edges. In Algorithm 4, n_u is the

Algorithm 4: TRW-S Forward Iteration,
c.f. [12, Fig. 3]

Input: Problem $f \in \mathbb{R}^{\mathcal{I}}$, reparametrization φ ;
Output: Updated reparametrization φ , LB_T , best labeling x , locally optimal labels $\mathcal{O}_u(\varphi)$;

```

1 for  $u \in \mathcal{V}$  do
2   for  $v \in \text{nb}(u) \mid uv \in \mathcal{E}$  do
3      $\varphi_{vu}(j) :=$ 
4      $\min_i \left[ \frac{1}{n_u} f_u^\varphi(i) - \varphi_{uv}(i) + f_{uv}(i, j) \right];$ 
5    $\mathcal{O}_u(\varphi) := \text{argmin}_i f_u^\varphi(i); x_u \in \mathcal{O}_u(\varphi);$ 
6    $LB_T(\varphi) := \sum_{u \in \mathcal{V}} \frac{n_{\text{term}}(u)}{n_u} \min_i f_u^\varphi(i);$ 

```

number of chains that meet in node u and $n_{\text{term}}(u)$ is the number of chains that terminate in u .

Viewed as computing the division of costs between the chains, TRW-S has the following properties. To a chain τ there is associated its oriented graph $(\mathcal{V}^\tau, \mathcal{E}^\tau)$ and its share of the unary terms in the decomposition, f^τ . We assume that each pairwise term f_{uv} is associated to exactly one chain passing through edge uv .

After the first backward pass, the algorithm maintains the following invariants. Consider the step of processing vertex $u^* \in \mathcal{V}$. For a chain τ which passes through edge $uv \in \mathcal{E}$ there holds:

- If $u^* < u$, message $\varphi_{uv}(i)$ is equal to the *right* min-marginals of the chain:

$$\varphi_{uv}(i) = \min_{\substack{(x_w \mid w > v) \\ x_u = i}} \left(\sum_{\substack{w \in \mathcal{V}^\tau \\ w > u}} f_w^\tau(x_w) + \sum_{\substack{u'v' \in \mathcal{E}^\tau \\ u' \leq u}} f_{u'v'}(x_{u'}, x_{v'}) \right). \quad (44)$$

- If $u^* > u$, message $\varphi_{vu}(j)$ is equal to the *left* min-

marginals of the chain:

$$\varphi_{vu}(j) = \min_{\substack{(x_w \mid w < v) \\ x_v = j}} \left(\sum_{\substack{w \in \mathcal{V}^\tau \\ w < v}} f_w^\tau(x_w) + \sum_{\substack{u'v' \in \mathcal{E}^\tau \\ u' \leq u}} f_{u'v'}(x_{u'}, x_{v'}) \right). \quad (45)$$

Weak tree agreement is the condition when the lower bound cannot be improved further by TRW-S [12]. It requires that among all optimal assignments of the chains there is a consistent subset.

We now introduce a measure of how far the TRW-S algorithm is from a fixed integer solution y . For a node u we define *node margin* of u as the value

$$m_u(\varphi) = \min_i (f_u^\varphi(i) - f_u^\varphi(y_u)). \quad (46)$$

The margin is related to the set $\mathcal{O}_u(\varphi)$ of local minimizers. If it is negative, $\mathcal{O}_u(\varphi)$ does not contain y_u and the negative value measures how well it is separated. Let us define the *problem margin* as the value

$$m(\varphi) = \min_{s \in \mathcal{V}} m_u(\varphi). \quad (47)$$

For the convenience of analysis, we continue to denote the cost vector to which Algorithm 4 is applied as f . In the context of Algorithm 3, we temporarily let $f := \tilde{g}$.

Theorem D.3 (Margin for TRW-S). For the reduced verification problem f and reparametrization φ build by TRW-S there holds:

$$LB_T(\varphi) = 0 \text{ iff } m_u(\varphi) \geq 0. \quad (48)$$

Proof. (\Rightarrow) Let $LB_T(\varphi) = 0$. By construction, $E_f(y) = 0$ so the duality gap is zero and therefore φ is dual optimal and $\delta(y)$ is primal optimal. By complementary slackness it must be that $(\forall u \in \mathcal{V}) f_u^\varphi(y_u)$ is locally minimal. Therefore problem margin is exactly zero. This part of the proof holds also for non-reduced verification problem and any reparametrization such that $LB_T(\varphi) = 0$.

(\Leftarrow) Let now the problem margin be non-negative. Let us consider edge $uv \in \mathcal{E}$ and let f^τ be the cost vector of the monotonic chain τ passing through edge uv at the beginning of iteration processing node v . Since u was already processed, the value f_u^φ is proportional to min-marginals of chain τ in u and we have $m_u(\varphi) \geq 0$. It follows that there is a minimizer $x \in \min_x \langle f^\tau, \delta(x) \rangle$ such that $x_u = y_u$. For this minimizer there holds

$$x \in \text{argmin}_{\tilde{x} \mid \tilde{x}_u = x_u} \langle f^\tau, \delta(\tilde{x}) \rangle. \quad (49)$$

This is because we constrained \tilde{x} to coincide with the global chain minimizer in v . This conditional minimizer however clearly does not depend on f_v^τ , the unary term associated to chain τ .

Consider the operation of averaging min-marginals over node v . By assumption, after averaging operation, $m_v \geq 0$ and therefore the local minimum over each chain is attained in y_v . The averaging operation only changes the value f_v^τ . The conditional minimizer

$$x' \in \operatorname{argmin}_{\tilde{x} | \tilde{x}_v = y_v} \langle f^\tau, \delta(\tilde{x}) \rangle \quad (50)$$

does not depend on f_v^τ .

Let r precede u in τ . Assume for contradiction that $x'_u \neq y_u$. Because u has been processed, the value $\varphi_{ur}(i)$ is the left-min-marginal for chain τ in node u and label $i \in \mathcal{X}_u$. Denote $\vec{f}_u^\tau = \varphi_{ur}(i) + f_u^\tau(i)$. The relations (49) and (50) can be expressed as

$$y_u \in \operatorname{argmin}_i (\vec{f}_u^\tau + f_{uv}(i, x_v)); \quad (51a)$$

$$x'_u \in \operatorname{argmin}_i (\vec{f}_u^\tau + f_{uv}(i, y_v)). \quad (51b)$$

From (51a) we have that $\vec{f}_u^\tau(y_u) + f_{uv}(y_u, x_v) \leq \vec{f}_u^\tau(x'_u) + f_{uv}(x'_u, x_v)$. We use now the partial submodularity fulfilled for f ,

$$f_{uv}(y_u, x_v) + f_{uv}(x'_u, y_v) \geq f_{uv}(x'_u, x_v) \quad (52)$$

and obtain that

$$\vec{f}_u^\tau(y_u) \leq \vec{f}_u^\tau(x'_u) + f_{uv}(x'_u, y_v). \quad (53)$$

Adding on the LHS the term $f_{uv}(y_u, y_v) = 0$, we get

$$\vec{f}_u^\tau(y_u) + f_{uv}(y_u, y_v) \leq \vec{f}_u^\tau(x'_u) + f_{uv}(x'_u, y_v). \quad (54)$$

This inequality allows to conclude that y_u is also a minimizer to (51b). Therefore either $x'_u = y_u$ or y_u is an equally good substitute.

We have shown that as soon as there is zero margin, for each chain the part of the minimizer over already processed nodes can be selected equal to y . By induction, at the end of the sweep, y is a minimizer for each chain. Hence WTA is achieved. In this case, there is zero integrality gap and $LB_T = E_f(y) = 0$. \square

It follows that on every iteration (but the first initializing one) of TRW-S either the problem margin is negative and therefore $(\exists u) \mathcal{O}_u(\varphi) \cap \mathcal{Y}_u \neq \emptyset$ (there is always something to prune) or WTA is achieved and the algorithm terminates with y being optimal integer solution. This proves Theorem D.1. Furthermore, if the stopping condition in line 12 is the iteration limit, Algorithm 3 runs in polynomial time.

D.3. Efficient Message Passing

Suppose that some pairwise potentials are specially structured, so that messages in message passing algorithms

$i \notin \mathcal{Y}_u$	$\bar{g}_u(i) = 0;$
$i \in \mathcal{Y}_u$	$\bar{g}_u(i) = f_u(i) - f_u(y_u);$
$i \notin \mathcal{Y}_u, j \notin \mathcal{Y}_v$	$\bar{g}_{uv}(i, j) = 0;$
$i \notin \mathcal{Y}_u, j \in \mathcal{Y}_v$	$\bar{g}_{uv}(i, j) = \Delta_{vu}(j),$ $\Delta_{vu}(j) = \min_{i' \notin \mathcal{Y}_u} [f_{uv}(i', j) - f_{uv}(i', y_v)];$
$i \in \mathcal{Y}_u, j \notin \mathcal{Y}_v$	$\bar{g}_{uv}(i, j) = \Delta_{uv}(i),$ $\Delta_{uv}(i) = \min_{j' \notin \mathcal{Y}_v} [f_{uv}(i, j') - f_{uv}(y_u, j')];$
$i \in \mathcal{Y}_u, j \in \mathcal{Y}_v$	$\bar{g}_{uv}(i, j) = \min \left\{ f_{uv}(i, j) - f_{uv}(y_u, y_v), \right.$ $\left. \Delta_{vu}(j) + \Delta_{uv}(i) \right\}.$

Table 6. Components of the Reduced Verification Problem

can be computed in linear time in the number of labels instead of quadratic time, see [5]. This is the case for many potentials with a linear ordering, e.g. absolute differences, squared distance and truncated versions thereof, including the Potts model.

It would be very helpful to be able to compute messages fast in these cases also for our reduced verification problem \bar{g} . The components of \bar{g} specified by Theorem 5.1 can be expressed directly in components of f as proposed in Table 6. Passing a message on edge uv amounts to calculate an expression of the form

$$\varphi_{vu}(j) := \min_{i \in \mathcal{X}_u} [a(i) + \bar{g}_{uv}(i, j)] \quad (55)$$

for some vector $a \in \mathbb{R}^{\mathcal{X}_u}$. For $j \notin \mathcal{Y}_v$, substituting pairwise terms of \bar{g} it is expanded as

$$\begin{aligned} \varphi_{vu}(j) &= \min_{i \in \mathcal{X}_u} [a(i) + \Delta_{uv}(i)] \\ &= \min_{i \in \mathcal{X}_u} [a(i) + \Delta_{uv}(i)] + \Delta_{vu}(j), \end{aligned} \quad (56)$$

where we defined that $\Delta_{uv}(U_u) = \Delta_{vu}(U_v) = 0$.

For $j \in \mathcal{Y}_v$, substituting pairwise terms of \bar{g} and denoting $c = f_{uv}(y_u, y_v)$ the message is expanded as

$$\begin{aligned} &\min \left\{ \min_{i \in U_u} a(i) + \Delta_{vu}(j), \right. \\ &\left. \min_{i \in \mathcal{Y}_u} \left[a(i) + \min \{ f_{uv}(i, j) - c, \Delta_{uv}(i) + \Delta_{vu}(j) \} \right] \right\} \\ &= \min \left\{ \min_{i \in \mathcal{Y}_u} [a(i) + f_{uv}(i, j)] - c, \right. \\ &\left. \min_{i \in \mathcal{X}_u \setminus \mathcal{Y}_u} [a(i) + \Delta_{uv}(i)] + \Delta_{vu}(j) \right\}. \end{aligned} \quad (57)$$

We therefore need to calculate (56) for all $j \in \mathcal{X}_u$ because the expression reoccurs in (57) for $j \in \mathcal{Y}_v$. Then for $j \in \mathcal{Y}_v$ it remains to take the minimum of a regular message and $\varphi_{vu}(j)$ by (56).

Algorithm 5: Message Passing for Pruning

Input: Unary term $a: \mathcal{Y}_u \cup \{y_u\} \rightarrow \mathbb{R}$;

Output: Message $\varphi_{vu}: \mathcal{Y}_v \cup \{y_v\} \rightarrow \mathbb{R}$;

/* Offset constant in (56) */

$$1 \quad m_1 := \min_{i \in \mathcal{Y}_u \cup \{y_u\}} [a(i) + \Delta_{uv}(i)];$$

/* Message passing for f */

$$2 \quad (\forall j \in \mathcal{Y}_v) \quad \varphi_{vu}(j) := \min_{i \in \mathcal{Y}_u \cup \{y_u\}} [a(i) + f_{uv}(i, j)];$$

/* Message correction for \bar{g} */

$$3 \quad (\forall j \in \mathcal{Y}_v) \quad \varphi_{vu}(j) := \min\{\varphi_{vu}(j) - c, \Delta_{vu}(j) + m_1\};$$

$$4 \quad \varphi_{vu}(y_v) := m_1;$$

We thus have reduced the message passing for \bar{g} to the message passing for f , $O(|\mathcal{X}_u| + |\mathcal{X}_v|)$ operations to calculate expression (56) and $O(|\mathcal{X}_v|)$ operations for the outer minimum. Therefore we can compute the message for the modified energy in time $O(|\mathcal{X}_u| + |\mathcal{X}_v|)$. We can take this complexity down to the theoretical perfection as follows. Recall that labels in $\mathcal{X}_u \setminus \mathcal{Y}_u$ can be contracted to a single one. In this case calculating (56) for all $j \in \mathcal{Y}_u$ takes only $O(|\mathcal{Y}_u| + |\mathcal{Y}_v|)$ time. Using the non-uniform min-convolution algorithm of [37] the message passing for the labels in the set \mathcal{Y}_u to labels in \mathcal{Y}_v can be implemented in $O(|\mathcal{Y}_u| + |\mathcal{Y}_v|)$ time. We have obtained the complexity matching to the total number of active labels in the problem. The more labels are marked as immovable in the course of the algorithm (sets \mathcal{Y}_u reduce), the less work is required. The final message update is specified in Algorithm 5, where the contracted labels $\mathcal{X}_u \setminus \mathcal{Y}_u$ are represented by y_u . In the implementation for each edge uv we need to store the offsets $\Delta_{uv}: \mathcal{Y}_u \rightarrow \mathbb{R}$ and $\Delta_{vu}: \mathcal{Y}_v \rightarrow \mathbb{R}$ and $c \in \mathbb{R}$.

E. Detailed Experimental Evaluation

Datasets and Evaluation. We give a brief characterization of all 38 test problem instances and report the obtained total percentage of persistent variables of our and competing methods in Table 2. The datasets `mrf-stereo` and `mrf-photomontage` originate from the Middlebury MRF benchmark [31]. The `color-seg` and `color-seg-n4` datasets were taken from the OpenGM MRF benchmark [10, 9], `ProteinFolding` originates from [16, 36]. All datasets are made available in the OpenGM-format [10, 9].

Detailed quantitative experimental evaluation can be found in Table 7. In addition to the per-label measure of partial optimality (15), to allow for future comparisons we report also the *logarithmic measure*. It is motivated by the fact that eliminating one label in a variable with say 2 states brings more information than eliminating one label in a variable with 100 states. We propose to measure the total decrease of the number of configurations of the search

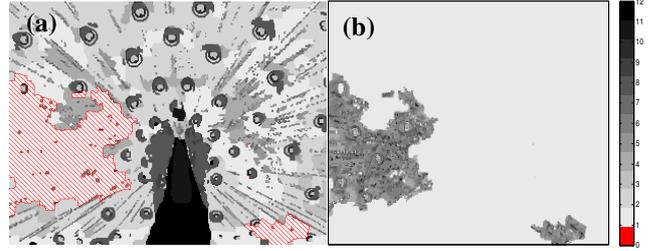


Figure 2. Instance `pfau`. (a) Proved part of optimal solution (red solution not determined / non-unique). (b) Reminder of the optimization problem: number of remaining labels in every pixel.

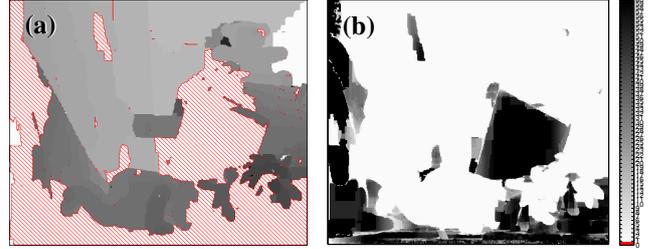


Figure 3. Instance `ted`. (a) Proved part of optimal solution (red solution not determined / non-unique). (b) Reminder of the optimization problem: number of remaining labels in every pixel.

space, e.g., from $|\mathcal{X}|$ to $p(|\mathcal{X}|)$, in the logarithmic domain:

$$1 - \frac{\log \prod_{v \in \mathcal{V}} |p_v(\mathcal{X}_v)|}{\log \prod_{v \in \mathcal{V}} |\mathcal{X}_v|} = 1 - \frac{\sum_{v \in \mathcal{V}} \log |p_v(\mathcal{X}_v)|}{\sum_{v \in \mathcal{V}} \log |\mathcal{X}_v|}. \quad (58)$$

In Figures 2-4 we give examples where the method was performing well. Figures 5 and 6, on the contrary reveal some cases of very poor performance. For example for `photomontage/pano` instance, we report 80% solution completeness, but these 80% only correspond to trivial hard constraints in the problem. Other methods perform worse mainly because they consider determining complete optimal labels only ([29]–TRWS) or intervals of labels (MQPBO).

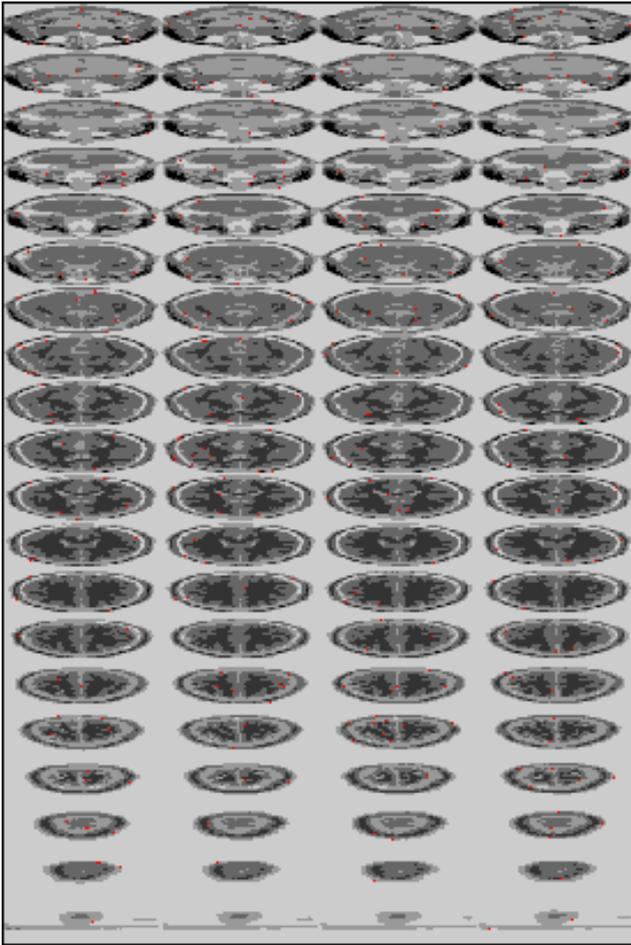


Figure 4. Instance `brain-9mm/brain-0`. Slices of the 3D volumetric problem. Proved part of optimal solution (red = solution not determined / non-unique).

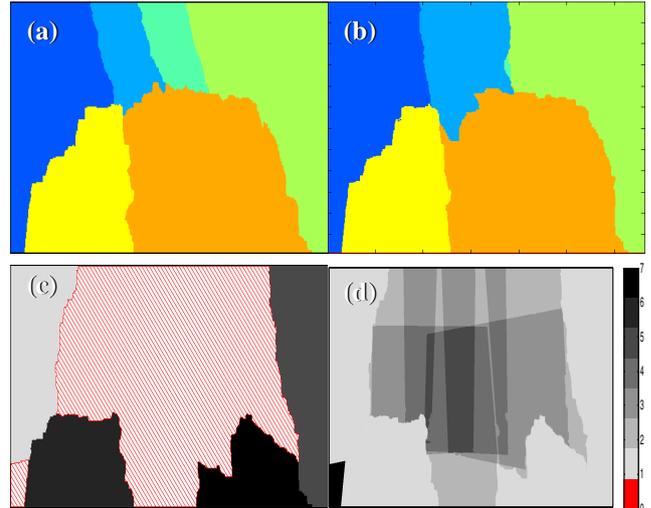


Figure 5. Instance `pano`: label encodes the image index for photomontage. (a) (b) Two labelings by TRW-S with slightly different initializations. It is clear that there is high ambiguity. (c) Part of the solution that was proved optimal and unique. (d) Remainder of the optimization problem (number of non-eliminated labels). It is clear that the method essentially removed hard constraints implied by different fields of view of images composing the panorama.

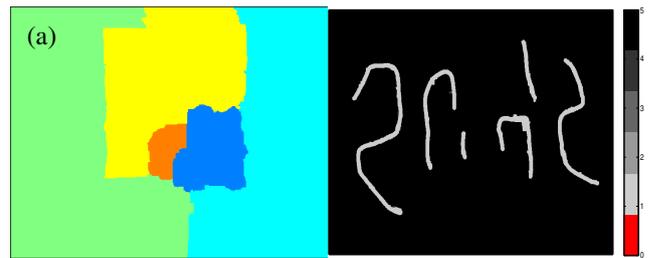


Figure 6. Instance `family`. (a) Labeling by TRW-S. (b) Remainder of the optimization problem (number of labels per pixel). It is clear that the method essentially only followed the hard constraints corresponding to the scribbles provided by the user and constrained very few pixels on top of that.

Instance	Algorithm	Time needed overall (s)	Time for initial solution (s)	#iterations Algorithm 1,2	#iterations TRWS	Logarithmic percentage partial optimality	Percentage excluded labels
ProteinFolding							
1CKK	Our-CPLEX	2757.62	1177.62	5	†	14.24%	27.04%
	Our-TRWS	5.76	5.00	3	1000+15	13.83%	26.53%
	MQPBO-10	5670.00	0.00	0	†	0.00%	0.00%
	MQPBO	825.00	0.00	0	†	0.00%	0.00%
	[29]-CPLEX	2502.69	2493.65	1	†	0.00%	0.00%
	[29]-TRWS	47.57	30.19	2	288+185	0.00%	0.00%
1CM1	Our-CPLEX	4070.00	992.15	7	†	8.38%	34.28%
	Our-TRWS	6.03	4.70	4	1000+65	8.07%	29.98%
	MQPBO-10	5520.00	0.00	0	†	0.00%	0.00%
	MQPBO	723.00	0.00	0	†	0.00%	0.00%
	[29]-CPLEX	2388.46	2198.04	3	†	0.00%	0.00%
	[29]-TRWS	51.33	21.60	3	242+358	0.00%	0.00%
1SY9	Our-CPLEX	2628.72	416.74	5	†	25.34%	51.30%
	Our-TRWS	6.88	5.50	4	1000+15	28.06%	57.98%
	MQPBO-10	7494.00	0.00	0	†	0.00%	0.00%
	MQPBO	2112.00	0.00	0	†	0.00%	0.11%
	[29]-CPLEX	1067.46	910.90	4	†	0.00%	0.00%
	[29]-TRWS	66.73	46.77	5	400+174	0.00%	0.00%
2BBN	Our-CPLEX	9677.42	5476.81	5	†	2.12%	8.58%
	Our-TRWS	10.00	8.60	3	1000+10	2.64%	14.17%
	MQPBO-10	1736.00	0.00	0	†	0.00%	0.00%
	MQPBO	2429.00	0.00	0	†	0.00%	0.00%
	[29]-CPLEX	9776.60	9771.18	1	†	0.00%	0.00%
	[29]-TRWS	54.21	42.90	2	242+146	0.00%	0.00%
2BCX	Our-CPLEX	36222.90	6998.66	5	†	4.81%	15.66%
	Our-TRWS	9.14	7.90	3	1000+55	4.39%	14.21%
	MQPBO-10	1008.00	0.00	0	†	0.00%	0.00%
	MQPBO	1288.00	0.00	0	†	0.00%	0.00%
	[29]-CPLEX	11419.60	11409.90	2	†	0.00%	0.00%
	[29]-TRWS	55.26	39.60	2	252+194	0.00%	0.00%
2BE6	Our-CPLEX	1381.60	765.84	4	†	9.14%	17.68%
	Our-TRWS	4.67	3.91	4	1000+60	8.96%	15.12%
	MQPBO-10	3728.00	0.00	0	†	0.00%	0.05%
	MQPBO	540.00	0.00	0	†	0.00%	0.00%
	[29]-CPLEX	1552.95	1552.88	1	†	0.00%	0.00%
	[29]-TRWS	40.12	28.12	2	363+230	0.00%	0.00%
2F3Y	Our-CPLEX	3628.90	2546.68	5	†	6.22%	10.66%
	Our-TRWS	5.83	5.20	3	1000+10	8.39%	13.74%
	MQPBO-10	5138.00	0.00	0	†	0.00%	0.00%
	MQPBO	928.00	0.00	0	†	0.00%	0.05%
	[29]-CPLEX	4618.78	4618.76	1	†	0.00%	0.00%
	[29]-TRWS	41.87	33.03	3	321+164	0.00%	0.00%
2FOT	Our-CPLEX	7458.75	1996.55	5	†	4.10%	11.64%
	Our-TRWS	6.25	5.30	4	1000+25	4.01%	11.01%
	MQPBO-10	4961.00	0.00	0	†	0.00%	0.09%
	MQPBO	1054.00	0.00	0	†	0.00%	0.07%
	[29]-CPLEX	4473.58	4440.51	1	†	0.00%	0.00%
	[29]-TRWS	61.92	44.42	2	398+222	0.00%	0.00%
2HQW	Our-CPLEX	5721.95	1946.20	6	†	10.30%	17.30%
	Our-TRWS	6.49	4.80	6	1000+160	8.33%	18.08%
	MQPBO-10	7228.00	0.00	0	†	0.00%	0.00%
	MQPBO	1193.00	0.00	0	†	0.00%	0.00%
	[29]-CPLEX	2163.98	2161.07	1	†	0.00%	0.00%
	[29]-TRWS	44.07	35.46	2	382+121	0.00%	0.00%
2O60	Our-CPLEX	12085.40	3007.95	6	†	4.22%	12.81%
	Our-TRWS	7.74	6.50	3	1000+55	4.94%	15.55%
	MQPBO-10	7516.00	0.00	0	†	0.00%	0.00%
	MQPBO	1997.00	0.00	0	†	0.00%	0.00%
	[29]-CPLEX	6137.07	6128.14	1	†	0.00%	0.00%
	[29]-TRWS	93.87	46.42	2	352+369	0.00%	0.00%
3BXL	Our-CPLEX	3247.11	915.86	7	†	4.97%	17.18%
	Our-TRWS	7.44	5.90	4	1000+60	4.66%	12.35%

Instance	Algorithm	Time needed overall (s)	Time for initial solution (s)	#iterations Algorithm 1,2	#iterations TRWS	Logarithmic percentage partial optimality	Percentage excluded labels
	MQPBO-10	6709.00	0.00	0	†	0.00%	0.00%
	MQPBO	1291.00	0.00	0	†	0.00%	0.00%
	[29]-CPLEX	1776.23	1598.07	2	†	0.00%	0.00%
	[29]-TRWS	44.71	25.52	2	227+216	0.00%	0.00%
pdb1b25	Our-CPLEX	1599.67	55.01	28	†	76.76%	84.05%
	Our-TRWS	5.18	2.92	18	530+150	83.00%	87.84%
	MQPBO-10	27.00	0.00	0	†	0.00%	2.53%
	MQPBO	2.00	0.00	0	†	0.00%	1.99%
	[29]-CPLEX	324.64	72.11	14	†	18.84%	22.32%
	[29]-TRWS	119.71	27.62	14	443+1238	18.92%	22.34%
pdb1d2e	Our-CPLEX	154.76	25.44	5	†	97.30%	97.98%
	Our-TRWS	1.67	1.13	7	420+75	96.97%	98.25%
	MQPBO-10	12.00	0.00	0	†	0.00%	4.61%
	MQPBO	0.00	0.00	0	†	0.00%	2.74%
	[29]-CPLEX	483.55	34.59	25	†	55.53%	58.94%
	[29]-TRWS	83.82	6.16	47	190+2775	55.69%	58.98%
pdb1fmj	Our-CPLEX	99.33	12.35	7	†	92.58%	94.90%
	Our-TRWS	1.05	0.60	14	540+135	83.18%	87.09%
	MQPBO-10	6.00	0.00	0	†	0.00%	2.92%
	MQPBO	0.00	0.00	0	†	0.00%	2.04%
	[29]-CPLEX	77.30	16.97	11	†	15.94%	18.83%
	[29]-TRWS	16.67	3.10	11	186+677	16.18%	18.91%
pdb1i24	Our-TRWS	0.06	0.02	2	60+5	99.73%	99.94%
	MQPBO-10	3.00	0.00	0	†	0.00%	2.85%
	MQPBO	0.00	0.00	0	†	0.00%	3.43%
	[29]-CPLEX	5.66	5.66	0	†	100.00%	100.00%
	[29]-TRWS	0.82	0.82	0	115+0	100.00%	100.00%
pdb1iqc	Our-CPLEX	111.58	18.51	5	†	99.10%	99.63%
	Our-TRWS	0.74	0.40	8	200+35	96.39%	97.10%
	MQPBO-10	8.00	0.00	0	†	0.00%	6.06%
	MQPBO	0.00	0.00	0	†	0.00%	4.90%
	[29]-CPLEX	229.09	24.36	20	†	35.32%	41.15%
	[29]-TRWS	36.03	4.06	28	169+2058	40.50%	45.56%
pdb1jmx	Our-CPLEX	142.20	15.52	9	†	97.24%	98.69%
	Our-TRWS	0.67	0.29	10	200+75	93.46%	95.83%
	MQPBO-10	8.00	0.00	0	†	0.00%	3.76%
	MQPBO	0.00	0.00	0	†	0.00%	3.73%
	[29]-CPLEX	121.71	16.21	19	†	35.86%	39.98%
	[29]-TRWS	20.02	3.59	24	188+1098	35.26%	39.12%
pdb1kgn	Our-CPLEX	196.03	17.97	10	†	89.22%	93.23%
	Our-TRWS	1.37	0.76	9	400+170	88.92%	93.16%
	MQPBO-10	9.00	0.00	0	†	0.00%	3.24%
	MQPBO	0.00	0.00	0	†	0.00%	2.27%
	[29]-CPLEX	161.57	24.37	12	†	39.42%	39.67%
	[29]-TRWS	53.49	6.45	17	268+1824	13.20%	13.36%
pdb1kwh	Our-CPLEX	105.77	9.63	10	†	79.09%	85.64%
	Our-TRWS	0.46	0.27	8	440+50	76.53%	83.26%
	MQPBO-10	5.00	0.00	0	†	0.00%	2.99%
	MQPBO	0.00	0.00	0	†	0.00%	3.43%
	[29]-CPLEX	51.33	12.89	9	†	25.54%	31.15%
	[29]-TRWS	9.15	2.43	8	208+401	25.43%	31.13%
pdb1m3y	Our-CPLEX	73.60	18.58	3	†	98.54%	99.47%
	Our-TRWS	0.79	0.65	3	340+10	97.53%	99.08%
	MQPBO-10	8.00	0.00	0	†	0.00%	6.38%
	MQPBO	0.00	0.00	0	†	0.00%	5.72%
	[29]-CPLEX	120.60	25.18	14	†	31.05%	27.97%
	[29]-TRWS	28.19	4.82	12	200+1135	31.08%	27.98%
pdb1qks	Our-CPLEX	138.12	15.19	8	†	98.30%	98.93%
	Our-TRWS	0.30	0.12	4	80+20	98.57%	99.38%
	MQPBO-10	9.00	0.00	0	†	0.00%	5.09%
	MQPBO	0.00	0.00	0	†	0.00%	3.68%
	[29]-CPLEX	96.77	15.82	12	†	28.18%	26.37%

Instance	Algorithm	Time needed overall (s)	Time for initial solution (s)	#iterations Algorithm 1,2	#iterations TRWS	Logarithmic percentage partial optimality	Percentage excluded labels
	[29]-TRWS	27.99	3.24	15	161+1154	30.63%	28.46%
color-seg							
colseg-cow3	Our-TRWS	66.30	48.10	6	1000+140	99.96%	99.97%
	Kovtun	1.00	0.00	0	†	0.00%	99.89%
	MQPBO-10	206.00	0.00	0	†	0.00%	43.55%
	MQPBO	24.00	0.00	0	†	0.00%	32.06%
	[29]-TRWS	7530.72	690.55	14	826+6056	99.95%	99.95%
colseg-cow4	Our-TRWS	91.26	48.31	13	1000+310	99.92%	99.93%
	Kovtun	2.00	0.00	0	†	0.00%	99.90%
	MQPBO-10	46.00	0.00	0	†	0.00%	0.56%
	MQPBO	40.00	0.00	0	†	0.00%	0.37%
	[29]-TRWS	7395.03	742.58	10	848+6349	99.80%	99.80%
colseg-garden4	Our-TRWS	0.49	0.15	5	70+20	99.91%	99.94%
	Kovtun	0.00	0.00	0	†	0.00%	94.96%
	MQPBO-10	14.00	0.00	0	†	0.00%	4.27%
	MQPBO	1.00	0.00	0	†	0.00%	0.21%
	[29]-TRWS	33.68	6.75	5	167+488	99.89%	99.89%
color-seg-n4							
clownfish-small	Our-TRWS	1.72	0.68	3	80+10	>99.99%	>99.99%
	Kovtun	1.00	0.00	0	†	0.00%	74.11%
	MQPBO-10	536.00	0.00	0	†	0.00%	15.83%
	MQPBO	41.00	0.00	0	†	0.00%	4.67%
	[29]-TRWS	151.98	30.01	6	223+610	99.97%	99.97%
crops-small	Our-TRWS	1.87	1.02	2	120+5	100.00%	100.00%
	Kovtun	1.00	0.00	0	†	0.00%	64.70%
	MQPBO-10	577.00	0.00	0	†	0.00%	14.32%
	MQPBO	33.00	0.00	0	†	0.00%	0.71%
	[29]-TRWS	677.08	34.88	40	260+3578	99.00%	99.00%
fourcolors	Our-TRWS	0.57	0.08	2	20+5	99.96%	99.97%
	Kovtun	0.00	0.00	0	†	0.00%	69.52%
	MQPBO-10	37.00	0.00	0	†	0.00%	0.00%
	MQPBO	3.00	0.00	0	†	0.00%	0.00%
	[29]-TRWS	31.28	2.60	8	34+238	99.92%	99.92%
lake-small	Our-TRWS	1.28	0.43	2	50+5	100.00%	100.00%
	Kovtun	1.00	0.00	0	†	0.00%	74.87%
	MQPBO-10	607.00	0.00	0	†	0.00%	15.31%
	MQPBO	31.00	0.00	0	†	0.00%	6.65%
	[29]-TRWS	13.75	13.75	0	95+-95	100.00%	100.00%
palm-small	Our-TRWS	2.48	1.37	3	160+10	>99.99%	>99.99%
	Kovtun	1.00	0.00	0	†	0.00%	68.65%
	MQPBO-10	510.00	0.00	0	†	0.00%	0.48%
	MQPBO	19.00	0.00	0	†	0.00%	0.00%
	[29]-TRWS	846.27	39.97	19	291+4582	98.20%	98.20%
penguin-small	Our-TRWS	1.21	0.54	2	90+5	100.00%	100.00%
	Kovtun	0.00	0.00	0	†	0.00%	91.99%
	MQPBO-10	193.00	0.00	0	†	0.00%	1.42%
	MQPBO	13.00	0.00	0	†	0.00%	1.03%
	[29]-TRWS	15.67	15.67	0	152+-152	100.00%	100.00%
pfau-small	Our-TRWS	18.77	7.22	48	950+470	89.43%	93.41%
	Kovtun	1.00	0.00	0	†	0.00%	5.59%
	MQPBO-10	591.00	0.00	0	†	0.00%	0.70%
	MQPBO	16.00	0.00	0	†	0.00%	0.00%
	[29]-TRWS	799.08	79.34	44	654+10857	10.43%	10.43%
snail	Our-TRWS	0.79	0.23	2	50+5	99.99%	99.99%
	Kovtun	0.00	0.00	0	†	0.00%	97.77%
	MQPBO-10	7.00	0.00	0	†	0.00%	77.91%
	MQPBO	1.00	0.00	0	†	0.00%	58.35%
	[29]-TRWS	46.20	6.47	5	83+332	99.98%	99.98%
strawberry-glass-2-small	Our-TRWS	1.35	0.60	2	80+5	100.00%	100.00%
	Kovtun	1.00	0.00	0	†	0.00%	54.99%
	MQPBO-10	528.00	0.00	0	†	0.00%	2.78%
	MQPBO	39.00	0.00	0	†	0.00%	0.00%

Instance	Algorithm	Time needed overall (s)	Time for initial solution (s)	#iterations Algorithm 1,2	#iterations TRWS	Logarithmic percentage partial optimality	Percentage excluded labels
	[29]-TRWS	311.54	31.00	11	259+1721	99.31%	99.31%
mrf-photomontage							
family-gm	Our-TRWS	286.40	93.08	77	1000+1265	4.75%	4.80%
	MQPBO-10	1087.00	0.00	0	†	0.00%	4.41%
	MQPBO	90.00	0.00	0	†	0.00%	4.34%
	[29]-TRWS	12726.45	1291.11	50	1015+22483	4.41%	4.41%
pano-gm	Our-TRWS	320.00	112.17	59	1000+1105	67.73%	79.17%
	MQPBO-10	646.00	0.00	0	†	0.00%	28.06%
	MQPBO	97.00	0.00	0	†	0.00%	40.37%
	[29]-TRWS	14360.45	1871.14	33	911+11193	27.55%	27.55%
mrf-stereo							
ted-gm	Our-TRWS	231.97	72.67	119	1000+715	67.27%	72.05%
	[29]-TRWS	3837.51	436.30	28	689+10383	38.13%	38.13%
tsu-gm	Our-TRWS	19.75	14.67	10	670+75	99.91%	99.94%
	[29]-TRWS	9277.99	267.55	54	377+17421	0.39%	0.39%
ven-gm	Our-TRWS	108.73	94.44	9	1000+40	0.01%	0.02%
	[29]-TRWS	14737.47	1451.83	55	993+16592	0.00%	0.00%

Table 7: Detailed experimental evaluation for Algorithm 1 utilising CPLEX [8] as a sub-solver, denoted as *Our-CPLEX*, Algorithm 2 utilising TRW-S [12] as a subsolver, denoted as *Our-TRWS*, their counterparts from [29] denoted by *[29]-CPLEX* and *[29]-TRWS* and MQPBO [11] run for one iteration with predefined label order, denoted by *MQPBO*, and run 10 iterations in 10 random label orders, denoted by *MQPBO-10*.