Efficient Message Passing for 0–1 ILPs with Binary Decision Diagrams

Jan-Hendrik Lange [1] **Paul Swoboda** [2]

Abstract

We present a message passing method for 0–1 integer linear programs. Our algorithm is based on a decomposition of the original problem into subproblems that are represented as binary decision diagrams. The resulting Lagrangean dual is solved iteratively by a series of efficient block coordinate ascent steps. Our method has linear iteration complexity in the size of the decomposition and can be effectively parallelized. The characteristics of our approach are desirable towards solving ever larger problems arising in structured prediction. We present experimental results on combinatorial problems from MAP inference for Markov Random Fields, quadratic assignment, discrete tomography and cell tracking for developmental biology and show promising performance.

1. Introduction

Structured prediction tasks in machine learning commonly require solving relaxations of NP-hard combinatorial optimization problems that are formulated as integer linear programs (ILPs). Examples include discrete graphical models (Werner, 2007), graph partitioning (Bansal et al., 2004), graph matchings (Torresani et al., 2008) and tracking problems (Luo et al., 2014). Commercial ILP solvers like Gurobi (Gurobi Optimization, LLC, 2020) or CPLEX (Cplex, IBM ILOG, 2019) rely on standard linear programming algorithms, such as the simplex and barrier method. These methods require matrix factorization and hence have super-linear complexity, which diminishes their competitiveness for very large problems that arise in structured prediction. Therefore, considerable research effort has been invested into efficient dedicated solvers for specific problem classes. Some of the most scalable methods exploit problem decompositions in order to solve Lagrangean relaxations. This family of algorithms includes subgradient, Frank-Wolfe and dual block coordinate ascent (DBCA) methods. The DBCA approach, also called *message passing* in the literature, exhibits very good performance for certain classes of combinatorial problems such as inference in Markov Random Fields, significantly outperforming general-purpose ILP solvers (Kappes et al., 2015). The drawback of DBCA solvers, however, is that they are only applicable to their dedicated problem class. In order to solve other problem classes, new specialized DBCA algorithms have to be developed. This hinders application of structured prediction in machine learning, since algorithm design is challenging and implementation time-consuming.

In this work we propose an efficient message passing method for solving relaxations of 0–1 ILPs. Our method (i) has linear iteration complexity unlike LP solvers, (ii) is not restricted to a narrow subclass of problems unlike specialized solvers and (iii) can be effectively parallelized and shows significant parallelization speedups. We demonstrate the potential of our method on a wide variety of structured prediction tasks.

Our algorithm works by decomposing any given 0–1 ILP into smaller subproblems represented by binary decision diagrams (BDDs) (Bryant, 1986). In the basic version we generate a BDD for each row of the constraint matrix. While general linear constraints may lead to BDDs of intractable sizes (e.g. for the NP-hard Knapsack problem), many inequalities commonly encountered in practice admit tractable BDD representations (Knuth, 2011; Wegener, 2000). Inequalities that do not admit a small BDD representation can be represented as multiple BDDs (Abío et al., 2012) efficiently. Hence, our BDD-representation can always be chosen bounded by the size of the original problem description. We combine the BDD subproblems via Lagrangean dual variables in order to obtain a convex relaxation for the ILP. The algorithm then updates dual variables iteratively by an operation called min-marginal averaging, which maximally improves the objective w.r.t. the current set of dual variables. Hence, our method belongs to the family of DBCA algorithms. Based on the computed dual solution, we compute a primal one via depth-first exploration of the search space. Our primal heuristic uses dual costs to guide the search towards high-quality solutions and individual subproblems inform the search so that feasible solutions are found fast and are of high quality. We show how BDDs

[1]University of Tübingen, Germany [2]Max Planck Institute for Informatics, Saarbrücken, Germany. Correspondence to: Paul Swoboda <pswoboda@mpi-inf.mpg.de>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

support an efficient implementation of our DBCA method.

We present all proofs in the appendix. The code and datasets are available on https://github.com/LPMP/BDD.

2. Related Work

Dual Block-Coordinate Ascent In machine learning DBCA algorithms for Lagrangean decompositions of combinatorial problems were successfully applied to a number of different tasks such as multiple target tracking (Arora & Globerson, 2013), graph matching (quadratic assignment problem) (Zhang et al., 2016; Swoboda et al., 2017b), multigraph matching (Swoboda et al., 2019), the multicut problem (Swoboda & Andres, 2017), cell tracking in biological image analysis (Haller et al., 2020) and MAP inference in MRFs (Kolmogorov, 2006; 2014; Globerson & Jaakkola, 2008; Werner, 2007; Savchynskyy et al., 2012; Jancsary & Matz, 2011; Meltzer et al., 2012; Wang & Koller, 2013; Johnson et al., 2007; Tourani et al., 2018; 2020).

The study (Swoboda et al., 2017a) presents general algorithmic principles on how to design efficient DBCA algorithms for arbitrary combinatorial subproblems. However, the decomposition into subproblems, the specific choice of updates and their efficient implementation are still left open for the algorithm designer to decide anew for each new problem class. The work (Werner et al., 2020) analyzes different update operations for DBCA problems and theoretically characterizes the resulting fixed points.

Optimization with Binary Decision Diagrams While binary decision diagrams (BDDs) have been used mostly to encode Boolean functions, finding optimal assignments w.r.t. a linear cost is a straightforward extension (Knuth, 2011). However, for NP-hard combinatorial problems the size of the BDD encoding increases exponentially, which makes a straightforward application of BDDs computationally intractable. In this context, two approaches have been proposed to limit BDD growth: (i) Solving a relaxation in which the set of feasible points is larger which leads to a lower bound (Andersen et al., 2007) or (ii) solving a restriction in which only a subset of feasible solutions is considered, leading to an upper bound (Bergman et al., 2016a;b). In any case, the constructed BDD is significantly smaller than the BDD encoding of the original problem.

Related to our work is the Lagrangean relaxation method by Bergman et al. (2015), who combine relaxed multi-valued decision diagrams (MDDs), an extension of BDDs, with additional constraints such as linear inequalities. The resulting Lagrangean dual problem is solved with subgradient ascent. In contrast, our method combines an arbitrary number of BDDs into a Lagrangean dual and applies the more efficient DBCA method. Hooker (2019) extends the work of Bergman et al. (2015) to provide improved bounds on job sequencing problems. Similarly, Castro et al. (2020) combine a relaxed decision diagram and linear constraints into a Lagrangean dual for a routing problem. The integration of techniques based on decision diagrams and (mixed-)integer programming is considered by (Tjandraatmadja & van Hoeve, 2020; González et al., 2020; González et al., 2020). The latter apply their hybrid approach to the (quadratic) stable set problem. Similar to our work, Bergman & Cire (2016; 2018); Lozano et al. (2018) consider a decomposition based on a collection of BDDs. Their approach is to derive a lifted linear formulation from the intersection of network flow polytopes associated with the individual BDDs solved subsequently by an ILP solver.

In contrast to prior BDD-based work, which is either applicable beyond narrow problem classes or standalone, the method we propose is both at once. It is (in principle) applicable to general 0–1 ILPs and does not rely on any external solvers.

Integer Linear Programming The integer linear programming (ILP) approach has been pioneered for the traveling salesman problem (Dantzig et al., 1954) and is successful in solving many large scale instances (Applegate et al., 2006). The input to ILP solvers consists of a description of the feasible set in terms of linear inequalities. In the first step, a linear programming (LP) relaxation obtained from relaxing the integrality constraints is solved. If the relaxed solution has fractional entries, then cutting plane methods seek to tighten the relaxation. Additionally, branch-and-bound steps are employed, which (in a nutshell) fix subsets of variables to integer values and resolve the modified LP. In this way, the search space of integer feasible solutions is traversed recursively. Branches of the search tree can be discarded if their associated lower bounds exceed the best found primal solution value.

Over the years, ILP technology has made spectacular progress. In combination with hardware improvements, current state-of-the-art solvers outperform earlier ones by at least 6 orders of magnitude. On a machine-independent basis, Bixby (2012) estimates a speedup of a factor of 29000 in the timespan 1991–2007 for CPLEX (Cplex, IBM ILOG, 2019) and of a factor of 16 in the timespan 2009–2012 for Gurobi (Gurobi Optimization, LLC, 2020). With further advancements since then, solvers are routinely capable of solving previously inaccessible instances in moderate time. Due to the considerable implementation effort, solvers written in academic projects seem not able to achieve state-of-the-art results (Mittelmann, 2017; 2020a;b).

In the context of very large scale instances, a major challenge for ILP solvers is their limited capability to utilize modern parallel CPU architectures. Solving LP relaxations, which consumes a considerable portion of the overall computation time, is particularly difficult to parallelize. Al-

though there have been efforts to exploit parallelization in the (dual) simplex method (Huangfu & Hall, 2018), the performance improvement over leading sequential implementations is relatively small (Gurobi Optimization, LLC, 2020). The barrier method can be more effectively parallelized (Gondzio & Sarkissian, 2003), but often falls short of the dual simplex method on large sparse problems. Moreover, the dual simplex method is more suitable for solving relaxations of ILPs due to its advantages in reoptimization. We show in the experimental section that our method benefits significantly from parallelization when solving large scale problems and scales favourably w.r.t. problem size.

3. Dual Decomposition of 0–1 Programs

Below we introduce a Lagrangean decomposition for binary programs and recapitulate the min-marginal averaging algorithm, the most commonly used DBCA approach.

Definition 1 (Binary program)**.** Consider m index subsets $\mathcal{I}_j \subset [n] = \{1, \dots, n\}$ and corresponding constraints $\mathcal{X}_j \subset \{0,1\}^{\mathcal{I}_j}$ for $j \in [m]$ together with a linear objective $c \in \mathbb{R}^n$. The corresponding binary program is defined as

$$\min \quad c^\top x \quad \text{s.t.} \quad x_{\mathcal{I}_j} \in \mathcal{X}_j \quad \forall j \in [m]. \tag{BP}$$

While we focus on 0–1 integer linear programs, binary programs can also encode, e.g., Max-SAT and weighted constraint satisfaction problems with finite domains.

Example 1. Consider the 0–1 integer linear program

$$\min \quad c^\top x \quad \text{s.t.} \quad Ax \leq b,\ x \in \{0,1\}^n. \tag{ILP}$$

The system of linear constraints $Ax \leq b$ may be split into m blocks, each block representing a single (or multiple) rows of the system. For instance, let $a_j^\top x \leq b_j$ denote the j-th row of $Ax \leq b$, then the problem can be written in the form (BP) by setting $\mathcal{I}_j = \{i \in [n] \mid a_{ji} \neq 0\}$ and $\mathcal{X}_j = \{x \in \{0,1\}^{\mathcal{I}_j} \mid \sum_{i \in \mathcal{I}_j} a_{ji} x_i \leq b_j\}$.

3.1. Lagrangean Dual

The problem (BP) is NP-hard and thus difficult to solve in general. However, optimization over a single constraint alone, i.e. $\min_{x \in \mathcal{X}_j} c^\top x$, may be much easier depending on the constraint (although still NP-hard in general). We exploit efficient methods for the individual subproblems in order to solve a Lagrangean dual problem of (BP) by block coordinate ascent.

Definition 2 (Lagrangean dual problem)**.** Define the set of subproblems that constrain variable x_i as $\mathcal{J}_i = \{j \in [m] \mid i \in \mathcal{I}_j\}$. Let the energy for subproblem $j \in [m]$ w.r.t. Lagrangean dual variables $\lambda^j \in \mathbb{R}^{\mathcal{I}_j}$ be $E^j(\lambda^j) = \min_{x \in \mathcal{X}_j} x^\top \lambda^j$. Then the Lagrangean dual problem is defined as

$$\max_{\lambda} \quad \sum_j E^j(\lambda^j) \quad \text{s.t.} \quad \sum_{j \in \mathcal{J}_i} \lambda_i^j = c_i \quad \forall i \in [n]. \tag{D}$$

If the optima of the individual subproblems $E^j(\lambda^j)$ agree with each other, then the consensus vector solves the original problem (BP) due to the constraints on λ. In general, it is a lower bound on (BP), since such a consensus need not hold. We provide a formal derivation of problem (D) in the appendix.

3.2. Min-Marginal Averaging

In this section we present the block coordinate ascent method to solve problem (D). The underlying algorithmic idea is to iterate over the variable indices $i \in [n]$ and update the associated dual variables such that in each subproblem the minima w.r.t. the current x_i agree with each other. This results in an algorithm which produces a monotonically non-decreasing sequence of lower bounds for (D). To this end, we define min-marginals and the min-marginal averaging update step.

Definition 3 (Min-marginal averaging)**.** For $i \in [n]$, $j \in \mathcal{J}_i$ and $\beta \in \{0,1\}$ let

$$m_{ij}^\beta = \min_{x \in \mathcal{X}_j} x^\top \lambda^j \quad \text{s.t.} \quad x_i = \beta \tag{1}$$

denote the *min-marginal* w.r.t. i, j and β. The *min-marginal averaging update* w.r.t. i is defined as

$$\lambda_i^j \leftarrow \lambda_i^j - (m_{ij}^1 - m_{ij}^0) + \frac{1}{|\mathcal{J}_i|} \sum_{k \in \mathcal{J}_i} m_{ik}^1 - m_{ik}^0 \tag{2}$$

for all $j \in \mathcal{J}_i$.

The quantity $|m_{ij}^1 - m_{ij}^0|$ indicates by how much $\min_{x \in \mathcal{X}_j} x^\top \lambda^j$ increases if x_i is fixed to 1 (if $m_{ij}^1 > m_{ij}^0$), respectively 0 (if $m_{ij}^1 < m_{ij}^0$). The min-marginal averaging update results in an equal distribution of min-marginal differences across each involved subproblem. Also, it results in a maximal improvement of the lower bound given all other dual variables are fixed.

Proposition 1. *The min-marginal averaging update w.r.t. $i \in [n]$ increases the dual bound by the non-negative value*

$$\min\left\{0, \sum_{j \in \mathcal{J}_i} m_{ij}^1 - m_{ij}^0\right\} - \sum_{j \in \mathcal{J}_i} \min\left\{0, m_{ij}^1 - m_{ij}^0\right\}.$$

Min-Marginal Averaging Algorithm Algorithm 1 computes a solution of the Lagrangean dual (D) by performing a series of min-marginal update steps (2). The order in which the variables are processed, i.e. an appropriate permutation of $[n]$, is fixed at the start of the algorithm. A suitable order

can be obtained, for instance, from bandwidth minimization of the constraint matrix, cf. Section A.3 in the appendix. In the appendix we also present an alternative averaging strategy (Section A.4) and a smoothed variant of our method (Section A.5) that is valuable for problems with bad fixed points of the non-smooth algorithm.

Algorithm 1: Min-marginal averaging

input objective vector $c \in \mathbb{R}^n$, constraint sets $\mathcal{X}_j \subset \{0,1\}^{\mathcal{I}_j}$ for $j \in [m]$
Find variable ordering $\{i_1, \dots, i_n\} = [n]$.
Initialize dual variables $\lambda_i^j = c_i/|\mathcal{J}_i|$ for all $i \in [n]$ and $j \in \mathcal{J}_i$.
while *(stopping criterion not met)* **do**
  Perform forward pass:
  for $i = i_1, \dots, i_n$ **do**
    for $j \in \mathcal{J}_i$ **do**
      Compute min-marginals for $\beta \in \{0,1\}$: $m_{ij}^\beta = \min_{x \in \mathcal{X}_j} x^\top \lambda^j \;\text{ s.t. }\; x_i = \beta$
    for $j \in \mathcal{J}_i$ **do**
      Update dual variable: $\lambda_i^j \leftarrow \lambda_i^j - (m_{ij}^1 - m_{ij}^0) + \frac{1}{|\mathcal{J}_i|} \sum_{k \in \mathcal{J}_i} m_{ik}^1 - m_{ik}^0$.
  Perform backward pass analogously (set variable order to $\{i_n, \dots, i_1\}$)

4. Primal Heuristic

In this section we present our approach to determine good primal solutions for the problem (BP) based on the dual solution that we obtained by block coordinate ascent. The basic idea is to iteratively fix primal variables and backtrack until we arrive at a feasible solution. The success of the search is determined by the order and the values of the variable fixations. In order to define a search strategy, we compute variable scores $S_i \in \mathbb{R}$ and preferred variable values $\beta_i \in \{0,1\}$ for all $i \in [n]$. Variables are fixed to preferred values β_i in descending order of their scores S_i.

We describe a straightforward choice for S_i and β_i here and discuss alternatives in the appendix. To this end, we compute for every index $i \in [n]$ the total min-marginal difference defined below.

Definition 4 (Total min-marginal difference)**.** The *total min-marginal difference* for $i \in [n]$ w.r.t. current dual variables λ is defined as $M_i = \sum_{j \in \mathcal{J}_i} m_{ij}^1 - m_{ij}^0$.

The quantity $|M_i|$ indicates by how much the dual bound increases if x_i is fixed to 1 (if $M_i > 0$), respectively 0 (if $M_i < 0$), in total across all individual subproblems, given that the dual variables remain unchanged. Thus, we prefer to fix x_i as indicated by the sign of M_i and define

$$\beta_i = 1 \quad \text{if} \quad M_i \leq 0 \quad \text{and} \quad \beta_i = 0 \quad \text{if} \quad M_i > 0. \tag{3}$$

The two variable fixation orders we employ are based on setting $S_i = |M_i|$ or $S_i = -M_i$. The second, less intuitive order introduces a bias to fix variables to 1, which, in our experiments, allows to find feasible solutions faster. This is due to feasible solutions typically having a small number of 1-entries.

Our primal heuristic is detailed in Algorithms 5 and 6 in the appendix. Algorithm 5 traverses the space of feasible solutions by depth-first search. We accelerate the search further by propagation of the restrictions to the feasible sets given the current partial assignment, see Algorithm 6.

5. Implementation with BDDs

We have described above a generic DBCA procedure for optimizing a Lagrangean relaxation of 0–1 integer linear programs and a primal search heuristic for finding solutions given dual variables. Below, we describe how reduced ordered binary decision diagrams (Bryant, 1986), a data-structure for representing Boolean functions, can provide efficient procedures for all operations that we require in the algorithms above. First we define the type of binary decision diagrams we employ.

5.1. Binary Decision Diagrams

Definition 5 (Binary decision diagram)**.** Given a set of ordered variable indices $\mathcal{I} = \{i_1, \dots, i_k\} \subseteq [n]$, a *binary decision diagram* (BDD) is a rooted, directed acyclic graph (V, A) with $A = A^1 \cup A^0$ such that:

(i) Every node $v \in V$ is labeled with an associated index $\mathrm{idx}(v) \in \mathcal{I}$ or is one of the two special nodes $\top$ or $\bot$, representing the outcomes true and false, respectively.
(ii) The root node r has index $\mathrm{idx}(r) = i_1$.
(iii) Each node $v \in V \backslash \{\top, \bot\}$ has two outgoing arcs, a 1-arc $(v, v_1^+) \in A^1$ and a 0-arc $(v, v_0^+) \in A^0$.
(iv) Every path from r to $\top$ or $\bot$ traverses a sequence of nodes $(v_1, \dots, v_\ell, \bot)$ for $\ell \leq k$ or $(v_1, \dots, v_k, \top)$ with consecutive indices $\mathrm{idx}(v_j) = i_j$ for all j.
(v) From every node $v \in V \backslash \{\top, \bot\}$ there exists a path from r to v and from v to $\top$.
(vi) The BDD is *reduced*, i.e. there are no isomorphic subgraphs.

Our definition of BDDs slightly deviates from the common one (Bryant, 1986), in that we mandate the indices to appear consecutively on each path (even if both outgoing arcs of some node point to the same node). This property helps us in devising simpler algorithms for min-marginal updates and variable restrictions. Still, the canonicity property holds, i.e. there is a one-to-one correspondence between BDDs and Boolean functions via the following path-property of BDDs.

Proposition 2 (Canonicity)**.** *There is a one-to-one corre-*

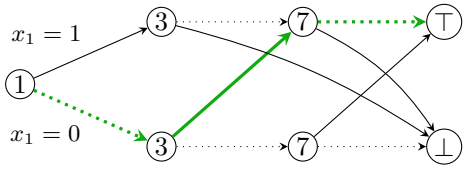

Figure 1. Binary decision diagram representing the indicator function of all binary triples (x_1, x_3, x_7) that satisfy the simplex constraint $x_1 + x_3 + x_7 = 1$. Outgoing arcs represent variable assignments to 0 (dotted) or 1 (solid). Feasible assignments are represented by paths to the truth symbol $\top$. For instance, the path marked by green arcs corresponds to $x_1 = 0, x_3 = 1, x_7 = 0$.

spondence between Boolean functions $f\colon \{0,1\}^{\mathcal{I}} \to \{0,1\}$ and BDDs such that $f(x_{i_1}, \dots, x_{i_k}) = 1$ if, and only if, the corresponding path to $\top$ in the BDD takes the x_{i_ℓ}-arc between v_ℓ and $v_{\ell+1}$ for $\ell < k$.

5.2. Subproblem Representation with BDDs

In our method we use BDDs to represent indicator functions $\mathbb{1}_{\mathcal{X}_j}(x)$. In order to obtain a viable implementation, the BDD size w.r.t. the number of variables needs to be moderate. On the one hand, there exist linear constraints whose corresponding BDD size is exponential irrespective of the variable order (Abío et al., 2012; Knuth, 2011). On the other hand, there exist polynomial upper bounds on the size of the BDD representation for many commonly encountered constraints. These include cardinality constraints, constraints with bounded coefficients and many others (Wegener, 2000; Knuth, 2011). See Figure 1 for an example where $\mathcal{X}$ is defined by a cardinality constraint which has size linear in the number of variables. For general inequalities there exist lifting techniques utilizing a coefficient splitting that result in multiple BDDs with overall polynomial size (Abío et al., 2012). Hence, the Lagrangean decomposition can be chosen such that the overall size of the BDD representation is polynomial. In most structured prediction problems and in particular in the ones we solve in our experiments the coefficients in the constraint matrix are bounded by a small constant. In this case Knuth (2011) guarantees BDDs of moderate size and we do not need the advanced techniques of Abío et al. (2012).

In order to transform linear inequalities into BDDs, we implement the efficient approach of Behle (2007; 2008), which is generalized by Serra (2020). Our implementation generates a BDD for a given linear inequality in time $\mathcal{O}(N \log W)$, where N is the number of nodes and W is the maximal width of any layer in the BDD representation.

The representation of $\mathbb{1}_{\mathcal{X}_j}$ as BDDs comes with the advantage that the min-marginals w.r.t. $\mathcal{X}_j$ can be computed by dynamic programming (shortest path search) in the associated BDD with appropriate arc costs. In fact, each forward and backward pass of Algorithm 1 requires traversing all BDDs only once to compute all min-marginals, as intermediate results from previous iterations can be reused. We detail the update steps of this procedure below.

5.3. Update Steps for Min-Marginal Computation

In order to compute min-marginals by shortest path search, we introduce the following BDD arc costs.

Definition 6 (BDD arc costs)**.** Let (V, A) be the BDD representing $\mathbb{1}_{\mathcal{X}_j}$. For every $uv \in A$ we define its arc cost

$$\theta_{uv} = \begin{cases} \lambda^j_{\mathrm{idx}(u)}, & uv \in A^1 \\ 0, & uv \in A^0 \,. \end{cases} \tag{4}$$

Algorithm 2: Forward BDD update step

Input: variable index $i \in \mathcal{I}$
for $v \in V$ *with* $\mathrm{idx}(v) = i$ **do**
  $\overrightarrow{m}_v = \min_{u:uv\in A} \{\overrightarrow{m}_u + \theta_{uv}\}$

Algorithm 3: Backward BDD update step

Input: variable index $i \in \mathcal{I}$
for $v \in V$ *with* $\mathrm{idx}(v) = i$ **do**
  $\overleftarrow{m}_v = \min\left\{\overleftarrow{m}_{v_0^+} + \theta_{vv_0^+}, \overleftarrow{m}_{v_1^+} + \theta_{vv_1^+}\right\}$

In Algorithms 2 and 3 we define dynamic programming update steps for computing min-marginals. For each node $u \in V$ we compute forward messages $\overrightarrow{m}_u$ and backward messages $\overleftarrow{m}_u$, which are the values of the shortest path between u and r, respectively u and $\top$. The forward messages are computed from first to last variable index using the already computed forward messages of preceding nodes. Similarly, the backward messages are computed from last to first variable index using the previously computed backward messages of successive nodes. Backward message values for terminal nodes are fixed to $\overleftarrow{m}_\bot = \infty$ and $\overleftarrow{m}_\top = 0$ and the forward message value of the root node r is fixed to $\overrightarrow{m}_r = 0$. Given an arc $uv \in A$, valid forward message $\overrightarrow{m}_u$ and valid backward message $\overleftarrow{m}_v$, we define the arc-marginal as

$$m_{uv} = \overrightarrow{m}_u + \theta_{uv} + \overleftarrow{m}_v \tag{5}$$

and for $i \in \mathcal{I} \cap [n]$, $\beta \in \{0,1\}$ the min-marginals as

$$m_i^\beta = \min_{uv \in A^\beta : \mathrm{idx}(u) = i} \{m_{uv}\}, \tag{6}$$

5.4. Incremental BDD Updates

With the above algorithms for BDD updates, we implement efficient incremental updates for the min-marginal averaging

operation in lines 8-9 of Algorithm 1. To this end, replace the box on lines 8-9 in the forward pass by

$$\boxed{\begin{array}{l}\text{Call Algorithm 2}(i, \text{BDD}_{\mathcal{X}_j});\\ \text{Compute min-marginals via (6)};\end{array}} \tag{7}$$

and similarly in the backward pass by

$$\boxed{\begin{array}{l}\text{Compute min-marginals via (6)};\\ \text{Call Algorithm 3}(i, \text{BDD}_{\mathcal{X}_j});\end{array}}\,. \tag{8}$$

Proposition 3. *After initialization of all messages, the incremental computation in (7) and (8) produces correct marginals when used in Algorithm 1.*

With this scheme, we can implement one forward resp. backward pass of Algorithm 1 in time proportional to visiting each BDD node exactly once. In other words, Algorithm 1 has iteration complexity linear in the size of the BDDs that encode the problem decomposition. For many problems with simple constraints (e.g. simplex constraints) the size of BDDs is proportional to the number of variables in the constraint, resulting in complexity linear in the size of the problem description.

6. Parallelization

In this section we present a concurrent extension of Algorithm 1 in order to speed up optimization of the dual problem (D).

The basic idea of our approach is to partition the variable indices into a number of intervals and perform min-marginal averaging for each interval in parallel. This requires that BDDs which straddle multiple intervals are split into several sub-BDDs, with additional Lagrange dual variables in order to enforce consistency of the representation. After each forward, resp. backward pass the dual variables between sub-BDDs are updated.

More precisely, in this section we assume a partition of $[n] = \{1, \dots, n\}$, into k intervals $\mathcal{I}^p = [n_p, n_{p+1})$ with $n_1 = 1$, $n_{k+1} = n + 1$ for $p = 1, \dots, k$. We choose the interval endpoints such that each interval corresponds to a similar number of BDD nodes, see below on how BDD nodes are distributed.

Definition 7 (Sub-BDD)**.** Given a BDD (V, A) we define its *sub-BDDs* (V_p, A_p) for $p = 1, \dots, k$ as follows. The node set V_p is given by

$$\begin{aligned} V_p = \{\top, \bot\} &\cup \{v \in V \mid \text{idx}(v) \in \mathcal{I}^p\} \\ &\cup \{v \in V \mid \exists vw \in A,\, \text{idx}(w) \in \mathcal{I}^p\}. \end{aligned} \tag{9}$$

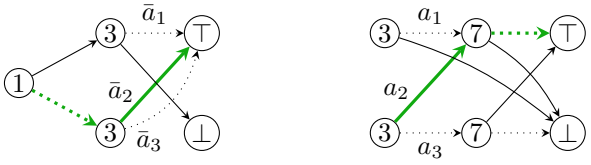

Figure 2. Given variable intervals $x_1, x_3 \in \mathcal{I}^p$, $x_7 \in \mathcal{I}^q$ with $p < q$, the BDD from Figure 1 is split into two sub-BDDs. For consistency between sub-BDDs, active paths must agree on the copy-arc pairs $(\bar{a}_1, a_1)$, $(\bar{a}_2, a_2)$ and $(\bar{a}_3, a_3)$. The two paths marked with green arcs correspond to the path marked in Figure 1.

The set of arcs A_p is given by

$$\begin{aligned} A_p = \{vw \mid vw \in A,\, \text{idx}(w) \in \mathcal{I}^p\} \\ \cup \{v\top \mid vw \in A,\, \text{idx}(v) \in \mathcal{I}^p,\, \text{idx}(w) \notin \mathcal{I}^p\} \\ \cup \{v\bot \mid v\bot \in A\}\,. \end{aligned} \tag{10}$$

The BDD splitting from Definition 7 is illustrated in Figure 2. Note that sub-BDDs do not necessarily fulfill condition (i) of Definition 5, as they may have multiple root nodes. Nonetheless, since they remain acyclic, the update steps detailed in Algorithms 7 and 8 can still be applied in order to compute min-marginals. Note that arc- and min-marginals now refer to costs w.r.t. the sub-BDD, not the original BDD from which those were derived.

Definition 8 (Copy-arc)**.** Let $j \in [m]$ be a subproblem index and (V, A) its associated BDD. Let $uv = a \in A_q$ be an arc of its q-th sub-BDD such that $\text{idx}(u) \notin \mathcal{I}^q$ and $v \neq \bot$. We call the arc $\bar{a} = u\top \in A_p$ for some $p < q$ the *copy-arc* of a. Let $\mathcal{C}_j = \{(\bar{a}, a)\}$ denote the set of all copy-arc pairs of the BDD indexed by j.

For consistency between the sub-BDDs and the original BDD, additional equality constraints between the copy-arcs of the sub-BDDs need to hold (cf. Figure 2). We relax these constraints and reparameterize the associated arc costs with additional dual variables μ.

Definition 9. For any copy-arc pair $(\bar{a}, a) \in \mathcal{C}_j$ of some BDD j we redefine their arc costs to the values

$$\theta^\mu_{\bar{a}} = \theta_{\bar{a}} + \mu_{\bar{a}} \qquad \text{and} \qquad \theta^\mu_a = \theta_a + \mu_a\,. \tag{11}$$

The additional dual variables $\mu_a, \mu_{\bar{a}}$ are constrained by

$$\mu_a + \mu_{\bar{a}} \leq 0\,. \tag{12}$$

The parallelized min-marginal averaging algorithm is summarized in Algorithm 4. The parallel algorithm works by performing min-marginal averaging in parallel on the intervals $\mathcal{I}^p$. After the forward pass, for each copy-arc pair $(\bar{a}, a) \in \mathcal{C}_j$, the dual variables $\mu_{\bar{a}}$ are updated with the arc-marginal differences from the forward and backward pass.

This step is possibly dampened by a factor $\gamma \in (0, 1]$. After the backward pass, analogous steps are performed. Note that the computation of the dual variable updates $\delta^{\rightarrow}$ and $\delta^{\leftarrow}$ in lines 9 and 16 of Algorithm 4 does not need additional calculations, since the needed arc-marginals are valid after each forward resp. backward pass.

Algorithm 4: Parallel min-marginal averaging

1 Set update step size $\gamma \in (0, 1]$
2 Initialize copy-arc updates $\delta^{\rightarrow}_{(\bar{a},a)} = \delta^{\leftarrow}_{(\bar{a},a)} = 0$
3 **while** *(stopping criterion not met)* **do**
4 **for** $p = 1, \dots, k$ *in parallel* **do**
5 Perform forward pass of Algorithm 1
6 **for** $p = 1, \dots, k$ **do**
7 **for** *all sub-BDDs j in the p-th interval* **do**
8 **for** *all copy-arc pairs* $(\bar{a}, a) \in \mathcal{C}_j$ **do**
9 $\delta^{\rightarrow}_{(\bar{a},a)} = m_{\bar{a}} - \min_{(\bar{a},a)\in\mathcal{C}_j} m_{\bar{a}}$
10 $\mu_{\bar{a}} \leftarrow \mu_{\bar{a}} - \gamma \cdot \delta^{\rightarrow}_{(\bar{a},a)} + \gamma \cdot \delta^{\leftarrow}_{(\bar{a},a)}$
11 **for** $p = 1, \dots, k$ *in parallel* **do**
12 Perform backward pass of Algorithm 1
13 **for** $p = 1, \dots, k$ **do**
14 **for** *all sub-BDDs j in the p-th interval* **do**
15 **for** *all copy-arc pairs* $(\bar{a}, a) \in \mathcal{C}_j$ **do**
16 $\delta^{\leftarrow}_{(\bar{a},a)} = m_a - \min_{(\bar{a},a)\in\mathcal{C}_j} m_a$
17 $\mu_a \leftarrow \mu_a - \gamma \cdot \delta^{\leftarrow}_{(\bar{a},a)} + \gamma \cdot \delta^{\rightarrow}_{(\bar{a},a)}$

Proposition 4. *Algorithm 4 is non-decreasing w.r.t. the dual bound.*

7. Experiments

We show competitiveness of our solver with Gurobi (Gurobi Optimization, LLC, 2020), a leading ILP solver, on a diverse set of structured prediction problems. More details on the corresponding ILP formulations and experimental results can be found in the appendix.

Datasets We selected a variety of problems from structured prediction and combinatorial optimization to demonstrate the versatility of our solver. Overall we run experiments on 3115 instances with 15k–26M variables and 10k–8M constraints. The statistics of the datasets in terms of number of variables and constraints are detailed in Table 3 in the appendix. Our benchmark problems can be categorized as follows.

Cell tracking Small and large cell tracking problems from the study (Haller et al., 2020).

Graph matching (GM) Quadratic assignment problems (often called graph matching in the literature) for correspondence in computer vision (Torresani et al., 2008) (`hotel`, `house`) and developmental biology (Kainmueller et al., 2014) (`worms`).

Markov Random Field (MRF) Several datasets from the OpenGM (Kappes et al., 2015) benchmark, containing both small and large instances with varying topologies and number of labels.

QAPLIB The widely used benchmark dataset for quadratic assignment problems used in the combinatorial optimization community (Burkard et al., 1997).

Discrete tomography The synthetic discrete tomography dataset introduced in (Kuske et al., 2017) consisting of a few thousand instances with a varying number of projections and object densities.

Algorithms Below we specify the algorithms for our empirical comparison. For the experiments we set a time limit of 10 minutes for *discrete tomography* and 1 hour for all other instances.

`Gurobi` (Gurobi Optimization, LLC, 2020) We run the dual simplex method in a single thread to first solve the root LP relaxation and afterwards perform branch-and-bound search. The dual simplex method was the overall best performing LP algorithm on the considered datasets. We further disable the presolve routine for better comparability. In fact, presolve shows little benefit for the considered instances but requires significant time to perform.

`BDD-MP` We create one BDD per linear inequality. For *cell-tracking-large* and *MRF* we use variable reordering and parallelization with $\gamma = 0.5$. We run Algorithm 1 until a minimal relative objective improvement of 10^{-6} and afterwards search for a primal solution with the primal heuristic Algorithm 5 from the appendix. We use the non-smooth optimization except for *discrete tomography*, where we use smoothing parameter $\alpha = 0.01$ (cf. Section A.5).

7.1. Results

In Table 1 we report averaged upper and lower bounds as well as running times for Gurobi and our method. Additional convergence plots are provided in the appendix.

Lower Bounds As can be seen in Table 1, for most datasets the lower bounds provided by our method are either equal or slightly weaker compared to those by Gurobi. The latter is expected, since our solver may converge to suboptimal points of the Lagrangean dual. Our method spends slightly more time in computing lower bounds for small instances than Gurobi, but is increasingly competitive for larger instances. In the case of *QAPLIB* both methods provide weak lower bounds while our method performs better than Gurobi.

Upper Bounds The results from Table 1 indicate that our rather simple primal heuristic achieves small optimality gaps for the considered problems (with the exception of *QAPLIB*).

Table 1. For each dataset the table shows average results obtained with `Gurobi` and our method `BDD-MP`. The lower bound denotes the *root LP relaxation* for `Gurobi` and the value of the Lagrangean relaxation computed by `BDD-MP`. The upper bound denotes the best primal solution value found or *n/a* if for at least one instance no feasible solution was found. We report the running times to compute the respective lower and upper bounds within the time limit. The best results for each dataset are highlighted in bold.

Dataset		Lower Bound (LB)		LB Time [s]		Upper Bound (UB)		UB Time [s]	
		Gurobi	BDD-MP	Gurobi	BDD-MP	Gurobi	BDD-MP	Gurobi	BDD-MP
Cell tracking – small		**–4.382e06**	–4.387e06	**0.42**	6.85	**–4.382e06**	–4.337e06	**0.69**	7.20
Cell tracking – large		**–1.545e08**	–1.549e08	262.0	**89.2**	**–1.524e08**	–1.515e08	1321.8	**127.1**
GM	*Hotel*	**–4.293e03**	**–4.293e03**	**1.78**	2.05	**–4.293e03**	**–4.293e03**	**1.93**	3.87
	House	**–3.778e03**	**–3.778e03**	**3.25**	4.96	**–3.778e03**	**–3.778e03**	**3.40**	6.79
	Worms	**–4.849e04**	–4.878e04	273.8	**262.8**	**–4.842e04**	–4.783e04	774.5	**264.7**
MRF	*Color-seg*	**3.085e08**	**3.085e08**	65.0	**28.4**	**3.085e08**	3.086e08	66.5	**41.9**
	Color-seg-n4	**1.976e04**	1.964e04	443.3	**49.2**	2.846e04	**2.179e04**	537.1	**57.5**
	Color-seg-n8	**1.973e04**	1.963e04	571.8	**97.1**	2.783e04	**2.238e04**	765.3	**120.7**
	Object-seg	**3.131e04**	3.125e04	752.6	**111.4**	1.498e05	**3.152e04**	753.1	**120.2**
QAPLIB	*small*	2.913e06	**3.675e06**	1780.3	**176.7**	**5.186e07**	5.239e07	2167.6	**180.7**
	large	4.512e04	**8.172e06**	3388.3	**2594.5**	**1.431e08**	1.452e08	**3353.2**	3357.6
Discrete tomography		**2.536e02**	2.394e02	321.65	**102.81**	*n/a*	*n/a*	*n/a*	*n/a*

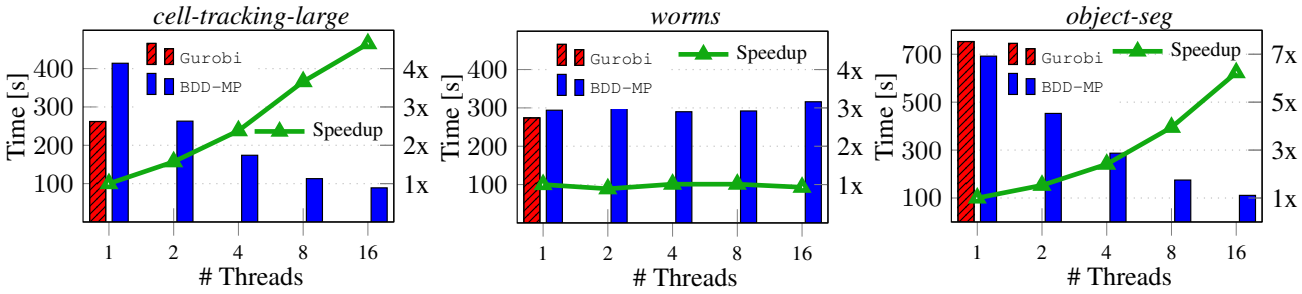

Figure 3. Running time until convergence (left axes) for Gurobi with dual simplex and our method with variable reordering and 1 to 16 threads. The right axes show the associated speedup factors of our method.

Generally, our solutions are only slightly worse than those provided by Gurobi. However, in some cases our method is able to find better primal solutions than Gurobi, which is a consequence of Gurobi's search being restricted by the given time limit.

Parallelization In Figure 3 we plot the improvement of running time until convergence for our parallelization scheme. It can be seen that for large instances of *cell tracking* as well as *MRF* the running times decrease considerably with an increasing number of threads. Naturally, due to significant overhead in the initialization, improvements due to parallelization are only exhibited for large enough problem sizes. For the denser quadratic assignment problems our current parallelization scheme seems to work less well and requires further research. In particular the number of iterations until convergence increases, which currently outweighs the fact that each iteration is performed faster. Generally, large sparse instances of the considered problems benefit from our parallelization scheme. Note that Gurobi does not make use of multiple threads when solving the relaxation with the dual simplex method. We provide further evaluation of our parallelization in the appendix.

Barrier Method In contrast to the dual simplex method, the barrier method can be parallelized more effectively. We tested the performance of Gurobi's barrier method with 16 threads when computing lower bounds for the instances in our study. For small and sparse problems we found that the dual simplex method is either faster or on par with the barrier method. For *graph matching* the barrier method ran into numerical problems and could not compute meaningful solutions. For the larger instances of *QAPLIB* the barrier method could not finish the first iteration within the time limit. Only for smaller instances of *QAPLIB* the barrier method exhibited superior performance, which we detail in Table 4 in the appendix. Note that starting branch-and-bound search with the dual simplex method given a barrier solution requires an additional crossover step.

Discussion For each of the structured prediction problems in this experimental comparison there are dedicated efficient

algorithms that exploit the problem structure, see related work. They outperform both our method and Gurobi significantly (by at least an order of magnitude), but are not generally applicable and hence unsuitable for new structured prediction problem formulations.

For small instances Gurobi outperforms our solver, while on larger ones we can observe that due to its scalability our solver achieves shorter running times. The performance of our method in relation to Gurobi is remarkable, as Gurobi and similar leading ILP solvers are highly sophisticated and complex pieces of software that have been developed for decades. In contrast, our implementation is an academic effort that leaves plenty of room for further improvements in the future. Our method shows the potential for solving very large scale sparse problems in moderate time due to its scalability and parallelization properties.

8. Conclusion

Our contribution serves as a first preliminary step towards more scalable ILP solvers, by providing an approach for solving relaxations of very large scale problems. Our preliminary results show the potential of our approach in this regard. However, more work needs to be done in order to make our method competitive, including (i) presolving, (ii) fine-tuning the initial BDD decomposition, (iii) advanced message passing techniques that lead to faster convergence, (iv) tightening of the Lagrangean relaxation in terms of additional BDDs, (v) integration with branch-and-bound search and (vi) parallelization that works uniformly well for sparse and dense problems.

Other open questions include applicability to relaxations of general 0–1 ILPs, i.e. the types of problems that can be effectively solved and their optimal BDD representation in terms of size and tightness.

References

Abío, I., Nieuwenhuis, R., Oliveras, A., Rodríguez-Carbonell, E., and Mayer-Eichberger, V. A new look at bdds for pseudo-boolean constraints. *Journal of Artificial Intelligence Research*, 45:443–480, 2012.

Andersen, H. R., Hadzic, T., Hooker, J. N., and Tiedemann, P. A constraint store based on multivalued decision diagrams. In *International Conference on Principles and Practice of Constraint Programming*, pp. 118–132. Springer, 2007.

Applegate, D. L., Bixby, R. E., Chvatal, V., and Cook, W. J. *The traveling salesman problem: a computational study*. Princeton university press, 2006.

Arora, C. and Globerson, A. Higher order matching for consistent multiple target tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 177–184, 2013.

Bansal, N., Blum, A., and Chawla, S. Correlation clustering. *Machine learning*, 56(1-3):89–113, 2004.

Behle, M. *Binary decision diagrams and integer programming*. PhD thesis, Saarland University, 2007.

Behle, M. On threshold bdds and the optimal variable ordering problem. *Journal of Combinatorial Optimization*, 16(2):107–118, 2008.

Bergman, D. and Cire, A. A. Decomposition based on decision diagrams. In Quimper, C.-G. (ed.), *Integration of AI and OR Techniques in Constraint Programming*, pp. 45–54, Cham, 2016. Springer International Publishing.

Bergman, D. and Cire, A. A. Discrete nonlinear optimization by state-space decompositions. *Management Science*, 64(10):4700–4720, 2018.

Bergman, D., Cire, A. A., and van Hoeve, W.-J. Lagrangian bounds from decision diagrams. *Constraints*, 20(3):346–361, 2015.

Bergman, D., Cire, A. A., Van Hoeve, W.-J., and Hooker, J. *Decision diagrams for optimization*, volume 1. Springer, 2016a.

Bergman, D., Cire, A. A., van Hoeve, W.-J., and Hooker, J. N. Discrete optimization with decision diagrams. *INFORMS Journal on Computing*, 28(1):47–66, 2016b.

Bixby, R. E. A brief history of linear and mixed-integer programming computation. *Documenta Mathematica*, Extra Volume: Optimization Stories:107–121, 2012.

Bryant, R. E. Graph-based algorithms for boolean function manipulation. *Computers, IEEE Transactions on*, 100(8): 677–691, 1986.

Burkard, R. E., Karisch, S. E., and Rendl, F. Qaplib–a quadratic assignment problem library. *Journal of Global optimization*, 10(4):391–403, 1997.

Castro, M. P., Cire, A. A., and Beck, J. C. An mdd-based lagrangian approach to the multicommodity pickup-and-delivery tsp. *INFORMS Journal on Computing*, 32(2): 263–278, 2020.

Cplex, IBM ILOG. Cplex optimization studio 12.10, 2019.

Cuthill, E. and McKee, J. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th National Conference*, ACM 69, pp. 157172, New York, NY, USA, 1969. Association for Computing Machinery. doi: 10.1145/800195.805928.

Dantzig, G., Fulkerson, R., and Johnson, S. Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, 2(4):393–410, 1954.

Dlask, T. and Werner, T. A class of linear programs solvable by coordinate-wise minimization. *arXiv preprint arXiv:2001.10467*, 2020.

Globerson, A. and Jaakkola, T. S. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *Advances in neural information processing systems*, pp. 553–560, 2008.

Gondzio, J. and Sarkissian, R. Parallel interior-point solver for structured linear programs. *Mathematical Programming*, 96(3):561–584, 2003.

González, J. E., Cire, A. A., Lodi, A., and Rousseau, L.-M. BDD-based optimization for the quadratic stable set problem. *Discrete Optimization*, pp. 100610, 2020.

González, J. E., Cire, A. A., Lodi, A., and Rousseau, L.-M. Integrated integer programming and decision diagram search tree with an application to the maximum independent set problem. *Constraints*, pp. 1–24, 2020.

Gurobi Optimization, LLC. Gurobi optimizer reference manual, 2020. URL http://www.gurobi.com.

Haller, S., Prakash, M., Hutschenreiter, L., Pietzsch, T., Rother, C., Jug, F., Swoboda, P., and Savchynskyy, B. A primal-dual solver for large-scale tracking-by-assignment. In *AISTATS*, 2020.

Hooker, J. N. Improved job sequencing bounds from decision diagrams. In Schiex, T. and de Givry, S. (eds.), *Principles and Practice of Constraint Programming*, pp. 268–283, Cham, 2019. Springer International Publishing. ISBN 978-3-030-30048-7.

Huangfu, Q. and Hall, J. A. J. Parallelizing the dual revised simplex method. *Math. Program. Comput.*, 10(1):119–142, 2018. doi: 10.1007/s12532-017-0130-5.

Jancsary, J. and Matz, G. Convergent decomposition solvers for tree-reweighted free energies. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 388–398, 2011.

Johnson, J. K., Malioutov, D. M., and Willsky, A. S. Lagrangian relaxation for MAP estimation in graphical models. *arXiv preprint arXiv:0710.0013*, 2007.

Kainmueller, D., Jug, F., Rother, C., and Myers, G. Active graph matching for automatic joint segmentation and annotation of C. elegans. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 81–88. Springer, 2014.

Kappes, J. H., Andres, B., Hamprecht, F. A., Schnörr, C., Nowozin, S., Batra, D., Kim, S., Kausler, B. X., Kröger, T., Lellmann, J., Komodakis, N., Savchynskyy, B., and Rother, C. A comparative study of modern inference techniques for structured discrete energy minimization problems. *International Journal of Computer Vision*, 115 (2):155–184, 2015. doi: 10.1007/s11263-015-0809-x.

Knuth, D. E. *The art of computer programming, volume 4A: combinatorial algorithms, part 1*. Pearson Education India, 2011.

Kolmogorov, V. Convergent tree-reweighted message passing for energy minimization. *IEEE transactions on pattern analysis and machine intelligence*, 28(10):1568–1583, 2006.

Kolmogorov, V. A new look at reweighted message passing. *IEEE transactions on pattern analysis and machine intelligence*, 37(5):919–930, 2014.

Kuske, J., Swoboda, P., and Petra, S. A novel convex relaxation for non-binary discrete tomography. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pp. 235–246. Springer, 2017.

Lozano, L., Bergman, D., and Smith, J. C. On the consistent path problem. *Optimization Online e-prints*, 2018.

Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W., Zhao, X., and Kim, T.-K. Multiple object tracking: A literature review. *arXiv preprint arXiv:1409.7618*, 2014.

Meltzer, T., Globerson, A., and Weiss, Y. Convergent message passing algorithms-a unifying view. *arXiv preprint arXiv:1205.2625*, 2012.

Mittelmann, H. D. Latest benchmarks of optimization software. In *INFORMS Annual Meeting. Houston, TX*, 2017.

Mittelmann, H. D. Benchmarking optimization software - a (hi)story. *SN Operations Research Forum*, 1, 2020a. doi: 10.1007/s43069-020-0002-0.

Mittelmann, H. D. Benchmarks for optimization software, 2020b. URL http://plato.asu.edu/bench.html.

Papadimitriou, C. H. The np-completeness of the bandwidth minimization problem. *Computing*, 16(3):263–270, 1976.

Savchynskyy, B., Schmidt, S., Kappes, J. H., and Schnörr, C. Efficient mrf energy minimization via adaptive diminishing smoothing. *UAI. Proceedings*, pp. 746–755, 2012. 1.

Serra, T. Enumerative branching with less repetition. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pp. 399–416, 2020.

Swoboda, P. and Andres, B. A message passing algorithm for the minimum cost multicut problem. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1617–1626, 2017.

Swoboda, P., Kuske, J., and Savchynskyy, B. A dual ascent framework for lagrangean decomposition of combinatorial problems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1596–1606, 2017a.

Swoboda, P., Rother, C., Abu Alhaija, H., Kainmuller, D., and Savchynskyy, B. A study of lagrangean decompositions and dual ascent solvers for graph matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1607–1616, 2017b.

Swoboda, P., Mokarian, A., Theobalt, C., Bernard, F., et al. A convex relaxation for multi-graph matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11156–11165, 2019.

Tjandraatmadja, C. and van Hoeve, W.-J. Incorporating bounds from decision diagrams into integer programming. *Mathematical Programming Computation*, pp. 1–32, 2020.

Torresani, L., Kolmogorov, V., and Rother, C. Feature correspondence via graph matching: Models and global optimization. In *European conference on computer vision*, pp. 596–609. Springer, 2008.

Tourani, S., Shekhovtsov, A., Rother, C., and Savchynskyy, B. Mplp++: Fast, parallel dual block-coordinate ascent for dense graphical models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 251–267, 2018.

Tourani, S., Shekhovtsov, A., Rother, C., and Savchynskyy, B. Taxonomy of dual block-coordinate ascent methods for discrete energy minimization. In *AISTATS*, 2020.

Wang, H. and Koller, D. Subproblem-tree calibration: A unified approach to max-product message passing. In *ICML (2)*, pp. 190–198, 2013.

Wegener, I. *Branching Programs and Binary Decision Diagrams: Theory and Applications*. Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 2000. ISBN 9780898719789. URL https://books.google.de/books?id=xqqJj42ZoXcC.

Werner, T. A linear programming approach to max-sum problem: A review. *IEEE transactions on pattern analysis and machine intelligence*, 29(7):1165–1179, 2007.

Werner, T., Průša, D., and Dlask, T. Relative interior rule in block-coordinate descent. In *Proceedings of the IEEE International Conference on Computer Vision*, 2020. To appear.

Zhang, Z., Shi, Q., McAuley, J., Wei, W., Zhang, Y., and Van Den Hengel, A. Pairwise matching through max-weight bipartite belief propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1202–1210, 2016.