

A Study of Lagrangean Decompositions and Dual Ascent Solvers for Graph Matching

Paul Swoboda¹, Carsten Rother², Hassan Abu Alhaija², Dagmar Kainmüller³ and Bogdan Savchynskyy²

¹IST Austria, ²TU Dresden, ³MPI-CBG

pswoboda@ist.ac.at

Abstract

We study the quadratic assignment problem, in computer vision also known as graph matching. Two leading solvers for this problem optimize the Lagrange decomposition duals with sub-gradient and dual ascent (also known as message passing) updates. We explore this direction further and propose several additional Lagrangean relaxations of the graph matching problem along with corresponding algorithms, which are all based on a common dual ascent framework. Our extensive empirical evaluation gives several theoretical insights and suggests a new state-of-the-art anytime solver for the considered problem. Our improvement over state-of-the-art is particularly visible on a new dataset with large-scale sparse problem instances containing more than 500 graph nodes each.

1. Introduction

In computer vision and beyond, the quadratic assignment problem, known also as *graph matching*, *feature correspondence* and *feature matching*, has attracted great interest. This problem is similar to Maximum-A-Posteriori (MAP) inference on a discrete pairwise graphical model, also called conditional random field (CRF) in the literature. It differs in an additional uniqueness constraint: Each label can be taken at most once. This uniqueness constraint makes it well-suited to attack e.g. tracking problems or shape matching. In both cases feature points or object parts have to be matched between multiple frames one-to-one. Unfortunately, the uniqueness constraint prevents naive application of efficient message passing solvers for MAP-inference to this problem. For this reason, many dedicated graph matching solvers were developed, see related work below.

On the other hand, efficient dual block-coordinate ascent (also known as message passing) algorithms like TRW-S [33] count among the most efficient solvers for MAP-inference in conditional random fields. Also, the graph matching problem, after possibly introducing many additional variables,

can be stated as a MAP-inference problem in a standard pairwise CRF. Such an approach already surpasses most state-of-the-art graph matching solvers.

Hence, it is desirable to devise specialized convergent message passing solvers exhibiting none of the drawbacks discussed above, i.e. (i) directly operating on a compact representation of the graph matching problem and (ii) using techniques from the MAP-inference community to gain computational efficiency.

To achieve this goal, we propose (i) several Lagrangean decompositions of the graph matching problem and (ii) novel efficient message passing solvers for these relaxations. We show their efficacy in an extensive empirical evaluation.

Related work The term *graph matching* refers to a number of different optimization problems in pattern recognition, see [18] for a review. We mean the special version known unambiguously as *quadratic assignment problem* (QAP) [36]. Recently, the graph matching was generalized to the *hyper-graph matching* problem (see [43] and references therein), which match between more than two graphs.

The quadratic assignment problem was first formulated in [13] back in 1957. Since a number of NP-complete problems such as traveling salesman, maximal clique, graph isomorphism and graph partitioning can be straightforwardly reduced to QAP, this problem is NP-hard itself. Its importance for numerous applications boosted its analysis a lot: The (already aged) overview [41] contains 362 references with over 150 works suggesting new algorithms and over 100 with new theoretical results related to this problem.

Nearly all possible solver paradigms were put to the test for QAP. These include, but are not limited to, convex relaxations based on Lagrangean decompositions [31, 52], linear [6, 22], convex quadratic [9] and semi-definite [46, 51, 63] relaxations, which can be used either directly to obtain approximate solutions or just to provide lower bounds. To tighten these bounds several cutting plane methods were proposed [11, 12]. On the other side, various primal heuristics, both (i) deterministic, such as local search [5, 44], graduated assignment methods [25], fixed point iterations [39], spectral

technique and its derivatives [17,38,53,62] and (ii) stochastic, like random walk [16] and Monte-Carlo sampling [37,49] were suggested to provide approximate solutions to the problem. Altogether these methods serve as building blocks for exact branch-and-bound [10,23,26] algorithms and other non-convex optimization methods [25,59,64]. The excellent surveys [15,41] contain further references.

As is usual for NP-hard problems, no single method can efficiently address all QAP instances. Different applications require different methods and we concentrate here on problem instances specific for computer vision. Traditionally within this community predominantly primal heuristics are used, since demand for low computational time usually dominates the need to obtain optimality guarantees. However, two recently proposed solvers [52,61] based on Lagrangean decomposition (also known as *dual decomposition* in computer vision) have shown superior results and surpassed numerous state-of-the-art primal heuristics.

The *dual decomposition solver* [52] represents the problem as a combination of MAP-inference for binary CRFs, the linear assignment problem and a number of small-sized QAPs over few variables; Lagrangean multipliers connecting these subproblems are updated with the sub-gradient method. Although the solver demonstrates superior results on computer vision datasets, we suspect that its efficiency can be further improved by switching to a different update method, such as bundle [30,32] or block-coordinate ascent [57]. This suspicion is based on comparison of such solvers for MAP-inference in CRFs [29] and similar observation related to other combinatorial optimization problems (see e.g. [45]).

Hungarian Belief Propagation (HBP) [61] considers a combination of a multilabel CRF and a linear assignment as subproblems; Lagrange multipliers are updated by a block-coordinate ascent (message passing) algorithm and the obtained lower bounds are employed inside a branch-and-bound solver. It is known [34], however, that efficiency of message passing significantly depends on the schedule of sending messages. Specifically, efficiency of dual ascent algorithms depends on selecting directions for the ascent (blocks of variables to optimize over) and the order in which these ascent operations are performed. Arguably, the underlying multilabel CRF subproblem is crucial and the message passing must deal with it efficiently. However, HBP [61] uses a message passing schedule similarly as in the MPLP algorithm [24], which was shown [29,34] to be significantly slower than the schedule of SRMP (TRW-S) [34].

Contribution We study several Lagrangean decompositions of the graph matching problem. Some of these are known, e.g. the one used in the HBP algorithm [61] and the one corresponding to the local polytope relaxation of the pairwise CRF representation of graph matching. The others have not been published so far, to our knowledge. For all these decompositions we provide efficient message passing

(dual ascent) algorithms based on a recent message passing framework [50]. In the case of the local polytope relaxation our algorithm coincides with the SRMP method [34], a higher-order generalization of the famous TRW-S algorithm [33].

Our experimental evaluation suggests a new state-of-the-art method for the graph matching problem, which outperforms both the dual decomposition [52] and the HBP [61] solvers. We propose tighter convex relaxations for all our methods. Also, we significantly improve performance of the HBP algorithm by changing its message passing schedule.

Proofs are given in the appendix. Code and datasets can be obtained at https://github.com/pawelwoboda/LP_MP.

Notation. Undirected graphs are denoted by $G = (V, E)$, where V is a finite *node set* and $E \subseteq \binom{V}{2}$ is the *edge set*. The set of neighboring nodes of $v \in V$ w.r.t. graph G is denoted by $\mathcal{N}_G(v) := \{u : uv \in E\}$. The convex hull of a set $X \subset \mathbb{R}^n$ is denoted by $\text{conv}(X)$.

2. CRFs and Graph Matching

First, we introduce conditional random fields and state the graph matching problem as one with additional uniqueness constraints. Second, we consider an inverse formulation of the graph matching problem, which, after being coupled with the original formulation, often leads to faster algorithms.

Conditional random fields (CRF). Let $G = (V, E)$ be an undirected graph. With each node $u \in V$ we associate a variable x_u taking its values in a finite *set of labels* $X_u \subseteq \{(1, 0, \dots, 0), (0, 1, 0, \dots, 0), \dots, (0, \dots, 0, 1)\}$. Hence, each label corresponds to a unit vector. Notation X_A denotes the Cartesian product $\prod_{u \in A \subseteq V} X_u$. A vector $x \in X_V$ with coordinates $(x_u)_{u \in V}$ is called a *labeling*. Likewise, we use the notation $x_A \in X_A$ (a special case being $x_{uv} \in X_{uv} \equiv X_u \times X_v$) to indicate part of a labeling. Functions $\theta_u : X_u \rightarrow \mathbb{R}$, $u \in V$, and $\theta_{uv} : X_{uv} \rightarrow \mathbb{R}$, $uv \in E$, are *potentials*, which define a local quality of labels and label pairs.

The *energy minimization* or *MAP-inference* problem for CRFs is

$$\min_{x \in X_V} \sum_{u \in V} \theta_u(x_u) + \sum_{uv \in E} \theta_{uv}(x_{uv}). \quad (1)$$

The objective in (1) is called *energy* of the CRF.

A great number of applied problems can be efficiently cast in the format (1), see e.g. [29,54]. This defines its importance for computer vision, machine learning and a number of other branches of science [54]. While problem (1) is NP-hard in general, many exact and approximate solvers were proposed [29].

Graph Matching. Although the format of Problem (1) allows us to express many practically important optimization tasks efficiently, some applications require the resulting labelings x to satisfy additional constraints. In particular, for the graph matching problem no label may be taken twice.

Let a *common universe* \mathcal{L} of labels be given such that $X_u \subseteq \mathcal{L} \forall u \in V$. We require each label $s \in \mathcal{L}$ to be taken at most once, i.e. $|\{u \in V : x_u = s\}| \leq 1$. In other words, we seek an injective mapping $(x_u)_{u \in V} : V \rightarrow \mathcal{L}$. This problem can be stated as

$$\min_{x \in X_V} \sum_{u \in V} \theta_u(x_u) + \sum_{uv \in E} \theta_{uv}(x_{uv}) \quad \text{s.t. } x_u \neq x_v \forall u \neq v. \quad (2)$$

Graph matching is NP-hard, since it is equivalent to MAP-inference for CRFs (1) in the trivial case, when nodes of the graph contain mutually non-intersecting sets of labels.

Inverse Graph Matching A special case arises if the universe \mathcal{L} of labels to be matched has the same size as the set of nodes of the graph $|\mathcal{L}| = |V|$. Then every injective mapping $(x_u)_{u \in V} : V \rightarrow \mathcal{L}$ must also be a bijection. Hence, every feasible labeling $x \in X_V$ corresponds to a permutation of V . The graph matching problem (2) can in this case also be approached in terms of the inverse permutation. To this end let the *inverse graph* $G' = (V', E')$ be given by $V' = \mathcal{L}$; the *inverse label set* $X'_s = \{v \in V : s \in X_v\}$ is associated with each node $s \in V'$; respectively $X'_\mathcal{L} = \prod_{s \in \mathcal{L}} X'_s$ is the set of *inverse labelings* and X'_{st} denotes $X'_s \times X'_t$; the set of edges of the inverse graph is defined as $E' = \{st \in V' \times V' : \exists x_{st} \in X'_{st} \text{ s.t. } x_s x_t \in E\}$. The *inverse costs* θ' for $s, s' \in V', x_s \in X'_s$ read:

$$\theta'_s(x_s) = \theta_{x_s}(s), \quad \theta'_{st}(x_{st}) = \begin{cases} \theta_{x_{st}}(s, t), & x_{st} \in E \\ 0, & \text{otherwise.} \end{cases}$$

Consider the resulting *inverse graph matching* problem

$$\min_{x \in X'_\mathcal{L}} \sum_{s \in \mathcal{L}} \theta'_s(x_s) + \sum_{st \in E'} \theta'_{st}(x_{st}) \quad \text{s.t. } x_s \neq x_t \forall s \neq t. \quad (3)$$

Labeling $x \in X_V$ and inverse labeling $y \in X'_\mathcal{L}$ correspond to each other iff $x_u = s \in V \Leftrightarrow y_s = u \in V$.

Note that when the edge set E is sparse, the inverse edge set E' may be not. In such a case, computational complexity of the inverse problem is higher than of the original one.

3. Lagrangean Decompositions

Since the graph matching problem (2) is NP-hard, it is common to consider convex relaxations. Below, we present three Lagrangean decomposition based relaxations of the problem. These can be applied to the original graph matching problem (2), to the inverse one (3) and to a combination of both. Since all these relaxations are based on the famous local polytope relaxation [47, 55] of the MAP-inference for CRFs (1), we give a short overview of this relaxation first.

Local Polytope for CRFs. The MAP-inference problem (1) can be represented as an integer linear program (ILP) [35] using an *overcomplete representation* [54] by grouping potentials corresponding to each node and edge into separate vectors. That is, $\theta_w(x_w)$, $w \in V \cup E$ stands for a vector with coordinates $(\theta_w(x_w))_{x_w \in X_w}$. The real-valued vectors μ_w have the same dimensionality as θ_w and stand for the "relaxed" version of x_w . The corresponding linear programming (LP) relaxation reads:

$$\begin{aligned} \min \quad & \sum_{w \in V \cup E} \langle \theta_w, \mu_w \rangle \\ \text{s.t.} \quad & \sum_{x_w \in X_w} \mu_w(x_w) = 1, \mu_w(s) \geq 0, w \in V \cup E, s \in X_w, \\ & \sum_{x_v \in X_v} \mu_{uv}(x_{uv}) = \mu_u(x_u), uv \in E, u \in uv, x_u \in X_u. \end{aligned} \quad (4)$$

Constraints of (4) define the *local polytope* L_G . Note that adding integrality constraints $\mu_w \in \{0, 1\}^{|X_w|}$ makes the problem (4) equivalent to its combinatorial formulation (1).

Integer Relaxed Pairwise Separable Linear Programs (IRPS-LP) Below we describe a general problem format studied in [50], which generalizes the local polytope relaxation (4). Importantly, the same format fits also the Lagrangean decompositions of the graph matching problem, which we consider below. This makes it possible to consider all these relaxations at once from a general viewpoint.

Let a *factor graph* $\mathcal{G} = (\mathbb{F}, \mathbb{E})$ consist of nodes $\mathbb{F} = \{1, \dots, k\}$, called *factors* and edges \mathbb{E} , called *factor-edges*. Let $X_i \subset \{0, 1\}^{\dim(X_i)}$, $i \in \mathbb{F}$, be sets of binary vectors and (ii) $A_{(i,j)} \subset \{0, 1\}^{\dim(X_i) \times K_{ij}}$, $ij \in \mathbb{E}$, $K_{ij} \in \mathbb{N}$ be matrices with binary entries, which map binary vectors from X_i into binary vectors from $\{0, 1\}^{K_{ij}}$, i.e., $A_{(i,j)} : X_i \rightarrow \{0, 1\}^{K_{ij}}$. The IRPS-LP is a class of problems, which factorize according to \mathcal{G} .

$$\min_{\mu \in \Lambda_{\mathcal{G}}} \sum_{i \in \mathbb{F}} \langle \theta_i, \mu_i \rangle \quad (5)$$

$$\Lambda_{\mathcal{G}} := \left\{ (\mu_1 \quad \dots \quad \mu_k) : \begin{array}{ll} \mu_i \in \text{conv}(X_i) & i \in \mathbb{F} \\ A_{(i,j)} \mu_i = A_{(j,i)} \mu_j & \forall ij \in \mathbb{E} \end{array} \right\}.$$

Constraints $A_{(i,j)} \mu_i = A_{(j,i)} \mu_j$ are associated with each factor-edge and are called *coupling constraints*. When representing the local polytope relaxation (4) as (5) we assume $\mathbb{F} = V \cup E$ and $\mathbb{E} = \{\{u, uv\}, \{v, uv\} : uv \in E\}$. The convex hull of X_w is fully defined by the first line of constraints in (4), since X_w constitutes a set of unit binary vectors. The second line of constraints in (4) defines the coupling constraints.

We use variable names μ for (in general) non-binary vectors $\mu_i \in \text{conv}(X_i)$ and x for binary ones $x_i \in X_i$, $i \in \mathbb{F}$.

3.1. Graph Matching Problem Relaxations.

Below, we describe three relaxations of the graph matching (2) problem, which fit the IRPS-LP (5) format. The first

one results in a standard local polytope relaxation (4) of a specially constructed CRF, the second one utilizes additional coupling constraints on top of (4), while the third approach uses a network flow subproblem. Additionally, we use the inverse formulation (3) and build two additional IRPS-LPs.

(R1) Graph Matching as CRF. To build a CRF equivalent to the graph matching we start with the underlying CRF as in (2) and express the uniqueness constraints in the edge factors. To this end we (i) extend the edge set E with new edges connecting any two nodes having at least one common label, i.e. $\hat{E} := E \cup \{uv \in \binom{V}{2} : X_u \cap X_v \neq \emptyset\}$; (ii) assign edge potentials $\theta_{uv} \equiv 0$ to all new edges $\hat{E} \setminus E$; (iii) for all $uv \in \hat{E}$ we assign $\theta_{uv}(x, x) := \infty \forall x \in X_u \cap X_v$. Any solution of the resulting CRF (1) with cost $< \infty$ is an assignment. The relaxation in terms of an IRPS-LP is the local polytope (4).

This approach results in general in a quadratic number of additional edge potentials, which may become intractable as the size of the graph matching problem grows.

(R2) Relaxation with Label Factors. For each label $s \in \mathcal{L}$ we introduce an additional *label factor*, which keeps track of nodes which assign label s . The label set of this factor $X_s := \{u \in V : s \in X_u\} \cup \{\#\}$ consists of those nodes $u \in V$ which can be assigned label s and an additional dummy node $\#$ representing non-assignment of label s . Label $\#$ is necessary, as not every label needs to be taken. The set of factors becomes $\mathbb{F} = V \cup E \cup \mathcal{L}$, with the coupling constraint set $\mathbb{E} = \{\{u, uv\}, \{v, uv\} : uv \in E\} \cup \{\{u, l\} : u \in V, l \in X_u\}$. The resulting IRPS-LP formulation reads

$$\begin{aligned} \min \quad & \sum_{w \in V \cup E} \langle \theta_w, \mu_w \rangle + \sum_{s \in \mathcal{L}} \langle \tilde{\theta}_s, \tilde{\mu}_s \rangle \quad (\text{R2}) \\ \mu & \in \mathbf{L}_G \\ \tilde{\mu}_s & \in \text{conv}(X_s), \quad s \in \mathcal{L} \\ \mu_u(s) & = \tilde{\mu}_s(u), \quad s \in X_u. \end{aligned}$$

Here we introduced additional potentials $\tilde{\theta}_s$ for the label factor. Initially, we set $\tilde{\theta}_s \equiv 0$.

(R3) Relaxation with a Network Flow Factor. If one ignores the edge potentials θ_{uv} in (2), the problem can be equivalently reformulated as bipartite matching [7]:

$$\begin{aligned} \min_{\mu \in \mathcal{M}} \quad & \sum_{u \in V} \langle \theta_u, \mu_u \rangle, \quad \text{where} \quad (6) \\ \mathcal{M} = \quad & \left\{ (\mu_u)_{u \in V} \geq 0 : \begin{array}{l} \sum_{s \in X_u} \mu_u(s) = 1, u \in V \\ \sum_{u \in V, s \in X_u} \mu_u(s) \leq 1, s \in \mathcal{L} \end{array} \right\} \end{aligned}$$

Here we substituted the uniqueness constraints with the linear inequalities $\sum_{u \in V, s \in X_u} \mu_u(s) \leq 1$, which is equivalent for $\mu_u \in \{0, 1\}^{|X_u|}$. It is known that \mathcal{M} is the convex hull of all binary vectors satisfying the conditions of \mathcal{M} [7], i.e. $\text{conv}(\mathcal{M} \cap \{0, 1\}^{\dim(\mathcal{M})}) = \mathcal{M}$. Therefore \mathcal{M} fits into the

IRPS-LP framework. Crucially for an efficient implementation, (6) can be efficiently solved by minimum cost flow solvers [7].

Below we treat (6) as a separate factor \mathcal{M} and link it with (4) to obtain an IRPS-LP. Its factor graph is defined by $\mathbb{F} = V \cup E \cup \{\mathcal{M}\}$ and $\mathbb{E} = \{\{u, uv\}, \{v, uv\} : uv \in E\} \cup \{\{u, \mathcal{M}\} : u \in V\}$. The resulting IRPS-LP formulation is

$$\begin{aligned} \min \quad & \sum_{w \in V \cup E} \langle \theta_w, \mu_w \rangle + \sum_{u \in V} \langle \tilde{\theta}_u, \tilde{\mu}_u \rangle \quad (\text{R3}) \\ \mu & \in \mathbf{L}_G, \quad \tilde{\mu} \in \mathcal{M}, \\ \tilde{\mu}_u(s) & = \mu_u(s), \quad u \in V, s \in X_u. \end{aligned}$$

Initially, we set $\tilde{\theta} \equiv 0$.

Representation (6) for the uniqueness constraints has been already used e.g., in [20]. However their optimization technique lacks both convergence guarantees and monotonicity of a lower bound, which our methods possess. The work [61] considered the Lagrange dual of (R3) as a relaxation the graph matching problem. Their relaxation is equivalent to (R3), but their algorithm differs from ours. We refer to Section 5 for a discussion of the differences and to Section 6 for an experimental comparison.

(R4-R5) Coupling Original Graph Matching (2) and its Inverse (3). In the special case when $|\mathcal{L}| = |V|$ we may solve the inverse graph matching problem (3) instead of the original one (2). Another alternative is to solve both problems simultaneously and couple them together by requiring that the labeling of (2) is the inverse permutation for the labeling from (3). Such an approach doubles the problem size, yet it may result in a smaller number of iterations required to obtain convergence. This approach works both for relaxations (R2) and (R3).

The resulting coupled IRPS-LP for (R2) reads

$$\begin{aligned} \min_{\mu, \mu'} \quad & \sum_{w \in V \cup E} \langle \theta_w, \mu_w \rangle + \sum_{w \in V' \cup E'} \langle \theta'_w, \mu'_w \rangle \quad (\text{R4}) \\ \mu & \in \mathbf{L}_G, \quad \mu' \in \mathbf{L}_{G'} \\ \forall u \in V, u' \in X_u : & \mu_u(u') = \mu'_{u'}(u). \end{aligned}$$

Here the role of label factors in (R2) has been taken over by the node factors of the inverse graph matching (3). We distribute the costs equally among θ and θ' initially.

Another coupled IRPS-LP, corresponding to (R3) reads

$$\begin{aligned} \min_{\mu, \mu', \tilde{\mu}} \quad & \sum_{w \in V \cup E} \langle \theta_w, \mu_w \rangle + \sum_{w \in V' \cup E'} \langle \theta'_w, \mu'_w \rangle + \sum_{u \in V} \langle \tilde{\theta}_u, \tilde{\mu}_u \rangle \quad (\text{R5}) \\ \mu & \in \mathbf{L}_G, \quad \mu' \in \mathbf{L}_{G'}, \quad \tilde{\mu} \in \mathcal{M} \\ \forall u \in V, u' \in X_u : & \mu_u(u') = \tilde{\mu}_u(u'), \quad \mu'_{u'}(u) = \tilde{\mu}_u(u') \end{aligned}$$

Here the network flow factor \mathcal{M} controls consistency of the original μ and inverse labelings μ' . Initially, we set $\tilde{\theta} \equiv 0$ and distribute costs in θ and θ' equally.

The optimal values obtained by relaxations (R1) – (R5) may deliver differing bounds to (2), as characterized below.

Proposition 1. (R2) = (R3) and (R4) = (R5). Relaxation (R1) is weaker than (R2) and (R3).

4. General Algorithm

In this section we define a general algorithm for IRPS-LP problems (5), which is applicable to the decompositions (R1)–(R5) of the graph matching problem considered in Section 3.1. Our algorithm is a simplified version of the algorithm [50], where we fixed several parameters to the values common to the relaxations (R1)–(R5).

Instead of optimizing IRPS-LP (5) directly, we consider its Lagrangean dual w.r.t. the coupling constraints $A_{(i,j)}\mu_i = A_{(j,i)}\mu_j$. The IRPS-LP problem (5) can be shortly written as $\min_{\mu} \{\langle \theta, \mu \rangle, \text{ s.t. } A\mu = 0, \mu \in P\}$, where μ stands for $(\mu_i)_{i=1}^k$, $A\mu = 0$ represents all coupling constraints $A_{(i,j)}\mu_i - A_{(j,i)}\mu_j = 0$ and P denotes a polytope encapsulating the rest of constraints. By dualizing $A\mu = 0$ with a vector of Lagrange multipliers Δ one obtains the Lagrange function $\langle \theta, \mu \rangle - \langle \Delta, A\mu \rangle = \langle \theta - A^\top \Delta, \mu \rangle$. After introducing $\theta^\Delta := \theta - A^\top \Delta$ the dual objective reads $D(\Delta) = \min_{\mu} \{\langle \theta^\Delta, \mu \rangle, \text{ s.t. } \mu \in P\}$. It is well-known [14] that $D(\Delta) \leq \langle \theta, \mu \rangle$ for any feasible μ and the dual problem consists in maximizing $D(\Delta)$ over Δ . Going from θ to θ^Δ is called an *equivalent transformation* or *reparametrization* in the literature or *message passing* in the CRF-community.

Now we apply the above considerations to the general IRPS-LP problem (5). Specifically, let $i, j \in \mathbb{F}$ be two neighboring factors in the factor-graph \mathbb{G} . Then for any μ_i and μ_j satisfying the coupling constraint for edge $ij \in \mathbb{E}$

$$\begin{aligned} & \langle \theta_i, \mu_i \rangle + \langle \theta_j, \mu_j \rangle \\ &= \langle \theta_i, \mu_i \rangle + \langle \theta_j, \mu_j \rangle + \underbrace{\langle \Delta_{(i,j)}, A_{(i,j)}\mu_i - A_{(j,i)}\mu_j \rangle}_{=0} \\ &= \langle \theta_i + A_{(i,j)}^\top \Delta_{(i,j)}, \mu_i \rangle + \langle \theta_j - A_{(j,i)}^\top \Delta_{(i,j)}, \mu_j \rangle. \end{aligned}$$

The values and sign of the Lagrange multipliers $\Delta_{(i,j)}$ define how much cost is "sent" from j to i or the other way around. When we consider a subset $J \subseteq \mathcal{N}_{\mathbb{G}}(i)$ of the neighboring factors for i , the resulting equivalent transformation reads:

$$\theta_i \rightarrow \theta_i + \sum_{j \in J} A_{(i,j)}^\top \Delta_{(i,j)} \text{ and } \theta_j \rightarrow \theta_j - A_{(j,i)}^\top \Delta_{(i,j)}. \quad (7)$$

Below we define a subclass of messages $\Delta_{(i,j)}$ in the setting (7) which improve the dual.

Definition 1. Messages $\Delta_{(i,j)}$, $j \in J$, are called *admissible*, if there exists $x_i^* \in \operatorname{argmin}_{x_i \in X_i} \langle \theta_i, \mu_i \rangle \cap \operatorname{argmin}_{x_i \in X_i} \langle \theta_i^\Delta, \mu_i \rangle$ and additionally

$$\Delta_{(i,j)}(s) \begin{cases} \geq 0, & \nu(s) = 1 \\ \leq 0, & \nu(s) = 0 \end{cases}, \text{ where } \nu := A_{(i,j)} x_i^*. \quad (8)$$

We denote the set of admissible vectors by $AD(\theta_i, x_i^*, J)$.

Lemma 1 ([50]). *Admissible messages do not decrease the dual value, i.e., $\Delta \in AD(\theta_i^\phi, x_i^*, J)$ implies $D(0) \leq D(\Delta)$.*

Example 1. Let us apply Definition 1 to the local polytope relaxation (4) of CRFs. Let ij correspond to $\{u, uv\}$, where $u \in \mathbb{V}$ is some node and $uv \in \mathbb{E}$ is any of its incident edges and $J = \{j\}$. Then x_i^* corresponds to a locally optimal label $x_u^* \in \operatorname{argmin}_{s \in X_u} \theta_u(s)$ and $\nu(s) = \llbracket s = x_u^* \rrbracket$. Therefore we may assign $\Delta_{u,uv}(s)$ to any value from $[0, \theta_u(x_u^*) - \theta_u(s)]$. This assures that (8) is fulfilled and x_u^* remains a locally optimal label after reparametrization even if there are multiple optima in X_u .

Sending Messages. Procedure 1 represents an elementary step of our optimization algorithm. It consists of sending messages from a node i to a subset of its neighbors J .

Procedure 1: Send messages from $i \in \mathbb{F}$ to $J \subseteq \mathcal{N}_{\mathbb{G}}(i)$

- 1 **Optimize factor:** $x_i^* \in \operatorname{argmin}_{x_i \in X_i} \langle \theta_i, \mu_i \rangle$
- 2 **Choose** $\delta \in \mathbb{R}^{d_i}$ s.t. $\delta(s) \begin{cases} \geq 0, & x_i^*(s) = 1 \\ \leq 0, & x_i^*(s) = 0 \end{cases}$
- 3 **Maximize admissible messages to J :**

$$\Delta_{(i,J)} := (\Delta_{(i,j)})_{j \in J} \in \operatorname{argmax}_{\hat{\Delta} \in D(\theta_i^\phi, x_i^*, J)} \langle \delta, \theta_i^{\hat{\Delta}} \rangle \quad (9)$$

- 4 **Update** θ^i and θ_j , $j \in J$, according to (7)
-

Procedure 1 first computes an optimal labeling for the factor i in line 1, then computes message updates in (9) and finally updates the costs θ in line 4. The costs δ in line 2 are chosen as ± 1 , except when $i = \mathcal{M}$ is the network flow factor for (R3) and (R5). In this case, we choose $\delta(u, x_u) = \begin{cases} 0, & x_u = x_u^* \\ 1 - |X_u|, & x_u \neq x_u^* \end{cases}$.

Computation (9) provides a maximally possible admissible message from i to $\{J\}$. Essentially, it makes the cost vector of the factor i as uniform as possible. So, in the setting of Example 1 $\Delta_{u,uv}(s)$ becomes equal to $\theta_u(x_u^*) - \theta_u(s)$ and therefore $\theta_i^\Delta(s) = \theta_u(x_u^*)$ for all $s \in X_u$. Since the result of (9) is an admissible message, Procedure 1 never decreases the dual objective, as follows from Lemma 1.

Dual Ascent Algorithm. Let the notation $\{j_1, \dots, j_n\}_<$ stand for an ordered set such that $j_k < j_{k+1}$, $k = 1, \dots, n$. Algorithm 2 below goes over some of the factors $i \in \mathbb{F}$ in a pre-specified order and calls Procedure 1 to send or receive messages to/from some of the neighbors.

Algorithm 2 works as follows: We choose an ordered subset of factors $\{i_1, \dots, i_k\}_<$. For each factor $i \in \mathbb{F}$ we

Algorithm 2: Dual Ascent for IRPS-LP

```
1 Input:  $I = \{i_1, \dots, i_k\}_< \subseteq \mathbb{F}$ ,  $(J_r(i) \subseteq \mathcal{N}_{\mathbb{G}}(i))_{i \in I}$ ,  
    $(J_s(i) \subseteq \mathcal{N}_{\mathbb{G}}(i))_{i \in I}$   
2 for  $iter = 1, \dots, \mathbf{do}$   
3   for  $i = i_1, \dots, i_k$  do  
4     Receive messages:  
5     for  $j \in J_r(i)$  do  
6       | Call Algorithm 1 with input  $(j, \{i\})$ .  
7     end  
8     Send messages:  
9     Call Algorithm 1 with input  $(i, J_s(i))$ .  
10  end  
11  Reverse the order of  $i_1, \dots, i_k$  and exchange  $J_r \leftrightarrow J_s$   
12 end
```

select a neighborhood $J_r(i) \subseteq \mathcal{N}_{\mathbb{G}}(i)$ of factors from which to receive messages and a neighborhood $J_s(i)$ to which messages are sent by Procedure 1. We run Algorithm 2 on $\{i_1, \dots, i_k\}_<$ (forward direction) and $\{i_k, \dots, i_1\}_<$ (backward direction) alternately until some stopping condition is met. Since Algorithm 2 reparametrizes the problem by Procedure 1 only and the latter is guaranteed to not decrease the dual, so is Algorithm 2. We refer to [50] for further theoretical properties of Algorithm 2.

5. Graph matching algorithms.

For each of the relaxations (R1)-(R5) of the graph matching problem we detail parameters of Algorithm 2 used in our experiments: we define the sets I , $J_r(i)$, $J_s(i)$.

Algorithm Names. We use the following shortcuts for specializations of Algorithm 2 to the relaxations (R1)-(R5): **GM** corresponds to (R1), **AMP** to (R2), **AMCF** to (R3). To obtain the relaxations (R1-R3) we use either the original graph, as in (2), or an inverse one, as in (3). These options are denoted by suffixes **-O** and **-I** respectively. Additionally, the two coupled relaxations (R4) and (R5), are addressed by algorithms **AMP-C** and **AMCF-C** respectively. All in all, we have eight algorithms **GM-O**, **GM-I**, **AMP-O**, **AMP-I**, **AMP-C**, **AMCF-O**, **AMCF-I** and **AMCF-C**.

The sets I , $J_r(i)$ and $J_s(i)$ are defined in Table 5. For algorithms with the suffix **-I** the values are the same as for those with **-O**, but corresponding to the inverse graph.

We assume the order of graph nodes $\mathbb{V} := \{u_1, \dots, u_n\}_<$ and labels $\mathcal{L} := \{s_1, \dots, s_{|\mathcal{L}|}\}_<$ to be given a priori. We define $u_n < \mathcal{M} < s_1$ for the matching factor \mathcal{M} and $u < uv < v$ for the edge factors $uv \in E$. Similarly, we define $s < ss' < s'$ for all edge factors $ss' \in E'$ in the inverse graph. We extend the resulting partial order to a total one, e.g., by topological sort. For $i \in \mathbb{F}$ we define $\mathcal{N}_{\mathbb{G}}(i)_< :=$

Algorithm	Ordered set I	$J_r(i)$	$J_s(i)$
GM-O	$\{u_1, \dots, u_n\}_<$	$\mathcal{N}_{\mathbb{G}}(i)_<$	$\mathcal{N}_{\mathbb{G}}(i)_>$
AMP-O	$\{u_1, \dots, u_n, s_1, \dots, s_{ \mathcal{L} }\}_<$	$\mathcal{N}_{\mathbb{G}}(i)_<$	$\mathcal{N}_{\mathbb{G}}(i)_>$
AMCF-O	$\{u_1, \dots, u_n, \mathcal{M}\}_<$	$\mathcal{N}_{\mathbb{G}}(i)_< \cap E$	$\mathcal{N}_{\mathbb{G}}(i)_>$
AMP-C	$\{u_1, \dots, u_n, s_1, \dots, s_{ \mathcal{L} }\}_<$	$\mathcal{N}_{\mathbb{G}}(i)_<$	$\mathcal{N}_{\mathbb{G}}(i)_>$
AMCF-C	$\{u_1, \dots, u_n, \mathcal{M}, l_1, \dots, l_{ \mathcal{L} }\}_<$	$\mathcal{N}_{\mathbb{G}}(i)_< \cap E$	$\mathcal{N}_{\mathbb{G}}(i)_>$

Table 1. Input sets for specializations of Algorithm 2. For algorithms with the suffix **-I** the sets are the same as for those with **-O**, but correspond to the inverse graph.

$\{j \in \mathcal{N}_{\mathbb{G}}(i) : j < i\}$ and $\mathcal{N}_{\mathbb{G}}(i)_> := \mathcal{N}_{\mathbb{G}}(i) \setminus J_r(i)$ as the sets of preceding and subsequent factors.

Sending a message by some factor automatically implies receiving this message by another, coupled factor. Therefore, there is no need to go over all factors in Algorithm 2. In particular, edge-factors are coupled to node-factors only, therefore processing all node factors in Algorithm 2 automatically means updating all edge-factors as well. In the processing order and selection of the sets $J_r(i)$ and $J_s(i)$ we follow the most efficient MAP-solvers TRWS [33] and SRMP [34] (the latter is a generalization of TRWS to higher order models and has a slightly different implementation for pairwise CRFs (1)). In the special case when all nodes contain disjoint subsets of labels the graph matching problem (2) turns into MAP-inference in CRFs (1). Then all our algorithms **GM**, **AMP** and **AMCF** reduce to SRMP [34].

It is worth mentioning that for CRFs there exist algorithms, such as MPLP [24], which go over edge-factors only and in this way implicitly process also node-factors. As empirically shown in SRMP [34], MPLP is usually slower than SRMP. In Section 6 we show that our methods also favorably compare to the recently proposed HBP [61], which is similar to **AMCF-O**, but uses an MPLP-like processing schedule.

Optimization Subproblems of Procedure 1. For each call of Procedure 1 one must find the best factor element in line 1 and compute the best messages by solving (9). The first subproblem is solved by explicitly scanning all elements of the factor for node-, edge- and label-factors. For optimizing over \mathcal{M} , we use a min-cost-flow solver. Solving (9) for all choices of factors and neighborhoods is possible through closed-form solutions or calling a minimum cost flow solver and is described in the appendix.

Primal Rounding. Algorithm 2 only provides lower bounds to the original problem (2). To obtain a primal solution one may ignore the edge potentials θ_{uv} and solve the resulting reparametrized bipartite matching problem (6) with a minimum cost flow solver, as done in [61]. Empirically we found that it is better to interleave rounding and message passing, similarly as in TRWS [33] and SRMP [34]. Assume we have already computed a primal integer solution x_v^* for all $v < u$ and we want to compute x_u^* . To this end, between

lines 4 and 5 of Algorithm 2 for $i = u$ we assign

$$x_u^* \in \operatorname{argmin}_{x_u: x_u \neq x_v^* \forall v < u} \theta_u(x_u) + \sum_{v < u: uv \in E} \theta_{uv}(x_u, x_v^*). \quad (10)$$

Time complexity If $X_u = \mathcal{L} \forall u \in V$, time complexity per iteration is $O(|\mathcal{L}||V| + |\mathcal{L}|^2|E|)$ for **GM**. For **AMP** we must add $|\mathcal{L}|^3$ and for **AMCF** the time to solve (6) (possible in $O(\mathcal{L}^3)$). Details and speedups are in the appendix.

Higher Order Extensions. Our approach is straightforwardly extendable to the graph matching problem with higher order factors, a special case being third order: Let $T \subseteq \binom{V}{3}$ be a subset of triplets of nodes and $\theta_{uvw} : X_{uvw} \rightarrow \mathbb{R}$ be corresponding triplet potentials. The corresponding third order graph matching problem reads

$$\begin{aligned} \min_{x \in \mathcal{X}_V} \sum_{u \in V} \theta_u(x_u) + \sum_{uv \in E} \theta_{uv}(x_{uv}) + \sum_{uvw \in T} \theta_{uvw}(x_{uvw}) \\ \text{s.t. } x_u \neq x_v \forall u \neq v. \end{aligned} \quad (11)$$

The associated IRPS-LP can be constructed by including additional factors for all triplets in an analogous fashion as in (4), see e.g. [56] for the corresponding relaxation.

For relaxations (R1) – (R5) we use third order factors to enforce cycle inequalities, which we add in a cutting plane approach as in [48]. For this we set $\theta_{uvw} \equiv 0$ at the beginning. By this construction (11) is equivalent to (2), however the corresponding IRPS-LP are not: Triplet potentials make the relaxation tighter.

6. Experiments

Algorithms. We compare against the two Lagrangean decomposition based solvers [52, 61] described in Section 1.

- The dual decomposition solver **DD** [52]. We use local subproblems containing 4 nodes. Note that the comparison in [61] was made with subproblems of size 3, hence **DD**'s relaxation was weaker there.
- “Hungarian belief propagation” **HBP** [61]. In [61] a branch and bound solver is used on top of the dual ascent solver. For a fair comparison our reimplementation uses only the dual ascent component. As for **AMP** and **AMCF**, we append to **HBP** the suffixes **-O** and **-C** to denote the relaxations we let **HBP** run on.

According to [52, 61], these two algorithms outperformed competitors [16, 19, 20, 25, 27, 38, 39, 42, 46, 51, 58, 64] at the time of their publication, hence we do not compare against the latter ones.

We set a hard threshold of 1000 iterations for each algorithm, exiting earlier when the primal/dual gap vanishes or no dual progress was observed anymore. We compute primal solutions every 5-th iteration in our algorithms. For **GM**, **AMP**, **AMCF** and **HBP** we use the tightening extension discussed in Section 5 to improve the dual lower bound. We tighten our relaxation whenever no dual progress occurs.

dataset	#I	#V	#L	C
house	105	30	30	dense
hotel	105	30	30	dense
car	30	19-49	19-49	dense
motor	20	15-52	15-52	dense
graph flow	6	60-126	2-126	sparse
worms	30	≤ 600	20-60	sparse

Table 2. Dataset description. #I denotes number of instances, #V the number of nodes $|V|$, #L the number $|X_u|$ of labels a node $u \in V$ can be matched to and C the connectivity of the graph.

Datasets. We have compared on six datasets:

- **house** [3] and **hotel** [2] with costs as in [52]. The task is to find matching feature points between images capturing an object from different viewpoints.
- **car** and **motor**, both used in [40], contain pairs of cars and motorbikes with keypoints to be matched. The images are taken from the VOC PASCAL 2007 challenge [21]. Costs are computed from features as in [40].
- The **graph flow** dataset [1] comes from a tracking problem with large displacements [8]. Keypoints in frames of RGB-D images obtained by a Kinect camera [60] are matched. The depth information provided by the Kinect camera is taken into account when computing the potentials θ .
- The **worms** dataset [4] from bioimaging [28]. The goal is to annotate nuclei of *C. elegans*, a famous model organism used in developmental biology, by finding the corresponding nuclei in a precomputed model. The instances of **worms** are, to our knowledge, the largest graph matching datasets ever investigated in the literature.

A summary of dataset characteristics can be found in Table 2. Previous computational studies concentrated on small-scale problems having up to 60 nodes and labels. We have included the **worms** dataset with up to 500 nodes and $|\mathcal{L}| = 1500$ labels.

Results. Fig. 2 shows performance of the algorithms on all 6 considered datasets. Among all variants **-O**, **-I**, **-C** corresponding respectively to the original, inverse and coupled formulations we plotted only the best one. As expected, for dense graphs (datasets **house**, **hotel**, **car**, **motor**) the variant **-C** with coupling provided most robust convergence, being similar to the best of **-O** either **-I** and therefore is presented on the plots. For sparse graphs, the inverse representation becomes too expensive, as the inverse edge set E' may be dense even though E is sparse in (3). Therefore we stick to the original problem **-O**.

- **hotel** and **house** are easy datasets, and many instances needed < 5 iterations for convergence. **AMP**, **AMCF** and **DD** were able to solve all instances to optimality within few seconds or even faster. However, **DD** is the fastest method for this data.

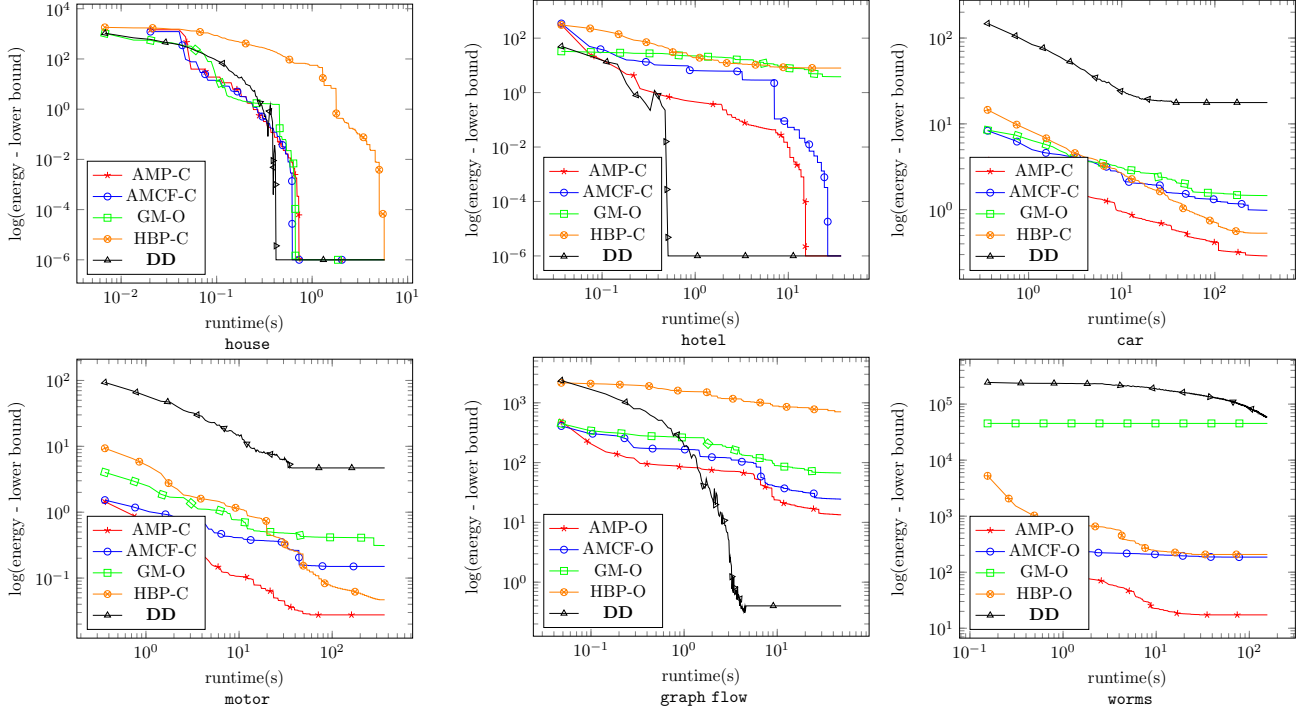


Figure 1. Plots showing convergence over time for house, hotel, car, motor, graph flow and worms datasets. Values denote $\log(\text{upper bound} - \text{lower bound})$. Values are averaged over all instances of the dataset. The x-axis and y-axis are logarithmic.

- `car` and `motor` were already harder and the 1000 iteration limit did not allow to ascertain optimality for all instances. **AMP** significantly outperforms its competitors, **DD** is significantly slower than the rest, whereas other algorithms show comparable results.
- on `worms` **AMP** significantly outperforms its competitors, **AMCF** and **HBP** converge to similar duality gap, although **AMCF** does it one-two orders of magnitude faster, **GM** and **DD** return results which are hardly competitive.
- `graph flow` is the only dataset, where **DD** clearly overcomes all competitors, followed by **AMP**. We attribute it to **DD**'s tighter relaxation, (its "local" subproblems contain 4 variables, whereas our subproblems have at most 3 variables after tightening.)

Insights and Conclusions

- **AMP** shows overall best performance for both small dense and large sparse datasets. It is the best anytime solver: it has the best performance in the first iterations. This is beneficial (i) if the run-time is limited or (ii) in branch-and-bound procedures, where a good early progress helps to efficiently eliminate non-optimal branches fast.
- Although **AMP**, **AMCF** and **HBP** address equivalent relaxations (having the same maximal dual value) their convergence speed is different. **AMCF** and **HBP** are generally slower than **AMP**, which we attribute to the sub-optimal redistribution of the costs by the min-cost-flow

- factors $\{\mathcal{M}\}$ when maximizing messages in (9).
- **DD**'s relatively good performance is probably due to the large subproblems used by this method. First, this decreases the number of dual variables, which accelerates bound convergence; second, this makes the relaxation tighter, which decreases the duality gap. We attribute slow convergence of **DD** to the subgradient method.
- Summarizing, larger subproblems are profitable for the sub-gradient method, but not for message passing.
- **AMCF** outperforms **HBP** due to better message scheduling.
- We attribute the inferior performance of **GM** mostly to the weakest relaxation it optimizes. Even under this condition, due to a good message scheduling and fast message passing it outperforms **DD** and **HBP** on several datasets. A detailed evaluation of all instances is in the appendix.

7. Acknowledgments

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

References

- [1] Graph flow dataset. <http://cvlab-dresden.de/research/iamge-matching/graphflow/>. 7
- [2] Hotel dataset. <http://vasc.ri.cmu.edu/idb/html/motion/hotel/>. 7
- [3] House dataset. <http://vasc.ri.cmu.edu/idb/html/motion/house/>. 7
- [4] Worms dataset. <https://datarep.app.ist.ac.at/57/>. 7
- [5] K. Adamczewski and Y. S. K. M. Lee. Discrete tabu search for graph matching. In *International Conference on Computer Vision*, 2015. 1
- [6] W. P. Adams and T. A. Johnson. Improved linear programming-based lower bounds for the quadratic assignment problem. *DIMACS series in discrete mathematics and theoretical computer science*, 16:43–75, 1994. 1
- [7] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1 edition, Feb. 1993. 4, 11, 13
- [8] H. A. Alhajja, A. Sellent, D. Kondermann, and C. Rother. Graphflow - 6d large displacement scene flow via graph matching. In J. Gall, P. V. Gehler, and B. Leibe, editors, *GCPR*, volume 9358 of *Lecture Notes in Computer Science*, pages 285–296. Springer, 2015. 7
- [9] K. M. Anstreicher and N. W. Brixius. A new bound for the quadratic assignment problem based on convex quadratic programming. *Mathematical Programming*, 89(3):341–357, 2001. 1
- [10] K. M. Anstreicher and N. W. Brixius. Solving quadratic assignment problems using convex quadratic programming relaxations. *Optimization Methods and Software*, 16(1-4):49–68, 2001. 2
- [11] M. Bazaraa and H. Sherali. New approaches for solving the quadratic assignment problem. In *Operations Research Verfahren*, volume 32, pages 29–46, 1979. 1
- [12] M. S. Bazaraa and H. D. Sherali. On the use of exact and heuristic cutting plane methods for the quadratic assignment problem. *Journal of the Operational Research Society*, 33(11):991–1003, 1982. 1
- [13] M. Beckman and T. Koopmans. Assignment problems and the location of economic activities. *Econometrica*, 25:53–76, 1957. 1
- [14] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004. 5
- [15] R. E. Burkard, E. Cela, P. M. Pardalos, and L. S. Pitsoulis. The quadratic assignment problem. In *Handbook of combinatorial optimization*, pages 1713–1809. Springer US, 1998. 2
- [16] M. Cho, J. Lee, and K. M. Lee. Reweighted random walks for graph matching. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *ECCV (5)*, volume 6315 of *Lecture Notes in Computer Science*, pages 492–505. Springer, 2010. 2, 7
- [17] M. Cho, J. Sun, O. Duchenne, and J. Ponce. Finding matches in a haystack: A max-pooling strategy for graph matching in the presence of outliers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2083–2090, 2014. 2
- [18] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *IJPRAI*, 18(3):265–298, 2004. 1
- [19] T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. In *NIPS*. MIT Press, Cambridge, MA, 2007. 7
- [20] J. Duchi, D. Tarlow, G. Elidan, and D. Koller. Using combinatorial optimization within max-product belief propagation. In *NIPS*, pages 369–376. MIT Press, 2006. 4, 7
- [21] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010. 7
- [22] A. Frieze and J. Yadegar. On the quadratic assignment problem. *Discrete applied mathematics*, 5(1):89–98, 1983. 1
- [23] J. W. Gavett and N. V. Plyter. The optimal assignment of facilities to locations by branch and bound. *Operations Research*, 14(2):210–232, 1966. 2
- [24] A. Globerson and T. S. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *NIPS*, pages 553–560, 2007. 2, 6
- [25] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(4):377–388, 1996. 1, 2, 7
- [26] P. Hahn, T. Grant, and N. Hall. A branch-and-bound algorithm for the quadratic assignment problem based on the hungarian method. *European Journal of Operational Research*, 108(3):629–640, 1998. 2
- [27] B. Jiang, J. Tang, C. Ding, and B. Luo. A local sparse model for matching problem. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI’15, pages 3790–3796. AAAI Press, 2015. 7
- [28] D. Kainmueller, F. Jug, C. Rother, and G. Myers. Active graph matching for automatic joint segmentation and annotation of *C. elegans*. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 81–88. Springer, 2014. 7
- [29] 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*, 115(2):155–184, 2015. 2
- [30] J. H. Kappes, B. Savchynskyy, and C. Schnörr. A bundle approach to efficient MAP-inference by Lagrangian relaxation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1688–1695. IEEE, 2012. 2
- [31] S. E. Karisch and F. Rendl. Lower bounds for the quadratic assignment problem via triangle decompositions. *Mathematical Programming*, 71(2):137–151, 1995. 1
- [32] K. C. Kiwiel. Proximity control in bundle methods for convex nondifferentiable minimization. *Mathematical programming*, 46(1-3):105–122, 1990. 2
- [33] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(10):1568–1583, 2006. 1, 2, 6
- [34] V. Kolmogorov. A new look at reweighted message passing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(5):919–930, 2015. 2, 6

- [35] B. Korte, J. Vygen, B. Korte, and J. Vygen. *Combinatorial optimization*, volume 2. Springer, 2012. 3
- [36] E. L. Lawler et al. The quadratic assignment problem. *Management Science*, 9(4):586–599, 1963. 1
- [37] J. Lee, M. Cho, and K. M. Lee. A graph matching algorithm using data-driven markov chain monte carlo sampling. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 2816–2819. IEEE, 2010. 2
- [38] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, pages 1482–1489. IEEE Computer Society, 2005. 2, 7
- [39] M. Leordeanu, M. Hebert, and R. Sukthankar. An integer projected fixed point method for graph matching and MAP inference. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *NIPS*, pages 1114–1122. Curran Associates, Inc., 2009. 1, 7
- [40] M. Leordeanu, R. Sukthankar, and M. Hebert. Unsupervised learning for graph matching. *International Journal of Computer Vision*, 96(1):28–45, 2012. 7
- [41] E. M. Loiola, N. M. M. de Abreu, P. O. Boaventura-netto, P. Hahn, and T. Querido. An analytical survey for the quadratic assignment problem. *EUROPEAN JOURNAL OF OPERATIONAL RESEARCH*, pages 657–690, 2007. 1, 2
- [42] J. Maciel and J. Costeira. A global solution to sparse correspondence problems. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(2):187–199, 2003. 7
- [43] Q. Nguyen, A. Gautier, and M. Hein. A flexible tensor block coordinate ascent scheme for hypergraph matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5270–5278, 2015. 1
- [44] P. M. Pardalos, K. A. Murthy, and T. P. Harrison. A computational comparison of local search heuristics for solving quadratic assignment problems. *Informatica*, 4(1-2):172–187, 1993. 1
- [45] C. Ribeiro and M. Minoux. Solving hard constrained shortest path problems by Lagrangean relaxation and branch-and-bound algorithms. *Methods of Operations Research*, 53:303–316, 1986. 2
- [46] C. Schellewald and C. Schnörr. Probabilistic subgraph matching based on convex relaxation. In *EMMCVPR*, volume 3757 of *Lecture Notes in Computer Science*, pages 171–186. Springer, 2005. 1, 7
- [47] M. I. Schlesinger. Syntactic analysis of two-dimensional visual signals in noisy conditions. *Kibernetika*, 4(113-130):1, 1976. 3
- [48] D. Sontag, D. K. Choe, and Y. Li. Efficiently searching for frustrated cycles in map inference. In *UAI*, pages 795–804. AUAI Press, 2012. 7
- [49] Y. Suh, M. Cho, and K. M. Lee. Graph matching via sequential monte carlo. In *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part III*, pages 624–637, 2012. 2
- [50] P. Swoboda, J. Kuske, and B. Savchynskyy. A dual ascent framework for Lagrangean decomposition of combinatorial problems. In *CVPR*, 2017. 2, 3, 5, 6
- [51] P. H. S. Torr. Solving Markov random fields using semi definite programming. In *In: AISTATS*, 2003. 1, 7
- [52] L. Torresani, V. Kolmogorov, and C. Rother. A dual decomposition approach to feature correspondence. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(2):259–271, 2013. 1, 2, 7
- [53] S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(5):695–703, 1988. 2
- [54] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008. 2, 3
- [55] T. Werner. A linear programming approach to max-sum problem: A review. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(7):1165–1179, 2007. 3
- [56] T. Werner. Revisiting the linear programming relaxation approach to Gibbs energy minimization and weighted constraint satisfaction. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(8):1474–1488, 2010. 7
- [57] S. J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015. 2
- [58] J. Yarkony, C. C. Fowlkes, and A. T. Ihler. Covering trees and lower-bounds on quadratic assignment. In *CVPR*, pages 887–894. IEEE Computer Society, 2010. 7
- [59] M. Zaslavskiy, F. Bach, and J.-P. Vert. A path following algorithm for the graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2227–2242, 2009. 2
- [60] Z. Zhang. Microsoft kinect sensor and its effect. *IEEE MultiMedia*, 19(2):4–10, Apr. 2012. 7
- [61] Z. Zhang, Q. Shi, J. McAuley, W. Wei, Y. Zhang, and A. van den Hengel. Pairwise matching through max-weight bipartite belief propagation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2, 4, 6, 7
- [62] G. Zhao, B. Luo, J. Tang, and J. Ma. *Using Eigen-Decomposition Method for Weighted Graph Matching*, pages 1283–1294. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. 2
- [63] Q. Zhao, S. E. Karisch, F. Rendl, and H. Wolkowicz. Semidefinite programming relaxations for the quadratic assignment problem. *Journal of Combinatorial Optimization*, 2(1):71–109, 1998. 1
- [64] F. Zhou and F. D. la Torre. Factorized graph matching. In *CVPR*, pages 127–134. IEEE Computer Society, 2012. 2, 7

8. Supplementary Material

8.1. Proofs

Proof of Proposition 1

Proof. • (R2) = (R3): As for both (R2) and (R3) we have $\mu \in \mathcal{L}$ and $\mu(s) = \tilde{\mu}_s(u)$, $s \in X_u$, we only need to prove that $\tilde{\mu}_s \in \text{conv}(X_s)$, $s \in \mathcal{L}$ is equivalent to $\tilde{\mu} \in \mathcal{M}$. By inspecting the definition (6) of \mathcal{M} we see that it is the intersection of (i) $\mu_u \in X_u$, $u \in \mathcal{V}$ and (ii) $\tilde{\mu}_s \in X_s$, $s \in \mathcal{L}$. However, by the equality constraints $\mu_u(s) = \tilde{\mu}_s(u)$ constraints (i) are already enforced, in (R2), hence equality follows.

- (R4) = (R5): Analogously to (R2) = (R3).
- Relaxation (R1) is weaker than (R2) and (R3): Relaxations (R2) and (R3) include additional constraints. □

8.2. Optimization Subproblems of Procedure 1

In Procedure 1 the two problem in lines 1 and 3 must be solved. Solution of the optimization problem in line 1 was discussed in the main part of the paper. Therefore, it only remains to show how to carry optimization of the problem in line 3 efficiently for all cases that can occur. This is shown in Table 3.

Checking validity of the operations in Table 3 for $i = u \in \mathcal{V}$ and $i = uv \in \mathcal{E}$ is straightforward. For $i = \mathcal{M}$ and $J = \mathcal{V}$, we prove correctness below. Correctness for $i = \mathcal{M}$ and $J = \mathcal{L}$ is analogous.

Lemma 2. *The reparametrization adjustment problem (9) for $i = \mathcal{M}$ and $J = \mathcal{V}$ is given by (14). Moreover it is the dual of a minimum cost network flow problem.*

Proof. Recall from network flow theory [7], that $x_u^* \in X_u$ is optimal for cost $\tilde{\theta}$, iff $\exists \pi \in \mathbb{R}^{|\mathcal{V}|}$, $\psi \in \mathbb{R}^{|\mathcal{L}|}$ such that

$$\tilde{\theta}_u(x_u) - \pi(u) + \psi(x_u) \begin{cases} \leq 0, & x_u^*(x_u) = 1 \\ \geq 0, & x_u^*(x_u) = 0 \end{cases} \forall u \in \mathcal{V}, x_u \in X_u.$$

Consider the primal/dual pair

$$\begin{aligned} \min_{(\mu_u)_{u \in \mathcal{V}}} & \sum_{u \in \mathcal{V}} \langle \tilde{\theta}_u, \mu_u \rangle & \max_{\Delta, \pi, \psi} & \sum_{u \in \mathcal{V}} \langle \Delta_u, \delta_u \rangle \\ \forall u \in \mathcal{V}, & \sum_{x_u \in X_u} \mu_u(x_u) = 0 & & \pi(u) \in \mathbb{R} \\ \forall l \in \mathcal{L}, & -\sum_{u: l \in X_u} \mu_u(l) = 0 & & \psi(l) \in \mathbb{R} \\ \mu_u(x_u) & \begin{cases} \geq \delta_u(x_u), & x_u \neq x_u^* \\ \leq \delta_u(x_u), & x_u = x_u^* \end{cases} & \Delta_u(x_u) \in & \begin{cases} \geq 0, & x_u^* = x_u \\ \leq 0, & x_u^* \neq x_u \end{cases} \\ \mu_u(x_u) & \in \mathbb{R} & \tilde{\theta}_u(x_u) + \Delta_u(x_u) + \pi(u) - \psi(x_u) & = 0 \end{aligned} \quad (12)$$

On the right side, the adjustment problem (3) is written down explicitly. The left hand side is a minimum cost flow problem, hence the second part of the claim is proven.

The last equality above on the right hand side ensures that $\Delta_u(x_u) = -\tilde{\theta}_u(x_u) - \pi(u) + \psi(x_u)$. Substituting this everywhere on the right hand side of (12) gives

$$\begin{aligned} \max_{\pi, \psi} & \sum_{u \in \mathcal{V}} \pi(u) \cdot \left(\sum_{x_u \in X_u} \delta_u(x_u) \right) + \sum_{l \in \mathcal{L}} \psi(l) \cdot \left(\sum_{u \in \mathcal{V}: l \in X_u} \delta_u(l) \right) \\ \text{s.t.} & \pi(u) \in \mathbb{R} \\ & \psi(l) \in \mathbb{R} \\ & \tilde{\theta}_u(x_u) + \pi(u) - \psi(x_u) \begin{cases} \leq 0, & x_u^* = x_u \\ \geq 0, & x_u^* \neq x_u \end{cases} \end{aligned} \quad (13)$$

This form matches the format given in (14). □

8.3. Time complexity

The time complexity of running one iteration of message passing is essentially the time to run all required invocations of Algorithm 1 via the routines described in Table 3. Total runtime per iteration for the various algorithms we have proposed can

be found in Table 4. We assume that $X_u = \mathcal{L} \forall u \in \mathcal{V}$. In sparse assignment problems, where this is not the case, run-time decreases according to sparsity.

If we hold the unary potentials $\theta_u, u \in \mathcal{V}$ in a heap, we can support operation $\min_{s' \in J \cap X_u} \theta_u(s')$ which is required in the third line in Table 3 in time $\log(|\mathcal{L}|)$, since either $J \cap X_u = X_u$ (sending) or $|J \cap X_u| = 1$ (receiving).

Hence, all our algorithms scale to realistic problem sizes.

8.4. Detailed Experimental Evaluation

Plots showing lower bound and primal solution energy per over time can be seen in Figure 2.

In Table 5 dataset statistics are given together with final upper and lower bound as well as runtime averaged over all instances in specific datasets are given.

A per-instance evaluation of all considered algorithms can be found in Table 6.

Algorithm 1 input $i \in \mathbb{V}$ $J \subseteq \mathcal{N}_{\mathbb{G}}(i)$	Solution $\Delta_{(i,j)}^* \forall j \in J$ of (9)
$i = u \in \mathbb{V}$ $J \subseteq \mathbb{E} \cup \{\mathcal{M}\} \cup \mathcal{L}$	$\Delta_{(u,uv)}^* = (\theta_u - \min_{x_u \in X_u} \theta_u(x_u)) / J \quad \forall uv \in \mathbb{E} \cap J$ $\Delta_{(u,\mathcal{M})}^* = (\theta_u - \min_{x_u \in X_u} \theta_u(x_u)) / J $ $\Delta_{(u,s)}^* = (\theta_u(s) - \min_{s' \in J \cap X_u} \theta_u(s')) / J \quad \forall s \in X_u \cap J$
$i = uv \in \mathbb{E}$ $J = \{u\}, u \in \mathbb{V}$	$\Delta_{(uv,u)}^*(x_u) = \min_{x_v \in X_v} \{\theta_{uv}(x_u, x_v)\} - \min_{x_{uv} \in X_{uv}} \{\theta_{uv}(x_{uv})\}$
$i = \mathcal{M}$ $J = \mathbb{V}$	$\Delta_u(x_u) = -\tilde{\theta}_u(x_u) - \pi^*(u) + \psi^*(x_u)$ $(\pi^*, \psi^*) \in \operatorname{argmax}_{\pi, \psi} \begin{aligned} & \sum_{u \in \mathbb{V}} \pi(u) \cdot \left(\sum_{x_u \in X_u} \delta_u(x_u) \right) \\ & + \sum_{l \in \mathcal{L}} \psi(l) \cdot \left(\sum_{u \in \mathbb{V}: l \in X_u} \delta_u(l) \right) \end{aligned} \quad (14)$ $\text{s.t. } \tilde{\theta}_u(x_u) + \pi(u) - \psi(x_u) \begin{cases} \leq 0, & x_u^* = x_u \\ \geq 0, & x_u^* \neq x_u \end{cases}$
$i = \mathcal{M}$ $J = \mathcal{L}$	$\Delta_s(u) = -\tilde{\theta}_u(s) - \pi^*(u) + \psi^*(s)$ $(\pi^*, \psi^*) \in \operatorname{argmax}_{\pi, \psi} \begin{aligned} & \sum_{u \in \mathbb{V}} \pi(u) \cdot \left(\sum_{x_u \in X_u} \delta_{x_u}(u) \right) \\ & + \sum_{l \in \mathcal{L}} \psi(l) \cdot \left(\sum_{u \in \mathbb{V}: l \in X_u} \delta_l(u) \right) \end{aligned} \quad (15)$ $\text{s.t. } \tilde{\theta}_u(x_u) + \pi(u) - \psi(x_u) \begin{cases} \leq 0, & x_u^* = x_u \\ \geq 0, & x_u^* \neq x_u \end{cases}$

Table 3. Message computation problems (9)

Algorithm	Time complexity
GM	$O(\mathcal{L} \cdot \mathbb{V} + \mathcal{L} ^2 \cdot \dot{\mathbb{E}})$
AMP	$O(\mathcal{L} \cdot \mathbb{V} + \mathcal{L} ^2 \cdot \mathbb{E} + \mathcal{L} ^3)$
AMP[†]	$O(\mathcal{L} \cdot \mathbb{V} + \mathcal{L} ^2 \cdot \mathbb{E} + \mathcal{L} ^2 \cdot \log \mathcal{L})$
AMCF	$O(\mathcal{L} \cdot \mathbb{V} + \mathcal{L} ^2 \cdot \mathbb{E} + MCF(\mathbb{V} , \mathbb{V} ^2))$

Table 4. Time complexity per iteration for the three proposed algorithms. $MCF(n, m)$ is the time to solve a min-cost-flow problem on a graph with n nodes and m edges: Orlin’s algorithm has time complexity $O(m^2 \log n + m \log^2 n)$ [7]. **AMP[†]** stores reparametrized unary costs in a heap to accelerate computation of the messages between label factors and unaries.

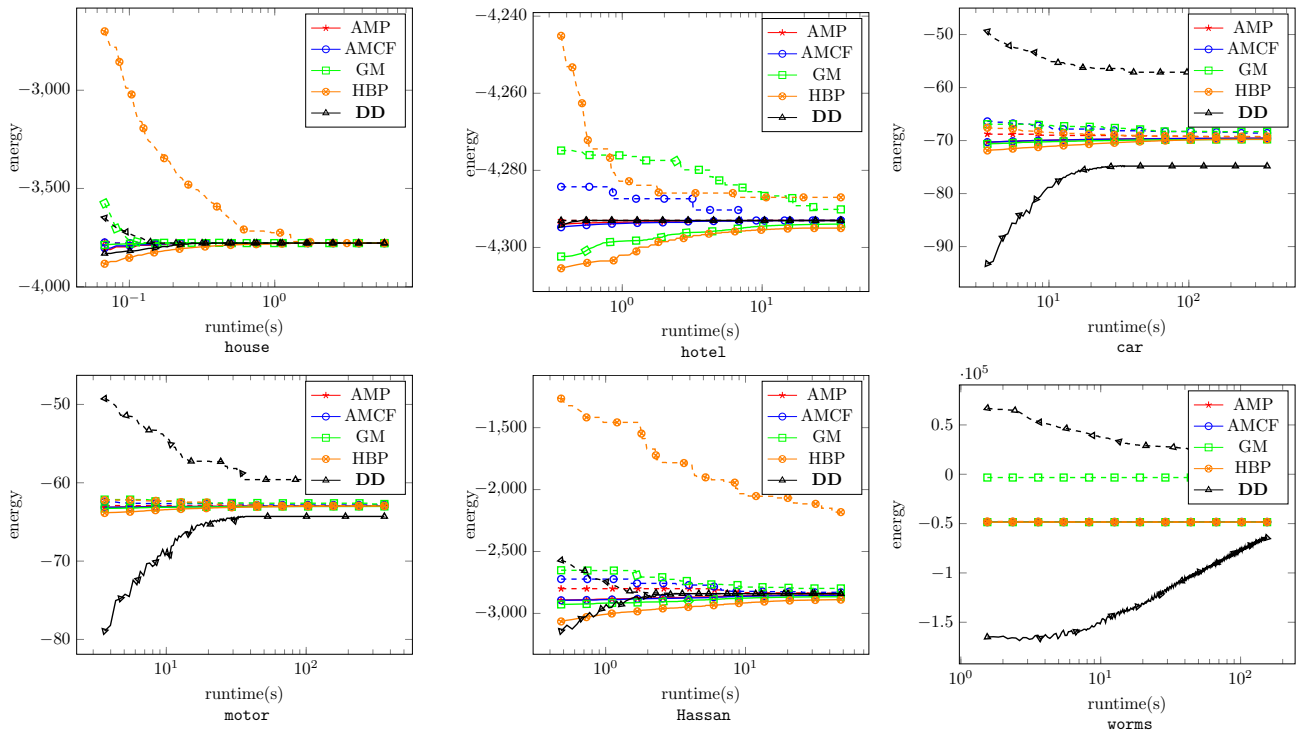


Figure 2. Runtime plots for house, hotel, car, motor, graph flow and worms datasets. Continuous lines denote dual lower bounds and dashed ones primal energies. Values are averaged over all instances of the dataset. The x-axis is logarithmic.

Dataset/Algorithm				AMP	AMCF	GM	HBP	DD
hotel	#I	105	LB	-4293.00	-4293.00	-4293.94	-4294.98	-4293.00
	#V	30	UB	-4293.00	-4293.00	-4290.12	-4287.00	-4293.00
	#L	30	time(s)	3.07	4.31	14.91	16.76	0.42
house	#I	105	LB	-3778.13	-3778.13	-3778.13	-3778.13	-3778.13
	#V	30	UB	-3778.13	-3778.13	-3778.13	-3778.13	-3778.13
	#L	30	time(s)	2.81	3.02	1.63	14.76	2.36
graph flow	#I	6	LB	-2849.96	-2853.95	-2865.20	-2888.88	-2840.29
	#V	≤ 126	UB	-2836.57	-2829.49	-2798.11	-2181.96	-2840.00
	#L	≤ 126	time(s)	218.22	231.53	216.39	194.43	17.92
car	#I	30	LB	-69.56	-69.57	-69.78	-69.77	-74.17
	#V	≤ 49	UB	-69.27	-68.58	-68.32	-69.23	-57.11
	#L	≤ 49	time(s)	656.86	698.74	776.58	572.41	83.71
motor	#I	20	LB	-62.97	-62.98	-63.02	-62.98	-64.25
	#V	≤ 52	UB	-62.95	-62.83	-62.71	-62.93	-59.60
	#L	≤ 52	time(s)	70.82	98.57	285.19	317.61	69.22
worms	#I	30	LB	-48471.21	-48491.81	-48517.50	-48495.92	-62934.54
	#V	≤ 605	UB	-48453.90	-48305.90	-3316.69	-48288.65	-5254.08
	#L	≤ 1500	time(s)	213.89	614.80	229.21	295.67	1471.95

Table 5. Description of datasets together with averaged algorithm results over all instances. #I means number of instances in dataset, #V the number of nodes per instance, #L the number of labels per instance, LB means lower bound, UB upper bound. **Bold numbers** indicate highest lower bound, lowest primal energy and smallest runtime.

Table 6: Per-instance results. UB means primal solution energy, LB dual lower bound and runtime(s) the runtime of the particular algorithm in seconds. **Bold numbers** indicate lowest primal energy, highest lower bound and smallest runtime.

Instance		AMP	AMCF	GM	HBP	DD
hotel						
energy_hotel_frame15frame22	UB	-4598.04	-4598.04	-4598.04	-4598.04	-4598.04
	LB	-4598.04	-4598.04	-4598.04	-4598.04	-4598.04
	runtime(s)	0.15	0.14	0.02	0.32	0.03
energy_hotel_frame15frame29	UB	-4540.81	-4540.81	-4540.81	-4540.81	-4540.81
	LB	-4540.81	-4540.81	-4540.81	-4540.81	-4540.81
	runtime(s)	0.23	0.14	0.02	0.32	0.04
energy_hotel_frame15frame36	UB	-4481.33	-4481.33	-4481.33	-4481.33	-4481.33
	LB	-4481.33	-4481.33	-4481.33	-4481.33	-4481.33
	runtime(s)	0.48	0.46	0.02	0.42	0.04
energy_hotel_frame15frame43	UB	-4377.31	-4377.31	-4377.31	-4377.31	-4377.31
	LB	-4377.31	-4377.31	-4377.31	-4377.31	-4377.31
	runtime(s)	0.47	0.46	0.13	0.52	0.06
energy_hotel_frame15frame50	UB	-4294.04	-4294.04	-4294.04	-4294.04	-4294.04
	LB	-4294.04	-4294.04	-4294.04	-4294.04	-4294.04
	runtime(s)	0.47	0.46	0.13	0.63	0.07
energy_hotel_frame15frame57	UB	-4244.75	-4244.75	-4244.75	-4244.75	-4244.75
	LB	-4244.75	-4244.75	-4244.75	-4244.75	-4244.75
	runtime(s)	0.48	0.46	0.13	0.72	0.12
energy_hotel_frame15frame64	UB	-4172.32	-4172.32	-4172.32	-4172.32	-4172.32
	LB	-4172.32	-4172.32	-4172.32	-4172.32	-4172.32
	runtime(s)	0.47	0.46	0.13	3.56	0.18
energy_hotel_frame15frame71	UB	-4135.78	-4135.78	-4135.78	-4135.78	-4135.78
	LB	-4135.78	-4135.78	-4135.78	-4135.78	-4135.78
	runtime(s)	0.80	0.79	0.35	4.83	0.30
energy_hotel_frame15frame78	UB	-4036.15	-4036.15	-4036.15	-4036.15	-4036.15
	LB	-4036.15	-4036.15	-4036.15	-4036.15	-4036.15
	runtime(s)	1.50	1.41	6.45	17.64	0.70
energy_hotel_frame15frame85	UB	-3985.99	-3985.99	-3985.99	-3985.99	-3985.99
	LB	-3985.99	-3985.99	-3985.99	-3985.99	-3985.99
	runtime(s)	1.92	2.04	14.64	28.40	0.66
energy_hotel_frame15frame92	UB	-3898.11	-3898.11	-3898.11	-3898.11	-3898.11
	LB	-3898.11	-3898.11	-3898.11	-3898.11	-3898.11
	runtime(s)	3.25	4.78	31.59	99.60	1.93
energy_hotel_frame15frame99	UB	-3860.42	-3860.42	-3860.42	-3860.42	-3860.42
	LB	-3860.42	-3860.42	-3860.42	-3869.73	-3860.42
	runtime(s)	11.24	11.89	160.82	192.49	1.15
energy_hotel_frame1frame15	UB	-4498.03	-4498.03	-4498.03	-4498.03	-4498.03
	LB	-4498.03	-4498.03	-4498.03	-4498.03	-4498.03
	runtime(s)	0.15	0.23	0.02	0.32	0.04
energy_hotel_frame1frame22	UB	-4438.33	-4438.33	-4438.33	-4438.33	-4438.33
	LB	-4438.33	-4438.33	-4438.33	-4438.33	-4438.33
	runtime(s)	0.47	0.65	0.13	0.32	0.05
energy_hotel_frame1frame29	UB	-4368.55	-4368.55	-4368.55	-4368.55	-4368.55
	LB	-4368.55	-4368.55	-4368.55	-4368.55	-4368.55
	runtime(s)	0.56	0.47	0.13	0.72	0.05
energy_hotel_frame1frame36	UB	-4306.23	-4306.23	-4306.23	-4306.23	-4306.23
	LB	-4306.23	-4306.23	-4306.23	-4306.23	-4306.23
	runtime(s)	0.48	0.64	0.13	0.61	0.06
energy_hotel_frame1frame43	UB	-4194.42	-4194.42	-4194.42	-4194.42	-4194.42
	LB	-4194.42	-4194.42	-4194.42	-4194.42	-4194.42
	runtime(s)	0.47	0.64	0.13	1.12	0.33
energy_hotel_frame1frame50	UB	-4125.68	-4125.68	-4125.68	-4125.68	-4125.68
	LB	-4125.68	-4125.68	-4125.68	-4125.68	-4125.68
	runtime(s)	0.48	0.46	0.13	1.12	0.26
energy_hotel_frame1frame57	UB	-4064.73	-4064.73	-4064.73	-4064.73	-4064.73
	LB	-4064.73	-4064.73	-4064.73	-4064.73	-4064.73
	runtime(s)	0.80	0.79	0.35	5.65	0.33
energy_hotel_frame1frame64	UB	-4021.19	-4021.19	-4021.19	-4021.19	-4021.19
	LB	-4021.19	-4021.19	-4021.19	-4021.19	-4021.19
	runtime(s)	1.26	1.09	0.58	6.41	0.67

Table 6: Per-instance results. UB means primal solution energy, LB dual lower bound and runtime(s) the runtime of the particular algorithm in seconds. **Bold numbers** indicate lowest primal energy, highest lower bound and smallest runtime.

Instance		AMP	AMCF	GM	HBP	DD
energy_hotel_frame1frame71	UB	-3969.29	-3969.29	-3969.29	-3969.29	-3969.29
	LB	-3969.29	-3969.29	-3969.29	-3969.29	-3969.29
	runtime(s)	1.12	1.42	1.66	12.79	0.98
energy_hotel_frame1frame78	UB	-3874.87	-3874.87	-3874.87	-3874.87	-3874.87
	LB	-3874.87	-3874.87	-3874.87	-3874.87	-3874.87
	runtime(s)	2.41	6.18	39.91	51.40	1.73
energy_hotel_frame1frame8	UB	-4570.58	-4570.58	-4570.58	-4570.58	-4570.58
	LB	-4570.58	-4570.58	-4570.58	-4570.58	-4570.58
	runtime(s)	0.15	0.14	0.02	0.32	0.03
energy_hotel_frame1frame85	UB	-3817.96	-3817.96	-3817.96	-3817.96	-3817.96
	LB	-3817.96	-3817.96	-3817.96	-3820.37	-3817.96
	runtime(s)	4.67	6.83	87.48	180.26	1.77
energy_hotel_frame1frame92	UB	-3728.34	-3728.34	-3728.34	-3728.34	-3728.34
	LB	-3728.34	-3728.34	-3748.95	-3770.59	-3728.34
	runtime(s)	20.19	39.39	365.63	190.72	2.34
energy_hotel_frame1frame99	UB	-3691.38	-3691.38	-3497.74	-3486.96	-3691.38
	LB	-3691.38	-3691.38	-3741.70	-3777.32	-3691.38
	runtime(s)	152.97	264.53	299.78	192.51	4.79
energy_hotel_frame22frame29	UB	-4615.38	-4615.38	-4615.38	-4615.38	-4615.38
	LB	-4615.38	-4615.38	-4615.38	-4615.38	-4615.38
	runtime(s)	0.15	0.14	0.02	0.22	0.03
energy_hotel_frame22frame36	UB	-4527.37	-4527.37	-4527.37	-4527.37	-4527.37
	LB	-4527.37	-4527.37	-4527.37	-4527.37	-4527.37
	runtime(s)	0.15	0.14	0.02	0.32	0.03
energy_hotel_frame22frame43	UB	-4428.42	-4428.42	-4428.42	-4428.42	-4428.42
	LB	-4428.42	-4428.42	-4428.42	-4428.42	-4428.42
	runtime(s)	0.48	0.46	0.02	0.42	0.06
energy_hotel_frame22frame50	UB	-4343.82	-4343.82	-4343.82	-4343.82	-4343.82
	LB	-4343.82	-4343.82	-4343.82	-4343.82	-4343.82
	runtime(s)	0.48	0.46	0.13	0.42	0.09
energy_hotel_frame22frame57	UB	-4302.35	-4302.35	-4302.35	-4302.35	-4302.35
	LB	-4302.35	-4302.35	-4302.35	-4302.35	-4302.35
	runtime(s)	0.47	0.46	0.13	0.61	0.07
energy_hotel_frame22frame64	UB	-4219.53	-4219.53	-4219.53	-4219.53	-4219.53
	LB	-4219.53	-4219.53	-4219.53	-4219.53	-4219.53
	runtime(s)	0.47	0.46	0.13	0.91	0.10
energy_hotel_frame22frame71	UB	-4188.30	-4188.30	-4188.30	-4188.30	-4188.30
	LB	-4188.30	-4188.30	-4188.30	-4188.30	-4188.30
	runtime(s)	0.47	0.46	0.24	3.76	0.26
energy_hotel_frame22frame78	UB	-4109.42	-4109.42	-4109.42	-4109.42	-4109.42
	LB	-4109.42	-4109.42	-4109.42	-4109.42	-4109.42
	runtime(s)	0.80	0.78	0.35	4.79	0.10
energy_hotel_frame22frame85	UB	-4063.39	-4063.39	-4063.39	-4063.39	-4063.39
	LB	-4063.39	-4063.39	-4063.39	-4063.39	-4063.39
	runtime(s)	1.12	1.10	1.11	8.04	0.37
energy_hotel_frame22frame92	UB	-3979.05	-3979.05	-3979.05	-3979.05	-3979.05
	LB	-3979.05	-3979.05	-3979.05	-3979.05	-3979.05
	runtime(s)	1.44	2.05	11.54	23.91	0.44
energy_hotel_frame22frame99	UB	-3956.74	-3956.74	-3956.74	-3956.74	-3956.74
	LB	-3956.74	-3956.74	-3956.74	-3956.74	-3956.74
	runtime(s)	2.73	3.65	36.03	31.84	0.67
energy_hotel_frame29frame36	UB	-4605.48	-4605.48	-4605.48	-4605.48	-4605.48
	LB	-4605.48	-4605.48	-4605.48	-4605.48	-4605.48
	runtime(s)	0.15	0.14	0.02	0.42	0.02
energy_hotel_frame29frame43	UB	-4493.70	-4493.70	-4493.70	-4493.70	-4493.70
	LB	-4493.70	-4493.70	-4493.70	-4493.70	-4493.70
	runtime(s)	0.47	0.46	0.02	0.32	0.04
energy_hotel_frame29frame50	UB	-4408.69	-4408.69	-4408.69	-4408.69	-4408.69
	LB	-4408.69	-4408.69	-4408.69	-4408.69	-4408.69
	runtime(s)	0.48	0.46	0.13	0.42	0.08

Table 6: Per-instance results. UB means primal solution energy, LB dual lower bound and runtime(s) the runtime of the particular algorithm in seconds. **Bold numbers** indicate lowest primal energy, highest lower bound and smallest runtime.

Instance		AMP	AMCF	GM	HBP	DD
energy_hotel_frame29frame57	UB	-4373.36	-4373.36	-4373.36	-4373.36	-4373.36
	LB	-4373.36	-4373.36	-4373.36	-4373.36	-4373.36
	runtime(s)	0.47	0.46	0.13	0.52	0.06
energy_hotel_frame29frame64	UB	-4295.62	-4295.62	-4295.62	-4295.62	-4295.62
	LB	-4295.62	-4295.62	-4295.62	-4295.62	-4295.62
	runtime(s)	0.47	0.46	0.13	0.52	0.07
energy_hotel_frame29frame71	UB	-4253.97	-4253.97	-4253.97	-4253.97	-4253.97
	LB	-4253.97	-4253.97	-4253.97	-4253.97	-4253.97
	runtime(s)	0.48	0.46	0.13	0.71	0.09
energy_hotel_frame29frame78	UB	-4167.70	-4167.70	-4167.70	-4167.70	-4167.70
	LB	-4167.70	-4167.70	-4167.70	-4167.70	-4167.70
	runtime(s)	0.80	0.78	0.24	3.82	0.32
energy_hotel_frame29frame85	UB	-4118.45	-4118.45	-4118.45	-4118.45	-4118.45
	LB	-4118.45	-4118.45	-4118.45	-4118.45	-4118.45
	runtime(s)	0.80	0.78	0.35	2.09	0.24
energy_hotel_frame29frame92	UB	-4037.12	-4037.12	-4037.12	-4037.12	-4037.12
	LB	-4037.12	-4037.12	-4037.12	-4037.12	-4037.12
	runtime(s)	1.12	1.09	2.87	9.02	0.34
energy_hotel_frame29frame99	UB	-4007.34	-4007.34	-4007.34	-4007.34	-4007.34
	LB	-4007.34	-4007.34	-4007.34	-4007.34	-4007.34
	runtime(s)	1.77	2.05	14.45	21.69	0.57
energy_hotel_frame36frame43	UB	-4571.30	-4571.30	-4571.30	-4571.30	-4571.30
	LB	-4571.30	-4571.30	-4571.30	-4571.30	-4571.30
	runtime(s)	0.15	0.14	0.02	0.32	0.06
energy_hotel_frame36frame50	UB	-4489.95	-4489.95	-4489.95	-4489.95	-4489.95
	LB	-4489.95	-4489.95	-4489.95	-4489.95	-4489.95
	runtime(s)	0.15	0.46	0.13	0.42	0.07
energy_hotel_frame36frame57	UB	-4451.32	-4451.32	-4451.32	-4451.32	-4451.32
	LB	-4451.32	-4451.32	-4451.32	-4451.32	-4451.32
	runtime(s)	0.47	0.46	0.02	0.52	0.08
energy_hotel_frame36frame64	UB	-4373.51	-4373.51	-4373.51	-4373.51	-4373.51
	LB	-4373.51	-4373.51	-4373.51	-4373.51	-4373.51
	runtime(s)	0.47	0.47	0.13	0.52	0.10
energy_hotel_frame36frame71	UB	-4326.74	-4326.74	-4326.74	-4326.74	-4326.74
	LB	-4326.74	-4326.74	-4326.74	-4326.74	-4326.74
	runtime(s)	0.48	0.46	0.13	2.86	0.17
energy_hotel_frame36frame78	UB	-4249.20	-4249.20	-4249.20	-4249.20	-4249.20
	LB	-4249.20	-4249.20	-4249.20	-4249.20	-4249.20
	runtime(s)	0.47	0.46	0.13	3.65	0.10
energy_hotel_frame36frame85	UB	-4192.38	-4192.38	-4192.38	-4192.38	-4192.38
	LB	-4192.38	-4192.38	-4192.38	-4192.38	-4192.38
	runtime(s)	0.47	0.78	0.24	1.62	0.32
energy_hotel_frame36frame92	UB	-4124.33	-4124.33	-4124.33	-4124.33	-4124.33
	LB	-4124.33	-4124.33	-4124.33	-4124.33	-4124.33
	runtime(s)	0.81	0.78	0.79	6.90	0.67
energy_hotel_frame36frame99	UB	-4094.55	-4094.55	-4094.55	-4094.55	-4094.55
	LB	-4094.55	-4094.55	-4094.55	-4094.55	-4094.55
	runtime(s)	1.12	1.12	9.03	21.35	0.78
energy_hotel_frame43frame50	UB	-4563.61	-4563.61	-4563.61	-4563.61	-4563.61
	LB	-4563.61	-4563.61	-4563.61	-4563.61	-4563.61
	runtime(s)	0.15	0.14	0.02	0.33	0.09
energy_hotel_frame43frame57	UB	-4532.17	-4532.17	-4532.17	-4532.17	-4532.17
	LB	-4532.17	-4532.17	-4532.17	-4532.17	-4532.17
	runtime(s)	0.15	0.14	0.02	0.32	0.05
energy_hotel_frame43frame64	UB	-4450.44	-4450.44	-4450.44	-4450.44	-4450.44
	LB	-4450.44	-4450.44	-4450.44	-4450.44	-4450.44
	runtime(s)	0.47	0.46	0.13	0.42	0.13
energy_hotel_frame43frame71	UB	-4422.17	-4422.17	-4422.17	-4422.17	-4422.17
	LB	-4422.17	-4422.17	-4422.17	-4422.17	-4422.17
	runtime(s)	0.48	0.46	0.13	2.68	0.16

Table 6: Per-instance results. UB means primal solution energy, LB dual lower bound and runtime(s) the runtime of the particular algorithm in seconds. **Bold numbers** indicate lowest primal energy, highest lower bound and smallest runtime.

Instance		AMP	AMCF	GM	HBP	DD
energy_hotel_frame43frame78	UB	-4351.89	-4351.89	-4351.89	-4351.89	-4351.89
	LB	-4351.89	-4351.89	-4351.89	-4351.89	-4351.89
	runtime(s)	0.48	0.46	0.13	0.71	0.10
energy_hotel_frame43frame85	UB	-4295.89	-4295.89	-4295.89	-4295.89	-4295.89
	LB	-4295.89	-4295.89	-4295.89	-4295.89	-4295.89
	runtime(s)	0.48	0.46	0.13	0.82	0.24
energy_hotel_frame43frame92	UB	-4221.52	-4221.52	-4221.52	-4221.52	-4221.52
	LB	-4221.52	-4221.52	-4221.52	-4221.52	-4221.52
	runtime(s)	0.80	0.78	0.24	3.58	0.13
energy_hotel_frame43frame99	UB	-4190.13	-4190.13	-4190.13	-4190.13	-4190.13
	LB	-4190.13	-4190.13	-4190.13	-4190.13	-4190.13
	runtime(s)	0.80	0.78	0.35	5.01	0.34
energy_hotel_frame50frame57	UB	-4566.79	-4566.79	-4566.79	-4566.79	-4566.79
	LB	-4566.79	-4566.79	-4566.79	-4566.79	-4566.79
	runtime(s)	0.15	0.14	0.02	0.41	0.05
energy_hotel_frame50frame64	UB	-4517.31	-4517.31	-4517.31	-4517.31	-4517.31
	LB	-4517.31	-4517.31	-4517.31	-4517.31	-4517.31
	runtime(s)	0.47	0.46	0.02	0.42	0.06
energy_hotel_frame50frame71	UB	-4463.52	-4463.52	-4463.52	-4463.52	-4463.52
	LB	-4463.52	-4463.52	-4463.52	-4463.52	-4463.52
	runtime(s)	0.47	0.46	0.13	0.52	0.10
energy_hotel_frame50frame78	UB	-4400.83	-4400.83	-4400.83	-4400.83	-4400.83
	LB	-4400.83	-4400.83	-4400.83	-4400.83	-4400.83
	runtime(s)	0.48	0.46	0.14	0.52	0.12
energy_hotel_frame50frame85	UB	-4342.35	-4342.35	-4342.35	-4342.35	-4342.35
	LB	-4342.35	-4342.35	-4342.35	-4342.35	-4342.35
	runtime(s)	0.47	0.46	0.13	0.72	0.15
energy_hotel_frame50frame92	UB	-4260.01	-4260.01	-4260.01	-4260.01	-4260.01
	LB	-4260.01	-4260.01	-4260.01	-4260.01	-4260.01
	runtime(s)	0.47	0.46	0.24	3.36	0.17
energy_hotel_frame50frame99	UB	-4240.95	-4240.95	-4240.95	-4240.95	-4240.95
	LB	-4240.95	-4240.95	-4240.95	-4240.95	-4240.95
	runtime(s)	0.80	0.46	0.24	4.63	0.14
energy_hotel_frame57frame64	UB	-4567.13	-4567.13	-4567.13	-4567.13	-4567.13
	LB	-4567.13	-4567.13	-4567.13	-4567.13	-4567.13
	runtime(s)	0.15	0.14	0.02	0.32	0.07
energy_hotel_frame57frame71	UB	-4508.58	-4508.58	-4508.58	-4508.58	-4508.58
	LB	-4508.58	-4508.58	-4508.58	-4508.58	-4508.58
	runtime(s)	0.15	0.15	0.02	0.32	0.11
energy_hotel_frame57frame78	UB	-4475.54	-4475.54	-4475.54	-4475.54	-4475.54
	LB	-4475.54	-4475.54	-4475.54	-4475.54	-4475.54
	runtime(s)	0.48	0.46	0.13	0.42	0.13
energy_hotel_frame57frame85	UB	-4398.90	-4398.90	-4398.90	-4398.90	-4398.90
	LB	-4398.90	-4398.90	-4398.90	-4398.90	-4398.90
	runtime(s)	0.48	0.46	0.13	0.62	0.19
energy_hotel_frame57frame92	UB	-4344.52	-4344.52	-4344.52	-4344.52	-4344.52
	LB	-4344.52	-4344.52	-4344.52	-4344.52	-4344.52
	runtime(s)	0.48	0.46	0.13	0.92	0.12
energy_hotel_frame57frame99	UB	-4332.33	-4332.33	-4332.33	-4332.33	-4332.33
	LB	-4332.33	-4332.33	-4332.33	-4332.33	-4332.33
	runtime(s)	0.48	0.46	0.13	3.06	0.23
energy_hotel_frame64frame71	UB	-4578.65	-4578.65	-4578.65	-4578.65	-4578.65
	LB	-4578.65	-4578.65	-4578.65	-4578.65	-4578.65
	runtime(s)	0.15	0.14	0.02	0.32	0.07
energy_hotel_frame64frame78	UB	-4545.63	-4545.63	-4545.63	-4545.63	-4545.63
	LB	-4545.63	-4545.63	-4545.63	-4545.63	-4545.63
	runtime(s)	0.47	0.14	0.02	0.41	0.07
energy_hotel_frame64frame85	UB	-4481.13	-4481.13	-4481.13	-4481.13	-4481.13
	LB	-4481.13	-4481.13	-4481.13	-4481.13	-4481.13
	runtime(s)	0.47	0.46	0.13	0.51	0.10

Table 6: Per-instance results. UB means primal solution energy, LB dual lower bound and runtime(s) the runtime of the particular algorithm in seconds. **Bold numbers** indicate lowest primal energy, highest lower bound and smallest runtime.

Instance		AMP	AMCF	GM	HBP	DD
energy_hotel_frame64frame92	UB	-4413.01	-4413.01	-4413.01	-4413.01	-4413.01
	LB	-4413.01	-4413.01	-4413.01	-4413.01	-4413.01
	runtime(s)	0.47	0.46	0.13	0.61	0.09
energy_hotel_frame64frame99	UB	-4385.51	-4385.51	-4385.51	-4385.51	-4385.51
	LB	-4385.51	-4385.51	-4385.51	-4385.51	-4385.51
	runtime(s)	0.47	0.46	0.13	0.81	0.11
energy_hotel_frame71frame78	UB	-4550.95	-4550.95	-4550.95	-4550.95	-4550.95
	LB	-4550.95	-4550.95	-4550.95	-4550.95	-4550.95
	runtime(s)	0.48	0.46	0.03	0.42	0.10
energy_hotel_frame71frame85	UB	-4552.71	-4552.71	-4552.71	-4552.71	-4552.71
	LB	-4552.71	-4552.71	-4552.71	-4552.71	-4552.71
	runtime(s)	0.15	0.14	0.02	0.41	0.17
energy_hotel_frame71frame92	UB	-4469.72	-4469.72	-4469.72	-4469.72	-4469.72
	LB	-4469.72	-4469.72	-4469.72	-4469.72	-4469.72
	runtime(s)	0.48	0.46	0.13	0.52	0.10
energy_hotel_frame71frame99	UB	-4413.34	-4413.34	-4413.34	-4413.34	-4413.34
	LB	-4413.34	-4413.34	-4413.34	-4413.34	-4413.34
	runtime(s)	0.47	0.46	0.13	0.52	0.17
energy_hotel_frame78frame85	UB	-4589.16	-4589.16	-4589.16	-4589.16	-4589.16
	LB	-4589.16	-4589.16	-4589.16	-4589.16	-4589.16
	runtime(s)	0.48	0.14	0.02	0.52	0.09
energy_hotel_frame78frame92	UB	-4545.04	-4545.04	-4545.04	-4545.04	-4545.04
	LB	-4545.04	-4545.04	-4545.04	-4545.04	-4545.04
	runtime(s)	0.15	0.14	0.02	0.41	0.10
energy_hotel_frame78frame99	UB	-4534.77	-4534.77	-4534.77	-4534.77	-4534.77
	LB	-4534.77	-4534.77	-4534.77	-4534.77	-4534.77
	runtime(s)	0.15	0.14	0.02	0.42	0.08
energy_hotel_frame85frame92	UB	-4578.16	-4578.16	-4578.16	-4578.16	-4578.16
	LB	-4578.16	-4578.16	-4578.16	-4578.16	-4578.16
	runtime(s)	0.15	0.14	0.02	0.32	0.07
energy_hotel_frame85frame99	UB	-4528.32	-4528.32	-4528.32	-4528.32	-4528.32
	LB	-4528.32	-4528.32	-4528.32	-4528.32	-4528.32
	runtime(s)	0.47	0.46	0.02	0.41	0.14
energy_hotel_frame8frame15	UB	-4572.64	-4572.64	-4572.64	-4572.64	-4572.64
	LB	-4572.64	-4572.64	-4572.64	-4572.64	-4572.64
	runtime(s)	0.15	0.14	0.02	0.22	0.05
energy_hotel_frame8frame22	UB	-4491.45	-4491.45	-4491.45	-4491.45	-4491.45
	LB	-4491.45	-4491.45	-4491.45	-4491.45	-4491.45
	runtime(s)	0.47	0.14	0.02	0.32	0.05
energy_hotel_frame8frame29	UB	-4424.96	-4424.96	-4424.96	-4424.96	-4424.96
	LB	-4424.96	-4424.96	-4424.96	-4424.96	-4424.96
	runtime(s)	0.47	0.48	0.13	2.48	0.07
energy_hotel_frame8frame36	UB	-4379.36	-4379.36	-4379.36	-4379.36	-4379.36
	LB	-4379.36	-4379.36	-4379.36	-4379.36	-4379.36
	runtime(s)	0.48	0.49	0.13	0.52	0.09
energy_hotel_frame8frame43	UB	-4262.09	-4262.09	-4262.09	-4262.09	-4262.09
	LB	-4262.09	-4262.09	-4262.09	-4262.09	-4262.09
	runtime(s)	0.48	0.46	0.13	1.12	0.23
energy_hotel_frame8frame50	UB	-4179.32	-4179.32	-4179.32	-4179.32	-4179.32
	LB	-4179.32	-4179.32	-4179.32	-4179.32	-4179.32
	runtime(s)	0.48	0.47	0.13	1.34	0.24
energy_hotel_frame8frame57	UB	-4131.18	-4131.18	-4131.18	-4131.18	-4131.18
	LB	-4131.18	-4131.18	-4131.18	-4131.18	-4131.18
	runtime(s)	0.81	0.78	0.24	3.57	0.40
energy_hotel_frame8frame64	UB	-4060.05	-4060.05	-4060.05	-4060.05	-4060.05
	LB	-4060.05	-4060.05	-4060.05	-4060.05	-4060.05
	runtime(s)	0.80	0.79	0.35	4.45	0.94
energy_hotel_frame8frame71	UB	-4021.39	-4021.39	-4021.39	-4021.39	-4021.39
	LB	-4021.39	-4021.39	-4021.39	-4021.39	-4021.39
	runtime(s)	1.12	1.10	0.79	10.87	0.58

Table 6: Per-instance results. UB means primal solution energy, LB dual lower bound and runtime(s) the runtime of the particular algorithm in seconds. **Bold numbers** indicate lowest primal energy, highest lower bound and smallest runtime.

Instance		AMP	AMCF	GM	HBP	DD
energy_hotel_frame8frame78	UB	-3931.28	-3931.28	-3931.28	-3931.28	-3931.28
	LB	-3931.28	-3931.28	-3931.28	-3931.28	-3931.28
	runtime(s)	2.42	3.03	28.11	28.70	1.78
energy_hotel_frame8frame85	UB	-3877.57	-3877.57	-3877.57	-3877.57	-3877.57
	LB	-3877.57	-3877.57	-3877.57	-3877.57	-3877.57
	runtime(s)	3.73	6.54	43.07	140.69	2.55
energy_hotel_frame8frame92	UB	-3802.05	-3802.05	-3802.05	-3802.05	-3802.05
	LB	-3802.05	-3802.05	-3802.05	-3816.65	-3802.05
	runtime(s)	14.01	16.47	105.07	190.19	2.82
energy_hotel_frame8frame99	UB	-3762.66	-3762.66	-3654.00	-3337.29	-3762.66
	LB	-3762.66	-3762.66	-3790.14	-3815.80	-3762.66
	runtime(s)	54.33	39.25	294.09	191.97	4.86
energy_hotel_frame92frame99	UB	-4593.17	-4593.17	-4593.17	-4593.17	-4593.17
	LB	-4593.17	-4593.17	-4593.17	-4593.17	-4593.17
	runtime(s)	0.15	0.14	0.02	0.31	0.06
house						
energy_house_frame10frame100	UB	-3720.20	-3720.20	-3720.20	-3720.20	-3720.20
	LB	-3720.20	-3720.20	-3720.20	-3720.20	-3720.20
	runtime(s)	7.25	6.19	2.75	21.87	2.53
energy_house_frame10frame95	UB	-3809.38	-3809.38	-3809.38	-3809.38	-3809.38
	LB	-3809.38	-3809.38	-3809.38	-3809.38	-3809.38
	runtime(s)	2.08	2.69	1.01	8.79	1.98
energy_house_frame10frame96	UB	-3786.86	-3786.86	-3786.86	-3786.86	-3786.86
	LB	-3786.86	-3786.86	-3786.86	-3786.86	-3786.86
	runtime(s)	2.08	2.36	1.01	9.27	2.66
energy_house_frame10frame97	UB	-3748.34	-3748.34	-3748.34	-3748.34	-3748.34
	LB	-3748.34	-3748.34	-3748.34	-3748.34	-3748.34
	runtime(s)	2.41	3.35	1.23	16.16	2.48
energy_house_frame10frame98	UB	-3766.04	-3766.04	-3766.04	-3766.04	-3766.04
	LB	-3766.04	-3766.04	-3766.04	-3766.04	-3766.04
	runtime(s)	2.73	3.33	1.01	9.16	2.07
energy_house_frame10frame99	UB	-3728.46	-3728.46	-3728.46	-3728.46	-3728.46
	LB	-3728.46	-3728.46	-3728.46	-3728.46	-3728.46
	runtime(s)	3.38	3.32	1.55	31.91	3.25
energy_house_frame11frame100	UB	-3739.39	-3739.39	-3739.39	-3739.39	-3739.39
	LB	-3739.39	-3739.39	-3739.39	-3739.39	-3739.39
	runtime(s)	4.02	4.91	1.88	16.19	2.54
energy_house_frame11frame101	UB	-3748.61	-3748.61	-3748.61	-3748.61	-3748.61
	LB	-3748.61	-3748.61	-3748.61	-3748.61	-3748.61
	runtime(s)	2.44	3.01	1.34	12.87	2.90
energy_house_frame11frame96	UB	-3809.96	-3809.96	-3809.96	-3809.96	-3809.96
	LB	-3809.96	-3809.96	-3809.96	-3809.96	-3809.96
	runtime(s)	1.76	2.37	1.12	8.69	2.19
energy_house_frame11frame97	UB	-3748.31	-3748.31	-3748.31	-3748.31	-3748.31
	LB	-3748.31	-3748.31	-3748.31	-3748.31	-3748.31
	runtime(s)	2.41	3.00	1.12	12.91	1.63
energy_house_frame11frame98	UB	-3781.51	-3781.51	-3781.51	-3781.51	-3781.51
	LB	-3781.51	-3781.51	-3781.51	-3781.51	-3781.51
	runtime(s)	2.42	3.32	0.79	6.22	2.27
energy_house_frame11frame99	UB	-3736.47	-3736.47	-3736.47	-3736.47	-3736.47
	LB	-3736.47	-3736.47	-3736.47	-3736.47	-3736.47
	runtime(s)	2.73	3.32	1.22	18.07	2.54
energy_house_frame12frame100	UB	-3768.48	-3768.48	-3768.48	-3768.48	-3768.48
	LB	-3768.48	-3768.48	-3768.48	-3768.48	-3768.48
	runtime(s)	6.60	5.53	5.37	24.18	2.38
energy_house_frame12frame101	UB	-3775.50	-3775.50	-3775.50	-3775.50	-3775.50
	LB	-3775.50	-3775.50	-3775.50	-3775.50	-3775.50
	runtime(s)	4.67	3.62	1.77	28.85	2.71
energy_house_frame12frame102	UB	-3783.42	-3783.42	-3783.42	-3783.42	-3783.42
	LB	-3783.42	-3783.42	-3783.42	-3783.42	-3783.42
	runtime(s)	3.37	3.01	1.01	8.71	2.62

Table 6: Per-instance results. UB means primal solution energy, LB dual lower bound and runtime(s) the runtime of the particular algorithm in seconds. **Bold numbers** indicate lowest primal energy, highest lower bound and smallest runtime.

Instance		AMP	AMCF	GM	HBP	DD
energy_house_frame12frame97	UB	-3780.35	-3780.35	-3780.35	-3780.35	-3780.35
	LB	-3780.35	-3780.35	-3780.35	-3780.35	-3780.35
	runtime(s)	3.05	2.99	1.45	15.89	2.68
energy_house_frame12frame98	UB	-3807.50	-3807.50	-3807.50	-3807.50	-3807.50
	LB	-3807.50	-3807.50	-3807.50	-3807.50	-3807.50
	runtime(s)	3.06	3.02	1.11	9.30	1.76
energy_house_frame12frame99	UB	-3766.79	-3766.79	-3766.79	-3766.79	-3766.79
	LB	-3766.79	-3766.79	-3766.79	-3766.79	-3766.79
	runtime(s)	4.34	3.31	1.89	28.90	2.49
energy_house_frame13frame100	UB	-3749.52	-3749.52	-3749.52	-3749.52	-3749.52
	LB	-3749.52	-3749.52	-3749.52	-3749.52	-3749.52
	runtime(s)	4.34	4.26	2.66	46.79	2.44
energy_house_frame13frame101	UB	-3773.17	-3773.17	-3773.17	-3773.17	-3773.17
	LB	-3773.17	-3773.17	-3773.17	-3773.17	-3773.17
	runtime(s)	2.73	2.35	1.33	18.01	2.78
energy_house_frame13frame102	UB	-3775.96	-3775.96	-3775.96	-3775.96	-3775.96
	LB	-3775.96	-3775.96	-3775.96	-3775.96	-3775.96
	runtime(s)	2.42	2.35	1.01	5.60	2.62
energy_house_frame13frame103	UB	-3749.50	-3749.50	-3749.50	-3749.50	-3749.50
	LB	-3749.50	-3749.50	-3749.50	-3749.50	-3749.50
	runtime(s)	2.41	2.68	1.12	17.98	2.61
energy_house_frame13frame98	UB	-3798.88	-3798.88	-3798.88	-3798.88	-3798.88
	LB	-3798.88	-3798.88	-3798.88	-3798.88	-3798.88
	runtime(s)	2.08	2.68	1.01	6.64	2.45
energy_house_frame13frame99	UB	-3754.97	-3754.97	-3754.97	-3754.97	-3754.97
	LB	-3754.97	-3754.97	-3754.97	-3754.97	-3754.97
	runtime(s)	2.73	2.36	1.45	13.06	2.36
energy_house_frame14frame100	UB	-3785.20	-3785.20	-3785.20	-3785.20	-3785.20
	LB	-3785.20	-3785.20	-3785.20	-3785.20	-3785.20
	runtime(s)	5.63	4.25	6.35	31.82	2.92
energy_house_frame14frame101	UB	-3796.54	-3796.54	-3796.54	-3796.54	-3796.54
	LB	-3796.54	-3796.54	-3796.54	-3796.54	-3796.54
	runtime(s)	4.01	3.32	1.55	12.88	2.68
energy_house_frame14frame102	UB	-3806.95	-3806.95	-3806.95	-3806.95	-3806.95
	LB	-3806.95	-3806.95	-3806.95	-3806.95	-3806.95
	runtime(s)	2.73	2.67	1.22	8.27	2.56
energy_house_frame14frame103	UB	-3769.12	-3769.12	-3769.12	-3769.12	-3769.12
	LB	-3769.12	-3769.12	-3769.12	-3769.12	-3769.12
	runtime(s)	3.06	2.99	1.35	12.88	3.29
energy_house_frame14frame104	UB	-3761.22	-3761.22	-3761.22	-3761.22	-3761.22
	LB	-3761.22	-3761.22	-3761.22	-3761.22	-3761.22
	runtime(s)	4.99	4.57	6.66	37.48	2.92
energy_house_frame14frame99	UB	-3788.42	-3788.42	-3788.42	-3788.42	-3788.42
	LB	-3788.42	-3788.42	-3788.42	-3788.42	-3788.42
	runtime(s)	3.69	2.68	1.55	14.36	2.65
energy_house_frame15frame100	UB	-3784.87	-3784.87	-3784.87	-3784.87	-3784.87
	LB	-3784.87	-3784.87	-3784.87	-3784.87	-3784.87
	runtime(s)	3.05	3.31	1.23	16.19	2.27
energy_house_frame15frame101	UB	-3796.63	-3796.63	-3796.63	-3796.63	-3796.63
	LB	-3796.63	-3796.63	-3796.63	-3796.63	-3796.63
	runtime(s)	2.09	2.36	0.79	8.64	2.19
energy_house_frame15frame102	UB	-3798.54	-3798.54	-3798.54	-3798.54	-3798.54
	LB	-3798.54	-3798.54	-3798.54	-3798.54	-3798.54
	runtime(s)	2.09	2.04	0.57	5.66	2.03
energy_house_frame15frame103	UB	-3774.99	-3774.99	-3774.99	-3774.99	-3774.99
	LB	-3774.99	-3774.99	-3774.99	-3774.99	-3774.99
	runtime(s)	2.09	2.69	0.68	12.87	1.98
energy_house_frame15frame104	UB	-3762.50	-3762.50	-3762.50	-3762.50	-3762.50
	LB	-3762.50	-3762.50	-3762.50	-3762.50	-3762.50
	runtime(s)	3.05	3.62	1.67	17.90	3.07

Table 6: Per-instance results. UB means primal solution energy, LB dual lower bound and runtime(s) the runtime of the particular algorithm in seconds. **Bold numbers** indicate lowest primal energy, highest lower bound and smallest runtime.

Instance		AMP	AMCF	GM	HBP	DD
energy_house_frame15frame105	UB	-3745.16	-3745.16	-3745.16	-3745.16	-3745.16
	LB	-3745.16	-3745.16	-3745.16	-3745.16	-3745.16
	runtime(s)	2.41	3.62	0.90	9.21	2.09
energy_house_frame16frame101	UB	-3804.83	-3804.83	-3804.83	-3804.83	-3804.83
	LB	-3804.83	-3804.83	-3804.83	-3804.83	-3804.83
	runtime(s)	1.76	2.36	4.38	9.34	1.44
energy_house_frame16frame102	UB	-3815.11	-3815.11	-3815.11	-3815.11	-3815.11
	LB	-3815.11	-3815.11	-3815.11	-3815.11	-3815.11
	runtime(s)	1.77	1.73	0.68	6.15	1.40
energy_house_frame16frame103	UB	-3787.76	-3787.76	-3787.76	-3787.76	-3787.76
	LB	-3787.76	-3787.76	-3787.76	-3787.76	-3787.76
	runtime(s)	1.76	2.05	0.78	8.06	2.01
energy_house_frame16frame104	UB	-3774.51	-3774.51	-3774.51	-3774.51	-3774.51
	LB	-3774.51	-3774.51	-3774.51	-3774.51	-3774.51
	runtime(s)	2.41	2.99	6.07	18.36	1.83
energy_house_frame16frame105	UB	-3752.97	-3752.97	-3752.97	-3752.97	-3752.97
	LB	-3752.97	-3752.97	-3752.97	-3752.97	-3752.97
	runtime(s)	2.09	2.99	4.81	12.93	1.62
energy_house_frame17frame102	UB	-3820.84	-3820.84	-3820.84	-3820.84	-3820.84
	LB	-3820.84	-3820.84	-3820.84	-3820.84	-3820.84
	runtime(s)	1.44	1.73	0.46	3.32	1.34
energy_house_frame17frame103	UB	-3799.56	-3799.56	-3799.56	-3799.56	-3799.56
	LB	-3799.56	-3799.56	-3799.56	-3799.56	-3799.56
	runtime(s)	1.77	2.04	0.57	6.48	1.97
energy_house_frame17frame104	UB	-3774.96	-3774.96	-3774.96	-3774.96	-3774.96
	LB	-3774.96	-3774.96	-3774.96	-3774.96	-3774.96
	runtime(s)	2.09	2.36	1.11	12.83	1.93
energy_house_frame17frame105	UB	-3756.27	-3756.27	-3756.27	-3756.27	-3756.27
	LB	-3756.27	-3756.27	-3756.27	-3756.27	-3756.27
	runtime(s)	2.41	2.99	0.79	9.35	2.97
energy_house_frame18frame103	UB	-3821.69	-3821.69	-3821.69	-3821.69	-3821.69
	LB	-3821.69	-3821.69	-3821.69	-3821.69	-3821.69
	runtime(s)	2.41	2.36	0.90	12.89	1.81
energy_house_frame18frame104	UB	-3794.07	-3794.07	-3794.07	-3794.07	-3794.07
	LB	-3794.07	-3794.07	-3794.07	-3794.07	-3794.07
	runtime(s)	3.38	3.64	2.10	24.01	2.99
energy_house_frame18frame105	UB	-3777.39	-3777.39	-3777.39	-3777.39	-3777.39
	LB	-3777.39	-3777.39	-3777.39	-3777.39	-3777.39
	runtime(s)	2.72	2.99	1.12	17.78	2.39
energy_house_frame19frame104	UB	-3799.19	-3799.19	-3799.19	-3799.19	-3799.19
	LB	-3799.19	-3799.19	-3799.19	-3799.19	-3799.19
	runtime(s)	3.37	3.50	2.21	12.89	3.46
energy_house_frame19frame105	UB	-3767.45	-3767.45	-3767.45	-3767.45	-3767.45
	LB	-3767.45	-3767.45	-3767.45	-3767.45	-3767.45
	runtime(s)	3.08	2.68	1.45	24.10	2.33
energy_house_frame1frame86	UB	-3833.13	-3833.13	-3833.13	-3833.13	-3833.13
	LB	-3833.13	-3833.13	-3833.13	-3833.13	-3833.13
	runtime(s)	1.44	1.41	0.35	2.72	1.74
energy_house_frame1frame87	UB	-3808.48	-3808.48	-3808.48	-3808.48	-3808.48
	LB	-3808.48	-3808.48	-3808.48	-3808.48	-3808.48
	runtime(s)	1.44	1.10	0.46	4.38	1.60
energy_house_frame1frame88	UB	-3758.22	-3758.22	-3758.22	-3758.22	-3758.22
	LB	-3758.22	-3758.22	-3758.22	-3758.22	-3758.22
	runtime(s)	1.76	1.92	0.68	7.04	2.06
energy_house_frame1frame89	UB	-3776.74	-3776.74	-3776.74	-3776.74	-3776.74
	LB	-3776.74	-3776.74	-3776.74	-3776.74	-3776.74
	runtime(s)	2.09	2.04	0.68	8.37	2.31
energy_house_frame1frame90	UB	-3710.78	-3710.78	-3710.78	-3710.78	-3710.78
	LB	-3710.78	-3710.78	-3710.78	-3710.78	-3710.78
	runtime(s)	2.40	2.04	1.78	56.87	3.92

Table 6: Per-instance results. UB means primal solution energy, LB dual lower bound and runtime(s) the runtime of the particular algorithm in seconds. **Bold numbers** indicate lowest primal energy, highest lower bound and smallest runtime.

Instance		AMP	AMCF	GM	HBP	DD
energy_house_frame1frame91	UB	-3761.74	-3761.74	-3761.74	-3761.74	-3761.74
	LB	-3761.74	-3761.74	-3761.74	-3761.74	-3761.74
	runtime(s)	2.08	2.36	0.79	5.97	2.66
energy_house_frame20frame105	UB	-3772.12	-3772.12	-3772.12	-3772.12	-3772.12
	LB	-3772.12	-3772.12	-3772.12	-3772.12	-3772.12
	runtime(s)	2.40	2.68	0.90	9.59	2.50
energy_house_frame2frame87	UB	-3837.08	-3837.08	-3837.08	-3837.08	-3837.08
	LB	-3837.08	-3837.08	-3837.08	-3837.08	-3837.08
	runtime(s)	1.44	1.10	0.35	2.01	1.32
energy_house_frame2frame88	UB	-3807.53	-3807.53	-3807.53	-3807.53	-3807.53
	LB	-3807.53	-3807.53	-3807.53	-3807.53	-3807.53
	runtime(s)	1.44	2.04	0.46	5.97	1.69
energy_house_frame2frame89	UB	-3807.82	-3807.82	-3807.82	-3807.82	-3807.82
	LB	-3807.82	-3807.82	-3807.82	-3807.82	-3807.82
	runtime(s)	1.76	2.06	0.68	7.47	2.05
energy_house_frame2frame90	UB	-3766.17	-3766.17	-3766.17	-3766.17	-3766.17
	LB	-3766.17	-3766.17	-3766.17	-3766.17	-3766.17
	runtime(s)	2.08	2.36	1.00	16.25	1.72
energy_house_frame2frame91	UB	-3791.43	-3791.43	-3791.43	-3791.43	-3791.43
	LB	-3791.43	-3791.43	-3791.43	-3791.43	-3791.43
	runtime(s)	1.76	2.67	0.81	6.42	2.12
energy_house_frame2frame92	UB	-3753.22	-3753.22	-3753.22	-3753.22	-3753.22
	LB	-3753.22	-3753.22	-3753.22	-3753.22	-3753.22
	runtime(s)	2.73	3.31	1.01	16.13	1.56
energy_house_frame3frame88	UB	-3808.14	-3808.14	-3808.14	-3808.14	-3808.14
	LB	-3808.14	-3808.14	-3808.14	-3808.14	-3808.14
	runtime(s)	1.76	2.04	0.68	7.18	1.95
energy_house_frame3frame89	UB	-3815.26	-3815.26	-3815.26	-3815.26	-3815.26
	LB	-3815.26	-3815.26	-3815.26	-3815.26	-3815.26
	runtime(s)	2.40	2.36	0.68	7.58	2.06
energy_house_frame3frame90	UB	-3761.33	-3761.33	-3761.33	-3761.33	-3761.33
	LB	-3761.33	-3761.33	-3761.33	-3761.33	-3761.33
	runtime(s)	2.41	2.68	1.22	24.28	2.13
energy_house_frame3frame91	UB	-3808.47	-3808.47	-3808.47	-3808.47	-3808.47
	LB	-3808.47	-3808.47	-3808.47	-3808.47	-3808.47
	runtime(s)	2.08	2.99	0.79	6.61	1.71
energy_house_frame3frame92	UB	-3769.27	-3769.27	-3769.27	-3769.27	-3769.27
	LB	-3769.27	-3769.27	-3769.27	-3769.27	-3769.27
	runtime(s)	3.05	3.64	1.12	11.88	1.87
energy_house_frame3frame93	UB	-3763.96	-3763.96	-3763.96	-3763.96	-3763.96
	LB	-3763.96	-3763.96	-3763.96	-3763.96	-3763.96
	runtime(s)	3.38	3.63	1.56	12.86	3.31
energy_house_frame4frame89	UB	-3826.43	-3826.43	-3826.43	-3826.43	-3826.43
	LB	-3826.43	-3826.43	-3826.43	-3826.43	-3826.43
	runtime(s)	2.08	2.36	1.00	9.00	2.12
energy_house_frame4frame90	UB	-3772.11	-3772.11	-3772.11	-3772.11	-3772.11
	LB	-3772.11	-3772.11	-3772.11	-3772.11	-3772.11
	runtime(s)	2.08	2.36	1.55	12.91	2.29
energy_house_frame4frame91	UB	-3813.78	-3813.78	-3813.78	-3813.78	-3813.78
	LB	-3813.78	-3813.78	-3813.78	-3813.78	-3813.78
	runtime(s)	2.08	2.68	1.00	8.87	2.99
energy_house_frame4frame92	UB	-3769.21	-3769.21	-3769.21	-3769.21	-3769.21
	LB	-3769.21	-3769.21	-3769.21	-3769.21	-3769.21
	runtime(s)	2.73	3.95	1.88	12.89	3.15
energy_house_frame4frame93	UB	-3770.02	-3770.02	-3770.02	-3770.02	-3770.02
	LB	-3770.02	-3770.02	-3770.02	-3770.02	-3770.02
	runtime(s)	3.05	3.63	2.21	12.91	1.78
energy_house_frame4frame94	UB	-3781.42	-3781.42	-3781.42	-3781.42	-3781.42
	LB	-3781.42	-3781.42	-3781.42	-3781.42	-3781.42
	runtime(s)	2.41	3.33	1.66	12.92	2.72

Table 6: Per-instance results. UB means primal solution energy, LB dual lower bound and runtime(s) the runtime of the particular algorithm in seconds. **Bold numbers** indicate lowest primal energy, highest lower bound and smallest runtime.

Instance		AMP	AMCF	GM	HBP	DD
energy_house_frame5frame90	UB	-3757.48	-3757.48	-3757.48	-3757.48	-3757.48
	LB	-3757.48	-3757.48	-3757.48	-3757.48	-3757.48
	runtime(s)	2.40	2.04	1.44	37.31	1.94
energy_house_frame5frame91	UB	-3801.56	-3801.56	-3801.56	-3801.56	-3801.56
	LB	-3801.56	-3801.56	-3801.56	-3801.56	-3801.56
	runtime(s)	2.09	2.36	0.90	8.11	1.78
energy_house_frame5frame92	UB	-3759.36	-3759.36	-3759.36	-3759.36	-3759.36
	LB	-3759.36	-3759.36	-3759.36	-3759.36	-3759.36
	runtime(s)	2.74	2.68	1.34	18.41	2.52
energy_house_frame5frame93	UB	-3764.02	-3764.02	-3764.02	-3764.02	-3764.02
	LB	-3764.02	-3764.02	-3764.02	-3764.02	-3764.02
	runtime(s)	3.05	3.00	1.66	12.93	2.15
energy_house_frame5frame94	UB	-3765.23	-3765.23	-3765.23	-3765.23	-3765.23
	LB	-3765.23	-3765.23	-3765.23	-3765.23	-3765.23
	runtime(s)	2.73	2.68	1.56	12.87	2.06
energy_house_frame5frame95	UB	-3773.62	-3773.62	-3773.62	-3773.62	-3773.62
	LB	-3773.62	-3773.62	-3773.62	-3773.62	-3773.62
	runtime(s)	3.37	3.01	1.55	11.99	1.72
energy_house_frame6frame91	UB	-3824.94	-3824.94	-3824.94	-3824.94	-3824.94
	LB	-3824.94	-3824.94	-3824.94	-3824.94	-3824.94
	runtime(s)	1.77	2.36	1.11	7.96	2.60
energy_house_frame6frame92	UB	-3779.95	-3779.95	-3779.95	-3779.95	-3779.95
	LB	-3779.95	-3779.95	-3779.95	-3779.95	-3779.95
	runtime(s)	3.05	3.32	1.56	12.95	2.34
energy_house_frame6frame93	UB	-3780.40	-3780.40	-3780.40	-3780.40	-3780.40
	LB	-3780.40	-3780.40	-3780.40	-3780.40	-3780.40
	runtime(s)	3.69	3.00	2.32	16.16	3.18
energy_house_frame6frame94	UB	-3787.27	-3787.27	-3787.27	-3787.27	-3787.27
	LB	-3787.27	-3787.27	-3787.27	-3787.27	-3787.27
	runtime(s)	2.09	2.78	1.77	17.81	2.27
energy_house_frame6frame95	UB	-3794.15	-3794.15	-3794.15	-3794.15	-3794.15
	LB	-3794.15	-3794.15	-3794.15	-3794.15	-3794.15
	runtime(s)	3.37	3.31	1.78	11.84	2.06
energy_house_frame6frame96	UB	-3770.58	-3770.58	-3770.58	-3770.58	-3770.58
	LB	-3770.58	-3770.58	-3770.58	-3770.58	-3770.58
	runtime(s)	3.70	3.00	2.03	28.82	2.20
energy_house_frame7frame92	UB	-3764.47	-3764.47	-3764.47	-3764.47	-3764.47
	LB	-3764.47	-3764.47	-3764.47	-3764.47	-3764.47
	runtime(s)	3.05	3.31	1.66	16.11	3.04
energy_house_frame7frame93	UB	-3768.98	-3768.98	-3768.98	-3768.98	-3768.98
	LB	-3768.98	-3768.98	-3768.98	-3768.98	-3768.98
	runtime(s)	3.38	3.63	2.09	28.11	2.29
energy_house_frame7frame94	UB	-3771.82	-3771.82	-3771.82	-3771.82	-3771.82
	LB	-3771.82	-3771.82	-3771.82	-3771.82	-3771.82
	runtime(s)	2.41	3.08	1.66	12.86	2.32
energy_house_frame7frame95	UB	-3780.66	-3780.66	-3780.66	-3780.66	-3780.66
	LB	-3780.66	-3780.66	-3780.66	-3780.66	-3780.66
	runtime(s)	3.38	3.19	1.67	16.48	2.16
energy_house_frame7frame96	UB	-3755.85	-3755.85	-3755.85	-3755.85	-3755.85
	LB	-3755.85	-3755.85	-3755.85	-3755.85	-3755.85
	runtime(s)	4.03	3.00	1.99	17.88	2.67
energy_house_frame7frame97	UB	-3712.31	-3712.31	-3712.31	-3712.31	-3712.31
	LB	-3712.31	-3712.31	-3712.31	-3712.31	-3712.31
	runtime(s)	3.37	3.63	2.54	17.82	4.17
energy_house_frame8frame93	UB	-3787.12	-3787.12	-3787.12	-3787.12	-3787.12
	LB	-3787.12	-3787.12	-3787.12	-3787.12	-3787.12
	runtime(s)	3.38	3.96	2.32	21.87	2.78
energy_house_frame8frame94	UB	-3794.54	-3794.54	-3794.54	-3794.54	-3794.54
	LB	-3794.54	-3794.54	-3794.54	-3794.54	-3794.54
	runtime(s)	3.05	3.96	2.32	16.17	2.13

Table 6: Per-instance results. UB means primal solution energy, LB dual lower bound and runtime(s) the runtime of the particular algorithm in seconds. **Bold numbers** indicate lowest primal energy, highest lower bound and smallest runtime.

Instance		AMP	AMCF	GM	HBP	DD
energy_house_frame8frame95	UB	-3800.01	-3800.01	-3800.01	-3800.01	-3800.01
	LB	-3800.01	-3800.01	-3800.01	-3800.01	-3800.01
	runtime(s)	4.02	4.58	2.54	28.82	2.29
energy_house_frame8frame96	UB	-3781.83	-3781.83	-3781.83	-3781.83	-3781.83
	LB	-3781.83	-3781.83	-3781.83	-3781.83	-3781.83
	runtime(s)	4.67	5.22	3.51	16.45	3.14
energy_house_frame8frame97	UB	-3738.27	-3738.27	-3738.27	-3738.27	-3738.27
	LB	-3738.27	-3738.27	-3738.27	-3738.27	-3738.27
	runtime(s)	4.35	5.24	6.04	16.24	2.53
energy_house_frame8frame98	UB	-3759.21	-3759.21	-3759.21	-3759.21	-3759.21
	LB	-3759.21	-3759.21	-3759.21	-3759.21	-3759.21
	runtime(s)	4.99	6.17	2.64	21.78	2.29
energy_house_frame9frame94	UB	-3806.80	-3806.80	-3806.80	-3806.80	-3806.80
	LB	-3806.80	-3806.80	-3806.80	-3806.80	-3806.80
	runtime(s)	1.76	2.38	0.90	8.27	2.47
energy_house_frame9frame95	UB	-3811.22	-3811.22	-3811.22	-3811.22	-3811.22
	LB	-3811.22	-3811.22	-3811.22	-3811.22	-3811.22
	runtime(s)	2.09	2.70	1.01	9.27	2.05
energy_house_frame9frame96	UB	-3797.97	-3797.97	-3797.97	-3797.97	-3797.97
	LB	-3797.97	-3797.97	-3797.97	-3797.97	-3797.97
	runtime(s)	2.08	2.36	1.01	9.29	2.55
energy_house_frame9frame97	UB	-3748.62	-3748.62	-3748.62	-3748.62	-3748.62
	LB	-3748.62	-3748.62	-3748.62	-3748.62	-3748.62
	runtime(s)	2.09	3.00	1.12	16.17	2.98
energy_house_frame9frame98	UB	-3770.40	-3770.40	-3770.40	-3770.40	-3770.40
	LB	-3770.40	-3770.40	-3770.40	-3770.40	-3770.40
	runtime(s)	2.40	3.63	0.90	9.01	2.53
energy_house_frame9frame99	UB	-3726.83	-3726.83	-3726.83	-3726.83	-3726.83
	LB	-3726.83	-3726.83	-3726.83	-3726.83	-3726.83
	runtime(s)	3.05	3.63	1.55	18.35	2.66
Hassan						
board_torresani	UB	-2262.66	-2262.66	-2262.66	-2236.29	-2262.66
	LB	-2262.66	-2262.66	-2262.66	-2263.27	-2262.66
	runtime(s)	2.58	2.57	3.24	54.25	3.23
books_torresani	UB	-4124.40	-4105.15	-4091.01	-2991.97	-4135.27
	LB	-4153.99	-4163.54	-4200.55	-4232.44	-4137.02
	runtime(s)	459.31	480.94	464.50	444.69	45.79
hammer_torresani	UB	-2097.78	-2092.19	-1901.89	-903.62	-2097.78
	LB	-2108.19	-2119.02	-2138.38	-2169.68	-2097.78
	runtime(s)	305.12	312.79	302.55	218.99	17.60
party_torresani	UB	-3629.91	-3629.91	-3629.91	-3005.44	-3629.91
	LB	-3643.57	-3639.94	-3662.25	-3671.11	-3629.91
	runtime(s)	292.56	334.90	258.55	220.28	16.76
table_torresani	UB	-3278.81	-3261.48	-3277.33	-2328.58	-3288.51
	LB	-3305.51	-3312.67	-3301.49	-3369.81	-3288.51
	runtime(s)	245.39	249.47	255.71	151.99	7.68
walking_torresani	UB	-1625.85	-1625.55	-1625.85	-1625.85	-1625.85
	LB	-1625.85	-1625.85	-1625.85	-1627.00	-1625.85
	runtime(s)	4.35	8.53	13.78	76.41	16.47
car						
car1	UB	-34.88	-34.88	-34.88	-34.88	-34.88
	LB	-34.88	-34.88	-34.88	-34.88	-34.88
	runtime(s)	1.48	1.94	20.52	5.13	8.56
car10	UB	-57.62	-57.62	-57.62	-57.62	-46.19
	LB	-57.62	-57.62	-57.62	-57.62	-58.66
	runtime(s)	5.38	11.43	13.41	17.78	60.73
car11	UB	-63.06	-63.06	-63.06	-63.06	-63.06
	LB	-63.06	-63.06	-63.06	-63.06	-63.06
	runtime(s)	2.20	5.58	4.19	13.39	18.39

Table 6: Per-instance results. UB means primal solution energy, LB dual lower bound and runtime(s) the runtime of the particular algorithm in seconds. **Bold numbers** indicate lowest primal energy, highest lower bound and smallest runtime.

Instance		AMP	AMCF	GM	HBP	DD
car12	UB	-57.36	-55.88	-57.36	-57.36	-37.26
	LB	-57.36	-57.36	-57.36	-57.36	-59.47
	runtime(s)	4.92	8.87	36.30	25.84	55.86
car13	UB	-72.15	-72.15	-72.15	-72.15	-58.93
	LB	-72.15	-72.15	-72.15	-72.15	-73.94
	runtime(s)	15.10	8.94	22.94	49.63	54.78
car14	UB	-97.96	-97.70	-97.52	-97.77	-60.72
	LB	-98.15	-98.02	-98.32	-98.77	-120.95
	runtime(s)	3614.22	3615.70	3610.92	2833.06	114.21
car15	UB	-66.89	-66.89	-66.89	-66.89	-66.89
	LB	-66.89	-66.89	-66.89	-66.89	-67.17
	runtime(s)	2.61	6.55	13.85	35.16	37.40
car16	UB	-68.21	-68.21	-68.21	-68.21	-68.21
	LB	-68.21	-68.21	-68.21	-68.21	-68.21
	runtime(s)	6.25	5.13	13.07	34.80	55.55
car17	UB	-57.09	-57.09	-57.09	-57.09	-57.09
	LB	-57.09	-57.09	-57.09	-57.09	-57.09
	runtime(s)	4.78	12.26	94.38	10.31	76.36
car18	UB	-92.18	-92.18	-92.18	-92.18	-92.18
	LB	-92.18	-92.18	-92.18	-92.18	-92.18
	runtime(s)	15.08	18.47	29.70	114.35	105.84
car19	UB	-115.11	-115.11	-115.11	-115.11	-80.58
	LB	-115.11	-115.11	-115.11	-115.11	-120.55
	runtime(s)	82.44	109.77	132.66	1604.25	349.92
car2	UB	-48.85	-48.85	-48.85	-48.85	-48.85
	LB	-48.85	-48.85	-48.85	-48.85	-48.85
	runtime(s)	93.72	138.94	215.20	67.66	32.59
car20	UB	-106.69	-106.69	-106.69	-106.69	-94.48
	LB	-106.69	-106.69	-106.69	-106.69	-107.46
	runtime(s)	20.42	20.15	17.38	56.33	163.27
car21	UB	-94.55	-94.55	-94.55	-94.55	-94.55
	LB	-94.55	-94.55	-94.55	-94.55	-94.55
	runtime(s)	63.10	57.48	86.08	302.83	145.09
car22	UB	-55.58	-55.58	-55.58	-55.58	-55.58
	LB	-55.58	-55.58	-55.58	-55.58	-55.58
	runtime(s)	1.69	1.47	1.72	3.43	27.37
car23	UB	-70.20	-64.70	-70.20	-70.20	-47.36
	LB	-70.20	-70.20	-70.20	-70.20	-75.29
	runtime(s)	29.92	61.96	41.27	165.15	63.77
car24	UB	-64.58	-64.58	-64.58	-64.58	-55.40
	LB	-64.58	-64.58	-64.58	-64.58	-66.32
	runtime(s)	146.68	139.16	392.19	1158.90	48.65
car25	UB	-34.19	-34.19	-34.19	-34.19	-34.19
	LB	-34.19	-34.19	-34.19	-34.19	-34.19
	runtime(s)	2.01	4.20	16.96	8.67	17.74
car26	UB	-59.92	-56.97	-59.92	-59.92	-59.92
	LB	-59.92	-59.92	-59.92	-59.92	-59.98
	runtime(s)	27.64	17.56	53.86	63.54	56.01
car27	UB	-67.95	-67.95	-67.95	-67.95	-67.95
	LB	-67.95	-67.95	-67.95	-67.95	-67.95
	runtime(s)	13.41	25.23	28.80	81.25	54.91
car28	UB	-74.92	-74.15	-71.70	-74.73	-38.23
	LB	-77.06	-76.90	-77.96	-78.18	-113.45
	runtime(s)	3630.63	3629.86	3621.74	2392.52	133.85
car29	UB	-84.31	-81.76	-84.31	-84.31	-48.33
	LB	-84.31	-84.40	-84.78	-84.73	-93.55
	runtime(s)	3300.78	3255.77	2735.09	2103.61	111.87
car3	UB	-86.55	-86.55	-86.55	-86.55	-54.10
	LB	-86.55	-86.55	-86.61	-86.64	-99.20
	runtime(s)	1231.46	340.40	3612.01	2044.42	158.59

Table 6: Per-instance results. UB means primal solution energy, LB dual lower bound and runtime(s) the runtime of the particular algorithm in seconds. **Bold numbers** indicate lowest primal energy, highest lower bound and smallest runtime.

Instance		AMP	AMCF	GM	HBP	DD
car30	UB	-58.85	-58.85	-58.85	-58.85	-52.63
	LB	-58.85	-58.85	-58.85	-58.85	-60.48
	runtime(s)	41.33	87.70	207.34	62.66	40.45
car4	UB	-51.17	-49.20	-49.38	-50.64	-33.85
	LB	-54.00	-54.36	-55.65	-55.69	-61.03
	runtime(s)	1539.03	1791.20	1689.60	720.89	77.74
car5	UB	-65.24	-63.51	-49.11	-66.25	-36.53
	LB	-68.15	-67.82	-69.00	-69.93	-83.96
	runtime(s)	2971.41	3614.74	3611.02	1389.58	121.17
car6	UB	-83.78	-83.10	-82.53	-82.52	-49.51
	LB	-83.88	-84.07	-84.14	-84.09	-90.13
	runtime(s)	2495.72	3477.40	2434.92	1403.09	238.56
car7	UB	-81.95	-81.95	-81.95	-81.95	-81.95
	LB	-81.95	-81.95	-81.95	-81.95	-81.95
	runtime(s)	14.95	13.28	14.09	46.10	43.76
car8	UB	-43.95	-41.22	-38.30	-44.00	-31.55
	LB	-44.42	-44.64	-46.72	-44.70	-52.72
	runtime(s)	326.15	480.89	526.10	356.44	28.05
car9	UB	-62.40	-62.40	-62.40	-62.40	-62.40
	LB	-62.40	-62.40	-62.40	-62.40	-62.40
	runtime(s)	1.37	0.22	0.26	1.55	10.20
motor						
motor1	UB	-94.18	-94.18	-94.18	-94.18	-46.36
	LB	-94.18	-94.18	-94.20	-94.23	-115.29
	runtime(s)	576.63	915.02	3625.61	3181.60	168.71
motor10	UB	-48.95	-48.95	-48.05	-48.95	-48.95
	LB	-48.95	-48.95	-48.95	-48.95	-48.95
	runtime(s)	0.98	0.87	3.68	7.00	10.45
motor11	UB	-50.16	-50.16	-50.16	-50.16	-50.16
	LB	-50.16	-50.16	-50.16	-50.16	-50.16
	runtime(s)	0.27	0.87	2.33	2.51	15.45
motor12	UB	-55.81	-55.81	-55.81	-55.81	-55.81
	LB	-55.81	-55.81	-55.81	-55.81	-55.81
	runtime(s)	5.08	5.65	14.42	21.04	22.16
motor13	UB	-36.35	-36.35	-36.35	-36.35	-36.35
	LB	-36.36	-36.35	-36.37	-36.35	-36.35
	runtime(s)	36.13	7.80	189.60	29.00	18.18
motor14	UB	-43.80	-41.49	-39.95	-43.47	-42.99
	LB	-44.35	-44.50	-45.21	-44.25	-44.77
	runtime(s)	492.28	453.56	670.01	393.21	57.10
motor15	UB	-32.30	-32.30	-32.30	-32.30	-27.19
	LB	-32.30	-32.30	-32.33	-32.30	-33.74
	runtime(s)	8.53	9.91	82.75	53.03	13.99
motor16	UB	-75.13	-75.13	-75.13	-75.13	-75.13
	LB	-75.13	-75.13	-75.13	-75.13	-75.13
	runtime(s)	33.22	19.71	19.04	114.67	123.36
motor17	UB	-84.38	-84.38	-84.38	-84.38	-83.99
	LB	-84.38	-84.38	-84.38	-84.41	-85.24
	runtime(s)	67.35	431.98	740.71	463.66	149.62
motor18	UB	-131.41	-131.41	-131.41	-131.41	-120.90
	LB	-131.41	-131.41	-131.41	-131.41	-131.86
	runtime(s)	63.01	46.24	44.81	424.98	365.89
motor19	UB	-75.29	-75.29	-75.29	-75.29	-75.29
	LB	-75.29	-75.29	-75.29	-75.29	-75.29
	runtime(s)	1.02	0.52	3.37	15.85	26.00
motor2	UB	-94.93	-94.93	-94.93	-94.93	-94.93
	LB	-94.93	-94.93	-94.93	-94.93	-94.93
	runtime(s)	9.53	1.00	0.92	32.43	57.88
motor20	UB	-82.10	-82.10	-82.10	-82.10	-79.77
	LB	-82.10	-82.10	-82.10	-82.18	-83.24
	runtime(s)	49.05	43.97	142.10	1155.51	145.62

Table 6: Per-instance results. UB means primal solution energy, LB dual lower bound and runtime(s) the runtime of the particular algorithm in seconds. **Bold numbers** indicate lowest primal energy, highest lower bound and smallest runtime.

Instance		AMP	AMCF	GM	HBP	DD
motor3	UB	-45.37	-45.37	-45.37	-45.37	-45.37
	LB	-45.37	-45.37	-45.37	-45.37	-45.37
	runtime(s)	1.12	2.59	19.42	5.99	9.11
motor4	UB	-27.50	-27.50	-27.50	-27.50	-27.50
	LB	-27.50	-27.50	-27.50	-27.50	-27.50
	runtime(s)	0.14	0.10	0.68	0.35	1.47
motor5	UB	-29.91	-29.91	-29.91	-29.91	-29.91
	LB	-29.91	-29.91	-29.91	-29.91	-29.91
	runtime(s)	0.04	0.02	0.12	0.12	1.05
motor6	UB	-51.51	-51.51	-51.51	-51.51	-51.51
	LB	-51.51	-51.51	-51.51	-51.51	-51.51
	runtime(s)	1.07	1.10	4.97	4.67	10.25
motor7	UB	-64.93	-64.93	-64.93	-64.93	-64.93
	LB	-64.93	-64.93	-64.93	-64.93	-64.93
	runtime(s)	6.37	3.32	21.00	21.14	52.96
motor8	UB	-79.71	-79.71	-79.71	-79.71	-79.71
	LB	-79.71	-79.71	-79.71	-79.71	-79.71
	runtime(s)	62.98	23.81	104.01	416.33	123.32
motor9	UB	-55.17	-55.17	-55.17	-55.17	-55.17
	LB	-55.17	-55.17	-55.17	-55.17	-55.17
	runtime(s)	1.57	3.31	14.15	9.02	11.89
worms						
C18G1_2L1_1-lowThresh-more-hyp.surf-16-03-11-1745	UB	-46310.40	-46070.09	inf	-45774.10	3396.88
	LB	-46310.87	-46339.35	-46366.44	-46334.22	-60642.07
	runtime(s)	207.40	593.32	194.37	289.49	1482.13
cnd1threeL1_1213061-lowThresh-more-hyp.surf-16-03-11-1745	UB	-49998.17	-49933.02	inf	-49979.79	-17409.57
	LB	-49998.17	-50000.88	-50003.44	-50002.52	-61715.51
	runtime(s)	14.33	570.65	22.99	204.90	1520.45
cnd1threeL1_1228061-lowThresh-more-hyp.surf-16-03-11-1745	UB	-50553.88	-50499.12	inf	-50528.66	-4741.50
	LB	-50553.88	-50555.70	-50576.63	-50567.57	-73855.69
	runtime(s)	69.22	771.24	59.42	404.33	1498.98
cnd1threeL1_1229061-lowThresh-more-hyp.surf-16-03-11-1745	UB	-49113.04	-49042.26	inf	-49024.03	-15101.58
	LB	-49119.90	-49133.09	-49177.09	-49143.20	-56976.00
	runtime(s)	208.76	484.13	66.55	207.18	1424.92
cnd1threeL1_1229062-lowThresh-more-hyp.surf-16-03-11-1745	UB	-49414.48	-49217.71	inf	-49159.46	-7233.37
	LB	-49427.52	-49446.99	-49452.32	-49441.35	-65090.03
	runtime(s)	249.81	615.87	141.67	280.93	1438.74
cnd1threeL1_1229063-lowThresh-more-hyp.surf-16-03-11-1745	UB	-50480.36	-50425.15	inf	-50427.31	-11408.78
	LB	-50480.36	-50492.21	-50499.47	-50489.68	-63543.89
	runtime(s)	133.26	637.62	39.45	296.92	1525.52
eft3RW10035L1_0125071-lowThresh-more-hyp.surf-16-03-11-1745	UB	-47038.08	-46898.52	inf	-46643.35	6389.96
	LB	-47075.02	-47113.47	-47120.22	-47110.52	-66393.07
	runtime(s)	264.14	374.47	614.80	361.89	1439.53
eft3RW10035L1_0125072-lowThresh-more-hyp.surf-16-03-11-1745	UB	-49243.13	-49128.56	inf	-49156.19	-10301.00
	LB	-49273.44	-49291.64	-49373.97	-49305.48	-61184.07
	runtime(s)	378.21	627.18	53.24	330.01	1415.11
eft3RW10035L1_0125073-lowThresh-more-hyp.surf-16-03-11-1745	UB	-45145.48	-44994.08	inf	-44960.05	4993.09
	LB	-45152.87	-45186.69	-45262.48	-45189.34	-55876.38
	runtime(s)	191.76	491.98	133.49	266.09	1391.44
egl5L1_0606074-lowThresh-more-hyp.surf-16-03-11-1745	UB	-42290.34	-41752.20	inf	-41510.29	25094.04
	LB	-42438.31	-42527.52	-42557.47	-42522.95	-62140.57
	runtime(s)	296.69	436.30	393.89	344.93	1447.15
elt3L1_0503071-lowThresh-more-hyp.surf-16-03-11-1745	UB	-48663.64	-48597.23	inf	-48553.52	-12597.30
	LB	-48665.33	-48682.63	-48685.64	-48683.32	-61435.00
	runtime(s)	286.57	514.79	165.60	298.29	1546.62
elt3L1_0503072-lowThresh-more-hyp.surf-16-03-11-1745	UB	-50403.27	-50323.34	inf	-50204.28	-17439.71
	LB	-50404.54	-50419.53	-50430.55	-50433.43	-64443.67
	runtime(s)	308.49	564.10	130.73	384.55	1542.32
elt3L1_0504073-lowThresh-more-hyp.surf-16-03-11-1745	UB	-45831.06	-45567.68	inf	-45728.81	8747.71
	LB	-45831.06	-45847.62	-45895.93	-45868.27	-63468.72
	runtime(s)	90.98	592.91	50.99	217.13	1392.60

Table 6: Per-instance results. UB means primal solution energy, LB dual lower bound and runtime(s) the runtime of the particular algorithm in seconds. **Bold numbers** indicate lowest primal energy, highest lower bound and smallest runtime.

Instance		AMP	AMCF	GM	HBP	DD
hlh1fourL1_0417071-lowThresh-more-hyp.surf-16-03-11-1745	UB	-46836.83	-46533.33	inf	-46408.66	854.23
	LB	-47032.22	-47086.83	-47119.47	-47082.80	-63780.52
	runtime(s)	262.59	543.30	483.25	313.96	1439.73
hlh1fourL1_0417075-lowThresh-more-hyp.surf-16-03-11-1745	UB	-49550.13	-49409.57	inf	-49530.58	-12058.87
	LB	-49550.13	-49566.47	-49641.38	-49558.24	-61647.72
	runtime(s)	18.79	789.50	73.30	382.49	1511.04
hlh1fourL1_0417076-lowThresh-more-hyp.surf-16-03-11-1745	UB	-48403.69	-48315.54	inf	-48321.19	-53.31
	LB	-48404.77	-48414.68	-48434.31	-48420.78	-63809.37
	runtime(s)	334.20	737.72	342.43	286.27	1471.41
hlh1fourL1_0417077-lowThresh-more-hyp.surf-16-03-11-1745	UB	-48071.87	-47907.39	inf	-48024.64	-1259.67
	LB	-48071.87	-48093.76	-48107.28	-48097.67	-63985.70
	runtime(s)	97.92	789.72	182.47	259.06	1500.05
hlh1fourL1_0417078-lowThresh-more-hyp.surf-16-03-11-1745	UB	-48236.10	-48133.31	inf	-48130.23	-10622.12
	LB	-48236.37	-48242.47	-48250.18	-48254.45	-61492.56
	runtime(s)	599.12	745.14	800.89	263.29	1445.43
mir61L1_1228061-lowThresh-more-hyp.surf-16-03-11-1745	UB	-48777.03	-48637.86	inf	-48677.84	-5805.11
	LB	-48788.87	-48800.51	-48842.39	-48807.78	-60780.51
	runtime(s)	319.10	542.99	302.49	280.86	1452.32
mir61L1_1228062-lowThresh-more-hyp.surf-16-03-11-1745	UB	-49415.47	-49315.92	inf	-49332.57	-14934.02
	LB	-49419.06	-49423.49	-49447.95	-49431.13	-59976.84
	runtime(s)	358.76	638.51	716.44	289.63	1482.86
mir61L1_1229062-lowThresh-more-hyp.surf-16-03-11-1745	UB	-49836.77	-49696.20	inf	-49717.00	-11816.15
	LB	-49836.77	-49851.68	-49860.50	-49859.68	-64448.89
	runtime(s)	277.70	703.77	397.94	283.92	1493.55
pha4A7L1_1213061-lowThresh-more-hyp.surf-16-03-11-1745	UB	-47991.95	-47816.55	inf	-47881.55	†
	LB	-47996.18	-48019.58	-48103.71	-48029.49	†
	runtime(s)	254.40	630.45	721.77	267.23	†
pha4A7L1_1213062-lowThresh-more-hyp.surf-16-03-11-1745	UB	-49985.66	-49909.30	inf	-49952.46	†
	LB	-49985.66	-49994.59	-49995.43	-49991.35	†
	runtime(s)	20.54	900.00	19.99	335.07	†
pha4A7L1_1213064-lowThresh-more-hyp.surf-16-03-11-1745	UB	-49284.01	-49161.13	inf	-49173.02	†
	LB	-49320.70	-49344.45	-49359.94	-49345.73	†
	runtime(s)	267.25	653.14	38.14	308.96	†
pha4B2L1_0125072-lowThresh-more-hyp.surf-16-03-11-1745	UB	-47230.88	-47070.83	inf	-47109.54	†
	LB	-47233.63	-47257.22	-47272.88	-47264.07	†
	runtime(s)	158.91	295.26	139.47	218.78	†
pha4I2L_0408071-lowThresh-more-hyp.surf-16-03-11-1745	UB	-46106.84	-45729.12	inf	-45602.49	2993.45
	LB	-46124.34	-46187.69	-46245.04	-46192.43	-61549.52
	runtime(s)	244.49	428.54	365.95	282.77	1420.16
pha4I2L_0408072-lowThresh-more-hyp.surf-16-03-11-1745	UB	-50062.40	-49999.81	-50004.15	-50051.26	-8030.13
	LB	-50062.40	-50067.32	-50082.97	-50067.73	-64464.55
	runtime(s)	129.34	576.88	122.65	257.30	1454.57
pha4I2L_0408073-lowThresh-more-hyp.surf-16-03-11-1745	UB	-49497.10	-49438.10	-49496.46	-49379.77	-5615.99
	LB	-49497.10	-49508.85	-49497.10	-49513.77	-69802.71
	runtime(s)	193.95	767.31	64.83	319.46	1548.78
unc54L1_0123071-lowThresh-more-hyp.surf-16-03-11-1745	UB	-50069.07	-50008.12	inf	-50008.27	-13325.80
	LB	-50069.17	-50073.81	-50076.97	-50080.87	-60277.54
	runtime(s)	64.45	680.90	16.82	299.76	1516.94
unc54L1_0123072-lowThresh-more-hyp.surf-16-03-11-1745	UB	-49775.89	-49645.88	inf	-49708.63	-9321.43
	LB	-49775.89	-49783.56	-49785.75	-49787.69	-63516.95
	runtime(s)	115.50	746.36	20.38	334.56	1468.27