A Comparative Study of Graph Matching Algorithms in Computer Vision – Supplement Material

Stefan Haller[1], Lorenz Feineis[1], Lisa Hutschenreiter[1], Florian Bernard[2], Carsten Rother[1], Dagmar Kainmüller[3], Paul Swoboda[4], Bogdan Savchynskyy[1]

[1]Heidelberg University, [2]University of Bonn, [3]MDC Berlin, [4]MPI-INF Saarbrücken

A1 Equivalence Proofs

Theorem 1. *The graph matching problem and the quadratic assignment problem (QAP) are polynomially reducible to each other.*

While the formal proof can be found below, let us develop an intuition for the problem behind Theorem 1. In general, the main difference between graph matching and the QAP is that while the QAP requires complete matchings, graph matching allows for incomplete matchings, i.e., not every element of the first set has to be assigned to one of the second and vice versa. So the equivalence construction mainly has to deal with these side constraints.

When transitioning from graph matching to the QAP, we go from incomplete to complete matchings. Therefore, the idea is to extend the two sets, usually called $\mathcal{V}$ and $\mathcal{L}$, by "dummy" elements to which all previously unassigned elements can be assigned without changing the total cost, see Figure A1 for an illustration.

On the other hand, when going from the QAP to graph matching, we switch from a complete to an incomplete matching. Here, the idea is to shift the cost structure in such a way that the cost difference of any two complete matchings stays the same, while at the same time guaranteeing that any incomplete matching can be improved by completing it, see Figure A2 for an illustration.

Proof. We will prove Theorem 1 by construction.
Reduction of graph matching to QAP. Consider the graph matching problem, c.f. (1),

$$\min_{x \in \{0,1\}^{\mathcal{V}\times\mathcal{L}}} \sum_{\substack{i,j \in \mathcal{V} \\ s,l \in \mathcal{L}}} c_{is,jl}\, x_{is}\, x_{jl} \tag{A1}$$

$$\text{subject to} \sum_{s \in \mathcal{L}} x_{is} \leq 1 \quad \text{for all } i \in \mathcal{V}, \tag{A2}$$

$$\sum_{i \in \mathcal{V}} x_{is} \leq 1 \quad \text{for all } s \in \mathcal{L}$$

for finite sets $\mathcal{V}$ and $\mathcal{L}$, and cost function $c\colon (\mathcal{V}\times\mathcal{L})^2 \to \mathbb{R}$.

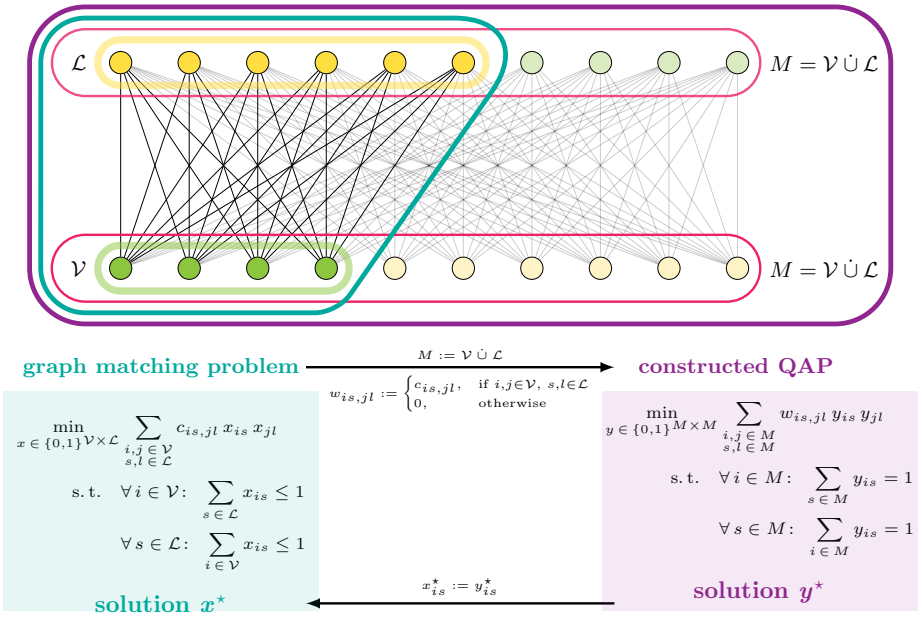

Fig. A1. From graph matching to the QAP. The notation corresponds to the notation used in the proof of Theorem 1. The idea behind reducing graph matching to quadratic assignment is to enable completion of the possibly incomplete matchings feasible for graph matching. This is done by providing "dummy" nodes to which those nodes can be matched that would otherwise be left unassigned. As potentially all nodes could be unassigned, we provide a corresponding dummy for each.

Let M be the disjoint union of $\mathcal{V}$ and $\mathcal{L}$, $M := \mathcal{V} \dot\cup \mathcal{L}$, and let $w\colon M^4 \to \mathbb{R}$ be defined as

$$w_{is,jl} = w(i,s,j,l) := \begin{cases} c_{is,jl}, & \text{if } i,j \in \mathcal{V},\ s,l \in \mathcal{L}, \\ 0, & \text{otherwise.} \end{cases}$$

We can then formulate the QAP

$$\min_{y \in \{0,1\}^{M\times M}} \sum_{\substack{i,j \in M \\ s,l \in M}} w_{is,jl}\, y_{is}\, y_{jl} \tag{A3}$$

$$\text{subject to} \sum_{s \in M} y_{is} = 1 \quad \text{for all } i \in M, \tag{A4}$$

$$\sum_{i \in M} y_{is} = 1 \quad \text{for all } s \in M.$$

Let $y^\star$ be a solution of (A3) satisfying (A4), i.e.,

$$y^\star \in \operatorname*{arg\,min}_{y \in \{0,1\}^{M\times M}} \sum_{\substack{i,j \in M \\ s,l \in M}} w_{is,jl}\, y_{is}\, y_{jl}$$
$$\text{s.t.} \sum_{s \in M} y_{is} = 1 \quad \text{for all } i \in M,$$
$$\sum_{i \in M} y_{is} = 1 \quad \text{for all } s \in M.$$

Now let $x^\star \in \{0,1\}^{\mathcal{V}\times\mathcal{L}}$ be defined as $x^\star_{is} = y^\star_{is}$ for all $i \in \mathcal{V}$, $s \in \mathcal{L}$. It remains to show that $x^\star$ then is a solution of (A1) satisfying (A2).

First note that $x^\star$ satisfies the constraints (A2) since for all $i \in \mathcal{V}$

$$\begin{aligned} \sum_{s \in \mathcal{L}} x^\star_{is} &= \sum_{s \in \mathcal{L}} y^\star_{is} \\ &\le \sum_{s \in M} y^\star_{is} && \text{as } \mathcal{L} \subseteq M \text{ and } y^\star \in \{0,1\}^{M\times M}, \\ &\le 1 && \text{as } y^\star \text{ satisfies (A4).} \end{aligned}$$

Analogously, $\sum_{i \in \mathcal{V}} x^\star_{is} \le 1$ for all $s \in \mathcal{L}$.

Suppose now that $x^\star$ is not a solution of (A1), i.e., there exists $x' \in \{0,1\}^{\mathcal{V}\times\mathcal{L}}$ satisfying (A2) with

$$\sum_{\substack{i,j \in \mathcal{V} \\ s,l \in \mathcal{L}}} c_{is,jl}\, x'_{is}\, x'_{jl} < \sum_{\substack{i,j \in \mathcal{V} \\ s,l \in \mathcal{L}}} c_{is,jl}\, x^\star_{is}\, x^\star_{jl}.$$

We can then define $y' \in \{0,1\}^{M \times M}$ by

$$y'_{is} := \begin{cases} x'_{is}, & \text{if } i \in \mathcal{V},\ s \in \mathcal{L}, \\ x'_{si}, & \text{if } i \in \mathcal{L},\ s \in \mathcal{V}, \\ 1, & \text{if } i \in \mathcal{V},\ s = i, \text{ and } \sum_{l \in \mathcal{L}} x'_{il} < 1, \\ 1, & \text{if } i \in \mathcal{L},\ s = i, \text{ and } \sum_{j \in \mathcal{V}} x'_{js} < 1, \\ 0, & \text{otherwise.} \end{cases}$$

Observe that y' satisfies the constraints (A4) since for all $i \in M$

$$\begin{aligned} \sum_{s \in M} y'_{is} &= \sum_{s \in \mathcal{V}} y'_{is} + \sum_{s \in \mathcal{L}} y'_{is} \\ &= \begin{cases} y'_{ii} + \sum_{s \in \mathcal{L}} x'_{is} & \text{if } i \in \mathcal{V} \\ \sum_{s \in \mathcal{V}} x'_{si} + y'_{ii} & \text{if } i \in \mathcal{L} \end{cases} \\ &= \begin{cases} 1 - \sum_{l \in \mathcal{L}} x'_{il} + \sum_{s \in \mathcal{L}} x'_{is} & \text{if } i \in \mathcal{V} \\ \sum_{s \in \mathcal{V}} x'_{si} + 1 - \sum_{j \in \mathcal{V}} x'_{ji} & \text{if } i \in \mathcal{L} \end{cases} \\ &= 1, \end{aligned}$$

and, analogously, $\sum_{i \in M} y'_{is} = 1$ for all $s \in M$. Then we obtain

$$\begin{aligned} &\sum_{\substack{i,j \in M \\ s,l \in M}} w_{is,jl}\, y'_{is}\, y'_{jl} \\ &\quad = \sum_{\substack{i,j \in \mathcal{V} \\ s,l \in \mathcal{L}}} c_{is,jl}\, y'_{is}\, y'_{jl} && \text{by definition of } w, \\ &\quad = \sum_{\substack{i,j \in \mathcal{V} \\ s,l \in \mathcal{L}}} c_{is,jl}\, x'_{is}\, x'_{jl} && \text{by definition of } y', \\ &\quad < \sum_{\substack{i,j \in \mathcal{V} \\ s,l \in \mathcal{L}}} c_{is,jl}\, x^\star_{is}\, x^\star_{jl} && \text{due to the choice of } x', \\ &\quad = \sum_{\substack{i,j \in \mathcal{V} \\ s,l \in \mathcal{L}}} c_{is,jl}\, y^\star_{is}\, y^\star_{jl} && \text{by definition of } x^\star, \\ &\quad = \sum_{\substack{i,j \in M \\ s,l \in M}} w_{is,jl}\, y^\star_{is}\, y^\star_{jl} && \text{by definition of } w, \\ &\quad \leq \sum_{\substack{i,j \in M \\ s,l \in M}} w_{is,jl}\, y'_{is}\, y'_{jl} && \text{since } y^\star \text{ solution of (A3)}, \end{aligned}$$

which is a contradiction. Hence, $x^\star$ is a solution of (A1).

Therefore, graph matching is polynomially reducible to the quadratic assignment problem as the size of the constructed QAP is polynomial in the size of the

initial graph matching problem, and each solution of the QAP directly induces a solution to the graph matching problem.

Reduction of QAP to graph matching. Consider the QAP

$$\min_{x \in \{0,1\}^{M \times M}} \sum_{\substack{i,j \in M \\ s,l \in M}} w_{is,jl}\, x_{is}\, x_{jl} \tag{A5}$$

$$\text{subject to} \sum_{s \in M} x_{is} = 1 \quad \text{for all } i \in M, \tag{A6}$$

$$\sum_{i \in M} x_{is} = 1 \quad \text{for all } s \in M,$$

for a finite set M and cost function $w\colon M^4 \to \mathbb{R}$.

We can now formulate the graph matching problem

$$\min_{x \in \{0,1\}^{M \times M}} \sum_{\substack{i,j \in M \\ s,l \in M}} c_{is,jl}\, x_{is}\, x_{jl} \tag{A7}$$

$$\text{subject to} \sum_{s \in M} x_{is} \leq 1 \quad \text{for all } i \in M, \tag{A8}$$

$$\sum_{i \in M} x_{is} \leq 1 \quad \text{for all } s \in M,$$

where $\mathcal{V} = \mathcal{L} = M$, and $c_{is,jl} := w_{is,jl} - \max(w) - 1$ for all $i, j, s, l \in M$. Note that by definition $c_{is,jl} < 0$ for all $i, j, s, l \in M$.

Let $x^\star$ be a solution of (A7) satisfying (A8). We now want to prove that $x^\star$ also satisfies (A6) and is a solution of (A5).

Suppose that $x^\star$ does not satisfy (A6), i.e., there exists $i' \in M$ with $\sum_{s \in M} x^\star_{i's} < 1$. Then there also exists $s' \in M$ with $\sum_{i \in M} x^\star_{is'} < 1$. Observe that $x^\star_{i's} = 0$ for all $s \in M$, and $x^\star_{is'} = 0$ for all $i \in M$ since $x^\star \in \{0,1\}^{M \times M}$.

We can now define x' as

$$x'_{is} = \begin{cases} 1, & \text{if } i = i',\ s = s', \\ x^\star_{is}, & \text{otherwise.} \end{cases}$$

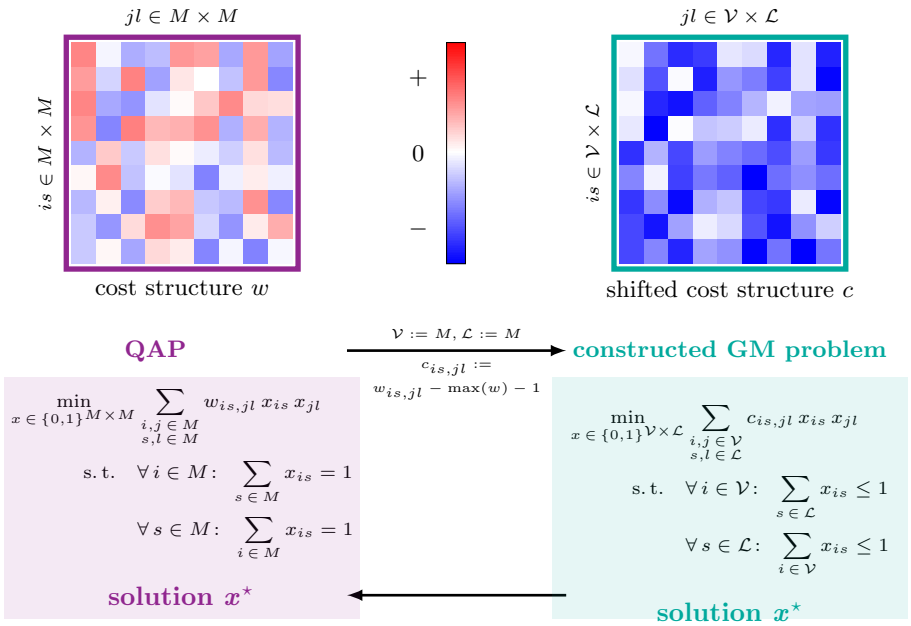

Fig. A2. From the QAP to graph matching (GM). The notation corresponds to the notation used in the proof of Theorem 1. The idea behind reducing quadratic assignment to graph matching is to make sure that while the requirement for complete matchings is dropped, a better objective value can always be obtained when completing a given incomplete matching. This is guaranteed by shifting the cost structure to incur negative costs for all assignments.

Due to the choice of i' and s', x' also satisfies the constraints (A8). Furthermore,

$$\begin{aligned}
&\sum_{\substack{i,j \in M \\ s,l \in M}} c_{is,jl}\, x'_{is}\, x'_{jl} \\
&\qquad = \sum_{\substack{i,j \in M \\ s,l \in M}} c_{is,jl}\, x^\star_{is}\, x^\star_{jl} + \sum_{\substack{i,s \in M \\ is \neq i's'}} c_{is,i's'}\, x^\star_{is} \\
&\qquad\quad + \sum_{\substack{j,l \in M \\ jl \neq i's'}} c_{i's',jl}\, x^\star_{jl} + c_{i's',i's'} \\
&\qquad < \sum_{\substack{i,j \in M \\ s,l \in M}} c_{is,jl}\, x^\star_{is}\, x^\star_{jl},
\end{aligned}$$

since $c_{is,jl} < 0$ for all $i, j, s, l \in M$. This contradicts $x^\star$ being a solution of (A7). Hence, $x^\star$ satisfies (A6).

Suppose now that $x^\star$ is not a solution of (A5), i.e., there exists $\overline{x} \in \{0,1\}^{M \times M}$ satisfying (A6) with

$$\sum_{\substack{i,j \in M \\ s,l \in M}} w_{is,jl}\, \overline{x}_{is}\, \overline{x}_{jl} < \sum_{\substack{i,j \in M \\ s,l \in M}} w_{is,jl}\, x^\star_{is}\, x^\star_{jl}.$$

Note that $\overline{x}$ also satisfies (A8) as it satisfies (A6).

Due to the constraints (A6), $x^\star$ and $\overline{x}$ have exactly $|M|$ nonzero entries, hence when summing over all $i, j, s, l \in M$, both, $x^\star_{is} x^\star_{jl}$ and $\overline{x}_{is} \overline{x}_{jl}$, are nonzero exactly $|M|^2$ times. Therefore, we obtain

$$\begin{aligned}
&\sum_{\substack{i,j \in M \\ s,l \in M}} w_{is,jl}\, \overline{x}_{is}\, \overline{x}_{jl} - |M|^2(\max(w) + 1) \\
&\qquad < \sum_{\substack{i,j \in M \\ s,l \in M}} w_{is,jl}\, x^\star_{is}\, x^\star_{jl} - |M|^2(\max(w) + 1) \\
&\sum_{\substack{i,j \in M \\ s,l \in M}} \big(w_{is,jl} - \max(w) - 1 \big)\, \overline{x}_{is}\, \overline{x}_{jl} \\
&\qquad < \sum_{\substack{i,j \in M \\ s,l \in M}} \big(w_{is,jl} - \max(w) - 1 \big)\, x^\star_{is}\, x^\star_{jl} \\
&\sum_{\substack{i,j \in M \\ s,l \in M}} c_{is,jl}\, \overline{x}_{is}\, \overline{x}_{jl} < \sum_{\substack{i,j \in M \\ s,l \in M}} c_{is,jl}\, x^\star_{is}\, x^\star_{jl}
\end{aligned}$$

This would contradict $x^\star$ being a solution of (A7). Thus, $x^\star$ is a solution of (A5).

This proves that the quadratic assignment problem is polynomially reducible to graph matching as the size of the constructed graph matching problem is

polynomial in the size of the initial QAP. Each solution of the graph matching problem directly yields a solution of the QAP.

Remark A1. Note that a polynomial reduction from the QAP to graph matching can also be obtained by shifting only the unary costs by a sufficiently large constant, i.e., by defining

$$c_{is,jl} = \begin{cases} w_{is,jl} - K, & \text{if } jl = is,\ i,j,s,l \in M \\ w_{is,jl}, & \text{otherwise.} \end{cases}$$

for sufficiently large K. This is relevant in practice as it only changes the diagonal of the cost matrix, and not the full matrix. In this way it does not as heavily influence the sparsity of the cost structure.

The statements in the proof of Theorem 1 hold in particular if all quadratic terms incur a cost of zero, so we can reduce the objective function to its linear component, i.e., the objective function

$$\sum_{\substack{i,j \in \mathcal{V} \\ s,l \in \mathcal{L}}} c_{is,jl}\, x_{is}\, x_{jl}$$

of the graph matching problem, c.f. (A1) in the proof above, can be reduced to

$$\sum_{\substack{i \in \mathcal{V} \\ s \in \mathcal{L}}} c_{is}\, x_{is},$$

which corresponds to the incomplete linear assignment problem. Similarly, we can replace the objective function

$$\sum_{\substack{i,j \in M \\ s,l \in M}} w_{is,jl}\, y_{is}\, y_{jl}$$

of the quadratic assignment problem, c.f. (A5), by

$$\sum_{i,s \in M} w_{is}\, y_{is},$$

which corresponds to the objective of the linear assignment problem. With this in mind, the corollary below directly follows from Theorem 1:

Corollary 1. *The incomplete linear assignment problem and the linear assignment problem (LAP) are polynomially reducible to each other.*

Table A1. Short specification of the dd file format. The format originated in [59]. It encodes a sparse representation of the graph matching problem (1). Unary costs are encoded as *assignments* and pairwise costs are encoded as *edges* between two assignments. Any lines marked as optional or comments are ignored when building the graph matching problem. They were introduced in [59] to ease visualization of matching problems between two images.

input line	description
`c comment line`	comments are ignored
`p <#V> <#L> <#A> <#E>`	prologue: $\|\mathcal{V}\|$, $\|\mathcal{L}\|$, number of assignments and edges to follow
`a <id> <i> <s> <cost>`	specifies possible assignment $i \to s$ for $i \in \mathcal{V}$, $s \in \mathcal{L}$ (unary terms $c_{is,is}x_{is}$)
`e <id1> <id2> <cost>`	specifies edge between two assignments (pairwise terms $(c_{is,jl} + c_{jl,is})x_{is}x_{jl}$)
`i0 <i> <x> <y>`	optional: coordinate of the node $i \in \mathcal{V}$ as a point in the left image
`i1 <s> <x> <y>`	optional: coordinate of the node $s \in \mathcal{L}$ as a point in the right image
`n0 <i> <j>`	optional: specifies that points $i, j \in \mathcal{V}$ are neighbors in the left image
`n1 <s> <l>`	optional: specifies that points $s, l \in \mathcal{L}$ are neighbors in the right image

A2 Symmetry of the Formulations

Note that formulations (3) and (4)-(6) are asymmetric, since the sets $\mathcal{V}$ and $\mathcal{L}$ play different roles in the formulations, in contrast to the symmetric formulation (1). Although swapping the roles of $\mathcal{V}$ and $\mathcal{L}$ does not influence the optimal solutions, it influences the problem structure, such as the sparsity of the resulting graphical model, as well as the tightness of the corresponding LP relaxation. Symmetrized formulations based on the graphical model representation use two graphical models, with the role of $\mathcal{V}$ and $\mathcal{L}$ being swapped in one of them [57]. These graphical models, with variables denoted correspondingly by x and $\hat{x}$, are coupled by additional constraints which enforce either the unary variables to be equal in both representations, i.e., $x_{is} = \hat{x}_{si}$, or the pairwise variables, i.e., $x_{is,jl} = \hat{x}_{si,lj}$. The first type of coupling constraints is computationally cheaper, whereas the second one leads to a tighter LP relaxation equivalent to the standard one considered in operations research [1]. In our experimental evaluation the second type of constraints is implicitly used by `dd-ls*` algorithms.

A3 Data Format (dd-format)

As a unified data format for all datasets we use the one introduced by [59]. It encodes a sparse representation of the graph matching problem (1) in plain text. Due to its origin we refer to it as *dd-* or *dual decomposition* data format. It is used by `dd-ls*`, as well as `fm`, `fm-bca` and `mp-*` algorithms as a native input. Table A1 presents an overview of the file format and shows a description of of all input line types.

With other algorithms suitable converters are used. The one for the graphical model representation (3) is readily available at [32]. For Matlab algorithms a matrix representation of (1) is required. We implemented this conversion by ourselves and will make the code available.

Algorithm 1: Transform into non-positive costs.

Input: Graph matching problem $(\mathcal{V}, \mathcal{L}, c)$
Output: Equivalent non-positive cost vector c'

`/* initialize */`
$c' \leftarrow (0 \dots 0)$

`/* first, shift unary costs */`
for $i \in \mathcal{V}$ *and* $s \in \mathcal{L}$ **do**
  $\alpha \leftarrow \max\{c_{is'} \mid s' \in \mathcal{L} \text{ and } c_{is'} \neq \infty\}$
  $c'_{is} \leftarrow c_{is} - \max\{0, \alpha\}$

`/* second, shift pairwise costs */`
for $i, j \in \mathcal{V}$, $i < j$ *and* $s, t \in \mathcal{L}$ **do**
  $\alpha \leftarrow \max\{c_{is',jl'} \mid s', l' \in \mathcal{L}\}$
  $c'_{is,jl} \leftarrow c_{is,jl} - \max\{0, \alpha\}$

A4 Cost Transformations

Different methods have different requirements for the cost matrix, see columns *bijective*, *non-positive* and *zero unary* in Table 1. For datasets where the costs do not fulfill those requirements, see equivalent columns in Table 3, the costs have to be transformed to make them amenable to those algorithms. The transformations are chosen to in a way such that **(i)** the cost difference between any two matchings stays the same; and **(ii)** the sparsity of the cost matrix is preserved as much as possible. In the following we describe each of the three transformations. The full implementation is part of our published source code for the benchmark. Without loss of generality we assume the cost matrix to be upper triangular, i.e. $c_{is,jl} = 0$ for $i, j \in \mathcal{V}$, $s, l \in \mathcal{L}$ and $i > j$.

Bijective Costs. Non-bijective problem instances are transformed into the bijective ones in a way similar to the described in the proof of Theorem 1 and illustrated in Fig. A1.

Non-positive Costs. Here we transform the cost matrix into a form where all *finite* costs are non-positive. Note that infinite costs will stay positive, as they are identified by corresponding algorithms and prohibit selection of the associated matchings. **??** 1 shows the transformation which shifts the costs of all bijective assignments by a fixed constant. Therefore, for non-bijective instances we first transform them into the bijective form as described in previous paragraph.

Remove Unary Costs. The `pm` algorithm does not take into account unary costs, see column *zero unary* in Table 1. We use **??** 2 to transform unary costs into pairwise costs.

For the maximization algorithms `ipfpu`, `ipfps`, `ga`, `rrwm`, `pm`, `sm`, `smac`, `lsm`, `mpm`, `fgmd` and `hbp` the sign of the elements in the cost matrix is flipped additionally after performing all other operations.

Algorithm 2: Transform unary into pairwise costs.

Input: Graph matching problem $(\mathcal{V}, \mathcal{L}, c)$
Output: Equivalent cost vector c' with empty diagonal

`/* initialize */`
for $i, j \in \mathcal{V}$ *and* $s, l \in \mathcal{L}$ **do**
  $c'_{is,jl} \leftarrow \begin{cases} c_{is,jl} & \text{if } (i,s) \neq (j,l) \\ 0 & \text{otherwise} \end{cases}$

`/* count how many additional cost entries are necessary; remember decision with least number of additional entries in array` J `*/`
for $i \in \mathcal{V}$ *and* $s \in \mathcal{L}$ **do**
  $J_{is} \leftarrow \arg\min_{j \in \mathcal{V}} |\{l \in \mathcal{L} \mid c_{is,jl} = 0 \text{ and } i \neq j, s \neq l\}|$

`/* distribute unaries across pairwise costs */`
for $i, j \in \mathcal{V}$, $i < j$ *and* $s, l \in \mathcal{L}$ **do**
  if $J_{is} = j$ **then**
    $c'_{is,jl} \leftarrow c'_{is,jl} + c_{is,is}$
  if $J_{jl} = i$ **then**
    $c'_{is,jl} \leftarrow c'_{is,jl} + c_{jl,jl}$

A5 Accuracy computation

For `hotel`, `house-sparse`, `house-dense`, `car` and `motor` the complete ground truth assignments are known. The accuracy is computed as the number of correctly assigned nodes over $n_{\mathcal{V}}$. The ground truth for `caltech-small`, `caltech-large`, `worms` and `pairs` is only partial i.e. not every node has a ground truth label. This is taken into account by computing the accuracy as the number of correctly assigned nodes over the number of nodes with available ground truth label.

A6 Run Time Measurement

To make the comparison as fair as possible, we exclude preprocessing steps and the time it takes to load the graph matching files into the solver. This means that the following steps are excluded from our run-time measurements: **(i)** Conversion process from the dd file format into the input format of the solvers; **(ii)** Loading the file from storage into memory; **(iii)** Parsing the input file; and **(iv)** Dataset transformation if needed.

Practically, we start the timer directly before the solver starts the optimization routine. For optimization methods that have not included this functionality, we have modified the methods accordingly.

Table A2. Command line parameters for all C++ methods.

method	command line
dd-ls0	tkrgm --linear --tree --max-iter N
dd-ls3	tkrgm --linear --local 2 --tree --max-iter N
dd-ls4	tkrgm --linear --local 3 --tree --max-iter N
fw	graph_matching_frank_wolfe_text_input input.dd
mp	graph_matching_mp -i input.dd --roundingReparametrization uniform:0.5
mp-mcf	graph_matching_mp_tightening -i input.dd --tighten --tightenInterval 50 \ --tightenIteration 200 --tightenConstraintsPercentage 0.01 \ --tightenReparametrization uniform:0.5 --graphMatchingRounding mcf
mp-fw	graph_matching_mp_tightening -i input.dd --tighten --tightenInterval 50 \ --tightenIteration 200 --tightenConstraintsPercentage 0.01 \ --tightenReparametrization uniform:0.5 --graphMatchingRounding fw
fm	qap_dd --max-batches N --batch-size 0 --generate 1
fm-bca	qap_dd --max-batches N --batch-size 10 --generate 10

A7 Reproducibility

Original Source Code. References with links to the original source code can be found in column *Matlab* and *C++* in Table 1. Note that for the `dd-ls*` methods we use the same command line wrapper as used in [31]. The wrapper allows to select the problem decomposition at run-time without need for recompilation. The specific command lines for the C++ programs (see column *C++* in Table 1) that have been used in the benchmark are listed in Table A2.

Matlab Wrappers. We incorporated the implementations of the Matlab algorithms in our own wrappers in order to be able to load the datasets and to make the output standardized. No parameters have to be specified.

Reproducible Containers. For all methods we build Linux containers so that the environment as well as the specific version of the solvers are reproducible. The container files contain instructions for downloading, processing and building a fixed version of the corresponding methods. The source code of each method is fetched from the Internet. For the case that source code is no longer online, we preserve a historical copy of the repositories. The container build scripts will pin each method to a specific version or commit id to preserve reproducibility.

HPC Scripts. For usage in high-performance compute clusters (HPC clusters) we provide a small script to transform the container images into singluarity image. Most HPC clusters provide tools to easily run these singularity images. Additionally, we provide SLURM scheduling scripts so that the benchmark can be reproduced easily and quickly.

Project Web Site. The wrappers, container files and other scripts are publicly available on the project web site, see https://vislearn.github.io/gmbench/.

A8 Detailed Evaluation Results

More detailed evaluation results are provided on the following pages. Figure A3 shows run-time performance profiles similar to Figure 1 but for each dataset

separately. Tables A3 to A13 show the average results of each method on each dataset for time limits 1s, 10s, 100s and 300s similar to Table 4 and Table 5.

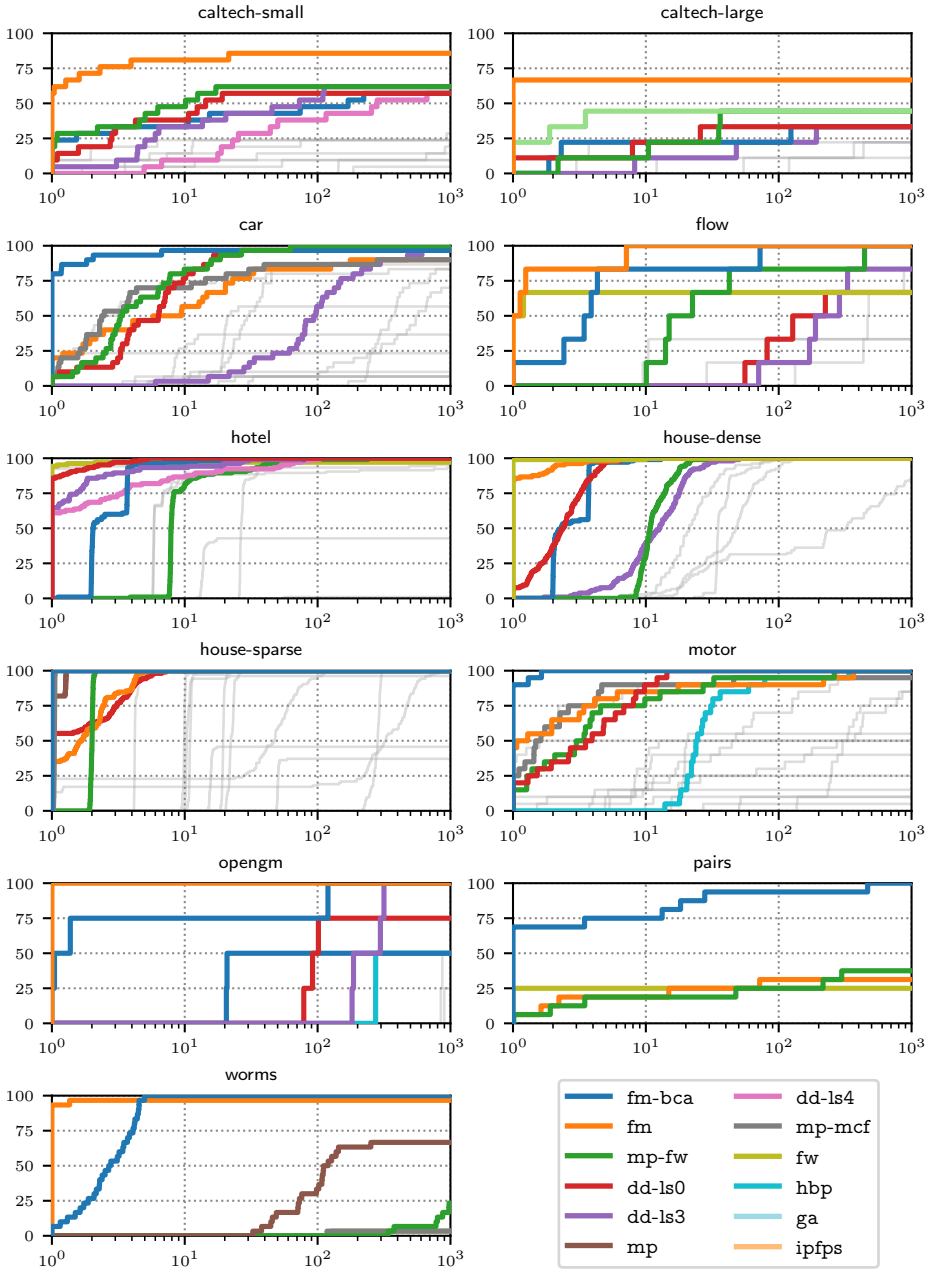

Fig. A3. Run-time performance profile [22] per dataset (X-axis: ratio to best performance τ; Y-axis: solving probability $\rho(\tau)$ in %).

Table A3. Detailed fixed-time evaluation for dataset caltech-small.

	caltech-small (1s)				caltech-small (10s)				caltech-small (100s)				caltech-small (300s)			
	opt	E	D	acc	opt	E	D	acc	opt	E	D	acc	opt	E	D	acc
fgmd	0		—*		14		—*		29		—*		43		—*	
fm	**52**	-8906	-43926	58	**57**	-9040	-43926	62	**57**	-9050	-43926	63	**57**	-9050	-43926	63
fw	0	0	–	0	0	0	–	0	0	0	–	0	0	0	–	0
ga	0		—*		5		—*		5		—*		5		—*	
ipfps	19	**-8983**	–	67	19	-8983	–	67	19	-8983	–	67	19	-8983	–	67
ipfpu	10	-8829	–	62	10	-8829	–	62	10	-8829	–	62	10	-8829	–	62
lsm	0		—*		0	0	–	0	0	0	–	0	0	0	–	0
mpm	0		—*		0		—*		0		—*		0		—*	
pm	0	-6510	–	51	0	-6510	–	51	0	-6510	–	51	0	-6510	–	51
rrwm	5		—*		5	-8848	–	61	5	-8848	–	61	5	-8848	–	61
sm	0	-3932	–	36	0	-3932	–	36	0	-3932	–	36	0	-3932	–	36
smac	0	-6196	–	42	0	-6196	–	42	0	-6196	–	42	0	-6196	–	42
dd-ls0	43	-7414	**-10658**	58	48	-8251	**-9502**	60	48	-8251	-9502	60	48	-8251	-9502	60
dd-ls3	33	-6842	-15159	57	52	-7862	-9904	60	52	-8357	**-9466**	61	52	-8357	-9466	61
dd-ls4	24	-6332	-18033	54	38	-6919	-13407	53	52	-7878	-9810	60	52	-8324	**-9443**	61
fm-bca	38	-8927	-12827	62	43	-8943	-12797	62	48	-8953	-12797	61	52	-8960	-12797	61
hbp	0		—*		5		—*		10		—*		10		—*	
mp	14	-7967	-12191	59	14	-8026	-12183	60	19	-8095	-12182	62	19	-8095	-12182	62
mp-fw	33	-8886	-12793	60	38	**-9052**	-12244	65	38	**-9056**	-11217	65	43	**-9057**	-10818	64
mp-mcf	5	-7882	-12380	60	10	-8030	-11737	59	14	-8324	-10987	57	14	-8435	-10731	57

opt: optimally solved instances (%); **E:** average best objective value; **D:** average best lower bound if applicable;
acc: average accuracy corresponding to best objective (%); **—*:** method yields no solution for at least one problem instance

Table A4. Detailed fixed-time evaluation for dataset caltech-large.

	caltech-large (1s)			caltech-large (10s)			caltech-large (100s)			caltech-large (300s)		
	E	D	acc	E	D	acc	E	D	acc	E	D	acc
fgmd		—*			—*			—*			—*	
fm	-33972	-142829	52	-34117	-142829	52	-34125	-142829	52	-34125	-142829	52
fw	0	–	0	0	–	0	0	–	0	0	–	0
ga		—*			—*			—*			—*	
ipfps		—*			—*		-33998	–	51	-33998	–	51
ipfpu		—*		-34216	–	52	-34216	–	52	-34216	–	52
lsm		—*			—*		0	–	0	0	–	0
mpm		—*			—*			—*			—*	
pm	-29106	–	43	-29106	–	43	-29106	–	43	-29106	–	43
rrwm		—*			—*			—*			—*	
sm		—*		-14423	–	28	-14423	–	28	-14423	–	28
smac		—*		-24183	–	39	-24183	–	39	-24183	–	39
dd-ls0	-26236	-56071	48	-32973	**-35007**	51	-33539	-34959	52	-33539	-34959	52
dd-ls3	-25226	-72766	44	-28653	-42079	49	-33552	**-34914**	49	-33557	**-34911**	49
dd-ls4	-25268	-79327	46	-25599	-62120	46	-30148	-38880	51	-32266	-35577	51
fm-bca	-33758	-48582	51	-34040	-48223	51	-34073	-48217	51	-34082	-48217	51
hbp		—*			—*			—*			—*	
mp	-31315	**-46170**	48	-32017	-46070	48	-32069	-46066	48	-32074	-46066	48
mp-fw	**-34227**	-51526	51	**-34237**	-48882	51	**-34277**	-45923	51	**-34287**	-44424	51
mp-mcf	-29813	-48168	45	-30362	-46630	47	-30737	-43833	47	-31230	-42564	48

opt: optimally solved instances (%); **E:** average best objective value; **D:** average best lower bound if applicable;
acc: average accuracy corresponding to best objective (%); **—*:** method yields no solution for at least one problem instance

Table A5. Detailed fixed-time evaluation for dataset car.

	car (1s)				car (10s)				car (100s)				car (300s)			
	opt	E	D	acc	opt	E	D	acc	opt	E	D	acc	opt	E	D	acc
fgmd	0		—*		83		—*		83	**-69**	–	89	83	**-69**	–	89
fm	77	**-69**	-107	88	90	**-69**	-107	89	90	**-69**	-107	89	90	**-69**	-107	89
fw	7	-63	–	63	7	-63	–	63	7	-63	–	63	7	-63	–	63
ga	57	-68	–	84	57	-68	–	84	57	-68	–	84	57	-68	–	84
ipfps	10	-65	–	80	10	-65	–	80	10	-65	–	80	10	-65	–	80
ipfpu	7	-60	–	69	7	-60	–	69	7	-60	–	69	7	-60	–	69
lsm	0	-51	–	52	0	-51	–	52	0	-51	–	52	0	-51	–	52
mpm	7		—*		7	-57	–	65	7	-57	–	65	7	-57	–	65
pm	0	-35	–	23	0	-35	–	23	0	-35	–	23	0	-35	–	23
rrwm	37	-68	–	87	37	-68	–	87	37	-68	–	87	37	-68	–	87
sm	7	-63	–	76	7	-63	–	76	7	-63	–	76	7	-63	–	76
smac	0	-52	–	52	0	-52	–	52	0	-52	–	52	0	-52	–	52
dd-ls0	**97**	**-69**	**-69**	91	**97**	**-69**	**-69**	91	97	**-69**	**-69**	91	97	**-69**	**-69**	91
dd-ls3	47	-57	-75	74	87	-68	**-70**	90	97	**-69**	**-69**	91	97	**-69**	**-69**	91
dd-ls4	3	-49	-78	59	57	-59	-73	77	87	-67	**-70**	89	97	**-69**	**-69**	92
fm-bca	93	**-69**	-71	92	**97**	**-69**	-71	92	97	**-69**	-71	92	97	**-69**	-71	92
hbp	77		—*		87		—*		87	**-69**	-73	91	87	**-69**	-73	91
mp	80	**-69**	-71	92	83	**-69**	-71	92	87	**-69**	-71	92	87	**-69**	-71	92
mp-fw	90	**-69**	-71	91	**97**	**-69**	-70	91	**100**	**-69**	**-70**	91	**100**	**-69**	**-70**	91
mp-mcf	87	**-69**	-70	91	90	**-69**	-70	91	93	**-69**	**-70**	91	93	**-69**	**-70**	91

opt: optimally solved instances (%); **E:** average best objective value; **D:** average best lower bound if applicable;
acc: average accuracy corresponding to best objective (%); **—*:** method yields no solution for at least one problem instance

Table A6. Detailed fixed-time evaluation for dataset flow.

	flow (1s)			flow (10s)			flow (100s)			flow (300s)		
	opt	E	D	opt	E	D	opt	E	D	opt	E	D
fgmd	0	—*		0	—*		0	—*		0	—*	
fm	**83**	**-2838**	-3436	**100**	**-2840**	-3436	**100**	**-2840**	-3436	**100**	**-2840**	-3436
fw	67	-2828	–	67	-2828	–	67	-2828	–	67	-2828	–
ga	0	—*		0	—*		0	-2469	–	0	-2469	–
ipfps	0	-766	–	0	-766	–	0	-766	–	0	-766	–
ipfpu	0	-512	–	0	-512	–	0	-512	–	0	-512	–
lsm	0	—*		0	—*		0	31042	–	0	31042	–
mpm	0	—*		0	—*		0	—*		0	—*	
pm	0	-32	–	0	-32	–	0	-32	–	0	-32	–
rrwm	0	—*		0	-226	–	0	-226	–	0	-226	–
sm	0	-139	–	0	-139	–	0	-139	–	0	-139	–
smac	0	0	–	0	0	–	0	0	–	0	0	–
dd-ls0	33	-2345	-2968	67	-2819	-2854	67	-2819	-2854	67	-2819	-2854
dd-ls3	17	-2059	-3030	83	-2821	**-2847**	83	-2834	-2844	83	-2834	-2844
dd-ls4	0	-2062	-3090	67	-2767	-2863	83	-2835	**-2843**	83	-2835	**-2843**
fm-bca	**83**	**-2838**	-2898	**100**	**-2840**	-2879	**100**	**-2840**	-2878	**100**	**-2840**	-2878
hbp	0	—*		0	—*		0	—*		0	—*	
mp	33	-2628	**-2887**	33	-2674	-2882	50	-2690	-2881	50	-2690	-2881
mp-fw	**83**	—*		83	—*		83	-2838	-2870	83	-2838	-2862
mp-mcf	33	-2521	-2892	33	-2719	-2869	50	-2749	-2854	50	-2789	-2851

opt: optimally solved instances (%); **E:** average best objective value; **D:** average best lower bound if applicable;
acc: average accuracy corresponding to best objective (%); **—*:** method yields no solution for at least one problem instance

Table A7. Detailed fixed-time evaluation for dataset hotel.

	hotel (1s)				hotel (10s)				hotel (100s)				hotel (300s)			
	opt	E	D	acc	opt	E	D	acc	opt	E	D	acc	opt	E	D	acc
fgmd	0	—*			0	—*			96	-4283	–	98	96	-4283	–	98
fm	97	-4292	-4406	100	97	-4292	-4406	100	97	-4292	-4406	100	97	-4292	-4406	100
fw	97	-4288	–	99	97	-4288	–	99	97	-4288	–	99	97	-4288	–	99
ga	0	947	–	15	0	947	–	15	0	947	–	15	0	947	–	15
ipfps	0	1051	–	14	0	1051	–	14	0	1051	–	14	0	1051	–	14
ipfpu	0	1062	–	15	0	1062	–	15	0	1062	–	15	0	1062	–	15
lsm	0	—*			0	1729	–	12	0	1729	–	12	0	1729	–	12
mpm	43	-2585	–	78	43	-2585	–	78	43	-2585	–	78	43	-2585	–	78
pm	0	775	–	33	0	775	–	33	0	775	–	33	0	775	–	33
rrwm	0	744	–	15	0	744	–	15	0	744	–	15	0	744	–	15
sm	0	1086	–	13	0	1086	–	13	0	1086	–	13	0	1086	–	13
smac	1	-1571	–	61	1	-1571	–	61	1	-1571	–	61	1	-1571	–	61
dd-ls0	**100**	**-4293**	**-4294**	100	**100**	**-4293**	-4294	100	**100**	**-4293**	-4294	100	**100**	**-4293**	-4294	100
dd-ls3	**100**	**-4293**	**-4294**	100	**100**	**-4293**	**-4293**	100	**100**	**-4293**	**-4293**	100	**100**	**-4293**	**-4293**	100
dd-ls4	98	-4291	-4297	100	**100**	**-4293**	**-4293**	100	**100**	**-4293**	**-4293**	100	**100**	**-4293**	**-4293**	100
fm-bca	**100**	**-4293**	-4300	100	**100**	**-4293**	-4300	100	**100**	**-4293**	-4300	100	**100**	**-4293**	-4300	100
hbp	97	—*			**100**	**-4293**	-4305	100	**100**	**-4293**	-4305	100	**100**	**-4293**	-4305	100
mp	93	-4280	-4299	99	96	-4285	-4299	99	98	-4289	-4299	100	98	-4289	-4299	100
mp-fw	99	-4292	-4306	100	**100**	**-4293**	-4299	100	**100**	**-4293**	-4296	100	**100**	**-4293**	-4295	100
mp-mcf	90	-4245	-4303	98	93	-4264	-4298	99	95	-4274	-4296	99	97	-4277	-4295	99

opt: optimally solved instances (%); **E:** average best objective value; **D:** average best lower bound if applicable;
acc: average accuracy corresponding to best objective (%); **—*:** method yields no solution for at least one problem instance

Table A8. Detailed fixed-time evaluation for dataset house-dense.

	house-dense (1s)				house-dense (10s)				house-dense (100s)				house-dense (300s)			
	opt	E	D	acc	opt	E	D	acc	opt	E	D	acc	opt	E	D	acc
fgmd	0	—*			0	—*			77	-3542	–	89	77	-3542	–	89
fm	**100**	**-3778**	-4078	100	**100**	**-3778**	-4078	100	**100**	**-3778**	-4078	100	**100**	**-3778**	-4078	100
fw	**100**	**-3778**	–	100	**100**	**-3778**	–	100	**100**	**-3778**	–	100	**100**	**-3778**	–	100
ga	0	3491	–	8	0	3491	–	8	0	3491	–	8	0	3491	–	8
ipfps	0	3654	–	8	0	3654	–	8	0	3654	–	8	0	3654	–	8
ipfpu	0	3659	–	8	0	3659	–	8	0	3659	–	8	0	3659	–	8
lsm	0	—*			0	2391	–	18	0	2391	–	18	0	2391	–	18
mpm	0	1260	–	53	0	1260	–	53	0	1260	–	53	0	1260	–	53
pm	0	3262	–	18	0	3262	–	18	0	3262	–	18	0	3262	–	18
rrwm	0	2895	–	10	0	2895	–	10	0	2895	–	10	0	2895	–	10
sm	0	3789	–	9	0	3789	–	9	0	3789	–	9	0	3789	–	9
smac	0	2817	–	31	0	2817	–	31	0	2817	–	31	0	2817	–	31
dd-ls0	**100**	**-3778**	-3779	100	**100**	**-3778**	-3779	100	**100**	**-3778**	-3779	100	**100**	**-3778**	-3779	100
dd-ls3	**100**	**-3778**	**-3779**	100	**100**	**-3778**	**-3778**	100	**100**	**-3778**	**-3778**	100	**100**	**-3778**	**-3778**	100
dd-ls4	92	-3763	-3795	99	**100**	**-3778**	**-3778**	100	**100**	**-3778**	**-3778**	100	**100**	**-3778**	**-3778**	100
fm-bca	**100**	**-3778**	**-3779**	100	**100**	**-3778**	**-3778**	100	**100**	**-3778**	**-3778**	100	**100**	**-3778**	**-3778**	100
hbp	98	—*			**100**	**-3778**	-3806	100	**100**	**-3778**	-3806	100	**100**	**-3778**	-3806	100
mp	99	-3777	-3782	100	99	-3777	-3780	100	99	-3777	-3780	100	99	-3777	-3780	100
mp-fw	**100**	**-3778**	-3843	100	**100**	**-3778**	-3791	100	**100**	**-3778**	-3779	100	**100**	**-3778**	**-3778**	100
mp-mcf	31	-3542	-3825	89	85	-3732	-3783	98	99	-3776	**-3778**	100	**100**	**-3778**	**-3778**	100

opt: optimally solved instances (%); **E:** average best objective value; **D:** average best lower bound if applicable;
acc: average accuracy corresponding to best objective (%); **—*:** method yields no solution for at least one problem instance

Table A9. Detailed fixed-time evaluation for dataset house-sparse.

	house-sparse (1s)				house-sparse (10s)				house-sparse (100s)				house-sparse (300s)			
	opt	E	D	acc	opt	E	D	acc	opt	E	D	acc	opt	E	D	acc
fgmd	0		—*		**100**	**-67**	–	100	**100**	**-67**	–	100	**100**	**-67**	–	100
fm	**100**	**-67**	-79	100	**100**	**-67**	-79	100	**100**	**-67**	-79	100	**100**	**-67**	-79	100
fw	0	0	–	0	0	0	–	0	0	0	–	0	0	0	–	0
ga	**100**	**-67**	–	100	**100**	**-67**	–	100	**100**	**-67**	–	100	**100**	**-67**	–	100
ipfps	**100**	**-67**	–	100	**100**	**-67**	–	100	**100**	**-67**	–	100	**100**	**-67**	–	100
ipfpu	**100**	**-67**	–	100	**100**	**-67**	–	100	**100**	**-67**	–	100	**100**	**-67**	–	100
lsm	46	-65	–	96	46	-65	–	96	46	-65	–	96	46	-65	–	96
mpm	0	-60	–	90	0	-60	–	90	0	-60	–	90	0	-60	–	90
pm	0	-54	–	83	0	-54	–	83	0	-54	–	83	0	-54	–	83
rrwm	**100**	**-67**	–	100	**100**	**-67**	–	100	**100**	**-67**	–	100	**100**	**-67**	–	100
sm	96	**-67**	–	100	96	**-67**	–	100	96	**-67**	–	100	96	**-67**	–	100
smac	37	-44	–	63	37	-44	–	63	37	-44	–	63	37	-44	–	63
dd-ls0	**100**	**-67**	**-67**	100	**100**	**-67**	**-67**	100	**100**	**-67**	**-67**	100	**100**	**-67**	**-67**	100
dd-ls3	96	**-66**	**-67**	100	**100**	**-67**	**-67**	100	**100**	**-67**	**-67**	100	**100**	**-67**	**-67**	100
dd-ls4	18	-56	-72	86	**100**	**-67**	**-67**	100	**100**	**-67**	**-67**	100	**100**	**-67**	**-67**	100
fm-bca	**100**	**-67**	**-67**	100	**100**	**-67**	**-67**	100	**100**	**-67**	**-67**	100	**100**	**-67**	**-67**	100
hbp	**100**	**-67**	**-67**	100	**100**	**-67**	**-67**	100	**100**	**-67**	**-67**	100	**100**	**-67**	**-67**	100
mp	**100**	**-67**	**-67**	100	**100**	**-67**	**-67**	100	**100**	**-67**	**-67**	100	**100**	**-67**	**-67**	100
mp-fw	**100**	**-67**	**-67**	100	**100**	**-67**	**-67**	100	**100**	**-67**	**-67**	100	**100**	**-67**	**-67**	100
mp-mcf	**100**	**-67**	**-67**	100	**100**	**-67**	**-67**	100	**100**	**-67**	**-67**	100	**100**	**-67**	**-67**	100

opt: optimally solved instances (%); **E:** average best objective value; **D:** average best lower bound if applicable;
acc: average accuracy corresponding to best objective (%); **—*:** method yields no solution for at least one problem instance

Table A10. Detailed fixed-time evaluation for dataset motor.

	motor (1s)				motor (10s)				motor (100s)				motor (300s)			
	opt	E	D	acc	opt	E	D	acc	opt	E	D	acc	opt	E	D	acc
fgmd	0		—*		85	**-63**	–	97	85	**-63**	–	97	85	**-63**	–	97
fm	90	**-63**	-94	93	**100**	**-63**	-94	97	**100**	**-63**	-94	97	**100**	**-63**	-94	97
fw	20	-58	–	70	20	-58	–	70	20	-58	–	70	20	-58	–	70
ga	55	-62	–	90	55	-62	–	90	55	-62	–	90	55	-62	–	90
ipfps	25	-61	–	85	25	-61	–	85	25	-61	–	85	25	-61	–	85
ipfpu	15	-58	–	77	15	-58	–	77	15	-58	–	77	15	-58	–	77
lsm	10	-52	–	64	10	-52	–	64	10	-52	–	64	10	-52	–	64
mpm	5		—*		5	-50	–	56	5	-50	–	56	5	-50	–	56
pm	0	-35	–	32	0	-35	–	32	0	-35	–	32	0	-35	–	32
rrwm	50	-62	–	89	50	-62	–	89	50	-62	–	89	50	-62	–	89
sm	40	-60	–	87	40	-60	–	87	40	-60	–	87	40	-60	–	87
smac	10	-52	–	66	10	-52	–	66	10	-52	–	66	10	-52	–	66
dd-ls0	**100**	**-63**	**-63**	97	**100**	**-63**	**-63**	97	**100**	**-63**	**-63**	97	**100**	**-63**	**-63**	97
dd-ls3	65	-57	-65	87	**100**	**-63**	**-63**	97	**100**	**-63**	**-63**	97	**100**	**-63**	**-63**	97
dd-ls4	30	-52	-68	78	70	-60	-64	93	**100**	**-63**	**-63**	97	**100**	**-63**	**-63**	97
fm-bca	**100**	**-63**	**-63**	97	**100**	**-63**	**-63**	97	**100**	**-63**	**-63**	97	**100**	**-63**	**-63**	97
hbp	95		—*		**100**	**-63**	-65	97	**100**	**-63**	-65	97	**100**	**-63**	-65	97
mp	90	**-63**	**-63**	96	95	**-63**	**-63**	98	95	**-63**	**-63**	98	95	**-63**	**-63**	98
mp-fw	95	**-63**	**-63**	98	**100**	**-63**	**-63**	97	**100**	**-63**	**-63**	97	**100**	**-63**	**-63**	97
mp-mcf	90	**-63**	**-63**	98	95	**-63**	**-63**	99	**100**	**-63**	**-63**	97	**100**	**-63**	**-63**	97

opt: optimally solved instances (%); **E:** average best objective value; **D:** average best lower bound if applicable;
acc: average accuracy corresponding to best objective (%); **—*:** method yields no solution for at least one problem instance

Table A11. Detailed fixed-time evaluation for dataset opengm.

	opengm (1s)			opengm (10s)			opengm (100s)			opengm (300s)		
	opt	E	D	opt	E	D	opt	E	D	opt	E	D
fgmd	0	—*		50	—*		75	-166	–	75	-166	–
fm	**100**	**-171**	-192	**100**	**-171**	-192	**100**	**-171**	-192	**100**	**-171**	-192
fw	0	-152	–	0	-152	–	0	-152	–	0	-152	–
ga	50	-167	–	50	-167	–	50	-167	–	50	-167	–
ipfps	0	-95	–	0	-95	–	0	-95	–	0	-95	–
ipfpu	0	-86	–	0	-86	–	0	-86	–	0	-86	–
lsm	0	-67	–	0	-67	–	0	-67	–	0	-67	–
mpm	0	-94	–	0	-94	–	0	-94	–	0	-94	–
pm	0	-83	–	0	-83	–	0	-83	–	0	-83	–
rrwm	0	-154	–	0	-154	–	0	-154	–	0	-154	–
sm	0	-101	–	0	-101	–	0	-101	–	0	-101	–
smac	0	-84	–	0	-84	–	0	-84	–	0	-84	–
dd-ls0	50	-160	**-172**	75	-161	**-171**	75	-161	**-171**	75	-161	**-171**
dd-ls3	0	-118	-185	**100**	**-171**	**-171**	**100**	**-171**	**-171**	**100**	**-171**	**-171**
dd-ls4	0	-105	-214	25	-141	-178	**100**	**-171**	**-171**	**100**	**-171**	**-171**
fm-bca	75	-170	-177	**100**	**-171**	-177	**100**	**-171**	-177	**100**	**-171**	-177
hbp	0	—*		50	-164	-185	50	-164	-185	50	-164	-185
mp	0	-57	-192	0	-57	-192	0	-57	-192	0	-57	-192
mp-fw	0	-150	-192	0	-150	-192	0	-150	-192	0	-150	-192
mp-mcf	0	-57	-192	0	-57	-192	0	-57	-192	0	-57	-192

opt: optimally solved instances (%); **E:** average best objective value; **D:** average best lower bound if applicable;
acc: average accuracy corresponding to best objective (%); **—*:** method yields no solution for at least one problem instance

Table A12. Detailed fixed-time evaluation for dataset pairs.

	pairs (1s)			pairs (10s)			pairs (100s)			pairs (300s)		
	E	D	acc	E	D	acc	E	D	acc	E	D	acc
fgmd		—*			—*			—*			—*	
fm	-64812	-76418	51	-65625	-76418	54	-65825	-76418	55	-65870	-76418	56
fw	**-65722**	–	54	**-65797**	–	54	-65802	–	54	-65802	–	54
ga		—*			—*			—*			—*	
ipfps		—*			—*		-35115	–	7	-35115	–	7
ipfpu		—*			—*		-35666	–	7	-35666	–	7
lsm		—*			—*			—*			—*	
mpm		—*			—*			—*			—*	
pm		—*			—*			—*			—*	
rrwm		—*			—*			—*			—*	
sm		—*			—*			—*		-196	–	0
smac		—*			—*			—*			—*	
dd-ls0	-61482	-76497	41	-61482	-73521	41	-62974	**-67306**	57	-63454	-67114	60
dd-ls3	-61638	**-75656**	41	-61638	-73528	41	-62426	-67599	50	-64556	**-66704**	60
dd-ls4	-61634	-75852	41	-61634	-74053	41	-61634	-70214	41	-62894	-67352	53
fm-bca		—*		-65567	-70163	55	**-65913**	-69003	58	**-65958**	-68909	58
hbp		—*			—*			—*			—*	
mp		—*		-64150	**-68255**	57	-64380	-68136	57	-64418	-68130	57
mp-fw		—*			—*			—*		-65794	-69623	55
mp-mcf		—*		-63990	-68318	56	-64174	-68053	57	-64208	-67748	57

opt: optimally solved instances (%); **E:** average best objective value; **D:** average best lower bound if applicable;
acc: average accuracy corresponding to best objective (%); **—*:** method yields no solution for at least one problem instance

Table A13. Detailed fixed-time evaluation for dataset worms.

	worms (1s)				worms (10s)				worms (100s)				worms (300s)			
	opt	E	D	acc	opt	E	D	acc	opt	E	D	acc	opt	E	D	acc
fgmd	0		—*		0		—*		0		—*		0		—*	
fm	**93**	-48457	-55757	89	**93**	-48458	-55757	89	**93**	-48461	-55757	89	**93**	-48461	-55757	89
fw	0	-46974	–	81	3	-48032	–	85	3	-48038	–	85	3	-48038	–	85
ga	0		—*		0		—*		0		—*		0		—*	
ipfps	0		—*		0		—*		0	-1147	–	1	0	-1147	–	1
ipfpu	0		—*		0		—*		0	0	–	0	0	0	–	0
lsm	0		—*		0		—*		0		—*		0		—*	
mpm	0		—*		0		—*		0		—*		0		—*	
pm	0		—*		0		—*		0		—*		0		—*	
rrwm	0		—*		0		—*		0		—*		0		—*	
sm	0		—*		0		—*		0	-6453	–	6	0	-6453	–	6
smac	0		—*		0		—*		0		—*		0		—*	
dd-ls0	0	60443	-163870	26	0	50517	-148830	25	0	-3982	-58449	58	0	-43824	-48682	87
dd-ls3	0	64017	-160520	24	0	49257	-144842	24	0	11744	-71523	46	0	-40882	-48943	85
dd-ls4	0	65731	-160409	24	0	58300	-153566	25	0	31066	-109434	33	0	12795	-75088	44
fm-bca	**93**	**-48460**	**-48514**	89	**93**	**-48464**	**-48498**	89	**93**	**-48464**	-48498	89	**93**	**-48464**	-48498	89
hbp	0		—*		0		—*		0		—*		0		—*	
mp	0		—*		67	-48391	**-48498**	89	70	-48393	**-48497**	89	70	-48393	**-48496**	89
mp-fw	0		—*		3		—*		57	-48402	-48759	88	83	-48435	-48631	88
mp-mcf	0		—*		3	-47942	-48588	88	3	-48027	-48558	88	3	-48057	-48552	88

opt: optimally solved instances (%); **E:** average best objective value; **D:** average best lower bound if applicable;
acc: average accuracy corresponding to best objective (%); **—*:** method yields no solution for at least one problem instance