A Comparative Study of Graph Matching Algorithms in Computer Vision

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
✉ stefan.haller@iwr.uni-heidelberg.de

Abstract. The graph matching optimization problem is an essential component for many tasks in computer vision, such as bringing two deformable objects in correspondence. Naturally, a wide range of applicable algorithms have been proposed in the last decades. Since a common standard benchmark has not been developed, their performance claims are often hard to verify as evaluation on differing problem instances and criteria make the results incomparable. To address these shortcomings, we present a comparative study of graph matching algorithms. We create a uniform benchmark where we collect and categorize a large set of existing and publicly available computer vision graph matching problems in a common format. At the same time we collect and categorize the most popular open-source implementations of graph matching algorithms. Their performance is evaluated in a way that is in line with the best practices for comparing optimization algorithms. The study is designed to be reproducible and extensible to serve as a valuable resource in the future.

Our study provides three notable insights: **(i)** popular problem instances are exactly solvable in substantially less than 1 second, and, therefore, are insufficient for future empirical evaluations; **(ii)** the most popular baseline methods are highly inferior to the best available methods; **(iii)** despite the NP-hardness of the problem, instances coming from vision applications are often solvable in a few seconds even for graphs with more than 500 vertices.

Keywords: Graph Matching · Optimization · Benchmark

1 Introduction

Finding correspondences between elements of two discrete sets, such as keypoints in images or vertices of 3D meshes, is a fundamental problem in computer vision and as such highly relevant for numerous vision tasks, including 3D reconstruction [49], tracking [64], shape model learning [29], and image alignment [7], among others. Graph matching [24, 55, 63] is a standard way to address such problems. In graph matching, vertices of the matched graphs correspond to the elements of the discrete sets to be matched. Graph edges define the cost structure of the problem: pairs of matched vertices are penalized in addition to the vertex-to-vertex matchings. This allows to take, e.g., the underlying geometrical relationship between vertices into account, but also makes the optimization problem NP-hard.

Deep graph matching [51] is a modern learning-based approach, that combines neural networks for computing matching costs with combinatorial graph matching algorithms to find a matching. The graph matching algorithm plays a crucial role in this context, as it has to provide high-quality solutions within a limited time budget. The high demand on run-time is also due to back-propagation learning and graph matching minimization being interleaved and executed together many times during training.

Hense, our work focuses on the *optimization* part of the graph matching pipeline. The modeling and learning aspects are beyond its scope. We evaluate a range of existing *open-source* algorithms. Our study compares their performance on a diverse set of computer vision problems. The focus of the evaluation lies on both, speed and objective value of the solution.

Why do we require a benchmark? Dozens of algorithms addressing the graph matching problem have been proposed in the computer vision literature, see, e.g., the surveys [24, 55, 63], and references therein. Most works promise state-of-the-art performance, which is persuasively demonstrated by experimental evaluation. However, **(i)** results from one article are often incomparable to results from another, since different problem instances with different costs are used, even if these instances are based on the same image data; **(ii)** not every existing method is evaluated on all available problem instances, even if open-source code is available. Some methods, especially those with poor performance on many instances, are very popular as baselines, whereas better performing techniques are hardly considered in comparisons; **(iii)** new algorithms are often only evaluated on easy, small-scale problems. This does not provide any information on how these algorithms perform on larger, more difficult problem instances. For these reasons, the field of graph matching has, in our view, not developed as well as it could have done. By providing a reproducible and extensible benchmark we hope to change this in the future. Such a benchmark is of particular importance for the fast-moving field of *deep graph matching*, as it helps to select an appropriate, fast solver for the combinatorial part of the learning pipeline.

Graph matching problem. Let $\mathcal{V}$ and $\mathcal{L}$ be the two finite sets, whose elements we want to match to each other. For each pair $i, j \in \mathcal{V}$ and each pair $s, l \in \mathcal{L}$ a *cost* $c_{is,jl} \in \mathbb{R}$ is given. Each pair can be interpreted as an *edge* between a pair of *vertices* of an underlying graph. This is where the term *graph matching* comes from. Note that direct *vertex-i-to-vertex-s* matching costs are defined by the *diagonal* elements $c_{is,is}$ of the resulting *cost* (or *affinity*) *matrix* $C = (c_{is,jl})$ with $is = (i, s) \in \mathcal{V} \times \mathcal{L}$, $jl = (j, l) \in \mathcal{V} \times \mathcal{L}$. The diagonal elements are referred to as *unary costs*, in contrast to the *pairwise costs* defined by non-diagonal entries. The goal of graph matching is to find a *matching*, or mutual *assignment*, between elements of the sets $\mathcal{V}$ and $\mathcal{L}$ that minimizes the total cost for all pairs of assignments. It can be formulated as the following integer quadratic problem[1]:

$$\min_{x\in\{0,1\}^{\mathcal{V}\times\mathcal{L}}} \sum_{i,j\in\mathcal{V}} \sum_{s,l\in\mathcal{L}} c_{is,jl}\, x_{is}\, x_{jl} \quad \text{s.t.} \begin{cases} \forall i \in \mathcal{V}\colon\ \sum_{s\in\mathcal{L}} x_{is} \leq 1\,, \text{ and} \\ \forall s \in \mathcal{L}\colon\ \sum_{i\in\mathcal{V}} x_{is} \leq 1\,. \end{cases} \tag{1}$$

The vector x defines the matching as $x_{is} = 1$ corresponds to assigning i to s. The inequalities in (1) allow for this assignment to be *incomplete*, i.e., some elements of both sets may remain unassigned. This is in contrast to *complete* assignments, where each element of $\mathcal{V}$ is matched to *exactly one* element of $\mathcal{L}$ and vice versa. Note that complete assignments require $|\mathcal{V}| = |\mathcal{L}|$.

Relation to the quadratic assignment problem. The graph matching problem (1) is closely related to the NP-hard [50] *quadratic assignment problem (QAP)* [12], which is well-studied in operations research [12, 15, 48]. The QAP only considers *complete* assignments, i.e., $|\mathcal{V}| = |\mathcal{L}|$ and equality is required in the constraints in (1). In contrast, in the field of computer vision incomplete assignments are often required in the model to allow for, e.g., outliers or matching of images with different numbers of features. Still, graph matching and the QAP are polynomially reducible to each other, see supplement for a proof.

The most famous QAP benchmark is the QAPLIB [11] containing 136 problem instances. However, the benchmark problems in computer vision (CV) substantially differ from those in QAPLIB both by the feasible set that includes incomplete assignments, as well as by the structure of the cost matrix C: **(i)** CV problems are usually of a general, more expressive *Lawler* form [42], whereas QAPLIB considers factorizable costs $c_{is,jl} = f_{ij} d_{sl}$ known as *Koopmans-Beckmann* form. The latter allows for more efficient specialized algorithms. **(ii)** the cost matrix C in QAPLIB is often dense, whereas in CV problems it is typically sparse, i.e., a large number of entries in C are 0; **(iii)** for CV problem instances the cost matrix C may contain *infinite* costs on the diagonal to prohibit certain vertex-to-vertex mappings; **(iv)** QAPLIB problems are different from an optimization point of view. For instance, while the classical LP relaxation [1] is often quite loose for QAPLIB problem instances, it is tight or nearly tight for typical instances considered in CV.

Consequently, comparison results on QAPLIB and CV instances differ significantly. It is also typical for NP-hard problems that instances coming from different applied areas require different optimization techniques. Therefore, a dedicated benchmarking on the CV datasets is required.

Contributions. Our contribution is three-fold: **(i)** Based on open source data, we collected, categorized and generated 451 *existing* graph matching instances grouped into 11 datasets in a common format. Most graph matching papers use only a small subset of these datasets for evaluation. Our format provides a *ready-to-use* cost matrix C and does not require any image analysis to extract the costs.

[1] For sets A and B the notation $x \in A^B$ denotes a vector x whose coordinates take on values from the set A and are indexed by elements of B, i.e., each element of B corresponds to a value from A.

(ii) We collected and categorized 20 open-source graph matching algorithms and evaluated them on the above datasets. During that we adapted the cost matrix to requirements of particular algorithms where needed. For each method we provide a brief technical description. We did not consider algorithms with no publicly available open source code. **(iii)** To allow our benchmark to grow further, we set up a web site[2] with all results. Our benchmark is reproducible, extensible and follows the best practices of [6]. We will maintain its web-page in the future and welcome scientists to add problem instances as well as algorithms.

Our work significantly excels evaluations in *all* the papers introducing the algorithms we study. This implies also to the largest existing comparison [31] so far. The latter considers only 8 out of the 11 datasets and evaluates 6 algorithms out of our 20.

2 Background to algorithms

In this section we briefly review the main theoretical concepts and building blocks of the considered approaches.

Linearization. In case all pairwise costs are zero, the objective in (1) linearizes to $\sum_{i\in\mathcal{V},s\in\mathcal{L}} c_{is,is}x_{is}$, turning (1) into the *incomplete linear assignment problem (iLAP)*. A typical way of how the iLAP is obtained in existing algorithms is by considering the Taylor expansion of the objective (1) in the vicinity of a given point x. The linear term of this expansion forms the iLAP objective. The iLAP can be reduced to a complete linear assignment problem (LAP) [10], see supplement, and addressed by, e.g., Hungarian [41] or auction [8] algorithms. Below, when we refer to LAP this includes both LAP and iLAP problems.

Birkhoff polytope and permutation matrices. For $|\mathcal{V}| = |\mathcal{L}|$ and $x \in [0,1]^{\mathcal{V}\times\mathcal{L}}$, where $[0,1]$ denotes the closed interval from 0 to 1, the constraints $\sum_{s\in\mathcal{L}} x_{is}=1$, $\forall i \in \mathcal{V}$, and $\sum_{i\in\mathcal{V}} x_{is}=1$, $\forall s \in \mathcal{L}$, define the set of *doubly-stochastic matrices* also known as *Birkhoff polytope*. Its restriction to binary vectors $x \in \{0,1\}^{\mathcal{V}\times\mathcal{L}}$ is called the *set of permutations* or *permutation matrices*.

Inequality to equality transformation. By adding *slack* or *dummy* variables, indexed by #, with zero cost in the objective in (1), the uniqueness constraints in (1) can be rewritten as equalities for $\mathcal{V}^{\#} := \mathcal{V} \mathbin{\dot\cup} \{\#\}$, $\mathcal{L}^{\#} := \mathcal{L} \mathbin{\dot\cup} \{\#\}$, and $x \in [0,1]^{\mathcal{V}^{\#}\times\mathcal{L}^{\#}}$, where $\dot\cup$ is the disjoint union:

$$\mathcal{B} := \left\{x \mid \forall i \in \mathcal{V}\colon\ \textstyle\sum_{s\in\mathcal{L}^{\#}} x_{is} = 1 \quad \text{and} \quad \forall s \in \mathcal{L}\colon\ \sum_{i\in\mathcal{V}^{\#}} x_{is} = 1\right\}. \qquad (2)$$

Here, $x_{i\#} = 1$ (or $x_{\#s} = 1$) means that the node i (or label s) is unassigned. Following [65], we refer to the elements of $\mathcal{B}$ as *doubly-semi-stochastic matrices*.

Doubly-stochastic relaxation. Replacing the *integrality constraints* $x \in \{0,1\}^{\mathcal{V}\times\mathcal{L}}$ in (1) with the respective *box constraints* $x \in [0,1]^{\mathcal{V}\times\mathcal{L}}$ leads to a *doubly-stochastic relaxation* of the graph matching problem.[3] Despite the con-

[2] The web site for the benchmark is available at https://vislearn.github.io/gmbench/.

[3] Strictly speaking, the term *doubly-stochastic* corresponds to the case when equality constraints are considered in (1). In [65] the inequality variant is called *doubly semi-stochastic* but we use *doubly-stochastic* in both cases.

vexity of its feasible set, the doubly-stochastic relaxation is NP-hard because of the non-convexity of its quadratic objective in general [53].

Probabilistic interpretation. Doubly-stochastic relaxations are often motivated from a probabilistic perspective, where the individual matrix entries represent matching probabilities. An alternative probabilistic interpretation is to consider the *product graph* between $\mathcal{V}$ and $\mathcal{L}$, in which the edge weights directly depend on the cost matrix C. This way, graph matching can be understood as selecting reliable nodes in the product graph, e.g., by random walks [17].

Injective and bijective formulations. Assume $|\mathcal{V}| \leq |\mathcal{L}|$. A number of existing approaches consider an asymmetric formulation of the graph matching problem (1), where the uppermost constraint in (1) is exchanged for equality, i.e., $\forall i \in \mathcal{V}\colon \sum_{s\in\mathcal{L}} x_{is} = 1$. We call this formulation *injective*. The strict inequality case $|\mathcal{V}| < |\mathcal{L}|$ is also referred to as an *unbalanced QAP* in the literature. Note that to address problems of the general form (1) by such algorithms, it is necessary to extend the set $\mathcal{L}$ with $|\mathcal{V}|$ dummy elements. This is similar to the reduction from graph matching to QAP described in the supplement. Since availiable implementations of multiple considered algorithms are additionally restricted to the case $|\mathcal{V}| = |\mathcal{L}|$, i.e. to the classical QAP as introduced in Section 1, we adopt the term *bijective* to describe the corresponding algorithms and datasets.

Spectral relaxation. The graph matching objective in (1) can be compactly written as $x^\top Cx$. Instead of the uniqueness constraints in (1) the *spectral relaxation* considers the non-convex constraint $x^\top x = n$. This constraint includes all matchings with exactly n assignments, which is of interest when the total number of assignments n is known, e.g., for the injective formulation where $n = |\mathcal{V}|$. The minimization of $x^\top Cx$ subject to $x^\top x = n$ reduces to an *eigenvector problem*, i.e., finding a vector x corresponding to the smallest eigenvalue of the matrix C. The latter amounts to minimizing a Rayleigh quotient [30].

Path following. Another way to deal with the non-convexity of the graph matching problem is *path-following*, represented by [69] in our study. The idea is to solve a sequence of optimization problems with objective $f_{\alpha^t}(x) = (1 - \alpha^t) f_{\mathrm{cvx}}(x) + \alpha^t f_{cav}(x)$ for α^t, $t \in 1, \ldots, N$, gradually growing from 0 to 1. The (approximate) solution of each problem in the sequence is used as a starting point for the next. The hope is that this iterative process, referred to as *following the convex-to-concave path*, leads to a solution with low objective value for the whole problem. The objective for $\alpha^1 = 0$ is equal to $f_{\mathrm{cvx}}(x)$ and is convex, therefore it can be solved to global optimality. The objective for $\alpha^N = 1$ is equal to $f_{\mathrm{cav}}(x)$ and is concave. Its local optima over the set of doubly-stochastic matrices are guaranteed to be binary, and, therefore, *feasible assignments*, i.e., they satisfy all constraints of (1).

Graphical model representation. The graph matching problem can be represented in the form of a *maximum a posteriori (MAP) inference* problem for discrete graphical models [54], known also as *Markov random field (MRF) energy minimization* and closely related to *valued and weighted constraint satisfaction* problems. As several graph matching works in computer vision [31, 57, 67], use this representation, we provide it below in more detail.

Let $(\mathcal{V},\mathcal{E})$ be an undirected graph, with the finite set $\mathcal{V}$ introduced above being the *set of nodes* and $\mathcal{E} \subseteq \binom{\mathcal{V}}{2}$ being the set of *edges*. For convenience we denote edges $\{i,j\} \in \mathcal{E}$ simply by ij. Let the finite set $\mathcal{L}$ introduced above be the set of *labels*. We associate with each node $i \in \mathcal{V}$ a set $\mathcal{L}_i^{\#} = \mathcal{L}_i \dot\cup \{\#\}$ with $\mathcal{L}_i \subseteq \mathcal{L}$. Like above, $\#$ stands for the *dummy label* distinct from all labels in $\mathcal{L}$ to encode that *no label* is selected. With each label $s \in \mathcal{L}_i^{\#}$ in each node $i \in \mathcal{V}$ the *unary cost* $\theta_{is} := c_{is,is}$ (0 for $s = \#$) is associated. The case $|\mathcal{L}_i| < |\mathcal{L}|$ corresponds to infinite unary costs $c_{is,is} = \infty$, $s \in \mathcal{L}\backslash\mathcal{L}_i$, as the respective assignments can be excluded from the very beginning. Likewise, with each edge $ij \in \mathcal{E}$ and each label pair $sl \in \mathcal{L}_i^{\#} \times \mathcal{L}_j^{\#}$, the pairwise cost $\theta_{is,jl} = c_{is,jl} + c_{jl,is}$ (0 for s or $l = \#$) is associated. The graph $(\mathcal{V},\mathcal{E})$ being undirected implies $ij = ji$ and $\theta_{is,jl} = \theta_{jl,is}$. An edge ij belongs to $\mathcal{E}$ only if there is a label pair $sl \in \mathcal{L}_i \times \mathcal{L}_j$ such that $\theta_{is,jl} \neq 0$. In this way a sparse cost matrix C may translate into a sparse graph $(\mathcal{V},\mathcal{E})$.

The problem of finding an optimal assignment of labels to nodes, equivalent to the *graph matching* problem (1), can thus be stated as

$$\min_{y\in\mathcal{Y}} \Big[E(y) := \sum_{i\in\mathcal{V}} \theta_{iy_i} + \sum_{ij\in\mathcal{E}} \theta_{iy_i,jy_j}\Big] \text{ s. t. } \forall\, i,j \in \mathcal{V}, i \neq j \colon y_i \neq y_j \text{ or } y_i = \# \quad (3)$$

where $\mathcal{Y}$ stands for the Cartesian product $\bigtimes_{i\in\mathcal{V}} \mathcal{L}_i^{\#}$, and $y_i = s$, $s \in \mathcal{L}_i$, is equivalent to $x_{is} = 1$ in terms of (1). Essentially, (3) corresponds to a *MAP inference problem for discrete graphical models* [54] with additional uniqueness constraints for the labels.

ILP representation and LP relaxations. Based on (3) the graph matching problem can be expressed by a linear objective subject to linear and integrality constraints by introducing variables $x_{is,jl} = x_{jl,is}$ for each pair of labels $sl \in \mathcal{L}_i^{\#} \times \mathcal{L}_j^{\#}$ in neighboring nodes $ij \in \mathcal{E}$, and enforcing the equality $x_{is,jl} = x_{is}x_{jl}$ with suitable linear constraints. An *integer linear program (ILP)* formulation of the graph matching problem (1) can then be written as:

$$\min_{x\in\{0,1\}^{\mathcal{J}}} \sum_{i\in\mathcal{V}, s\in\mathcal{L}_i^{\#}} c_{is}x_{is} + \sum_{ij\in\mathcal{E}, sl\in\mathcal{L}_i^{\#}\times\mathcal{L}_j^{\#}} (c_{is,jl} + c_{jl,is})x_{is,jl} \quad (4)$$

$$\forall i \in \mathcal{V}\colon \sum_{s\in\mathcal{L}_i^{\#}} x_{is} = 1\,, \quad \forall s \in \mathcal{L}\colon \sum_{i\in\mathcal{V}} x_{is} \leq 1\,, \quad (5)$$

$$\forall ij \in \mathcal{E},\ l \in \mathcal{L}_j^{\#}\colon \sum_{s\in\mathcal{L}_i^{\#}} x_{is,jl} = x_{jl}\,. \quad (6)$$

Here $\mathcal{J} = \{(i,s)\colon i \in \mathcal{V},\ s \in \mathcal{L}_i^{\#}\} \cup \{(is,jl)\colon ij \in \mathcal{E}, sl \in \mathcal{L}_i^{\#} \times \mathcal{L}_j^{\#}\}$ denotes the set of coordinates of the vector x. The formulation (4)-(6) differs from the standard ILP representation for discrete graphical models by the label uniqueness constraints (5, rightmost). Substitution of the integrality constraints $x \in \{0,1\}^{\mathcal{J}}$ in (4) with the box constraints $x \in [0,1]^{\mathcal{J}}$ results in the respective *LP relaxation*.

Table 1. Method properties. Purely primal heuristics are separated from the dual methods by a horizontal line.

method	IQP	ILP	bijective	non-pos.	0-unary	lineariz.	norm	doubly	spectral	discret.	path fol.	fusion	duality	SGA	BCA	Matlab	C++
fgmd [69]	+		+								+					[68]	
fm [31]		+										+					[32]
fw [62]	+					+		+									[56]
ga [27]	+		+			+		+		+						[20]	
ipfps [46]	+		+	+		+		+		+						[44]	
ipfpu [46]	+		+			+		+		+						[44]	
lsm [33]	+		+	+			+			+						[66]	
mpm [18]	+		+	+			+			+						[19]	
pm [65]	+		+	+	+			+		+						[68]	
rrwm [17]	+		+			+		+		+						[16]	
smac [21]	+		+	+			+		+	+						[20]	
sm [43]	+		+	+			+		+	+						[44]	
dd-ls(0/3/4) [59]		+											+	+			[38]
fm-bca [31]		+										+	+		+		[32]
hbp [67]		+	+										+		+	[66]	
mp(-mcf/-fw) [57]		+											+		+		[56]

Meaning of properties ('+' indicates presence): *IQP*: addresses IQP formulation; *ILP*: addresses ILP formulation; *bijective*: addresses bijective formulation; *non-pos.*: requires non-positive costs, see Remark 1; *0-unary*: requires zero unary costs; *lineariz.*: linearization-based method; *norm*: imposes norm-constraints; *doubly*: addresses doubly-stochastic relaxation; *spectral*: solves spectral relaxation; *discret.*: discretization as in Remark 2; *path fol.*: path following method; *fusion*: utilizes fusion; *duality*: Lagrange duality-based; *SGA*: uses dual sub-gradient ascent; *BCA*: uses dual block-coordinate ascent; *Matlab/C++*: implemented in Matlab/C++ [reference to code]

3 Graph matching algorithms

Below we summarize the graph matching methods that we consider in our comparison, see Table 1 for an overview of their characteristics and references.

3.1 Primal heuristics

Linearization based. These methods are based on iterative linearizations of the quadratic objective (1) derived from its Taylor expansion.

Iterated projected fixed point (`ipfp`) [46] solves on each iteration the LAP obtained through linearization in the vicinity of a current, in general non-integer, assignment. Between iterations the quadratic objective is optimized along the direction to the obtained LAP solution, which yields a new, in general non-integer assignment. We evaluate two versions of `ipfp` which differ by their initialization: `ipfpu` is initialized with $x^0 \in [0,1]^{\mathcal{V}\times\mathcal{L}}$, where $x^0_{is} = 1/\sqrt{N}$ if $c_{is,is} < \infty$, and $x^0_{is} = 0$ otherwise. Here, $N := |\{is \in \mathcal{V}\times\mathcal{L} \mid c_{is,is} < \infty\}|$. `ipfps` starts from the result of the spectral matching `sm` [43] described below.

Graduated assignment (`ga`) [27] optimizes the doubly-stochastic relaxation. On each iteration it approximately solves the LAP obtained through linearization in the vicinity of a current, in general non-integer, assignment utilizing the Sinkhorn algorithm [40] for a given fixed temperature. The obtained approximate solution is used afterwards as the new assignment. The temperature is decreased over iterations to gradually make the solutions closer to integral.

Fast approximate quadratic programming (`fw`) [62] considers the Frank-Wolfe method [25] for optimizing over the set $\mathcal{B}$, c.f. (2). Each iteration first solves a LAP to find the optimum of the linearization at the current solution, followed by a line search in order to find the best convex combination of the current and the new solution. To obtain an integer solution, the objective of the LAP solution is evaluated in each iteration, and the lowest one among all solutions is kept. The initial LAP is based on the unary costs only. The implementation [56] we evaluate is applicable to the general Lawler form of the problem (1), in contrast to the Koopmans-Beckmann form addressed in [62].

Norm constraints based. *Spectral matching* (`sm`) [43] uses a spectral relaxation that amounts to a Rayleigh quotient problem [30] which can be optimized by the power iteration method. Here, each update comprises of a simple matrix multiplication and a subsequent normalization, so that x^t is iteratively updated via $x^{t+1} = -Cx^t / \|Cx^t\|_2$.

Spectral matching with affine constraints (`smac`) [21] is similar to `sm`, but additionally takes into account affine equality constraints that enforce one-to-one matchings. The resulting formulation amounts to a Rayleigh quotient problem under affine constraints, that can efficiently be computed in terms of the eigenvalue decomposition.

Max-pooling matching (`mpm`) [18] resembles `sm`, but it replaces the sum-pooling implemented in terms of the matrix multiplication $-Cx$ in the power iteration update of `SM` by a max-pooling operation. With that, only candidate matches with the smallest costs are taken into account.

Local sparse model (`lsm`) [33] solves the relaxation $\max_x x^\top Cx$, s.t. $||x||_{1,2}^2 = \sum_{i=1}^{|\mathcal{V}|} \left(\sum_{k=1}^{|\mathcal{L}|} |x_{ik}| \right)^2 = 1$, $x \geq 0$. The $l_{1,2}$-norm $||x||_{1,2}$ should encourage the solution of the above relaxation to be sparse in each row when treating x as a matrix. This resembles the sparsity property of permutation matrices, which satisfy $||x||_{1,2} = |\mathcal{V}|$.

Remark 1. All of the norm constraints based algorithms described above require non-positive[4] costs in order to guarantee convergence of the underlying iterative techniques. This condition can be w.l.o.g. assumed for any graph matching problem. The corresponding cost transformation is described in the supplement.

Probabilistic interpretation based. *Reweighted Random Walks Matching* (`rrwm`) [17] interprets graph matching as the problem of selecting reliable nodes in an *association graph*, whose weighted adjacency matrix is given by $-C$. Nodes are selected through a random walk that starts from one node and randomly visits nodes according to a Markov transition matrix derived from the edge weights of the association graph. In order to take into account matching constraints, the authors of [17] consider a reweighted random walk strategy.

Probabilistic matching (`pm`) [65] considers a probabilistic formulation of graph matching in which the quadratic objective is replaced by a relative entropy objective. It is shown that by doing so one can obtain a convex problem formulation via marginalization, which is optimized in terms of an iterative successive projection algorithm.

Remark 2. Most of the primal heuristics considered above aim to optimize the quadratic objective (1) over a continuous set such as, e.g., the Birkhoff polytope. The resulting assignment $x \in \mathbb{R}^{\mathcal{V} \times \mathcal{L}}$ is, therefore, not guaranteed to be integer.

[4] Non-negative in original maximization formulations

As suggested in [17], to obtain an integer assignment we solve a LAP with $(-x)$ treated as the cost matrix. We apply this procedure as a postprocessing step for `ipfp`, `ga`, `sm`, `smac`, `mpm`, `lsm`, `rrwm`, and `pm`. Note that this postprocessing does not change an integer assignment.

Path following based. *Factorized graph matching* (`fgmd`) [69] proposes an efficient factorization of the cost matrix to speed-up computations, and is based on the convex-concave path following strategy, see Section 2. Individual problems from the path are solved with the Frank-Wolfe method [25].

Randomized generation and fusion based. *Fusion moves with a greedy heuristic* (`fm`) [31] is based on the graphical model representation and consists of two parts: A randomized greedy assignment generation, and *fusion* of the assignments. The randomized generator greedily fixes labels in the nodes in a way that minimizes the objective value restricted to the already fixed labels. The fusion procedure merges the current assignment with the next generated one by approximately solving an auxiliary *binary* MAP inference problem utilizing QPBO-I [52]. The merged solution is guaranteed to be at least as good as the two input assignments. This property guarantees monotonic improvement of the objective value.

3.2 Lagrange duality-based techniques

The methods below consider the Lagrange decompositions [28] of the graph matching problem (1) [59], or its graphical model representation (3) [31, 57, 67], and optimize the corresponding dual. The methods differ in the dual optimization and chosen primal solution reconstruction algorithms.

Block-coordinate methods (`hbp`, `mp-*`, `fm-bca`). The works [31, 57, 67] employ a block-coordinate ascent (BCA) technique to optimize the dual problem obtained by relaxing the coupling (6) and label uniqueness constraints (5, rightmost). Since the dual is piece-wise linear, BCA algorithms may not attain the dual optimum, but may get stuck in a sub-optimal fixed point [9, 54].

Although the elementary operations performed by these algorithms are very similar, their convergence speed and attained fixed points differ drastically. In a nutshell, these methods decompose the problem (3) into the graphical model without uniqueness constraints, and the LAP problem, as described, e.g., in [31]. Dual algorithms reparametrize the problem making it more amenable to primal techniques [54]. Table 2 gives an overview of the evaluated combinations for **(i)** optimizing the dual of the graphical model, **(ii)** optimizing the LAP, and **(iii)** obtaining the primal solution from the reparametrized costs[5], which influence the practical performance of BCA solvers. Additionally, `mp-mcf` and `mp-fw` tighten the relaxation by considering triples of graph nodes as subproblems.

Subgradient method (`dd-ls*`). The algorithms denoted as `dd-ls*` with `*` being `0`, `3` or `4` represent different variants of a dual subgradient optimization method [59]. The variant `dd-ls0` addresses the relaxation equivalent to a symmetrized graphical model formulation, see supplement for a description. This is

[5] *Reparametrized* costs are also known as *reduced* costs, e.g., in the simplex tableau.

Table 2. Characterization of dual BCA algorithms.

	(i) graphical model	(ii) LAP	(iii) primal
hbp	MPLP [26]	Hungarian [41]	branch & bound
mp(-mcf)	anisotropic diffusion [54]	network simplex [2]	LAP
mp-fw			fw
fm-bca	MPLP++ [60]	BCA	fm

Algorithms used for optimizing the *graphical model* and *LAP* part, as well as technique used to obtain a *primal* solution. For *LAP* in the *primal* column the solution of the LAP subproblem is reused as feasible assignment. Instead of solving the LAP subproblem, fm-bca performs a series of *BCA* steps wrt. the LAP dual variables.

achieved by considering the Lagrange decomposition of the problem into two graphical models, with $\mathcal{V}$ and $\mathcal{L}$ being the set of nodes, respectively, and a LAP subproblem. The graphical models are further decomposed into acyclic ones, i.e. trees, solvable by dynamic programming, see, e.g., [54, Ch.9]. The *tree decomposition* is not described in [59], and we reconstructed it based on the source code [38] and communication with the authors. As we observed it to be more efficient than the *max-flow subproblems* suggested in the paper [59] the latter were not used in our evaluation.

Variants dd-ls3 and dd-ls4 tighten the relaxation of dd-ls0 by considering *local subproblems* of both graphical models in the decomposition. These are obtained by reducing the node sets $\mathcal{V}$ and $\mathcal{L}$ to 3 or respectively 4 elements inducing a connected subgraph of the graphical model, see [59] for details.

4 Benchmark

Datasets. The 11 datasets we collected for evaluation of the graph matching algorithms stem from applications in computer vision and bio-imaging. All existing graph matching papers use only a subset of these datasets for evaluation purposes. Together these datasets contain 451 problem instances. Table 3 gives an overview of their characteristics. We modified costs in several datasets to make them amenable to some algorithms, see supplement. Our modification results in a constant shift of the objective value for each feasible assignment, and, therefore, does not influence the quality of the solution.

Below we give a brief description of each dataset. Along with the standard computer vision datasets with small-sized problems, hotel, house-dense/sparse, car, motor and opengm with $|\mathcal{V}|$ up to 52, our collection contains the middle-sized problems flow, with $|\mathcal{V}|$ up to 126, and the large-scale worms and pairs problems with $|\mathcal{V}|$ up to 565.

Wide baseline matching (hotel, house-dense/sparse) is based on a series of images of the same synthetic object with manually selected landmarks from different viewing angles based on the work by [14]. For hotel and house-dense we use the same models as in [57] published in [58]. house-sparse consists of the same image pairs as house-dense, but the cost structure is derived following the approach of [67] that results in significantly sparser problem instances. Graphs with the landmarks as nodes are obtained by Delaunay triangulation. The costs are set to $c_{is,jl} = -\exp(-(d_{ij} - d_{sl})^2/2500)A^1_{ij}A^2_{sl}$ where d_{ij}, d_{sl} are Euclidean

Table 3. Dataset properties. A '+' indicates that all problem instances of the dataset have the respective property.

dataset	#inst.	#opt.	bijective	injective	non-pos.	0-unary	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{L}\rvert/\lvert\mathcal{V}\rvert$	density (%)	diagonal dens. (%)	data
caltech-large [17]	9	1		+	+	+	36-219	0.4-2.3	0.55	18.1	[16]
caltech-small [17]	21	12		+	+	+	9-117	0.4-3	0.99	26.8	[16]
car [23, 47]	30	30	+		+		19-49	1	2.9	100	[45]
flow [3, 57]	6	6					48-126	≈ 1	0.39	15.8	[4, 5]
hotel [14, 59]	105	105	+				30	1	12.8	100	[13, 58]
house-dense [14, 59]	105	105	+				30	1	12.6	100	[13, 58]
house-sparse [14, 67]	105	105	+		+	+	30	1	1.5	100	[13]
motor [23, 47]	20	20	+		+		15-52	1	3.8	100	[45]
opengm [36, 39]	4	4	+				19-20	1	74.8	100	[37]
pairs [31, 34]	16	0					511-565	≈ 1	0.0019	3.7	[32]
worms [34]	30	28					558	≈ 2.4	0.00038	1.6	[35]

Meaning of properties:
#inst.: number of problem instances; *#opt.*: number of known optima; *bijective/injective*: bi-/injective assignment is assumed; *non-pos.*: all costs are non-positive; *0-unary*: datasets with zero unary costs; $|\mathcal{V}|$: number of elements in $\mathcal{V}$; $|\mathcal{L}|/|\mathcal{V}|$: ratio of the number of elements in $\mathcal{L}$ to the number of elements in $\mathcal{V}$; *density (%)*: percentage of non-zero elements in the cost matrix C; *diag. dens. (%)*: percentage of non-infinite elements on the diagonal of C; *data*: [references] to problem instances, images, feature coordinates or ground truth.

distances between two landmarks and $A^1 \in \{0,1\}^{\mathcal{V}\times\mathcal{V}}$, $A^2 \in \{0,1\}^{\mathcal{L}\times\mathcal{L}}$ are adjacency matrices of the corresponding graphs. The unary costs are zero.

Keypoint matching (car, motor) contains *car* and *motor*bike images from the PASCAL VOC 2007 Challenge [23] with the features and costs from [47]. We use the instances available from [32].

Large displacement flow (flow) was introduced by [3] for key point matching on scenes with large motion. We use the instances from [32] which use keypoints and costs as in [57].

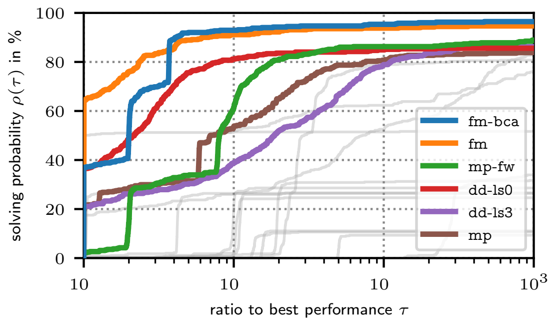
OpenGM matching (opengm) is a set of non-rigid point matching problems by [39], now part of the *OpenGM* Benchmark [36]. We use the instances from [32].

The caltech dataset was proposed in [17]. The data available at the project page [16] contains the *mutual projection error* matrix $D = (d_{is,jl})$, lists of possible assignments, and partial ground truth. We reconstructed the dataset from this data. Unary costs are set to zero. Pairwise costs for pairs of possible assignments are set to $c_{is,jl} = -\max(50 - d_{is,jl}, 0)$. We divided the dataset into caltech-small and caltech-large, where all instances with more than 40000 non-zero pairwise costs are considered as large.

Worm atlas matching (worms) has the goal to annotate nuclei of *C. elegans*, a famous model organism used in developmental biology, by assigning nuclei names from a known atlas of the organism. A detailed description can be found in [34]. We use the instances obtained from [32] which are originally from [35].

Worm-to-worm matching (pairs) directly matches the cell nuclei of individual *C. elegans* worms to each other. The resulting models are much coarser than those of the worms dataset. We consider the same 16 problem instances as [31] using the models from [32].

Evaluation metrics. For *fixed-time* performance evaluation [6] we restrict run-time (1, 10, 100 s) and evaluate attained objective values E, lower bound D and, for datasets with ground truth available, accuracy *acc*. We also report the number of optimally solved instances per dataset.



Methods solving the largest number of instances, see $\rho(\tau = 10^3)$, are highlighted in color. Other methods are shown as "ghosts", i.e., unlabeled in gray. fm is the best solver in 65% of all cases, see $\rho(\tau = 1)$. fm-bca outperforms fm when the allowed performance ratio is increased to $\tau \geq 3.7$. Overall, fm-bca solves ≈97% and fm solves ≈95% of all instances. Following are duality-based methods like mp-fw and dd-ls0.

Fig. 1. Run-time performance profile [22] across all 451 instances.

For *fixed-target* performance evaluation [6] we measure the time $t_s(p)$ until each solver s solves the problem p within an optimality tolerance of 0.1%. For instances with unknown optimum, we consider the best achieved objective value across all methods as optimum as suggested in [6]. The performance ratio to the best solver is computed by $r_s(p) = \frac{t_s(p)}{\min\{t_s(p):\forall s\}}$. We create a performance profile [6, 22] by computing $\rho_s(\tau) = \frac{1}{|P|} \cdot |\{r_s(p) \leq \tau : \forall p\}|$ for each solver s where $|P|$ denotes the total number of problem instances. Intuitively, $\rho_s(\tau)$ is the probability of solver s being at most τ times slower than the fastest solver.

5 Empirical Results

Fixed-time evaluation presented in Table 4 addresses small problem instances, whereas Table 5 addresses mid-size and large problem instances. The performance profile for fixed-target evaluation is presented in Figure 1. More detailed results are available in the supplement. Results have been obtained by taking the minimum run-time across five trials on an AMD EPYC 7702 2.0 GHz processor. Randomized alogrithms were made deterministic by fixing their random seed (`fm` and `fm-bca`). We equally treat Matlab and C++ implementations, in spite of the apparent efficiency considerations, because the solution quality of *all* Matlab algorithms is inferior to the C++ techniques, even if run-time is ignored.

For **small problems** we show results for 1 second in Table 4, as the best methods already solve almost all instances to optimality within this time. The best methods on these datasets are `fm`, `fm-bca` and `dd-ls0`. `dd-ls3/4` have higher costs per iteration, and require more than 1 second to arrive at the solution quality of `dd-ls0`. The other dual BCA-based methods perform almost as good on all but the `opengm` dataset, which seems to be the most difficult dataset amongs the one in Table 4. Apart from `fm` pure primal heuristics are unable to compete with duality-based techniques. The comparison of the results for `house-dense` and `house-sparse` shows that most of the primal heuristics perform much better on sparse problems.

For **larger problems** the most representative times shown in Table 5 are 1, 10 and 100 seconds, depending on the dataset. Again, the duality-based methods

Table 4. Fixed-time evaluation of small problem instances. Maximal run-time per problem instance is 1 second. Boldface marks best values, except for accuracy since algorithms do not optimize it explicitly and do not have access to a ground truth. Horizontal line separates purely primal from duality-based methods. Accuracy omitted for opengm as no ground truth available. Dual bounds omitted as most problems are solved optimally.

	hotel (1s)			house-dense (1s)			house-sparse (1s)			car (1s)			motor (1s)			opengm (1s)		caltech-small (1s)		
	opt	*E*	acc	opt	*E*	acc	opt	*E*	acc	opt	*E*	acc	opt	*E*	acc	opt	*E*	opt	*E*	acc
fgmd	0	—*		0	—*		0	—*		0	—*		0	—*		0	—*	0	—*	
fm	97	-4292	100	**100**	**-3778**	100	**100**	**-67**	100	77	**-69**	88	90	**-63**	93	**100**	**-171**	**52**	-8906	58
fw	97	-4288	99	**100**	**-3778**	100	0	0	0	7	-63	63	20	-58	70	0	-152	0	0	0
ga	0	947	15	0	3491	8	**100**	**-67**	100	57	-68	84	55	-62	90	50	-167	0	—*	
ipfps	0	1051	14	0	3654	8	**100**	**-67**	100	10	-65	80	25	-61	85	0	-95	19	**-8983**	67
ipfpu	0	1062	15	0	3659	8	**100**	**-67**	100	7	-60	69	15	-58	77	0	-86	10	-8829	62
lsm	0	—*		0	—*		46	-65	96	0	-51	52	10	-52	64	0	-67	0	—*	
mpm	43	-2585	78	0	1260	53	0	-60	90	7	—*		5	—*		0	-94	0	—*	
pm	0	775	33	0	3262	18	0	-54	83	0	-35	23	0	-35	32	0	-83	0	-6510	51
rrwm	0	744	15	0	2895	10	**100**	**-67**	100	37	-68	87	50	-62	89	0	-154	5	—*	
sm	0	1086	13	0	3789	9	96	**-67**	100	7	-63	76	40	-60	87	0	-101	0	-3932	36
smac	1	-1571	61	0	2817	31	37	-44	63	0	-52	52	10	-52	66	0	-84	0	-6196	42
dd-ls0	**100**	**-4293**	100	**100**	**-3778**	100	**100**	**-67**	100	**97**	**-69**	91	**100**	**-63**	97	50	-160	43	-7414	58
dd-ls3	**100**	**-4293**	100	**100**	**-3778**	100	96	**-66**	100	47	-57	74	65	-57	87	0	-118	33	-6842	57
dd-ls4	98	-4291	100	92	-3763	99	18	-56	86	3	-49	59	30	-52	78	0	-105	24	-6332	54
fm-bca	**100**	**-4293**	100	**100**	**-3778**	100	**100**	**-67**	100	93	**-69**	92	**100**	**-63**	97	75	-170	38	-8927	62
hbp	97	—*		98	—*		**100**	**-67**	100	77	—*		95	—*		0	—*	0	—*	
mp	93	-4280	99	99	-3777	100	**100**	**-67**	100	80	**-69**	92	90	**-63**	96	0	-57	14	-7967	59
mp-fw	99	-4292	100	**100**	**-3778**	100	**100**	**-67**	100	90	**-69**	91	95	**-63**	98	0	-150	33	-8886	60
mp-mcf	90	-4245	98	31	-3542	89	**100**	**-67**	100	87	**-69**	91	90	**-63**	98	0	-57	5	-7882	60

opt: optimally solved instances (%); **E:** average best objective value; **acc:** average accuracy corresponding to best objective (%)
—*: method yields no solution for at least one problem instance within the given time interval.

and the fm heuristic lead the table. The fm-bca method consistently attains the best or close to best objective and accuracy values on all datasets, whereas its lower bound is often worse than the lower bounds obtained by the mp-* and dd-ls* methods. In contrast, most of the primal heuristics as well as hbp fail, and, for brevity, are omitted in Table 5.

Algorithms dd-ls3/4 consider tighter relaxations than dd-ls0, but are slower, therefore lose in the competition on short time intervals. However, they have the ability to attain the best lower bounds given longer runs ($\gg 100$s).

There is a significant performance gap between the closely related hbp, mp-* and fm-bca methods. Foremost, this is explained by the method for reconstructing the primal solution: The fm algorithm used in the fm-bca solver is solid also as a stand-alone technique, and significantly outperforms the fw and LAP heuristics used in the mp-* algorithms. The branch-and-bound solver used in hbp is quite slow and does not scale well. The second reason for different performance of these methods is the specific BCA algorithm used for the underlying discrete graphical model, c.f. Table 2. According to the recent study [61], which provides a unified treatment of the dual BCA methods for dense[6] graphical models, MPLP ++ performs best, followed by anisotropic diffusion and MPLP as the slowest method. Table 5 shows that there is no solution suitable for every purpose: The speed of fm-bca comes at the price of a looser lower bound. Nonetheless, combining a primal heuristic with a dual optimizer consistently improves upon the

[6] Most of the considered graphical models are dense in terms of [61].

Table 5. Fixed-time evaluation of mid-size and large problem instances. Only the best performing algorithms are shown. Notation is the same as in Table 4. For each dataset the maximum allowed run-time per instance is given in parentheses. For flow no ground truth is available, so the column *acc* is omitted. For caltech-large and pairs no global optima are known, and the column *opt* is omitted.

	flow (1s)			worms (1s)				caltech-large (10s)			caltech-large (100s)			pairs (10s)			pairs (100s)		
	opt	*E*	*D*	opt	*E*	*D*	acc	*E*	*D*	acc	*E*	*D*	acc	*E*	*D*	acc	*E*	*D*	acc
fm	**83**	**-2838**	-3436	**93**	-48457	-55757	89	-34117	-142829	52	-34125	-142829	52	-65625	-76418	54	-65825	-76418	55
fw	67	-2828	–	0	-46974	–	81	0	–	0	0	–	0	**-65797**	–	54	-65802	–	54
dd-ls0	33	-2345	-2968	0	60443	-163870	26	-32973	**-35007**	51	-33539	-34959	52	-61482	-73521	41	-62974	**-67306**	57
dd-ls3	17	-2059	-3030	0	64017	-160520	24	-28653	-42079	49	-33552	**-34914**	49	-61638	-73528	41	-62426	-67599	50
dd-ls4	0	-2062	-3090	0	65731	-160409	24	-25599	-62120	46	-30148	-38880	51	-61634	-74053	41	-61634	-70214	41
fm-bca	**83**	**-2838**	-2898	**93**	**-48460**	**-48514**	89	-34040	-48223	51	-34073	-48217	51	-65567	-70163	55	**-65913**	-69003	58
mp	33	-2628	**-2887**	0		—*		-32017	-46070	48	-32069	-46066	48	-64150	**-68255**	57	-64380	-68136	57
mp-fw	**83**	—*		0		—*		**-34237**	-48882	51	**-34277**	-45923	51		—*			—*	
mp-mcf	33	-2521	-2892	0		—*		-30362	-46630	47	-30737	-43833	47	-63990	-68318	56	-64174	-68053	57

opt, E, acc, —*: same as in Table 4; **D:** best attained lower bound if applicable, i.e., for dual methods, otherwise –

results obtained by the heuristic alone. This holds for fm, but the effect is even more pronounced for the fw and LAP heuristics.

The fixed-target evaluation in Figure 1 confirms that the fm and fm-bca method are amongst the best performing solvers. While fm-bca uses fm as primal heuristics with additional dual BCA updates, the overhead of the latter is visible. After increasing the allowed performance ratio for fm-bca to a factor of 3.7, we can expect better solutions than fm alone. Other top performers are duality-based algorithms with mp-fw and dd-ls0 being the closest followers.

6 Conclusions

Our evaluation shows that: **(i)** Most instances from the popular datasets hotel, house, car and motor can be solved to optimality in well below a second by several optimization techniques. opengm can also be solved to optimality in under a second, although it turns out to be hard for many methods. Therefore, we argue that *these datasets alone are not sufficient anymore to empirically show efficiency of new algorithms.* The most difficult in our collection are the datasets caltech-* and pairs. For a comprehensive evaluation of new methods more datasets are required. **(ii)** The most popular comparison baselines like ipfp, ga, rrwm, pm, sm, smac, lsm, mpm and fgmd are not competitive, and, therefore, *comparison to these alone should not anymore be considered as sufficient.* **(iii)** The most efficient methods are duality-based techniques equipped with efficient primal heuristics. In particular, the fm/fm-bca method currently attains the best or nearly best objective values for most problem instances in the shortest time. **(iv)** Although being NP-hard in general, the graph matching problem can be often efficiently solved in computer vision practice. For many of the considered datasets, including those with $|\mathcal{L}| > 1000$ and $|\mathcal{V}| > 500$, a reasonable approximate solution can be attained in less than a second.

Acknowledgements. This work was supported by the DFG grant SA 2640/2-1 and the Helmholtz Information & Data Science School for Health. We thank the ZIH at TU Dresden for providing high performance computing resources.

References

1. Adams, W.P., Johnson, T.A.: Improved Linear Programming-based Lower Bounds for the Quadratic Assignment Problem. Discrete Mathematics and Theoretical Computer Science (1994)
2. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms and Applications. Prentice Hall (1993)
3. Alhaija, H.A., Sellent, A., Kondermann, D., Rother, C.: GraphFlow – 6D Large Displacement Scene Flow via Graph Matching. In: Proceedings of the DAGM German Conference on Pattern Recognition (2015)
4. Alhaija, H.A., Sellent, A., Kondermann, D., Rother, C.: Graph Matching Problems for GraphFlow – 6D Large Displacement Scene Flow Problem Instances (2018), https://research-explorer.app.ist.ac.at/record/5573
5. Alhaija, H.A., Sellent, A., Kondermann, D., Rother, C.: Project GraphFlow – 6D Large Displacement Scene Flow Images (2018), https://hci.iwr.uni-heidelberg.de/vislearn/research/image-matching/graphflow/
6. Beiranvand, V., Hare, W., Lucet, Y.: Best practices for comparing optimization algorithms. Optimization and Engineering (2017)
7. Bernard, F., Thunberg, J., Gemmar, P., Hertel, F., Husch, A., Goncalves, J.: A Solution for Multi-Alignment by Transformation Synchronisation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2015)
8. Bertsekas, D.P.: A Distributed Algorithm for the Assignment Problem. Lab. for Information and Decision Systems Working Paper, MIT (1979)
9. Bertsekas, D.P.: Nonlinear programming, second edition. Athena scientific (1999)
10. Burkard, R., Dell'Amico, M., Martello, S.: Assignment Problems. SIAM (2009)
11. Burkard, R., Karisch, S., Rendl, F.: QAPLIB – A Quadratic Assignment Problem Library. Journal of Global Optimization (1997)
12. Burkard, R.E., Cela, E., Pardalos, P.M., Pitsoulis, L.S.: The Quadratic Assignment Problem, pp. 1713–1809. Springer (1998)
13. Caetano, T.: Data for Learning Graph Matching (2011), https://www.tiberiocaetano.com/data/
14. Caetano, T.S., McAuley, J.J., Cheng, L., Le, Q.V., Smola, A.J.: Learning Graph Matching. IEEE Transactions on Pattern Analysis and Machine Intelligence (2009)
15. Cela, E.: The Quadratic Assignment Problem: Theory and Algorithms, vol. 1. Springer Science & Business Media (2013)
16. Cho, M., Jungmin, L., Kyoung, M.L.: Reweighted Random Walks for Graph Matching: Project Page (2010), https://cv.snu.ac.kr/research/~RRWM/
17. Cho, M., Lee, J., Lee, K.M.: Reweighted Random Walks for Graph Matching. In: Proceedings of the European Conference on Computer Vision (2010)
18. Cho, M., Sun, J., Duchenne, O., Ponce, J.: 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 (2014)
19. Cho, M., Sun, J., Duchenne, O., Ponce, J.: Finding Matches in a Haystack Source Code (2014), https://www.di.ens.fr/willow/research/maxpoolingmatching/
20. Cour, T.: Graph Matching Toolbox in MATLAB (2010), http://www.timotheecour.com/software/graph_matching/graph_matching.html
21. Cour, T., Srinivasan, P., Shi, J.: Balanced Graph Matching. In: Advances in Neural Information Processing Systems (2007)
22. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Mathematical Programming (2002)

23. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2007 Results (2007), http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html
24. Foggia, P., Percannella, G., Vento, M.: Graph Matching and Learning in Pattern Recognition in the last 10 Years. International Journal of Pattern Recognition and Artificial Intelligence (2014)
25. Frank, M., Wolfe, P., et al.: An Algorithm for Quadratic Programming. Naval Research Logistics Quarterly (1956)
26. Globerson, A., Jaakkola, T.S.: Fixing Max-Product: Convergent Message Passing Algorithms for MAP LP-relaxations. In: Advances in Neural Information Processing Systems (2008)
27. Gold, S., Rangarajan, A.: A Graduated Assignment Algorithm for Graph Matching. IEEE Transactions on Pattern Analysis and Machine Intelligence (1996)
28. Guignard, M., Kim, S.: Lagrangean Decomposition: A Model Yielding Stronger Lagrangean Bounds. Mathematical Programming (1987)
29. Heimann, T., Meinzer, H.P.: Statistical Shape Models for 3D Medical Image Segmentation: A Review. Medical Image Analysis (2009)
30. Horn, R.A., Johnson, C.R.: Matrix Analysis. Cambridge University Press (2012)
31. Hutschenreiter, L., Haller, S., Feineis, L., Rother, C., Kainmüller, D., Savchynskyy, B.: Fusion Moves for Graph Matching. In: Proceedings of the IEEE International Conference on Computer Vision (2021)
32. Hutschenreiter, L., Haller, S., Feineis, L., Rother, C., Kainmüller, D., Savchynskyy, B.: Fusion Moves for Graph Matching Website (2021), https://vislearn.github.io/libmpopt/iccv2021/
33. Jiang, B., Tang, J., Ding, C., Luo, B.: A Local Sparse Model for Matching Problem. In: Proceedings of the AAAI Conference on Artificial Intelligence (2015)
34. Kainmueller, D., Jug, F., Rother, C., Myers, G.: Active Graph Matching for Automatic Joint Segmentation and Annotation of C. elegans. In: Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention (2014)
35. Kainmueller, D., Jug, F., Rother, C., Myers, G.: Graph Matching Problems for Annotating C. elegans (2017), https://doi.org/10.15479/AT:ISTA:57
36. Kappes, J.H., Andres, B., Hamprecht, F.A., Schnörr, C., Nowozin, S., Batra, D., Kim, S., Kausler, B.X., Kröger, T., Lellmann, J., Komodakis, N., Savchynskyy, B., Rother, C.: A Comparative Study of Modern Inference Techniques for Structured Discrete Energy Minimization Problems. International Journal of Computer Vision (2015)
37. Kappes, J.H., Andres, B., Hamprecht, F.A., Schnörr, C., Nowozin, S., Batra, D., Kim, S., Kausler, B.X., Kröger, T., Lellmann, J., Komodakis, N., Savchynskyy, B., Rother, C.: OpenGM Benchmark (2015), http://hciweb2.iwr.uni-heidelberg.de/opengm/index.php?l0=benchmark
38. Kolmogorov, V.: Feature Correspondence via Graph Matching Source Code (2015), https://pub.ist.ac.at/~vnk/software.html#GRAPH-MATCHING
39. Komodakis, N., Paragios, N.: Beyond Loose LP-Relaxations: Optimizing MRFs by Repairing Cycles. In: Proceedings of the European Conference on Computer Vision (2008)
40. Kosowsky, J., Yuille, A.: The Invisible Hand Algorithm: Solving the Assignment Problem with Statistical Physics. Neural Networks (1994)
41. Kuhn, H.W.: The Hungarian Method for the Assignment Problem. Naval Research Logistics Quarterly (1955)

42. Lawler, E.L.: The Quadratic Assignment Problem. Management Science (1963)
43. Leordeanu, M., Hebert, M.: A Spectral Technique for Correspondence Problems Using Pairwise Constraints. In: Proceedings of the IEEE International Conference on Computer Vision (2005)
44. Leordeanu, M.: Efficient Methods for Graph Matching and MAP Inference (2013), https://sites.google.com/site/graphmatchingmethods/
45. Leordeanu, M., Hebert, M.: Cars and Motor Models, https://datasets.d2.mpi-inf.mpg.de/discrete_cv_problems/car_motor_graph_matching.zip
46. Leordeanu, M., Hebert, M., Sukthankar, R.: An Integer Projected Fixed Point Method for Graph Matching and MAP Inference. In: Advances in Neural Information Processing Systems (2009)
47. Leordeanu, M., Sukthankar, R., Hebert, M.: Unsupervised Learning for Graph Matching. International Journal of Computer Vision (2012)
48. Loiola, E.M., Maia de Abreu, N.M., Boaventura-Netto, P.O., Hahn, P., Querido, T.: An Analytical Survey for the Quadratic Assignment Problem. European Journal of Operational Research (2007)
49. Ma, J., Jiang, X., Fan, A., Jiang, J., Yan, J.: Image Matching from Handcrafted to Deep Features: A Survey. International Journal of Computer Vision (2021)
50. Pardalos, P.M., Rendl, F., Wolkowicz, H.: The Quadratic Assignment Problem - A Survey and Recent Developments. Quadratic Assignment and Related Problems (1993)
51. Rolínek, M., Swoboda, P., Zietlow, D., Paulus, A., Musil, V., Martius, G.: Deep Graph Matching via Blackbox Differentiation of Combinatorial Solvers. In: Proceedings of the European Conference on Computer Vision (2020)
52. Rother, C., Kolmogorov, V., Lempitsky, V.S., Szummer, M.: Optimizing Binary MRFs via Extended Roof Duality. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2007)
53. Sahni, S.: Computationally Related Problems. SIAM Journal on Computing (1974)
54. Savchynskyy, B.: Discrete Graphical Models – An Optimization Perspective. Foundations and Trends in Computer Graphics and Vision (2019)
55. Sun, H., Zhou, W., Fei, M.: A Survey On Graph Matching in Computer Vision. In: International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (2020)
56. Swoboda, P.: LPMP Source Code (2021), https://github.com/LPMP/LPMP
57. Swoboda, P., Rother, C., Abu Alhaija, H., Kainmuller, D., Savchynskyy, B.: A Study of Lagrangean Decompositions and Dual Ascent Solvers for Graph Matching. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017)
58. Torresani, L., Kolmogorov, V., Rother, C.: Hotel and House-sparse Models, https://datasets.d2.mpi-inf.mpg.de/discrete_cv_problems/graph_matching_hotel_house.zip
59. Torresani, L., Kolmogorov, V., Rother, C.: A Dual Decomposition Approach to Feature Correspondence. IEEE Transactions on Pattern Analysis and Machine Intelligence (2013)
60. Tourani, S., Shekhovtsov, A., Rother, C., Savchynskyy, B.: MPLP++: Fast, Parallel Dual Block-Coordinate Ascent for Dense Graphical Models. In: Proceedings of the European Conference on Computer Vision (2018)
61. Tourani, S., Shekhovtsov, A., Rother, C., Savchynskyy, B.: Taxonomy of Dual Block-Coordinate Ascent Methods for Discrete Energy Minimization. In: Proceedings of the Conference on Artifical Intelligence and Statistics (2020)

62. Vogelstein, J.T., Conroy, J.M., Lyzinski, V., Podrazik, L.J., Kratzer, S.G., Harley, E.T., Fishkind, D.E., Vogelstein, R.J., Priebe, C.E.: Fast Approximate Quadratic Programming for Graph Matching. PLOS ONE (2015)
63. Yan, J., Yin, X.C., Lin, W., Deng, C., Zha, H., Yang, X.: A Short Survey of Recent Advances in Graph Matching. In: Proceedings of the ACM International Conference on Multimedia Retrieval (2016)
64. Yilmaz, A., Javed, O., Shah, M.: Object Tracking: A Survey. ACM Computing (2006)
65. Zass, R., Shashua, A.: Probabilistic Graph and Hypergraph Matching. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2008)
66. Zhang, Z.: HungarianBP: Pairwise Matching Through Max-Weight Bipartite Belief Propagation Source Code (2016), https://github.com/zzhang1987/HungarianBP
67. Zhang, Z., Shi, Q., McAuley, J., Wei, W., Zhang, Y., van den Hengel, A.: Pairwise Matching Through Max-Weight Bipartite Belief Propagation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2016)
68. Zhou, F.: Implementation of Factorized Graph Matching (2018), https://github.com/zhfe99/fgm
69. Zhou, F., la Torre, F.D.: Factorized Graph Matching. IEEE Transactions on Pattern Analysis and Machine Intelligence (2016)