Fusion Moves for Graph Matching

Lisa Hutschenreiter*, Stefan Haller*, Lorenz Feineis*,
Carsten Rother*, Dagmar Kainmüller†, Bogdan Savchynskyy*

* Heidelberg University, † Max Delbrück Center for Molecular Medicine, Berlin

Abstract

We contribute to approximate algorithms for the quadratic assignment problem also known as graph matching. Inspired by the success of the fusion moves technique developed for multilabel discrete Markov random fields, we investigate its applicability to graph matching. In particular, we show how fusion moves can be efficiently combined with the dedicated state-of-the-art dual methods that have recently shown superior results in computer vision and bio-imaging applications. As our empirical evaluation on a wide variety of graph matching datasets suggests, fusion moves significantly improve performance of these methods in terms of speed and quality of the obtained solutions. Our method sets a new state-of-the-art with a notable margin with respect to its competitors.

1. Introduction

The *quadratic assignment problem* also known as *graph matching* is one of the most prominent combinatorial problems having numerous applications. In computer vision it is predominantly used for feature matching [47]. The modern approach to this application is *deep graph matching*, see *e.g*. [40, 42, 55], which enjoys constantly growing attention in the community. As follows from the name, deep graph matching combines neural networks with combinatorial matching techniques for inference and joint learning. Whereas most of earlier deep graph matching approaches [17, 24, 51, 52, 53] employed a linear assignment problem (LAP) solver[1] to obtain matchings in their pipeline, the most promising state-of-the-art method [40] uses a fully featured graph matching solver. Being called in a loop on each training iteration, this solver must provide high-quality solutions within a very restricted time budget, typically less than a second. Modern state-of-the-art methods [46, 47, 56] satisfy this requirement only if applied to relatively small problems, with a few dozen feature points at most. Hence, scalability of deep graph matching critically depends on the existence of highly-efficient graph matching solvers.

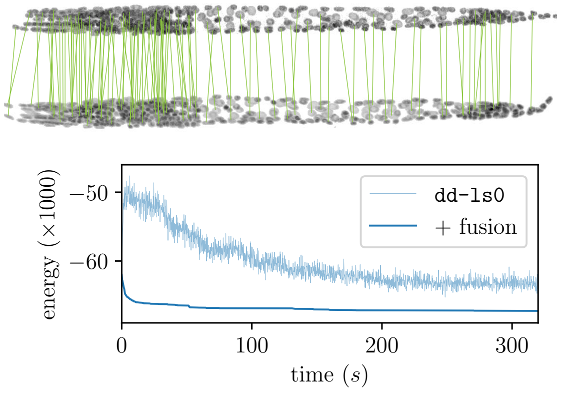

Figure 1. **(Top)** Scalable graph matching is especially important for bio-imaging, where hundreds or even thousands of cells on different images must be matched to each other. An instance from the `pairs` dataset (see Sec. 6), only each 5th matching is shown. **(bottom)** Convergence of the state-of-the-art method `dd-ls0` [47] (see Sec. 6) without and with fusion moves. Note that fusion moves attain much better energy in notably shorter time.

In this work we address this problem by introducing a new graph matching technique, which notably improves the state-of-the-art in terms of speed and attained accuracy. In particular, it provides highly accurate solutions for problems with more than 500 features in less than a second.

Related work. First formulated in 1957 [6], the graph matching problem plays a central role in combinatorial optimization. Due to its importance, nearly all possible optimization techniques were put to the test for it, see the surveys [10, 12, 38] and references therein.

As usual for NP-hard problems, no single method can efficiently address all graph matching instances. Different applications require different methods, and we concentrate

Acknowledgements. This work was supported by the European Research Council (ERC European Unions Horizon 2020 research and innovation program, grant 647769), the German Research Foundation (Exact Relaxation-Based Inference in Graphical Models, DFG SA 2640/1-1), and the Helmholtz Information & Data Science School for Health (HIDSS4Health). The computations were performed on an HPC Cluster at the Center for Information Services and High Performance Computing (ZIH) at TU Dresden.

[1] A polynomial subclass of graph matching without quadratic costs.

here on problem instances specific to computer vision. Traditionally, within this community primal heuristics[2] [1, 14, 19, 25, 35, 36, 50, 54, 57, 58] were used predominantly, since demand for low computation times usually dominates the need for optimality guarantees. These also include methods that build upon spectral relaxations [35, 36, 50, 57], or convex-to-concave path-following procedures [7, 15, 58]. However, recent works [46, 47, 56] have shown that Lagrange duality-based methods attain significantly better accuracy, especially as problem size and complexity grow. It is important to note that in operations research such Lagrange dual methods for graph matching are known at least since the 90s [2, 22, 30], and are widely used in branch-and-bound solvers. Although they address similar relaxations as [46, 56], their iteration complexity is an order of magnitude higher than those of [46, 56]. This makes them prohibitively expensive for use in typical computer vision applications.

While branch-and-bound remains the main tool to obtain exact solutions, it has an exponential worst-case complexity and is often too expensive. Hence, dual methods like [46, 47] use simple primal heuristics with low computational cost that can be called after each dual iteration. Improving such heuristics to obtain high-quality primal solutions already after few dual iterations would allow to outperform the purely primal methods not only in accuracy but also in runtime.

Fusion moves, as introduced by [34], is a primal heuristic proposed for maximum a posteriori inference in Markov random fields, known also as *discrete labeling* or *energy minimization problem*, see *e.g.* [43]. For brevity we will refer to it as the *MRF problem*.

In its most common setting the fusion moves method tries to improve a current approximate primal assignment by merging it with another assignment proposal. The merging constitutes a comparatively small two-label MRF problem, for which efficient exact and approximate techniques exist. As noted in [34], success of the method significantly depends on the *quality* and *diversity* of proposals. A number of ways of generating generic proposals for MRF problems and (approximate) solvers for the corresponding auxiliary problem have been evaluated by [29]. They also considered several instances of the graph matching problem treated as MRF. However, they found fusion moves with their, typically infeasible, proposals to be inferior to other methods. A similar negative result was reported by [47] with a simple but low quality local search-based proposal generator.

In operations research fusion moves is known since 1997 as *optimized crossover* or *recombination*, when it was proposed to address the independent set problem [3]. However, for the quadratic assignment problem it was reported as being inefficient, when used as a building block of a greedy genetic algorithm [4]. This was attributed to the lack of diversity of the solution population resulting from this method.

[2]A common name for algorithms missing optimality guarantees.

Contribution. We show how to use fusion moves to efficiently solve graph matching problems, and provide a theoretical rationale that efficient proposals for fusion moves must be feasible, *i.e.* satisfy the uniqueness constraints of the graph matching problem. We ensure *quality* of our proposals by generating them based on reparametrized costs improved in the course of dual optimization, and enforce *diversity* of proposals by making use of either *oscillating* dual updates, as in the dual subgradient method, or our proposed efficient *randomized* greedy algorithm. Altogether, our method combines the accuracy of dual solvers with the speed of dedicated primal heuristics. We demonstrate the superior performance of our technique on multiple datasets. Our code and datasets we used are available at https://vislearn.github.io/libmpopt/iccv2021.

The supplement, referred to as §A1–§A7 contains detailed proofs, dataset, experiment and algorithm descriptions.

2. Preliminaries

Graph matching problem. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph, where $\mathcal{V}$ is the finite set of *nodes* and $\mathcal{E} \subseteq \binom{\mathcal{V}}{2}$ the set of *edges*. For convenience we denote edges $\{u, v\} \in \mathcal{E}$ simply by uv. Let $\mathcal{L}$ be a finite set of *labels*. We associate with each node $u \in \mathcal{V}$ a subset of labels $\mathcal{L}_u \subseteq \mathcal{L}$, and a *unary cost function* $\theta_u \colon \mathcal{L}_u^\# \to \mathbb{R}$, where $\mathcal{L}_u^\# := \mathcal{L}_u \cup \{\#\}$. Here, $\#$ denotes a *dummy label* distinct from all labels in $\mathcal{L}$ to encode that no label is selected. Likewise, for each edge $uv \in \mathcal{E}$ let $\theta_{uv} \colon \mathcal{L}_u^\# \times \mathcal{L}_v^\# \to \mathbb{R}$ be a *pairwise cost function*.

Then the problem of finding an optimal assignment of labels to nodes, known as *graph matching* or *quadratic assignment* problem, can be stated as

$$\min_{x \in X} \left[E(x) := \sum_{u \in \mathcal{V}} \theta_u(x_u) + \sum_{uv \in \mathcal{E}} \theta_{uv}(x_u, x_v) \right] \tag{1}$$
$$\text{s.t.} \quad \forall\, u, v \in \mathcal{V}, u \neq v : \; x_u \neq x_v \text{ or } x_u = \#,$$

where X stands for the Cartesian product $\bigtimes_{u \in \mathcal{V}} \mathcal{L}_u^\#$. The objective E is referred to as *energy*, and the constraints in (1) are known as *uniqueness constraints*. They allow each non-dummy label to be selected at most once. The number of selected dummy labels is not limited. Elements $x \in X$ are called *assignments*. An assignment is *feasible* if it satisfies all uniqueness constraints. So, essentially, (1) corresponds to an *MRF problem with uniqueness constraints* for the labels.

Note that this formulation generalizes the classical quadratic assignment problem, see *e.g.* [10], by allowing for *incomplete* assignments, *i.e.* not every label in $\mathcal{L}$ has to be assigned to a node, and not necessarily every node is assigned a label in $\mathcal{L}$. Instead, nodes can be assigned the dummy label. Choosing a large constant as unary cost for the dummy label in each node enforces a *complete* assignment.

Without pairwise costs θ_{uv} the quadratic assignment problem (1) reduces to the well-known *linear assignment problem* (LAP). While the quadratic assignment problem is in general NP-hard, the LAP can be solved in polynomial time by *e.g.* the Hungarian method.

Fusion moves [34] address the, compared to (1) unconstrained, MRF problem $\min_{x\in X} E(x)$. In the simplest, but most widely used scenario, on each iteration of the algorithm the currently best assignment $x' \in X$ is *fused* with another candidate assignment $x'' \in X$ by solving the *auxiliary* minimization problem

$$\min_{x\in X_{\text{aux}}} E(x)\,, \tag{2}$$

where $X_{\text{aux}} := \{x \in X \mid x_u \in \{x'_u, x''_u\}, u \in \mathcal{V}\}$. Due to the considerably smaller size of the restricted label space X_{aux} the auxiliary problem (2) can often be efficiently solved approximately, or even exactly. The solver only has to guarantee *monotone improvement* of the best assignment by assuring

$$E(x^*) \le \min(E(x'), E(x'')) \tag{3}$$

for its output x^*, which is then further considered as the best assignment, *i.e.* in the next iteration $x' := x^*$. Note that the monotonicity condition (3) automatically holds for any x^* that is an exact solution of (2). For approximate methods the inequality (3) can be enforced by assigning x^* to the proposal with lower energy if needed. Each fusion operation is also referred to as a *fusion move*.

We adopt this method to the graph matching problem (1) by extending the auxiliary problem (2):

$$\begin{aligned}&\min_{x\in X_{\text{aux}}} E(x) \\ &\text{s.t.}\quad \forall\, u,v \in \mathcal{V}, u \ne v:\ x_u \ne x_v \text{ or } x_u = \#\,.\end{aligned} \tag{4}$$

That is, compared to (2), the uniqueness constraints are taken into account during fusion, which guarantees feasibility of the current best assignment.

There are two main questions that have to be answered to apply fusion moves: **(i)** How to generate proposals? **(ii)** How to solve the auxiliary problem (4)? Starting with the second, we address these questions below.

3. Solving the auxiliary problem

ILP formulation. The auxiliary problem (4) can be formulated as an *integer linear program* (ILP) as follows. For all $u \in \mathcal{V}$ let $\hat{\mathcal{L}}_u := \{x'_u, x''_u\}$ be the restricted set of labels.[3] We introduce binary variables $\mu_{u,s} \in \{0,1\}$ for each node $u \in \mathcal{V}$ and label $s \in \hat{\mathcal{L}}_u$, and $\mu_{uv,st} \in \{0,1\}$ for each edge $uv \in \mathcal{E}$ and each label pair $(s,t) \in \hat{\mathcal{L}}_u \times \hat{\mathcal{L}}_v$. Setting $\mu_{u,s} = \mu_{v,t} = \mu_{uv,st} = 1$ corresponds to assigning coordinates $x_u = s$ and $x_v = t$ of the solution labeling x. Together these variables form a vector $\mu \in \{0,1\}^N$, where $N = 2|\mathcal{V}| + 4|\mathcal{E}|$. Then the ILP

$$\min_{\mu\in\{0,1\}^N} \sum_{\substack{u\in\mathcal{V}\\ s\in\hat{\mathcal{L}}_u}} \mu_{u,s}\theta_u(s) + \sum_{\substack{uv\in\mathcal{E}\\ (s,t)\in\hat{\mathcal{L}}_u\times\hat{\mathcal{L}}_v}} \mu_{uv,st}\theta_{uv}(s,t) \tag{5}$$

$$\begin{aligned}\text{s.t. } &\forall\, u \in \mathcal{V}\colon \mu_{u,x'_u} + \mu_{u,x''_u} = 1\\ &\forall\, uv \in \mathcal{E}, (s,t) \in \hat{\mathcal{L}}_u \times \hat{\mathcal{L}}_v\colon\\ &\quad \mu_{uv,st} \le \mu_{u,s},\ \mu_{uv,st} \le \mu_{v,t}\,,\\ &\quad \mu_{uv,st} \ge \mu_{u,s} + \mu_{v,t} - 1\,,\\ &\forall\, u,v \in \mathcal{V}, u \ne v, s \in (\hat{\mathcal{L}}_u \cap \hat{\mathcal{L}}_v)\backslash\{\#\}\colon \qquad (6)\\ &\quad \mu_{u,s} + \mu_{v,s} \le 1\end{aligned}$$

is equivalent to (4). In particular, the inequalities in (6) enforce the uniqueness constraints. Clearly, problem (5) without the uniqueness constraints (6) constitutes an ILP representation of the MRF auxiliary problem (2).

The ILP problem (5)-(6) can be addressed by off-the-shelf ILP solvers like Gurobi [21]. However, with growing problem size, such solvers become prohibitively slow, as they have exponential worst-case complexity. Therefore, one has to resort to other exact or approximate optimization techniques, which we review now.

Elimination of uniqueness constraints. The uniqueness constraints (6) between nodes u and v can be eliminated by assigning a very large cost C_∞ to the pairwise cost function on the corresponding edge, *i.e.*

$$\theta_{uv}(s,s) := C_\infty,\ \forall s \in (\hat{\mathcal{L}}_u \cap \hat{\mathcal{L}}_v)\backslash\{\#\}\,. \tag{7}$$

If $uv \notin \mathcal{E}$, the edge uv is added to $\mathcal{E}$ together with pairwise costs $\theta_{uv}(s,t) := C_\infty \cdot [\![s = t \ne \#]\!]$, where $[\![A]\!]$ is equal to 1 if A holds, and 0 otherwise.

This way the graph matching auxiliary problem (4) is reduced to the MRF auxiliary problem (2), on a, possibly different, graph. This allows considering dedicated methods addressing the MRF auxiliary problem (2). Efficiency of these methods is very much dependent on the *submodularity* of the pairwise costs θ_{uv}. We review this property and the corresponding optimization methods below.

Submodular case. In general, two-label MRF problems like (2) are NP-hard [8]. However, they become efficiently solvable, if for all $u \in \mathcal{V}$ there exists a bijective mapping $\delta_u\colon \{0,1\} \to \hat{\mathcal{L}}_u$ called *ordering*, such that all pairwise costs θ_{uv}, $uv \in \mathcal{E}$, in problem (2) are *submodular*, *i.e.*

$$\theta_{uv}(0,0) + \theta_{uv}(1,1) \le \theta_{uv}(0,1) + \theta_{uv}(1,0)\,, \tag{8}$$

where we abbreviate $\theta_{uv}(\delta_u(0), \delta_v(1))$ by $\theta_{uv}(0,1)$. It is known that in this case the natural linear program (LP) relaxation of (5) is tight, and, moreover, reducible to the efficiently solvable min-cut/max-flow problem [31].[4]

[3] Without loss of generality we assume the non-trivial case $x'_u \ne x''_u$ for all $u \in \mathcal{V}$.

[4] The orderings δ_u can also be found explicitly [44], allowing for a more efficient min-cut/max-flow reduction [32].

Non-submodular case. The pairwise costs not fulfilling the submodularity condition (8) for a given mapping δ_u are called *supermodular*. Inequality (8) implies that swapping the "labels" 1 and 0 turns submodular pairwise costs into supermodular ones and vice versa. However, since a swap in one node changes sub-/supermodularity of all incident pairwise costs, we cannot always turn all supermodular pairwise costs into submodular ones. This is already impossible if the graph contains a triangular subgraph with all pairwise costs being supermodular.

In these cases the mentioned LP relaxation is in general not tight. However, it has the important *persistency* property, *i.e.* all integer coordinates of a relaxed solution belong to an optimal integer solution [31]. This allows for building efficient approximate methods for (2) applicable also to the non-submodular case [41]. These methods are known in the literature as *quadratic pseudo-boolean optimization* (QPBO) or *roof duality*. As an alternative, *trust region-based* approximate optimization algorithms for (2) have been suggested by [20]. They are based on an iterative approximation of the problem by submodular problems. To this end the supermodular pairwise costs are approximated with unary costs. Similar to the QPBO techniques, performance of trust-region methods drops as the number of supermodular pairwise costs increases. Contrary to the QPBO techniques they require an explicit ordering of the label sets.

4. Feasibility of proposals

Before we address the generation of proposals, we theoretically substantiate the main property of fusion move proposals for graph matching problems: *feasibility*. In other words, proposals should satisfy the uniqueness constraints to allow the method to perform well.

Size of the search space. In a nutshell, fusion moves is a local search method, with the search space defined by proposals. Performance of such methods critically depends on the size of the search space. Assuming that a better, or even the best, solution within this space can be found efficiently, this search space should be as large as possible to allow for better approximations. The following proposition sets the bounds on the size of the search space:

Proposition 1. *Let x' be a feasible, and x'' a possibly infeasible assignment for the graph matching problem (1). Let m be the number of dummy, and n the number of different non-dummy labels in x''. Then the auxiliary problem (4) has at most $2^m\left(\frac{|\mathcal{V}|}{n}+1\right)^n$ feasible solutions.*

In other words, for a fixed number of dummy labels in x'' the size of the search space exponentially increases with the number of different labels in x''. Feasible assignments maximize this number, see §A1 for a proof of Prop. 1.

The need for feasible assignments distinguishes graph matching from the MRF problem, where the space of possible solutions always grows as 2^n, where n is the *total* number of nodes where the proposals differ. Therefore, a popular and quite efficient way to generate MRF proposals known as α-expansion [9], where $x''_u = \alpha$ for all $u \in \mathcal{V}$, is completely ineffective for graph matching: According to Proposition 1 the search space reduces to $|\mathcal{V}| + 1$ solutions. Another popular method [29, 34] suggests constructing proposals from locally best labels returned by, *e.g.*, loopy belief propagation. As empirically observed by [29], for the graph matching problem such proposals typically do not satisfy the uniqueness constraints and, therefore, lead to a non-competitive performance of fusion moves.

Efficiency of approximate solvers. As noted in Section 3, performance of approximate solvers for the auxiliary problem (2) drops with an increasing proportion of supermodular pairwise costs. Since the uniqueness constraints for the auxiliary problem (4) are translated into large pairwise costs, it is important to find an ordering where these large costs do not lead to a violation of the submodularity constraint (8).

Let the proposal x'' be infeasible, *i.e.* there exist $u, v \in \mathcal{V}$, $u \neq v$, with $x''_u = x''_v \neq \#$. Consider now the ordering where labels x''_u are mapped to 0 for all $u \in \mathcal{V}$, and all x'_u to 1. Then, according to (7), $\theta_{uv}(0,0) = C_\infty$, which would lead to a supermodular pairwise cost θ_{uv}. Should there be multiple nodes with equal labels, *i.e.* $x''_u = x''_v = x''_w$, this would lead to a fully connected subgraph with supermodular costs. As discussed in Section 3, these costs cannot all be turned into submodular ones by swapping the labels 0 and 1. As a consequence, this leads to a deterioration in performance of approximate methods for the graph matching auxiliary problem (4). This case can be avoided by requiring x'' to be feasible.

Conversely, consider the practically inevitable case of equal labels in *different* proposals, *i.e.* $x'_u = x''_v$ for some $u, v \in \mathcal{V}$, $u \neq v$. According to (7), with the same initial ordering as above, $\theta_{uv}(1,0) := C_\infty$. This, however, renders the corresponding pairwise cost submodular, *c.f.* (8), which simplifies optimization.

***To summarize**, the feasibility of proposals increases the search space for each fusion, while at the same time allowing for efficient approximate solvers for the auxiliary problem.*

5. Proposal generation

As mentioned in Section 1 fusion moves work best if the proposals are of *high quality* and *diverse*. Essentially, *high quality* means low corresponding energy E, and *diversity* can be quantified by counting the number of nodes where two proposals differ.

How to obtain high quality proposals? The natural idea to get high quality proposals is to employ some iterative optimization process which outputs solutions on each iteration. As discussed in Section 4, in the case of graph matching, these proposals should be feasible. Dual methods equipped

Algorithm 1: Randomized greedy heuristic.

```
Input: graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, labels $\mathcal{L}$ and costs $\theta$
initialize $\mathcal{V}' := \emptyset$ and $\mathcal{L}' := \emptyset$
while $\mathcal{V}' \neq \mathcal{V}$ do
    randomly select $u \in \mathcal{N}(\mathcal{V}')$ or $u \in \mathcal{V} \backslash \mathcal{V}'$ if $\mathcal{N}(\mathcal{V}') = \emptyset$
    set
      $x_u := \operatorname*{arg\,min}_{s \in \mathcal{L}_u \backslash \mathcal{L}' \cup \{\#\}} \Big[ \theta_u(s) + \sum_{v \in \mathcal{N}(u) \cap \mathcal{V}'} \theta_{uv}(s, x_v) \Big]$
    update $\mathcal{V}' := \mathcal{V}' \cup \{u\}$ and $\mathcal{L}' := \mathcal{L}' \cup \{x_u\}$
Output: feasible assignment $x = (x_u)_{u \in \mathcal{V}}$
```

with efficiently computable primal heuristics are therefore natural candidates for proposal generators. In Section 5.2 we briefly describe two types of such methods, block-coordinate ascent- and subgradient-based ones.

How to obtain diverse proposals? Diversity of proposals based on dual optimization can either be induced by noisy dual updates, or must be an intrinsic property of the primal heuristic. We utilize both strategies.

The subgradient method is a representative of the first type. Due to non-optimal step-sizes and update directions it usually demonstrates a "zig-zag" progress of the dual value that induces similar behavior in the assignment scores obtained by a primal heuristic.

In contrast, block-coordinate ascent methods are based on optimal updates and guarantee a monotone improvement of the dual value. As a consequence, the corresponding assignments computed by deterministic primal heuristics often lack diversity.

To address this issue, we suggest to use the *randomized greedy heuristic* described in Section 5.1 as a generic method to generate diverse proposals. It combines *diversity* due to randomization of the node selection order with *high quality* due to taking locally optimal labels. Another important advantage of this method is that it can profit from the dual optimization, and provides qualitatively better proposals as the dual optimization progresses. In particular, it returns a globally optimal assignment if the latter is unique and the dual bound is tight. We describe its use in connection with a dual BCA solver in Section 5.2.

5.1. Randomized greedy heuristic

Let $\mathcal{N}(u) := \{v \in \mathcal{V} \mid uv \in \mathcal{E}\}$ be the neighborhood of u, and $\mathcal{N}(\mathcal{V}') := \big(\bigcup_{u \in \mathcal{V}'} \mathcal{N}(u)\big) \backslash \mathcal{V}'$ the neighborhood of $\mathcal{V}' \subseteq \mathcal{V}$. Note, $\mathcal{N}(\emptyset) = \emptyset$. The *randomized greedy heuristic* is defined by Algorithm 1. In each step an unassigned node is randomly selected from the neighborhood of the assigned nodes, and a label is assigned to it such that the uniqueness constraints are satisfied and the sum of its unary cost and all pairwise costs on edges connecting it with assigned nodes minimized.

5.2. Dual solvers

Dual problem. The graph matching problem (1) can be represented in an ILP form similar to that of the auxiliary problem (2). We define binary variables $\mu_{u,s}$ and $\mu_{uv,st}$ analogously. The number of such variables is $M := \sum_{u \in \mathcal{V}} |\mathcal{L}_u^{\#}| + \sum_{uv \in \mathcal{E}} |\mathcal{L}_u^{\#}||\mathcal{L}_v^{\#}|$. By denoting the set of nodes containing a particular non-dummy label s as $\mathcal{V}(s) := \{u \in \mathcal{V} \mid s \in \mathcal{L}_u\}$, problem (1) can be written as:

$$\min_{\mu \in \{0,1\}^M} \sum_{\substack{u \in \mathcal{V} \\ s \in \mathcal{L}_u^{\#}}} \mu_{u,s} \theta_u(s) + \sum_{\substack{uv \in \mathcal{E} \\ (s,t) \in \mathcal{L}_u^{\#} \times \mathcal{L}_v^{\#}}} \mu_{uv,st} \theta_{uv}(s,t) \quad (9)$$

$$\text{s.t.}\ \forall\, uv \in \mathcal{E},\ t \in \mathcal{L}_v^{\#}\colon \sum_{s \in \mathcal{L}_u^{\#}} \mu_{uv,st} = \mu_{v,t} \quad (10)$$

$$\forall\, u \in \mathcal{V}\colon \sum\nolimits_{s \in \mathcal{L}_u^{\#}} \mu_{u,s} = 1 \quad (11)$$

$$\forall\, s \in \mathcal{L}\colon \sum\nolimits_{u \in \mathcal{V}(s)} \mu_{u,s} \leq 1 \quad (12)$$

By introducing $\xi_u^{\lambda}(s) := \frac{\theta_u(s)}{2} + \lambda_{u,s}$ and $\hat{\xi}_u^{\lambda}(s) := \frac{\theta_u(s)}{2} - \lambda_{u,s}$ for $u \in \mathcal{V}$, $s \in \mathcal{L}_u^{\#}$, and arbitrary $\lambda_{u,s} \in \mathbb{R}$, we can equivalently rewrite the objective in (9) as a sum of objectives of MRF and LAP subproblems denoted as E^{MRF} and E^{LAP}, respectively:

$$\underbrace{\sum_{\substack{u \in \mathcal{V} \\ s \in \mathcal{L}_u^{\#}}} \mu_{u,s} \xi_u^{\lambda}(s) + \sum_{\substack{uv \in \mathcal{E} \\ (s,t) \in \mathcal{L}_u^{\#} \times \mathcal{L}_v^{\#}}} \mu_{uv,st} \theta_{uv}(s,t)}_{=:E^{\text{MRF}}(\mu,\lambda)} + \underbrace{\sum_{\substack{u \in \mathcal{V} \\ s \in \mathcal{L}_u^{\#}}} \mu_{u,s} \hat{\xi}_u^{\lambda}(s)}_{=:E^{\text{LAP}}(\mu,\lambda)}.$$

Let Λ be the set of all binary vectors $\mu \in \{0,1\}^M$ satisfying constraints (10)-(11), and B the set of those satisfying (11)-(12). Then the sum

$$\min_{\mu \in \Lambda} E^{\text{MRF}}(\mu, \lambda) + \min_{\hat{\mu} \in B} E^{\text{LAP}}(\hat{\mu}, \lambda) \quad (13)$$

of independent minimizations of the MRF and LAP subproblems constitutes a lower bound for (9)-(12).

While the second term can be minimized efficiently, *e.g.* by the Hungarian method, the first term is an NP-hard problem by itself. By dualizing the constraints (10) one obtains its Lagrange dual lower bound, *c.f.* [43, Ch.6],

$$D(\phi, \lambda) := \sum_{u \in \mathcal{V}} \min_{s \in \mathcal{L}_u^{\#}} \xi_u^{\phi,\lambda}(s) + \sum_{uv \in \mathcal{E}} \min_{(s,t) \in \mathcal{L}_u^{\#} \times \mathcal{L}_v^{\#}} \theta_{uv}^{\phi}(s,t)\,,$$

where

$$\xi_u^{\phi,\lambda}(s) := \xi_u^{\lambda}(s) - \sum_{v \in \mathcal{N}(u)} \phi_{u,v}(s)\,, \quad (14)$$

$$\theta_{uv}^{\phi}(s,t) := \theta_{uv}(s,t) + \phi_{u,v}(s) + \phi_{v,u}(t)\,,$$

are commonly referred to as *reparametrized costs*.

All in all, the dual problem of (9)-(12) consists in the lower bound maximization

$$\max_{\phi,\lambda} \Big[D(\phi, \lambda) + \min_{\hat{\mu} \in B} E^{\text{LAP}}(\hat{\mu}, \lambda) \Big]\,. \quad (15)$$

Dual block-coordinate ascent, §A2. Based on the ideas of [45, 56], and the recent progress in development of dual solvers for MRFs [48, 49], we implemented a *block-coordinate ascent* (BCA) solver that also allows to output assignment proposals.

Our solver monotonically improves the dual bound (15) by interleaving maximization w.r.t. ϕ and λ. Each step on ϕ consists of maximizing the bound w.r.t. the block of variables $(\phi_{u,v}(s), \phi_{v,u}(t))$, $(s,t) \in \mathcal{L}_u^\# \times \mathcal{L}_v^\#$ associated with one edge $uv \in \mathcal{E}$. A sequence of these steps addressing all edges is equivalent to one iteration of the MPLP++ algorithm of [48], that notably outperforms the MPLP algorithm [18] used by [56]. Each step on λ consists of maximizing the dual objective w.r.t. blocks $(\lambda_{u,s})$, $u \in \mathcal{V}(s)$, for each label $s \in \mathcal{L}$ similar to how it was done by [45].

For primal estimates we either use the exact solution of the LAP term $\min_{\hat\mu \in B} E^{\text{LAP}}(\hat\mu, \lambda)$ in (15) for the current value of λ, or run our randomized greedy Algorithm 1 on the graph matching problem with unary and pairwise costs $\xi_u^{\phi,\lambda}$ and θ_{uv}^{ϕ}, for current values of ϕ and λ. For the LAP heuristic we use Gurobi [21] as a solver. While using *e.g.* the Hungarian method for solving LAPs would be faster, we found that the greedy heuristics combined with fusion moves consistently outperforms its LAP counterpart in all our experiments in terms of run time and quality.

Subgradient method. We use the code of [47] as a representative of the dual subgradient methods. In its basic version denoted as *dd-ls0*, which stands for *dual decomposition with no local subproblems*, it optimizes the same bound as (15) with the difference that instead of the dual MRF-term $D(\phi, \lambda)$ it uses an equivalent *tree-decomposition* of the problem $\min_{\mu \in \Lambda} E^{\text{MRF}}(\mu, \lambda)$, see *e.g.* [43, Ch.9] for details. As the primal bound it uses a solution of the LAP problem $\min_{\hat\mu \in B} E^{\text{LAP}}(\hat\mu, \lambda)$ for the current value of λ.

Two other versions of the solver we use in our experiments, denoted as *dd-ls3* and *dd-ls4*, additionally consider *local subproblems* on subgraphs of $\mathcal{G}$ consisting of 3 or 4 neighboring nodes of the original graph, respectively. These modifications require more time per iteration, but optimize tighter bounds than (15). Additionally, these variants estimate primal solutions based on solutions of the local subproblems. We refer to [47] for further details.

Dual BCA algorithm complexity per iteration is $O(\sum_{uv \in \mathcal{E}} |\mathcal{L}_u^\#||\mathcal{L}_v^\#|)$, *e.g.* linear in the size of the problem. For a fully connected graph with $\mathcal{L}_u^\# = \mathcal{L} \cup \{\#\}$, $\forall u \in \mathcal{V}$, this turns to $O(|\mathcal{V}|^4)$. This is one degree of power less than the iteration complexity $O(|\mathcal{V}|^5)$ of the dual ascent algorithms [2, 22, 30] known in operations research.

6. Experiments and analysis

Experimental setup. We evaluate the performance of all tested algorithms by measuring their total run time and the obtained solution quality. Our experiments were run on a compute cluster equipped with AMD EPYC 7702 2.0 GHz processors and 512 GB main memory. For a fair comparison we used efficient implementations of all discussed algorithms, and report the minimal runtime of 5 independently scheduled trials.

Datasets, §A3. Our experimental evaluation was conducted on 8 datasets with overall 316 problem instances from computer vision and bio-imaging described in detail below. To demonstrate the scalability of our approach, along with the standard small-scale datasets for computer vision `hotel`, `house`, `car`, `motor` and `opengm` with $|\mathcal{V}| \leq 52$, we consider the middle-sized ones `flow`, $|\mathcal{V}| \leq 126$, and the large-scaled `worms` and `pairs` datasets with $|\mathcal{V}| \leq 565$. The latter are, to our knowledge, the *largest graph matching problem instances ever considered in computer vision.*

Wide baseline matching (`hotel`, `house`) is based on a series of images of the same object from different view angles. We use the same image pairs, landmarks, and cost structure as in [47] based on the work by [11].

Keypoint matching (`car`, `motor`) contains *car* and *motor*bike instances from the PASCAL VOC 2007 Challenge [16] with the features and costs from [37]. We preprocessed the models by removing edges with zero cost, thereby reducing graph density substantially.

Large displacement flow (`flow`) was introduced by [5] for key point matching on scenes with large displacement flow. We use the keypoints and costs from [46].

OpenGM matching (`opengm`) is a set of non-rigid point matching problems by [33], now part of the *OpenGM* Benchmark [28].

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 precomputed atlas of the organism. We use the models from [26, 27].

Worm-to-worm matching (`pairs`), see Fig. 1 for illustration, in contrast to the `worms` dataset, directly matches the cell nuclei of individual *C. elegans* worms to each other. This alleviates the need to precompute an atlas based on manual annotations. Unary and pairwise costs of the respective graph matching problems are derived by averaging the nucleus-(pair-)specific covariance matrices captured by the atlas over all nuclei. This coarsens the model to a level achievable without manual annotations. For our experiments we randomly chose 16 instances out of the $30 \cdot 29 = 870$ non-trivial pairs of worms based on the same data as `worms`.

Algorithms. As proposal generators we evaluate the three dual subgradient-based algorithms `dd-ls0`, `dd-ls3`, `dd-ls4` and our BCA solver `bca` described in Section 5.2. The latter is used with either the primal heuristics based on the LAP solution or on the greedy Algorithm 1, denoted as `bca-lap` and `bca-greedy`, respectively. We also use the `greedy` Algorithm 1 as a standalone baseline.

As proposal fusing methods we evaluate *Gurobi* [21] as

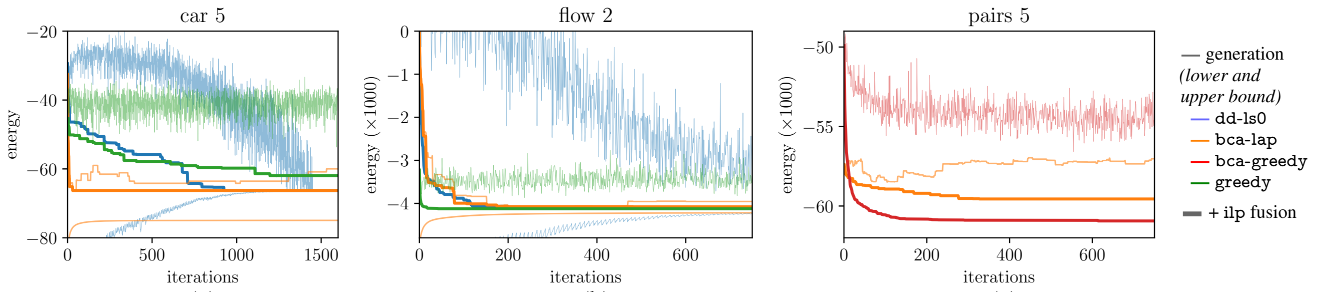

Figure 2. **(a-b) Influence of fusion.** The plots show the energy of assignments generated by dd-ls0 (blue), bca-lap (orange) and greedy (green) algorithms together with the dual bound where applicable. The thick line in matching color shows for each algorithm the achieved energy when using an ilp solver for fusion on top. Notably, fusion achieves very good quality with much less iterations. For some datasets, even greedily generated proposals suffice to obtain (almost) optimal solutions when fused. **(c) LAP vs. greedy heuristic.** The plot shows the quality of proposals generated by bca-lap (orange) and bca-greedy (red) for an exemplary instance of pairs. The fused solutions on top of these generators are shown in the same color as a thick line. Fusion moves applied to bca-greedy yield significantly better results than when applied to bca-lap, even though the bca-greedy proposals are visibly worse than those of bca-lap.

an exact ILP solver for the auxiliary problem (5) denoted as ilp, the *trust region*-based method of [20] denoted as lsatr, as well as different *QPBO* variants denoted as qpbo-XX. For the description of these variants see [41], and the corresponding source code.

Additionally, in Section 6.2 we compare our method to a number of state-of-the-art algorithms.

6.1. Influence of different components

Influence of fusion on solution quality, Fig. 2 (a-b), §A4,A6. For our first experiment we ran three methods to generate proposals: bca-lap, dd-ls0 and greedy. The algorithms bca-lap and dd-ls0 represent standard dual techniques with a LAP-based primal heuristic, and greedy constitutes a baseline. We fuse the generated proposals with the ilp method.

Fig. 2 (a-b) shows the results for these three proposal generators before and after fusion for two exemplary instances from the considered datasets. Although the energy of dd-ls0 proposals is far from optimal, their fusion immediately leads to much better results.

Although the energy of bca-lap proposals is often much lower than that of dd-ls0, the energy of the fused solutions is not necessarily lower. We explain this by lacking diversity in the bca-lap proposals. This explanation is confirmed by the performance of the fused greedy proposals. Even though the proposal quality for greedy is very dataset dependent, in combination with fusion it often leads to good results. While they are still worse than those obtained by dual methods in Fig. 2 (a), they can be very competitive as seen in Fig. 2 (b).

This experiment clearly shows that the overall solution quality can be substantially improved by fusing generated proposals. Since fusion provides already very good results with relatively few proposals, it promises a significant speed-up compared to fusion-free methods as this can significantly reduce the number of necessary iterations to achieve a certain solution quality.

Exact vs. approximate fusion, §A4,A6. To estimate the speed-up in runtime obtained by fusion we compared the exact ilp solver with the approximate qpbo-i, qpbo-p, qpbo-pi and lsatr solvers. Among them we found qpbo-i to be best performing in terms of consistent quality and speed. Despite a worst-case computational complexity of $O(|\mathcal{V}||\mathcal{E}|)$, qpbo-i was 10–50 times faster than the dual updates.

LAP vs. greedy heuristic for BCA, Fig. 2 (c). As noted above, fusion moves only marginally improve performance of bca-lap because of the low diversity of proposals generated by this method. This is easy to see if we compare it to bca-greedy, where the LAP heuristic is replaced by the greedy Algorithm 1. Indeed, for all datasets we observed that fusion of bca-greedy proposals produced results at least as good as fusion of bca-lap proposals, even when the bca-greedy proposals themselves had higher energies than those of bca-lap.

Effect of relaxation tightening, §A4,A6. In general, tighter relaxations provide better bounds in the long run. However, one pays with a higher runtime per iteration for this. Interestingly, due to fusion all three subgradient methods, dd-ls0, dd-ls3 and dd-ls4, get close or even attain the global optimum in most of the datasets. Therefore, due to lower iteration time the method dd-ls0 corresponding to the weakest relaxation converges first, and, hence, is preferable. Since fusion notably improves the energy of the found solutions, we claim that without fusion one would have to use tighter relaxations to attain the same result quality.

Summary. Table 1 summarizes our performance study. We include dd-ls0 and bca-greedy as the best representatives of their algorithm classes. We observed qpbo-i-based fusion to achieve solutions with the same or lower energy as the underlying proposal generator, while also on average converging faster than the proposal generator without fusion. In other words, *it is always sensible to use fusion moves*.

Table 1. **Summary of fusion moves performance.** Averaged energy of the best proposals for each dataset (*best gen.*), and time needed on average to generate it (t_{gen}) are shown. Furthermore, it shows for the `qpbo-i` fusion algorithm how long it took on average to beat the `dd-ls0` or `bca-greedy` proposals when fusing (t_{beat}), the average energy of the best proposal generated by fusion (*best fused*), and the average time after which this was obtained (t_{fuse}). All times are in seconds. The small numbers in front of the energies represent the number of instances solved to optimality by the respective method. Methods with fusion attain better energy values and are faster on average.

dataset (number of instances)	`dd-ls0`			+ `qpbo-i`				`bca-greedy`			+ `qpbo-i`			
		best gen.	$t_{\text{gen.}}$	t_{beat}		*best fused*	t_{fuse}		*best gen.*	$t_{\text{gen.}}$	t_{beat}		*best fused*	t_{fuse}
`hotel` (105)	105	**-4293.00**	0.07	0.04	105	**-4293.00**	0.04	103	-4291.21	0.81	0.03	105	**-4293.00**	**0.03**
`house` (105)	105	**-3778.13**	**0.02**	0.02	105	**-3778.13**	**0.02**	105	**-3778.13**	0.94	0.09	105	**-3778.13**	0.09
`car` (30)	29	-69.34	**0.17**	0.17	29	**-69.37**	**0.17**	27	-69.19	1.28	0.12	29	**-69.37**	**0.17**
`motor` (20)	20	**-62.95**	0.06	0.03	20	**-62.95**	0.03	19	-62.93	0.80	0.02	20	**-62.95**	**0.02**
`flow` (6)	3	-2818.83	2.79	1.12	5	-2835.84	1.91	4	-2837.82	9.65	0.65	5	**-2840.00**	**0.66**
`opengm` (4)	3	31.42	0.94	0.77	3	26.18	0.87	4	**21.22**	1.29	0.04	4	**21.22**	**0.04**
`worms` (30)	0	-43824.08	492.00	41.57	1	-48347.09	428.79	9	-48454.89	54.53	15.72	27	**-48465.28**	**17.77**
`pairs` (16)	0	-63453.77	348.33	7.87	0	-65936.89	**343.49**	0	-62696.28	783.83	1.96	0	**-66005.55**	953.09

Table 2. **Comparison table.** *our* denotes the proposed `bca-greedy+qpbo-i` method. For each method we state *opt/t* denoting the number of optimally solved instances together with the average time in seconds to attain the optimal solutions ("–" if no instance was solved to optimality), the average solution energy E (lower is better), and the average solution accuracy *acc* in percent. The sign "–" in the E or *acc* column means that at least for one problem instance the respective method yielded no assignment. For datasets indicated by † no ground truth is known and, therefore, no accuracy reported. The best accuracy is not highlighted in bold, since algorithms do not have access to the ground truth and hence do not maximize accuracy explicitly. The relatively low accuracy of 86% attained for `worms` is explained by model misspecification. The original work [27] reports 83% accuracy achieved by `dd-ls4` without time restrictions. Since *our* method is randomized, we report ranges where appropriate.

dataset (number of instances)	time budget	`dd-ls0` [47]			`dd-ls3` [47]			`HBP` [56]			`AMP` [46]			`AMP-tight` [46]			*our*		
		opt / t	E	acc	opt / t	E	acc	opt / t	E	acc	opt / t	E	acc	opt / t	E	acc	opt / t	E	acc
`hotel` (105)	1 s	**105/0.01**	**-4293**	100	**105**/0.04	**-4293**	100	102/0.11	—	—	98/0.11	-4280	99	104/0.13	-4292	100	100–**105/0.01**	**-4291±2**	100
`house` (105)	1 s	**105**/0.03	**-3778**	100	**105**/0.13	**-3778**	100	104/0.20	—	—	102/0.30	-3773	100	**105**/0.19	**-3778**	100	**105/0.01**	**-3778**	100
`car` (30)	1 s	**28**/0.13	**-69**	92	14/0.55	-57	74	23/0.12	—	—	24/0.11	**-69**	92	26/0.12	**-69**	91	27–**28/0.01**	**-69**	91±1
`motor` (20)	1 s	**20**/0.07	**-63**	97	13/0.25	-57	87	19/0.14	—	—	18/0.04	**-63**	96	17/0.08	**-63**	97	19–**20/0.01**	**-63**	98±1
`opengm`† (4)	1 s	1/0.81	-151		0/—	-118		0/—	—		0/—	-57		0/—	-150		**4/0.004**	**-171**	
	10 s	3/1.08	-161		**4**/2.61	**-171**		2/2.71	-164		0/—	-57		0/—	-150		**4/0.004**	**-171**	
`flow`† (6)	1 s	2/0.79	-2089		1/0.90	-1962		0/—	—		1/0.13	-2628		3/0.16	—		4–**5/0.06**	**-2837±1**	
	10 s	3/1.66	-2819		**5**/2.81	-2821		0/—	—		1/0.13	-2674		3/0.16	-2838		**5/0.06**	**-2838**	
`worms` (30)	1 s	0/—	60597	26	0/—	64158	23	0/—	—	—	0/—	—	—	0/—	—	—	10–**22/0.23**	**-48461±3**	86
	10 s	0/—	50578	24	0/—	49610	24	0/—	—	—	1/6.45	-48389	86	0/—	—	—	16–**25/0.39**	**-48464±1**	86
`pairs`† (16)	10 s	0/—	-61482		0/—	-61638		0/—	—		0/—	-64130		0/—	—		0/—	**-65259±133**	
	30 s	0/—	-61482		0/—	-61638		0/—	—		0/—	-64319		0/—	—		0/—	**-65594±120**	

Although `bca-greedy+qpbo-i` outperforms `dd-ls0+qpbo-i`, the latter one is very competitive and notably outperforms its basic variant `dd-ls0`.

6.2. Comparisons and conclusions

Table 2 compares our `bca-greedy+qpbo-i` method to several state-of-the-art techniques, see also §A5. We omitted a detailed comparison to [11, 13, 19, 35, 36, 58], since the accuracy they attain is notably lower than that of the dual methods [47, 56] as is shown in the latter papers. This conclusion is also supported by our own experiments. The more recent works [25, 54] unfortunately compare only to the weak baselines above, and do not make their code publicly available. Therefore, we restrict our comparison to the duality-based techniques, as they have been shown to perform best on the computer vision datasets. Note that the `AMP` method [46] recently pushed up the state-of-the-art within a deep graph matching approach [40].

We distinguish between *easy* problem instances (`hotel`, `house`, `motor`, `car`), *mid-difficult* problems (`opengm`, `flow`, `worms`) and *difficult* ones (`pairs`). For the easy datasets we provide results in 1 second, for mid-difficult in 1 and 10 and for difficult in 10 and 30 seconds respectively, see also §A5 for other run-time settings and a comparison of memory consumption.

Conclusions. As Table 2 shows our method notably outperforms its competitors in terms of speed and accuracy. Since it practically solves all easy and mid-difficult problem instances in significantly less than a second, it can be efficiently used in deep graph matching pipelines. The easy datasets `hotel`, `house`, `motor`, `car` are largely solved by all state-of-the-art methods and cannot be used to show progress of the solvers anymore.

References

[1] Kamil Adamczewski and Yumin Suh Kyoung Mu Lee. Discrete tabu search for graph matching. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015. 2

[2] Warren P. Adams and Terri A. Johnson. Improved linear programming-based lower bounds for the quadratic assignment problem. *Discrete Mathematics and Theoretical Computer Science*, 1994. 2, 6

[3] Charu C. Aggarwal, James B. Orlin, and Ray P. Tai. Optimized crossover for the independent set problem. *Operations research*, 1997. 2

[4] Ravindra K. Ahuja, James B. Orlin, and Ashish Tiwari. A greedy genetic algorithm for the quadratic assignment problem. *Computers & Operations Research*, 2000. 2

[5] Hassan Abu Alhaija, Anita Sellent, Daniel Kondermann, and Carsten Rother. GraphFlow – 6D large displacement scene flow via graph matching. In *Proceedings of the DAGM German Conference on Pattern Recognition*, 2015. 6, 13

[6] Martin Beckman and Tjalling C. Koopmans. Assignment problems and the location of economic activities. *Econometrica*, 1957. 1

[7] Florian Bernard, Christian Theobalt, and Michael Moeller. DS*: Tighter lifting-free convex relaxations for quadratic matching problems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2

[8] Endre Boros and Peter L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 2002. 3

[9] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2001. 4

[10] Rainer E. Burkard, Eranda Cela, Panos M. Pardalos, and Leonidas S. Pitsoulis. *The quadratic assignment problem*. 1998. 1, 2

[11] Tibério S. Caetano, Julian J. McAuley, Li Cheng, Quoc V. Le, and Alexander J. Smola. Learning graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009. 6, 8, 13

[12] Eranda Cela. *The quadratic assignment problem: theory and algorithms*, volume 1. Springer Science & Business Media, 2013. 1

[13] Minsu Cho, Jungmin Lee, and Kyoung Mu Lee. Reweighted random walks for graph matching. In *Proceedings of the European Conference on Computer Vision*, 2010. 8

[14] Minsu Cho, Jian Sun, Olivier Duchenne, and Jean Ponce. Finding matches in a haystack: A max-pooling strategy for graph matching in the presence of outliers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 2

[15] Nadav Dym, Haggai Maron, and Yaron Lipman. DS++: A flexible, scalable and provably tight relaxation for matching problems. *ACM Transactions on Graphics (TOG)*, 2017. 2

[16] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The PASCAL visual object classes challenge 2007 results, 2007. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. 6, 13

[17] Matthias Fey, Jan E. Lenssen, Christopher Morris, Jonathan Masci, and Nils M. Kriege. Deep graph matching consensus. In *Proceedings of the International Conference on Learning Representations*, 2020. 1

[18] Amir Globerson and Tommi S. Jaakkola. Fixing Max-Product: Convergent message passing algorithms for MAP LP-relaxations. In *Advances in Neural Information Processing Systems*, 2008. 6

[19] Steven Gold and Anand Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1996. 2, 8

[20] Lena Gorelick, Yuri Boykov, Olga Veksler, Ismail Ben Ayed, and Andrew Delong. Local submodularization for binary pairwise energies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 4, 7

[21] Gurobi. Gurobi optimization, 2018. http://www.gurobi.com. 3, 6

[22] Peter Hahn and Thomas Grant. Lower bounds for the quadratic assignment problem based upon a dual formulation. *Operations Research*, 1998. 2, 6

[23] Stefan Haller, Paul Swoboda, and Bogdan Savchynskyy. Exact map-inference by confining combinatorial search with LP relaxation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. 17

[24] Bo Jiang, Pengfei Sun, Jin Tang, and Bin Luo. Glmnet: Graph learning-matching networks for feature matching. *arXiv preprint arXiv:1911.07681*, 2019. 1

[25] Bo Jiang, Jin Tang, Chris Ding, Yihong Gong, and Bin Luo. Graph matching via multiplicative update algorithm. In *Advances in Neural Information Processing Systems*, 2017. 2, 8

[26] Dagmar Kainmueller, Florian Jug, Carsten Rother, and Gene Meyers. Graph matching problems for annotating C. elegans, 2017. https://doi.org/10.15479/AT:ISTA:57. 6, 13

[27] Dagmar Kainmueller, Florian Jug, Carsten Rother, and Gene Myers. 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. 6, 8, 13, 15

[28] Jörg H. Kappes, Björn Andres, Fred A. Hamprecht, Christoph Schnörr, Sebastian Nowozin, Dhruv Batra, Sungwoong Kim, Bernhard X. Kausler, Thorben Kröger, Jan Lellmann, Nikos Komodakis, Bogdan Savchynskyy, and Carsten Rother. A comparative study of modern inference techniques for structured discrete energy minimization problems. *International Journal of Computer Vision*, 2015. 6, 13, 17

[29] Jörg H. Kappes, Thorsten Beier, and Christoph Schnörr. MAP-inference on large scale higher-order discrete graphical models by fusion moves. In *Proceedings of the European Conference on Computer Vision*, 2014. 2, 4

[30] Stefan E. Karisch, Eranda Cela, Jens Clausen, and Torben Espersen. A dual framework for lower bounds of the quadratic assignment problem based on linearization. *Computing*, 1999. 2, 6

[31] Vladimir Kolmogorov and Carsten Rother. Minimizing non-submodular functions with graph cuts-a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007. 3, 4

[32] Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004. 3

[33] Nikos Komodakis and Nikos Paragios. Beyond loose LP-

relaxations: Optimizing MRFs by repairing cycles. In *Proceedings of the European Conference on Computer Vision*, 2008. 6, 13

[34] Victor Lempitsky, Carsten Rother, Stefan Roth, and Andrew Blake. Fusion moves for Markov random field optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010. 2, 3, 4

[35] Marius Leordeanu and Martial Hebert. A spectral technique for correspondence problems using pairwise constraints. In *Proceedings of the IEEE International Conference on Computer Vision*, 2005. 2, 8

[36] Marius Leordeanu, Martial Hebert, and Rahul Sukthankar. An integer projected fixed point method for graph matching and MAP inference. In *Advances in Neural Information Processing Systems*, 2009. 2, 8

[37] Marius Leordeanu, Rahul Sukthankar, and Martial Hebert. Unsupervised learning for graph matching. *International Journal of Computer Vision*, 2012. 6, 13

[38] Eliane Maria Loiola, Nair Maria Maia de Abreu, Paulo Oswaldo Boaventura-Netto, Peter Hahn, and Tania Querido. An analytical survey for the quadratic assignment problem. *European Journal of Operational Research*, 2007. 1

[39] Fuhui Long, Hanchuan Peng, Xiao Liu, Stuart K Kim, and Eugene Myers. A 3d digital atlas of c. elegans and its application to single-cell analyses. *Nature methods*, 2009. 15

[40] Michal Rolínek, Paul Swoboda, Dominik Zietlow, Anselm Paulus, Vít Musil, and Georg Martius. Deep graph matching via blackbox differentiation of combinatorial solvers. In *Proceedings of the European Conference on Computer Vision*. Springer, 2020. 1, 8

[41] Carsten Rother, Vladimir Kolmogorov, Victor S. Lempitsky, and Martin Szummer. Optimizing binary MRFs via extended roof duality. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007. 4, 7

[42] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 1

[43] Bogdan Savchynskyy. Discrete graphical models – An optimization perspective. *Foundations and Trends in Computer Graphics and Vision*, 2019. 2, 5, 6

[44] Dmitrij Schlesinger. Exact solution of permuted submodular minsum problems. In *Proceedings of the International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, 2007. 3

[45] Paul Swoboda, Jan Kuske, and Bogdan Savchynskyy. A dual ascent framework for Lagrangean decomposition of combinatorial problems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 6, 13

[46] Paul Swoboda, Carsten Rother, Hassan Abu Alhaija, Dagmar Kainmuller, and Bogdan Savchynskyy. 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. 1, 2, 6, 8, 13, 16, 17, 184, 185

[47] Lorenzo Torresani, Vladimir Kolmogorov, and Carsten Rother. A dual decomposition approach to feature correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013. 1, 2, 6, 8, 13, 15, 16, 184, 185

[48] Siddharth Tourani, Alexander Shekhovtsov, Carsten Rother, and Bogdan Savchynskyy. MPLP++: Fast, parallel dual block-coordinate ascent for dense graphical models. In *Proceedings of the European Conference on Computer Vision*, 2018. 6, 13

[49] Siddharth Tourani, Alexander Shekhovtsov, Carsten Rother, and Bogdan Savchynskyy. Taxonomy of dual block-coordinate ascent methods for discrete energy minimization. In *Proceedings of the Conference on Artifical Intelligence and Statistics*, 2020. 6

[50] Shinji Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1988. 2

[51] Runzhong Wang, Junchi Yan, and Xiaokang Yang. Learning combinatorial embedding networks for deep graph matching. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 1

[52] Runzhong Wang, Junchi Yan, and Xiaokang Yang. Neural graph matching network: Learning lawler's quadratic assignment problem with extension to hypergraph and multiple-graph matching. *arXiv preprint arXiv:1911.11308*, 2019. 1

[53] Tianshu Yu, Runzhong Wang, Junchi Yan, and Baoxin Li. Learning deep graph matching with channel-independent embedding and hungarian attention. In *Proceedings of the International Conference on Learning Representations*, 2019. 1

[54] Tianshu Yu, Junchi Yan, and Baoxin Li. Determinant regularization for gradient-efficient graph matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2, 8

[55] Andrei Zanfir and Cristian Sminchisescu. Deep learning of graph matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1

[56] Zhen Zhang, Qinfeng Shi, Julian McAuley, Wei Wei, Yanning Zhang, and Anton van den Hengel. Pairwise matching through max-weight bipartite belief propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1, 2, 6, 8, 16, 17, 184

[57] Guoxing Zhao, Bin Luo, Jin Tang, and Jinxin Ma. Using eigen-decomposition method for weighted graph matching. In *Proceedings of the International Conference on Intelligent Computing*, 2007. 2

[58] Feng Zhou and Fernando De la Torre. Factorized graph matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012. 2, 8