Towards Optimizing Large-Scale Multi-Graph Matching in Bioimaging

Max Kahl[*1] Sebastian Stricker[*1] Lisa Hutschenreiter[1] Florian Bernard[2]
Carsten Rother[1] Bogdan Savchynskyy[1]
[1]Heidelberg University [2]University of Bonn

Abstract

Multi-graph matching is an important problem in computer vision. Our task comes from bioimaging, where a set of 100 3D-microscopic images of worms have to be brought into correspondence. Surprisingly, virtually all existing methods are not applicable to this large-scale, real-world problem since they either assume a complete or dense problem setting, and they have so far only been applied to small-scale, toy or synthetic problems. Despite claims in literature that methods addressing complete multi-graph matching are applicable in an incomplete setting, our first contribution is to prove that their runtime would be excessive and impractical. Our second contribution is a new method for incomplete multi-graph matching that applies to real-world, larger-scale problems. We experimentally show that for our bioimaging application we are able to attain results in less than two minutes, whereas the only competing approach requires at least half an hour while producing far worse results. Furthermore, even for small-scale, dense or complete problem instances we achieve results that are at least on par with the leading methods, but an order of magnitude faster.

1. Introduction

Establishing correspondences across multiple finite sets is a fundamental combinatorial problem important for 3D model retrieval [32], shape matching [16, 38], statistical shape modeling [18, 47], federated learning [28], and genomic data analysis [11]. Typically, each *object*, such as an image or shape, is represented by a set of *keypoints*, which stand for semantically meaningful parts of an object. The task is to bring these keypoints into correspondence. Fig. 2 shows the primary application of our work, where 100 C. elegans worms have to be matched, and the keypoints are candidates for nuclei. This is a large-scale problem with about 45M possible keypoint correspondences, a factor of 89 more than previously considered. Given a matching, biologist can extract various statistics [29] about the worm. The matching must satisfy several conditions, see Fig. 1:

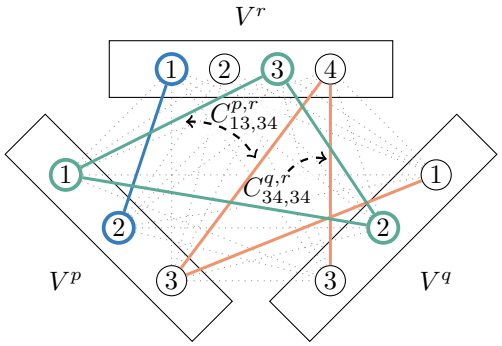

Figure 1. **Multi-graph matching and cycle consistency.** Shown are three objects (solid rectangles) V^p, V^q, V^r, whose keypoints (circles) must be matched (edges). By construction, the setting is incomplete since V^r has 4 keypoints, whereas V^p, V^q only 3. The blue and green matchings are cycle-consistent and form cliques in the shown 3-partite graph. The orange matching is not cycle-consistent and as such does not form a clique. Examples of linear and quadratic costs (dashed arrows) and their associated matchings are also shown.

- *Uniqueness*. Each keypoint of a given object can be matched to *at most* one keypoint of any other object. If "*at most*" is substituted by "*exactly*", one speaks of a *complete* and otherwise of an *incomplete* matching. Due to occlusions or noise during the keypoint extraction process, the incomplete setting is prevalent in practice.
- *Cycle consistency*. If keypoint 1 in object V^p is matched to keypoint 2 in object V^q and keypoint 3 in object V^r, then keypoint 2 must be matched to keypoint 3. Similar transitivity conditions must hold for all matching cycles across arbitrary object subsets.
- *Costs*. Matchings must be *minimal* w.r.t. the given costs, quantifying the similarity between all keypoints. Costs for d objects decompose into a sum of costs for each of the $d(d-1)/2$ object pairs. The latter, in turn, are sums of *linear*, keypoint-to-keypoint costs and *quadratic*, keypoint-pair-to-keypoint-pair costs. Quadratic costs allow the modeling of mutual geometric relations between keypoints, considerably improving matching accuracy [17].

[*]The two authors contributed equally to this paper.

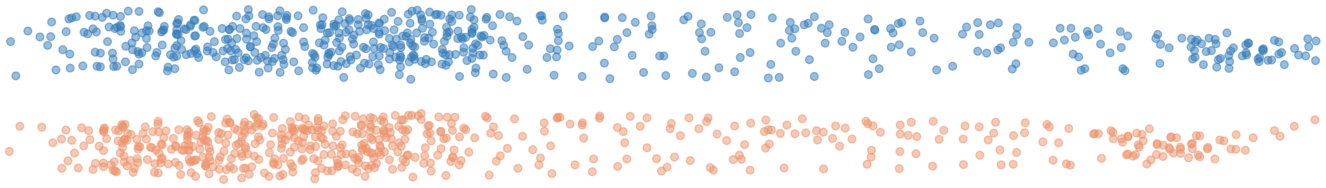

Figure 2. **Two exemplary worms** from the considered datasets, side view. We visualize the keypoints as small circles, which represent candidates for a possible nuclei at the respective 3D location. This forms the input to the multi-graph matching problem, and it can be observed that even for humans it is a very challenging task to bring the correct keypoints into correspondence. The problem setting is incomplete since the blue worm has 555 keypoints, the orange one 565 and the total number of nuclei in each worm is exactly 558. Furthermore, for computational reasons the problem setting has to be sparse. Only about 3% of possible keypoint-to-keypoint matchings are allowed, *i.e.*, have finite costs. For pre-processing steps like straightening of the worms and cost definitions see [22].

The problem combining these conditions is known as *multi-graph matching (MGM)*, and it is NP-hard [14, 43] in general. Note, the term *graph* originally stems from interpreting objects as graphs and deriving quadratic costs from differences of respective edge attributes. We call an MGM problem *sparse* if most keypoint-to-keypoint matchings are *forbidden* (otherwise *allowed*). An MGM problem where all matchings are allowed is called *dense*.

Fig. 2 shows two exemplary worms with a different number of keypoints, moreover, only a small fraction of keypoint-to-keypoint matchings are allowed. Hence, by construction, we are given an incomplete MGM problem with a sparse cost structure. Despite finding a good matching, certainly runtime also plays a key role. Firstly, a fast method may scale better for problems with a larger number of worms. Secondly, the cost structure has so far been designed, and partly learned, using biologically motivated prior knowledge. In the future, a next step is to learn, at least, some hyper-parameters of the model with e.g. Bayesian optimization [40], such as the cost of forcing keypoints to be matched. This means that the MGM task is inside a learning-loop and hence must be solved in reasonable runtime. Thirdly, MGM has so far been used rarely in practical applications, apart from bioimaging. We conjecture that the reason is a lack of good and fast MGM solvers.

Our contributions are as follows: (1) We prove that methods addressing the *complete* multi-graph matching problem are not practically applicable in an *incomplete* setting, since their runtime becomes excessive. (2) We present a new algorithm, *GREEDA*, for efficiently solving the incomplete multi-graph matching problem with sparse costs, which can handle linear, quadratic, and, theocratically, any higher-order costs. (3) We considerably outperform the state-of-the-art methods for matching the large-scale *worms-100* dataset, both in terms of runtime and total costs. Furthermore, for small-scale toy and synthetic data we are better or on par with the leading methods but at least an order of magnitude faster, depending on problem size. Our code is available at https://github.com/vislearn/multi-matching-optimization.

2. Real-world applicability of MGM methods

Large-scale MGMs in bioimaging applications, as in Fig. 2, have two decisive properties hardly covered by existing MGM algorithms: *Incompleteness* and *sparsity*. Whereas incomplete MGMs naturally arise due to noise in images and pre-processing steps like segmentation, sparsity allows to limit the size of the problem and encode important spatial relationships. Limiting the size is important as the number of quadratic costs grows as $O(n^4d^2)$, where n is the number of keypoints and d the number of objects. Hence, dense MGMs with $n > 100$ become practically infeasible. Additionally, *e.g.*, forbidding the matching of nuclei in the head of a worm to those in its tail is an important natural prior (see Fig. 2).

Complete vs. incomplete MGM. Many approaches [5, 43, 48, 50, 51] only treat complete MGM. Despite popular claims to the contrary, there is no straightforward and efficient way to apply them to the incomplete case. While a polynomial reduction from incomplete to complete graph matching exists [17], its often-mentioned generalization to MGM, *e.g.*, [48, 50, 51], is prohibitively expensive. Transforming an *incomplete* MGM problem with d objects, each with n keypoints, into a *complete* one results in nd keypoints in *each* of the d objects, see Suppl. D. This makes any complete MGM algorithm impractical. The following theorem proven in Suppl. D essentially states that there exists *no significantly better* reduction.

Theorem 1. *Let B be a complete MGM problem instance that is a clique-wise reduction of an incomplete MGM problem instance A. Let A have $|V|$ vertices in total. Than* each object *in B has at least $|V|$ vertices.*

In turn, our approach applies to incomplete MGMs *directly*.

Sparse problems. As defined above, an MGM problem is sparse if most keypoint-to-keypoint matchings are forbidden. In practice, forbidden matches are modeled by *infinite* costs that are only implicitly present in the problem description [17, 42, 44], rendering the latter *sparse*. However, many methods, especially the solution construction

and synchronization ones, discussed in the following paragraph, do not apply to infinite costs. Even worse, they often fail to reconcile infinite costs with their assumptions, *e.g.*, by heavily relying on spectral methods [1, 8, 34]. In contrast, our method guarantees to return only *allowed*, *finite-cost* matchings for *sparse* problems.

3. Related work

Graph matching (GM) refers to the well-studied [17] special case of matching *two* objects. Similar to MGM, one distinguishes *complete* and *incomplete* GM. Complete GM is also known as the NP-hard [35] *quadratic assignment problem (QAP)* [26] or, if quadratic costs are zero, the polynomially solvable [25] *linear assignment problem (LAP)* [10]. MGM can be viewed as $d(d-1)/2$ GM problems between all object pairs, coupled via cycle consistency constraints. If we wish to distinguish between cost orders similar to GM, we refer to (complete *or* incomplete) MGM as Multi-QAP and, respectively, as Multi-LAP if quadratic costs are zero. Unlike the LAP, the Multi-LAP is NP-hard [14, 43]. As a result of this coupled structure, GM solvers are an essential but interchangeable component of many MGM methods, including those considered in this work. We utilize the GM solver [20] as a GM subroutine since it shows the best results in the recent GM benchmark [17].

Multi-graph matching Although MGM has not yet received the same attention as GM, several types of solvers have been proposed in recent years. In the following we review their ability to deal with incomplete and sparse MGMs. The works most closely related to our algorithm are additionally addressed in Sec. 4.

GM-solver-based primal heuristics is probably the largest class of MGM algorithms including ours. These methods usually consist of *construction* and *local search* subroutines that employ a GM solver as their decisive component. Construction methods are usually based on the *composition principle*: They assign a matching between two objects by composition via a third one. This third object is either iteratively chosen w.r.t. a metric combining costs *and* a (pairwise) consistency measure [21, 49] or is a node in a *spanning tree* [43, 50, 51] of the complete graph connecting all objects as its nodes. Whereas the first subclass does not even guarantee cycle consistency of the result, the second cannot address sparse problems as forbidden matchings are often selected through composition. Moreover, existing methods of this type consider only complete MGMs.

In contrast, our approach extends a feasible solution consisting of $k \leq d$ objects by one object in each iteration until the solution includes all objects, *i.e.*, $k = d$. In combination with the incomplete GM solver [20], it guarantees a cycle-consistent and allowed matching.

Local search subroutines are usually based on the observation that cycle-consistent matchings can be improved by re-matching one object to the remaining, already matched $(d-1)$ objects, see Fig. 4. It turns out that this re-matching constitutes a GM problem. Hence, one can iteratively re-match different objects utilizing a GM solver. We refer to this algorithm as *GM local search* (GM-LS), see Sec. 4.3 for details. This idea was initially proposed by [4] for the closely related *multi-dimensional assignment problem (MDAP)* and then independently for different MGM variants [43, 48, 50, 51]. Similarly to the construction subroutine, existing works explicitly consider local search only for the complete problem.

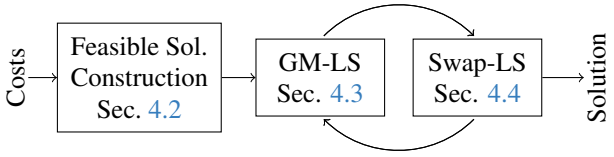

Figure 3. **Conceptual diagram of our MGM method GREEDA.** *GREEDA* is composed of three parts – a construction heuristic (Sec. 4.2) and two local search heuristics *GM-LS* (Sec. 4.3), *SWAP-LS* (Sec. 4.4).

In this work we *parallelize* GM local search and make it applicable to incomplete sparse MGMs. We also propose another local search method, referred to as *(clique) swap local search*, and so far unknown in the computer vision literature. The related work from the operations research is discussed in Sec. 4.4.

Synchronization approaches [12, 13, 39] first independently solve all pairwise GM problems, which induces a *cycle-inconsistent* matching. Afterward, they try to find a *cycle-consistent approximation* by changing a *minimal* number of matchings [1, 8, 30]. Such synchronization methods often serve as a subroutine for convex-relaxation-based MGM solvers, *e.g.*, [5, 41]. However, they are *expensive* and prone to *suboptimal* decisions as they require the solution of $d(d-1)/2$ GM problems and ignore costs during the approximation. Due to the latter, existing synchronization methods also fail to deal with sparse problems because they often introduce *"blunders"* in choosing forbidden matchings. In contrast, our algorithm considers the problem costs in any of its stages and thus avoids blunders.

Convex relaxation-based methods is probably the smallest subclass among existing MGM techniques. The work [5] considers lifting-free quadratic relaxation, but addresses complete problems only. The work [24] proposes a powerful semi-definite relaxation for incomplete MGMs, but its scalability is strongly limited due to variable lifting, as mentioned by the authors. The closest competitor for our method is the Lagrange relaxation-based method [41], able to deal with incomplete sparse problems. In Sec. 5 we demonstrate, however, that our algorithm significantly outperforms [41] in terms of runtime and objective value.

4. Our method

In Sec. 4.1, we formalize the MGM problem. In Secs. 4.2 to 4.4, we describe each individual building block of our method, which is summarized in Fig. 3, along with the respective technical contribution. We term our method *GREEDA* alluding to the greedy nature of our construction algorithm, see Sec. 4.2.

4.1. Formal problem definition

Graph matching concerns itself with matching two finite keypoint sets V^1 and V^2, further referred to as *vertex sets* or *objects*. It considers the undirected complete bipartite graph $\overline{G} = \left(V \coloneqq V^1 \cup V^2, \overline{E} \coloneqq V^1 \times V^2\right)$ with objects V^1 and V^2 as independent sets, where an edge $ij \coloneqq (i,j) \in \overline{E}$ corresponds to matching vertex $i \in V^1$ to vertex $j \in V^2$. Although discussing undirected graphs, we write the edge set $\overline{E}$ as a Cartesian product $V^1 \times V^2$ to emphasize the two independent sets V^1 and V^2. Due to their independence, we can always identify directed with undirected edges. An *incomplete* matching between object V^1 and V^2 is defined as subset of edges $E \subset \overline{E}$ containing *at most one* incident edge for each vertex. *Complete* matchings, conversely, contain *exactly one* incident edge for each vertex, which demands equal cardinalities of both objects $|V^1| = |V^2|$. The goal of GM is to find minimal matchings w.r.t. given costs $C : \left(V^1 \times V^2\right)^2 \mapsto \mathbb{R}$. In the matrix identification $C(i,s,j,t) = C_{is,jt}$, diagonal entries $C_{is,is}$ describe *linear* costs, and off-diagonal entries *quadratic* costs, see Fig. 1. Linear costs $C_{is,is}$ penalize *vertex-to-vertex* correspondences, *i.e.*, matching vertex $i \in V^1$ to vertex $s \in V^2$, whereby infinite costs $C_{is,is} = \infty$ *forbid* such a matching. Quadratic costs $C_{is,jt}$, in turn, penalize *vertex-pair-to-vertex-pair* correspondences, *i.e.*, matching the pair $(i,j) \in V^1 \times V^1$ to the pair $(s,t) \in V^2 \times V^2$.

Multi-graph matching generalizes GM by matching multiple, $d \in \mathbb{N}_{\geq 3}$ objects V^p, $p \in [d] \coloneqq [1,d] \cap \mathbb{N}$, w.r.t. costs between all pairs $C^{p,q} : \left(V^p \times V^q\right)^2 \mapsto \overline{\mathbb{R}}$, $p \neq q$, $p,q \in [d]$. Instead of a bipartite graph, it considers the undirected complete *d-partite* graph $\overline{G} = \left(V \coloneqq \cup_{p\in[d]} V^p, \overline{E}\right)$ with objects V^p as independent sets. *Incomplete* multi-matchings are subsets of edges $E \subset \overline{E}$ *s.t.* any vertex is incident to *at most one* edge connecting the same objects. Similar to GM, *complete* multi-matchings require *exactly one* such edge and equal cardinalities of objects.

Cycle consistency. A multi-matching $E \subset \overline{E}$ is *cycle-consistent* if each *path* in E can be extended within E to a *cycle* with at most one vertex per object. As shown in [41], enforcing this for all 3-cycles is sufficient. That is, a multi-matching $E \subset \overline{E}$ is cycle-consistent iff $ij \in E$ and $jk \in E$ imply $ik \in E$ for all $i,j,k \in V$, see Fig. 1.

Clique representation. In [45], it was shown that a multi-matching $E \subset \overline{E}$ is cycle-consistent iff the corresponding subgraph $G = (V,E)$ is a *union of cliques*, *i.e.*, there exist partitions $\{Q_l\}_{l\in L}$ of vertices V and $\{E_l\}_{l\in L}$ of edges E indexed via the same finite set L, *s.t.* for each index $l \in L$ the subgraph $G|_{Q_l}$ restricted to part Q_l is a *clique* with edges E_l. Therefore, cycle-consistent multi-matchings are induced by vertex partitions $\mathcal{Q}$ where any part $Q \in \mathcal{Q}$ contains at most one element per object V^p, *i.e.*, $|Q \cap V^p| \leq 1$, see Fig. 1. They translate to multi-matchings by considering elements of the same part Q as matched to each other. We call such vertex partitions *feasible* and denote the set of feasible vertex partitions or *solutions* as $\mathbb{Q}$. The set of feasible partitions over the vertices $V^D \coloneqq \bigcup_{p\in D} V^p$ of an object *subset* $D \subseteq [d]$ is denoted by $\mathbb{Q}^D$. For simplicity, partitions permit the empty set. Abusing terminology, we refer to parts $Q \in \mathcal{Q}$ of a solution as *cliques*. The object subset actually covered by such a clique $Q \in \mathcal{Q}$ is denoted as $D(Q) \subseteq [d]$, and a clique's vertex belonging to object $p \in D(Q)$ as Q^p, *i.e.*, $\{Q^p\} \coloneqq Q \cap V^p$, see Fig. 4.

The MGM objective is to find cycle-consistent multi-matchings minimizing the sum of all costs, *i.e.*,

$$\min_{\mathcal{Q}\in\mathbb{Q}} \left[C(\mathcal{Q}) \coloneqq \sum_{Q,R\in\mathcal{Q}} \sum_{\substack{p,q\in D(Q)\cap D(R)\\ p<q}} C^{p,q}_{Q^pQ^q,R^pR^q} \right], \quad (1)$$

where we assume $C^{p,q} = C^{q,p}$ and count this cost for each pair of objects $p,q \in [d]$, only once by requiring $p < q$. Note that the formulation in Eq. (1) implicitly assumes zero costs for *not* matching a vertex.

Alternative formulations. The clique formulation from Eq. (1) is non-standard. Many formulations [13, 41, 52] represent (multi-)matchings $E \subset \overline{E}$ by *partial permutation matrices* X, where $X_{is} = 1$ iff $is \in E$. Others [7, 33, 34] view MGM as a *labeling problem*, where each vertex of the same object must be assigned a different label – commonly called *universe point*. Vertices with the same label are matched to each other, *i.e.*, comprise a clique. We use the clique formulation because it allows the most natural description of our algorithm.

4.2. Feasible solution construction

Basic algorithm. The basic construction Alg. 1 obtains feasible solutions by solving a *chain* of GM problems. Given a *random ordering* of objects $[d]$, it iteratively extends a *partial solution*, which, in the k-th iteration, matches the objects $V^1, \dots, V^k$. The matching E is the solution of the GM problem with costs $C^{k+1,\mathcal{Q}}_{iS,jT}$, $i,j \in V^{k+1}$, $S,T \in \mathcal{Q}$, stemming from the *summation* over a clique's individual elements

$$C^{k+1,\mathcal{Q}}_{iS,jT} \coloneqq \sum_{q\in D(S)\cap D(T)} C^{k+1,q}_{iS^q,jT^q}. \quad (2)$$

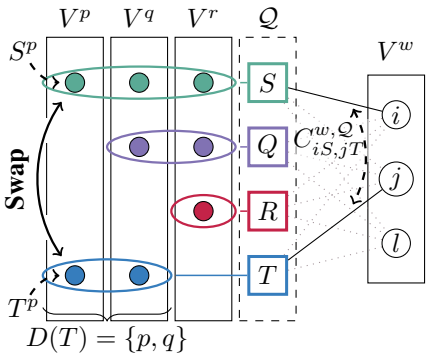

Figure 4. **Feasible solution construction and swap local search.** Depicted are three cycle-consistently matched objects V^p, V^q, V^r. Matched vertices are of the same color, horizontally aligned, and decomposed into cliques S, Q, R, T (ellipses), yielding a partial solution $\mathcal{Q}$ (dashed rectangle). As example, the clique T spans the objects $D(T) = \{p, q\}$, where its vertex T^p belongs to object V^p. The costs $C^{w,\mathcal{Q}}_{iS,jT}$ for matching vertex i and j of object V^w to cliques S and T of the partial solution $\mathcal{Q}$ are also shown. We consider two types of swaps (solid arrow), see Sec. 4.4: The *vertex swap* fixes the object p and interchanges the vertices S^p and T^p. The *2-clique swap* is more global and jointly optimizes over all possible vertex swaps between cliques S and T.

The partial solution is extended by adding matched vertices $i \in V^{k+1}, iS \in E$ to their assigned cliques $\{i\} \cup S \in \mathcal{Q}'$. If a vertex $i \in V^{k+1}$ is unmatched, *i.e.*, $\forall jS \in E : j \neq i$, it is added as a singleton $\{i\} \in \mathcal{Q}'$. If a clique $S \in \mathcal{Q}$ is unmatched, *i.e.*, $\forall iT \in E : S \neq T$, it remains unchanged $S \in \mathcal{Q}'$. We denote the (partial) solution resulting from this *merge* as $\mathcal{Q}' = \mathbf{merge}(V^{k+1}, \mathcal{Q}; E)$.

Alg. 1 has six notable properties: (1) It guarantees cycle-consistency. (2) It can use virtually any GM solver as a black-box. (3) It is independent of the cost order and thus applicable to higher-order problems as long as the underlying GM solver is applicable. (4) It is parameter-free w.r.t. the number of cliques. (5) It scales linearly in the number of objects $d \in \mathbb{N}_{\geq 3}$. (6) It is randomized, hence the best solution from its multiple parallel runs can be kept, see [15].

Extensions and parallelization. We propose two extensions of Alg. 1: *Incremental* and *parallel* construction.

Incremental construction addresses the problem of *error propagation*. Errors during early matchings "propagate" along the chain of pairwise problems, sway later matchings, and worsen the final solution. Because MGM problems' restrictions to object subsets are again MGM problems, we propose to weaken such propagations by using a *better* (but potentially more expensive) *multi-matching solver* for the first $s \in [d]$ objects $[s]$.

Parallel construction improves the linear scaling w.r.t. the number of objects. It generalizes the chain of pairwise problems to *binary trees*, specifically *leaf-labeled, ordered, binary trees* with *leaf label set* $[d]$. Leaf labels associate leaves with objects. Each tree vertex corresponds to a

Algorithm 1: Basic solution construction.

$\mathcal{Q} \leftarrow \bigcup_{i \in V^1} \{i\}$
for $k \in \{1, \dots, d-1\}$ **do**
  `// compute matching` E `between object` $(k+1)$ `and the already matched` $[k]$
  $E \leftarrow$ (Approx.) solve GM with costs $C^{k+1,\mathcal{Q}}$ (Eq. (2))
  $\mathcal{Q} \leftarrow \mathbf{merge}(V^{k+1}, \mathcal{Q};\ E)$

(partial) solution $\mathcal{Q} \in \mathbb{Q}^D$ matching its descendant leaves $D \subseteq [d]$. In analogy to the basic variant, a tree vertex's (partial) solution is obtained by first solving a GM problem that matches the cliques of its children $\mathcal{A} \in \mathbb{Q}^{D_A}$, $\mathcal{B} \in \mathbb{Q}^{D_B}$ and then merging matched cliques. Pairwise costs $C^{\mathcal{A},\mathcal{B}}_{IS,JT}, I, J \in \mathcal{A}, S, T \in \mathcal{B}$, are again obtained by summing costs of a clique's individual elements,

$$C^{\mathcal{A},\mathcal{B}}_{IS,JT} \coloneqq \sum_{p \in D(I) \cap D(J)} \sum_{q \in D(S) \cap D(T)} C^{p,q}_{I^p S^q, J^p T^q}\,. \quad (3)$$

Ultimately, the root corresponds to a final solution. GM problems of tree vertices at the same level can be solved in *parallel*. Therefore, *balanced* trees yield the best acceleration – in sequence only $O(\log(d))$ instead of $O(d)$ problems need solving. Additionally, most properties of the basic construction transfer to the parallel version: It is *randomized* w.r.t. the leaf order, allows *plug-&-play* of pairwise solvers, guarantees *cycle-consistency*, and is *parameter-free* w.r.t. the number of cliques. Pseudocode and further details are provided in Suppl. A.

Our novelty and related work. The basic solution construction Alg. 1 was first mentioned in [3] and later used in [4] for complete Multi-LAPs seen as a special case of the MDAP. Experiments in [4] also demonstrate its superiority over star-shaped spanning tree approaches. Its application to *Multi-QAPs* and its *parallelization* using binary trees are new. Incremental construction conceptually resembles an idea proposed in [43], where, as part of a spanning tree heuristic, a local search is run on the hitherto constructed partial solution. To summarize, our contribution is to introduce the prior art from operations research [3, 4], applying it to *Multi-QAPs* instead of Multi-LAPs, and proposing new extension and parallelization schemes.

4.3. GM local search

Basic algorithm. The basic GM local search step comprises *splitting* and *merging* an object V^p from and to a given solution $\mathcal{Q} \in \mathbb{Q}$ as defined by Alg. 2. An object V^p is split by *subtracting* $Q \setminus V^p$ its vertices from all cliques $Q \in \mathcal{Q}$. Merging it back to the split solution works just as in Sec. 4.2 by solving a GM problem with costs from Eq. (2) whose solution dictates the merge. While only accepting improvements, the basic GM local search Alg. 2 re-

Algorithm 2: Basic GM local search.

Input: Solution $\mathcal{Q} \in \mathbb{Q}$, Object Sequence $(p_k)_{k\in\mathbb{N}}$
$k \leftarrow 1$
while stopping criterion not met **do**
  `// split object k from the matching Q`
  $\mathcal{Q}' \leftarrow \{\, Q \setminus V^{p_k} | Q \in \mathcal{Q}\}$
  `// compute matching E between object k and the rest`
  $E \leftarrow$ (Approx.) solve GM with costs $C^{p_k,\mathcal{Q}'}$ (Eq. (2))
  $\mathcal{Q}' \leftarrow \mathbf{merge}(V^{p_k}, \mathcal{Q}';\ E)$
  if $C(\mathcal{Q}') < C(\mathcal{Q})$ **then**
    $\mathcal{Q} \leftarrow \mathcal{Q}'$ `// accept if profitable`
  $k \leftarrow k + 1$

peats this step along an *object sequence* $(p_k)_{k\in\mathbb{N}}, p_k \in [d]$, until a suitable stopping criterion is met. In practice, we run Alg. 2 cyclically along the random object ordering $[d]$ used in Alg. 1 as long as the solution improves. Like Alg. 1, the pairwise solver is *blackboxed* and solutions are always *cycle-consistent*.

Extensions and parallelization. A common extension [4] of this local search is to split a solution $\mathcal{Q} \in \mathbb{Q}$ along a subset of objects $D \subset [d]$. As a result, two partial solutions $\{Q \cap V^D | Q \in \mathcal{Q}\} \in \mathbb{Q}^D$, $\{Q \cap V^{[d]\setminus D} | Q \in \mathcal{Q}\} \in \mathbb{Q}^{[d]\setminus D}$ are merged using Eq. (3). Since these extensions performed worse than the base in preliminary experiments, we did not investigate them further.

Our proposed *parallelized* search step consists of two passes. First, it runs the basic search step for *all* objects $p \in [d]$ in *parallel*, splitting d objects $\mathcal{Q}^p := \{Q \setminus V^p | Q \in \mathcal{Q}\}$, generating d matchings $E^p \subset V^p \times \mathcal{Q}^p$, yielding d proposed solutions, but ultimately leaving the input solution $\mathcal{Q} \in \mathbb{Q}$ untouched. Second, it splits and merges objects according to the collected matchings $E^p, p \in [d]$, in *ascending* order of their proposed solutions' objective value – only accepting profits. The more matchings yield profit despite being "outdated" (because prior matchings changed the solution), the more *acceleration* is generated over the basic search step. Both the basic and parallel search step *sequentially* solve only *one* GM problem. Additionally, the parallelized step always accepts the *most profitable* proposal though this is only a *secondary* source of acceleration, as demonstrated in Sec. 5.1. Further details can be found in Suppl. B.

Our novelty and related work. Again, the basic Alg. 2 was first mentioned in [4] targeting complete Multi-LAPs as special case of the MDAP. In [4], its extension to object subsets is investigated and our preliminary findings are confirmed for the Multi-LAP. In computer vision, it was independently proposed in [43, 51] for Multi-LAPs and in [48, 50] for Multi-QAPs, both only complete. Our parallelized extension is novel.

Algorithm 3: Swap local search.

Input: Solution $\mathcal{Q} \in \mathbb{Q}$
while stopping criterion not met **do**
  for $Q, R \in \mathcal{Q}$ **do**
    $x^* \leftarrow$ Approximately solve Eq. (5) for cliques Q, R
    `// swap the vertices according to` x^*
    $Q', R' \leftarrow \mathrm{swap}(x^*, Q, R)$
    `// update the clique representation`
    $\mathcal{Q}' \leftarrow (\mathcal{Q} \setminus \{Q, R\}) \cup \{Q', R'\}$
    if $C(\mathcal{Q}') < C(\mathcal{Q})$ **then**
      $\mathcal{Q} \leftarrow \mathcal{Q}'$ `// accept if profitable`

4.4. Swap local search

Vertex swap. Given a solution $\mathcal{Q} \in \mathbb{Q}$, *swapping* vertices of the same object between two cliques $Q, R \in \mathcal{Q}$ maintains feasibility. While fixing one object $p \in [d]$, a *vertex swap* subtracts a vertex covered by either clique from said clique and adds it to the other, yielding two new cliques, Q' and R', see Fig. 4. For example, if both cliques Q and R cover the fixed object $p \in [d]$, the vertices Q^p and R^p are interchanged, *i.e.*, $Q' = (Q \setminus \{Q^p\}) \cup \{R^p\}$ and $R' = (R \setminus \{R^p\}) \cup \{Q^p\}$. If only one clique covers the fixed object, only one vertex is exchanged. Finally, all cliques remain unchanged if none covers the object. After a swap, cliques still cover at most one vertex per object. The new solution $\mathcal{Q}' = (\mathcal{Q} \setminus \{Q, R\}) \cup \{Q', R'\}$ is, thus, also feasible. Furthermore, the objective value change induced by a vertex swap decomposes into a sum of objective value changes $\delta C^{p,q}_{\text{swap}}(Q, R)$ between the fixed object $p \in [d]$ and all other objects $q \in [d] \setminus \{p\}$, *i.e.*,

$$C(\mathcal{Q}') - C(\mathcal{Q}) = \sum_{q\in[d]\setminus\{p\}} \delta C^{p,q}_{\text{swap}}(Q, R)\,. \tag{4}$$

A detailed formulation is given in Suppl. C.1.

2-clique swap. We leverage Eq. (4) to *jointly* consider *all possible* swaps between two cliques $Q, R \in \mathcal{Q}$. A 2-clique swap is a set of binary decision variables $x = \{x^p\}_{p\in[d]} \in \{0, 1\}^d$, deciding for every object $p \in [d]$, whether the vertex swap fixing it should be performed ($x^p = 1$), or not ($x^p = 0$). Performing all vertex swaps specified by a 2-clique swap $x \in \{0, 1\}^d$ yields, again, two new cliques Q' and R', which we denote by $Q', R' = \mathrm{swap}(x; Q, R)$. The optimal 2-clique swap x^* solves the following quadratic pseudo-boolean optimization problem [9]:

$$x^* \in \operatorname*{arg\,min}_{x\in\{0,1\}^d} \sum_{p\in[d]} x^p \left(\sum_{q\in[d]} \overline{x^q} \cdot \delta C^{p,q}_{\text{swap}}(Q, R) \right), \tag{5}$$

where $\overline{x} = (1 - x)$ denotes negation. This problem is NP-hard [37], which is why we address it using the state of the

art [20] approximative solver QPBO-I [37]. Our swap local search Alg. 3 iterates over all clique pairs, solving the problem from Eq. (5) with QPBO-I and performing the obtained 2-clique swap if profitable.

Our novelty and related work. The vertex swap was originally introduced for the MDAP with three objects [2] and later extended [23, 31, 36] to consider multiple swaps at the same time. However, these propositions come without a method to solve the 2-clique swap problem efficiently. Our contribution is formulating and solving the 2-clique swap as a quadratic pseudo-boolean optimization problem.

5. Experimental validation

Our experiments comprise an *ablation study* (Sec. 5.1) and *a comparison to other methods* (Sec. 5.2). Whereas the ablation study compares different variants of our *GREEDA* algorithm's *construction heuristic* and GM-LS, see Fig. 3, in the comparison to others, we utilize their *sequential* variants. We solve GM subproblems of *GREEDA* using the state-of-the-art GM solver [19, 20], which we turn from a randomized algorithm into a deterministic one by fixing its randomization seed. Further details about the experimental setup and solver settings are given in Suppl. E.

Contrary to most existing works, we only compare *objective values* or *costs* instead of *ground-truth-based accuracies* since none of the considered algorithms explicitly optimize the latter or has access to the ground truth. We leave accuracy comparisons to the works addressing modeling and learning questions.

Datasets. We evaluate methods on the established datasets: *synthetic*, CMU *hotel* and *house*, and *worms*. All of them are widely used in MGM literature and accessible through the archive accompanying [42]. Additionally, to showcase our approach, we extended the *worms* dataset with a new, large-scale problem instance, *worms-29*, by combining all 29 worm-based objects available in the *worms* dataset. In this instance, we reduced the size of each pairwise problem by forbidding many high-cost vertex-to-vertex matchings, making the problem even sparser: Each vertex is allowed to match 14 instead of 23 other vertices on average. All other characteristics, including costs and number of vertices per object, are the same as in *worms*. Finally, based on the recently published work [27], we constructed the *worms-100* problem instance with about 45M keypoint correspondences (compared to the 3M of the *worms-29* dataset).

5.1. Ablation study

We compare the *sequential*, *parallel*, and *incremental* solution construction methods from Sec. 4.2 in all possible combinations with the *sequential* and *parallel GM local search* from Sec. 4.3. For lack of a better solver, *incremental* construction uses *GREEDA* for the first 5 objects. To showcase our parallelization, we additionally consider a straightforward *"best-improvement"* parallelization of the GM local search: Although it performs all GM-LS steps in parallel, it accepts only the most profitable one. In this respect it differs from the parallel local search as described in Sec. 4.3, which aims to accept *all possible* profitable changes. The experiments were run on *worms-10*, containing the largest MGM instances addressed in computer vision [41] so far.

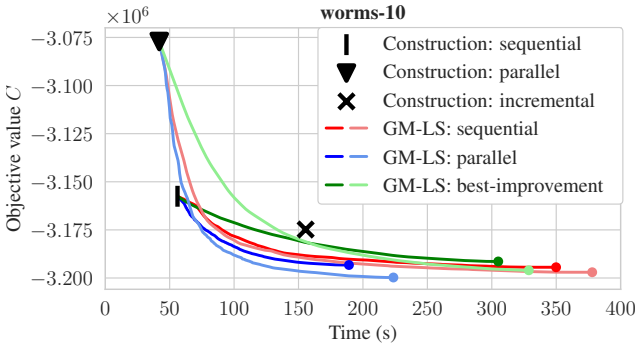

Figure 5. **Ablation study.** Convergence of sequential and parallel *GM local search* variants after sequential and parallel construction, respectively. For the incremental construction only the objective value is given to keep readability of the plot. Though taking notably more time, this method does not result in a better solution. The result is a single run on the worms-10 dataset and averaged over the first 10 problem instances.

Convergence speed. Fig. 5 depicts costs with respect to runtime. While all methods converge to virtually the same value, *parallel* GM-LS converges fastest. It also notably outperforms the *"best-improvement"* parallelization.

Distribution of objective value. To assess the influence of randomization, we run each algorithm 100 times using different object orderings. We found the differences to be negligible and refer to Suppl. F for a discussion and details.

5.2. Comparison to other methods

Algorithms. We compare our sequential variant of *GREEDA* to the state of the art MGM methods *MP-T* [41] and *DS** [5]. While the latter only applies to complete and dense MGM problems, the former can solve incomplete and sparse MGM problems. We also compare to the following synchronization methods: The projected power iteration (*PPI*) [12], Sparse *Stiefel* Synchronization [8], and the *Spectral* approach [34] extended by the Successive Block Rotation Algorithm [6] to yield incomplete matchings. We do *not* compare to the recently released *pygmtools* package [46], as the respective MGM algorithms only apply to complete problems and do not guarantee cycle consistency.

Implementation Details. While *GREEDA* and *MP-T* are available in C++, *DS**, *Stiefel*, *PPI*, and *Spectral* are Matlab implementations. We use the original software from the authors for all but *PPI*, which we re-implemented ourselves.

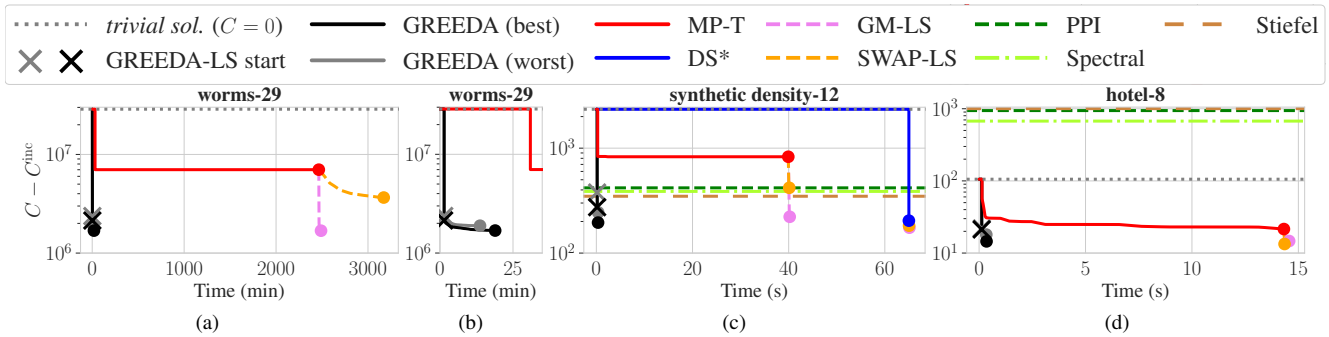

Figure 6. **Method Comparison.** Objective with respect to runtime comparison. Objective plotted in log-scale and offset by the *inconsistent* solution's objective C^{inc} obtained from independently solving $d(d-1)/2$ pairwise GM problems. The value C^{inc} approximates, since [20] is approximative, a lower bound to Eq. (1) provided by the MGM relaxation that ignores cycle consistency.
For our sequential variant of *GREEDA*, the best and worst result over 10 runs is given. (b) The first minutes of Fig. 6a, illustrating the considerable difference in construction time between *GREEDA* and *MP-T* for large problems. (c) and (d) provide averaged results over all instances of the 12 object *synthetic density* and the 8 object *hotel* dataset respectively. (a), (b) compare only methods able to find allowed solutions of sparse problems. *DS** results are shown for (c) only, as the other datasets contain incomplete problems. Note, that even 10 sequential runs of *GREEDA* to attain *GREEDA (best)*'s results would still render the fastest algorithm.

Results. For the *worms-100* dataset, the only true competitor *MP-T* runs out of memory on a machine with 250GB of RAM. In contrast, *GREEDA* (without *SWAP-LS*) constructs a feasible solution in 10 minutes and converges after about an hour, using 65GB of RAM.

Otherwise, for brevity, we only provide the comparison for three representative datasets, and refer to Suppl. H for a full list of results with all algorithms and datasets (in total 321 problem instances shown in 33 tables).

Fig. 6 shows the costs-runtime plots for different methods. Because of incomparable implementations (C++, Python, Matlab), we only consider the final objective value for synchronization methods and not their runtime. In general, the need to compute the initial cycle inconsistent solution by solving all $d(d-1)/2$ pairwise problems, though completely parallelizable, makes runtime comparisons with *direct* MGM methods difficult.

Overall, *GREEDA* consistently outperforms or is on par with competitors in terms of final objective value, and is an order of magnitude faster in terms of runtime (where applicable). In particular, for our new, large-scale *worms-29* dataset, see Fig. 6a, *GREEDA* achieves a better solution much faster than the direct competitor *MP-T*, improving construction time from half an hour to just two minutes (see zoom in Fig. 6b and Suppl. G). While *MP-T* fails to improve upon its initial solution significantly, our local search subroutines *GM-LS* and *SWAP-LS* quickly improve the result but converge slowly. In our current implementation, *SWAP-LS* is the main bottleneck due to its quadratic scaling in the number of vertices, which is particularly large for the *worms-29* dataset.

As mentioned in Sec. 3, synchronization methods may introduce blunders that substantially impact the resulting cost. This can be seen in Fig. 6d, where all synchronization methods return a solution that is worse than the trivial one, which would leave all vertices unmatched.

Fig. 6c includes a comparison to *DS**, which yields high-quality results but is substantially slower than the other methods, partially due to its Matlab implementation. Since it is a solver for complete problems, it cannot be applied to the *hotel*, *house*, or *worms* datasets, as the incomplete to complete problem transformation would increase the runtime even further, see Suppl. D.

All plots in Fig. 6 additionally show the application of *GM-LS* and *SWAP-LS* to the results of the other methods. Although the resulting objective values are comparable to *GREEDA*, optimization takes significantly longer.

6. Discussion and limitations

In this work we haven proven that methods for the complete multi-graph matching (MGM) problem are impractical for the incomplete setting. We proposed a new method for large-scale MGM problems that are incomplete and sparse. In particular, for the large-scale *worms* problems we obtain *better* results in considerably less time than competitors. But even for small-scale instances we outperform competitors in terms of speed, while having at least on par results.

Our algorithm's good performance is due to two reasons: Firstly, it decomposes the MGM problem into a series of fast to optimize GM problems, while using *all* costs and keeping cycle consistency at *all* times. Secondly, our two local search methods efficiently explore the exponentially large neighborhood of a current solution.

The modern approach to MGM includes learning costs with neural networks. Although a detailed discussion of such methods is beyond the scope of this work, our solver can be used within them, due to its reasonable runtime.

Acknowledgements

This work was supported by the German Research Foundation projects 498181230 and 539435352. Authors further acknowledge facilities for high throughput calculations bwHPC of the state of Baden-Württemberg (DFG grant INST 35/1597-1 FUGG) as well as Center for Information Services and High Performance Computing (ZIH) at TU Dresden.

References

[1] Federica Arrigoni, Eleonora Maset, and Andrea Fusiello. Synchronization in the symmetric inverse semigroup. In *Image Analysis and Processing-ICIAP 2017: 19th International Conference, Catania, Italy, September 11-15, 2017, Proceedings, Part II 19*, pages 70–81. Springer, 2017. 3

[2] Egon Balas and Matthew J Saltzman. An algorithm for the three-index assignment problem. *Operations Research*, 39 (1):150–161, 1991. 7

[3] Hans-Jürgen Bandelt, Yves Crama, and Frits CR Spieksma. Approximation algorithms for multi-dimensional assignment problems with decomposable costs. *Discrete Applied Mathematics*, 49(1-3):25–50, 1994. 5

[4] Hans-Jürgen Bandelt, Arjan Maas, and Frits CR Spieksma. Local search heuristics for multi-index assignment problems with decomposable costs. *Journal of the Operational Research Society*, 55(7):694–704, 2004. 3, 5, 6

[5] 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, 3, 7, 13, 14, 15, 16

[6] Florian Bernard, Johan Thunberg, Jorge Goncalves, and Christian Theobalt. Synchronisation of partial multi-matchings via non-negative factorisations. *Pattern Recognition*, 92:146–155, 2019. 7, 13, 14, 15, 16, 17, 18, 19, 20, 21

[7] Florian Bernard, Johan Thunberg, Paul Swoboda, and Christian Theobalt. Hippi: Higher-order projected power iterations for scalable multi-matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10284–10293, 2019. 4

[8] Florian Bernard, Daniel Cremers, and Johan Thunberg. Sparse quadratic optimisation over the stiefel manifold with application to permutation synchronisation. *Advances in Neural Information Processing Systems*, 34:25256–25266, 2021. 3, 7, 13, 14, 15, 16, 17, 18, 19, 20, 21

[9] E. Boros and P.L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 2002. 6

[10] Rainer Burkard, Mauro Dell'Amico, and Silvano Martello. *Assignment Problems*. SIAM, 2009. 3

[11] Jiazhou Chen, Guoqiang Han, Aodan Xu, and Hongmin Cai. Identification of multidimensional regulatory modules through multi-graph matching with network constraints. *IEEE Transactions on Biomedical Engineering*, 67(4):987–998, 2019. 1

[12] Yuxin Chen and Emmanuel J Candès. The projected power method: An efficient algorithm for joint alignment from pairwise differences. *Communications on Pure and Applied Mathematics*, 71(8):1648–1714, 2018. 3, 7, 13, 14, 15, 16, 17, 18, 19, 20, 21

[13] Yuxin Chen, Leonidas J Guibas, and Qi-Xing Huang. Near-optimal joint object matching via convex relaxation. *arXiv preprint arXiv:1402.1473*, 2014. 3, 4

[14] Yves Crama and Frits CR Spieksma. Approximation algorithms for three-dimensional assignment problems with triangle inequalities. *European Journal of Operational Research*, 60(3):273–279, 1992. 2, 3

[15] Thomas A Feo and Mauricio GC Resende. Greedy randomized adaptive search procedures. *Journal of global optimization*, 6:109–133, 1995. 5

[16] Maolin Gao, Zorah Lahner, Johan Thunberg, Daniel Cremers, and Florian Bernard. Isometric multi-shape matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14183–14193, 2021. 1

[17] Stefan Haller, Lorenz Feineis, Lisa Hutschenreiter, Florian Bernard, Carsten Rother, Dagmar Kainmüller, Paul Swoboda, and Bogdan Savchynskyy. A comparative study of graph matching algorithms in computer vision. In *European Conference on Computer Vision*, pages 636–653. Springer, 2022. 1, 2, 3

[18] Tobias Heimann and Hans-Peter Meinzer. Statistical shape models for 3D medical image segmentation: A review. *Medical Image Analysis*, 2009. 1

[19] Lisa Hutschenreiter, Stefan Haller, Lorenz Feineis, Carsten Rother, Dagmar Kainmüller, and Bogdan Savchynskyy. Fusion moves for graph matching website, 2021. https://vislearn.github.io/libmpopt/iccv2021/. 7

[20] Lisa Hutschenreiter, Stefan Haller, Lorenz Feineis, Carsten Rother, Dagmar Kainmüller, and Bogdan Savchynskyy. Fusion moves for graph matching. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021. 3, 7, 8

[21] Zetian Jiang, Tianzhe Wang, and Junchi Yan. Unifying offline and online multi-graph matching via finding shortest paths on supergraph. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3648–3663, 2020. 3

[22] 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. 2

[23] Daniel Karapetyan and Gregory Gutin. Local search heuristics for the multidimensional assignment problem. *Journal of Heuristics*, 17:201–249, 2011. 7

[24] Itay Kezurer, Shahar Z Kovalsky, Ronen Basri, and Yaron Lipman. Tight relaxation of quadratic matching. In *Computer graphics forum*, pages 115–128. Wiley Online Library, 2015. 3

[25] Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 1955. 3

[26] Eugene L. Lawler. The quadratic assignment problem. *Management Science*, 1963. 3

[27] Yongbin Li, Siyu Chen, Weihong Liu, Di Zhao, Yimeng Gao, Shipeng Hu, Hanyu Liu, Yuanyuan Li, Lei Qu, and Xiao Liu. A full-body transcription factor expression atlas with completely resolved cell identities in c. elegans. *Nature Communications*, 15(1):358, 2024. 7

[28] Chang Liu, Chenfei Lou, Runzhong Wang, Alan Yuhan Xi, Li Shen, and Junchi Yan. Deep neural network fusion via graph matching with applications to model ensemble and federated learning. In *International Conference on Machine Learning*, pages 13857–13869. PMLR, 2022. 1

[29] 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*, 6(9):667–672, 2009. 1

[30] Eleonora Maset, Federica Arrigoni, and Andrea Fusiello. Practical and efficient multi-view matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4568–4576, 2017. 3

[31] Robert A Murphey, Panos M Pardalos, and Leonidas S Pitsoulis. A greedy randomized adaptive search procedure for the multitarget multisensor tracking problem. *Network design: Connectivity and facilities location*, 40:277–302, 1997. 7

[32] Weizhi Nie, Anan Liu, Yahui Hao, and Yuting Su. View-based 3d model retrieval via multi-graph matching. *Neural Processing Letters*, 48:1395–1404, 2018. 1

[33] Zhakshylyk Nurlanov, Frank R Schmidt, and Florian Bernard. Universe points representation learning for partial multi-graph matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1984–1992, 2023. 4

[34] Deepti Pachauri, Risi Kondor, and Vikas Singh. Solving the multi-way matching problem by permutation synchronization. In *Advances in Neural Information Processing Systems*, pages 1860–1868. Citeseer, 2013. 3, 4, 7, 13, 14, 15, 16, 17, 18, 19, 20, 21

[35] Panos M. Pardalos, Franz Rendl, and Henry Wolkowicz. The quadratic assignment problem - a survey and recent developments. *Quadratic Assignment and Related Problems*, 1993. 3

[36] Alexander J Robertson. A set of greedy randomized adaptive local search procedure (grasp) implementations for the multidimensional assignment problem. *Computational Optimization and Applications*, 19(2):145–164, 2001. 7

[37] 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. 6, 7

[38] Yusuf Sahillioğlu. Recent advances in shape correspondence. *The Visual Computer*, 36(8):1705–1721, 2020. 1

[39] Yanyao Shen, Qixing Huang, Nati Srebro, and Sujay Sanghavi. Normalized spectral map synchronization. *Advances in neural information processing systems*, 29, 2016. 3

[40] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012. 2

[41] Paul Swoboda, Dagmar Kainmüller, Ashkan Mokarian, Christian Theobalt, and Florian Bernard. A convex relaxation for multi-graph matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11156–11165, 2019. 3, 4, 7, 13, 14, 15, 16, 17, 18, 19, 20, 21

[42] Paul Swoboda, Bjoern Andres, Andrea Hornakova, Florian Bernard, Jannik Irmai, Paul Roetzer, Bogdan Savchynskyy, David Stein, and Ahmed Abbas. Structured prediction problem archive. *arXiv preprint arXiv:2202.03574*, 2022. 2, 7

[43] Da Tang and Tony Jebara. Initialization and coordinate optimization for multi-way matching. In *Artificial Intelligence and Statistics*, pages 1385–1393. PMLR, 2017. 2, 3, 5, 6

[44] Lorenzo Torresani, Vladimir Kolmogorov, and Carsten Rother. A dual decomposition approach to feature correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013. 2

[45] Roberto Tron, Xiaowei Zhou, Carlos Esteves, and Kostas Daniilidis. Fast multi-image matching via density-based clustering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4057–4066, 2017. 4

[46] Runzhong Wang, Ziao Guo, Wenzheng Pan, Jiale Ma, Yikai Zhang, Nan Yang, Qi Liu, Longxuan Wei, Hanxue Zhang, Chang Liu, Zetian Jiang, Xiaokang Yang, and Junchi Yan. Pygmtools: A python graph matching toolkit. *Journal of Machine Learning Research*, 25(33):1–7, 2024. 7

[47] Rohit Yadav, François-Xavier Dupé, Sylvain Takerkart, and Guillaume Auzias. Population-wise labeling of sulcal graphs using multi-graph matching. *Plos one*, 18(11):e0293886, 2023. 1

[48] Junchi Yan, Yu Tian, Hongyuan Zha, Xiaokang Yang, Ya Zhang, and Stephen M Chu. Joint optimization for consistent multiple graph matching. In *Proceedings of the IEEE international conference on computer vision*, pages 1649–1656, 2013. 2, 3, 6

[49] Junchi Yan, Minsu Cho, Hongyuan Zha, Xiaokang Yang, and Stephen M Chu. Multi-graph matching via affinity optimization with graduated consistency regularization. *IEEE transactions on pattern analysis and machine intelligence*, 38(6): 1228–1242, 2015. 3

[50] Junchi Yan, Jun Wang, Hongyuan Zha, Xiaokang Yang, and Stephen Chu. Consistency-driven alternating optimization for multigraph matching: A unified approach. *IEEE Transactions on Image Processing*, 24(3):994–1009, 2015. 2, 3, 6

[51] Junchi Yan, Zhe Ren, Hongyuan Zha, and Stephen Chu. A constrained clustering based approach for matching a collection of feature sets. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 3832–3837. IEEE, 2016. 2, 3, 6

[52] Xiaowei Zhou, Menglong Zhu, and Kostas Daniilidis. Multi-image matching via fast alternating minimization. In *Proceedings of the IEEE international conference on computer vision*, pages 4032–4040, 2015. 4