Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 29, no. 1, pp. 289–320 (2025)
DOI: 10.7155/jgaa.v29i1.2979

Graph Matching Based on Similarities in Structure and Attributes

Raphaël Candelier

Sorbonne Université, CNRS, Institut de Biologie Paris-Seine (IBPS), Laboratoire Jean Perrin (LJP), F-75005, Paris

Submitted: Sept. 2024 Accepted: Oct. 2025 Published: Nov. 2025

Article type: Regular paper Communicated by: Henning Meyerhenke

Abstract.
Finding vertex-to-vertex correspondences in real-world graphs is a challenging task with applications in various domains. Building upon classical iterative methods, we present the Graph Attributes and Structure Matching (GASM) algorithm, which provides high-quality solutions by integrating in a unified framework the information from graph connectivity and from all attributes of vertices and edges, and by using a minute noise to lift the degeneracies due to local symmetries. Contrarily to graph neural network methods, this approach allows to dynamically adjust the number of message passing iterations to each data sample. In addition, we introduce parameters quantifying the uncertainty of the attributes to tune how much the solutions should rely on the structure or on the attributes. We further show that even without attributes GASM consistently finds as-good-as or better solutions than state-of-the-art algorithms, with similar processing times.

1 Introduction

The importance of graph matching comes from the fact that a considerable amount of phenomena with very diverse nature can be represented with the same concept of graph (or networks), and the ability to find correspondences between their atomic elements – vertices and edges – has concrete application in several domains including computational biology [22, 13, 10], neuroscience [35, 33, 36], chemoinformatics [37, 49, 28], medical imaging [26], computer vision [5, 19, 43], machine learning [27, 50] and linguistics [31].

Simple local optimizers, like 2opt which swaps pairs of edges recursively [8], have been used for a long time to solve many graph-related problems including matching graph structures. As this problem is NP-complete [3], there is no known algorithm to obtain the optimal solution in less than

E-mail address: raphael.candelier@sorbonne-universite.fr (Raphaël Candelier)

exponential time and space and an abundant literature now aims at finding in a reasonable time either isomorphic relations for exact graph matching or approximated solutions for matching graphs in an error-tolerant way (see [3, 6, 14] for reviews). Actually, the graph matching problem can be recast as a special case of *quadratic assignment problem* (QAP) [45], like the famous traveling salesman problem and many other problems in combinatorial optimization and distributed resource allocation. Many approaches try to reduce the problem down to a *linear assignment problem* (LAP) [2, 52, 34], for instance by defining explicitly a score matrix representing the similarity between graph elements and applying a LAP solver like the popular Jonker-Volgenant algorithm [21], which operates in polynomial time [9]. Vogelstein *et al.* [48] have proposed a very efficient algorithm for approximated solutions, *Fast Approximate QAP* (FAQ), that first solves a relaxed, linearized version of the QAP and subsequently projects the solution back onto the permutation space. FAQ has then been implemented in Scipy and it is still considered today as a standard.

In addition, most real-world graphs have *attributes* attached to their edges or vertices, accounting for virtually any property with values that can be numerical or not [14]. For instance, in the context of protein-protein interaction networks the vertices are characterized by a unique protein identifier while the edges may bear multiple association weights for different quantification of the interactions [44]. Matching the graph structure (*i.e.* just the connection backbone) can be done independently of the attributes, but the information bore in attributes can significantly improve both the solutions and the searching time; for instance Dickinson *et al.* [11] proved that for graphs possessing unique vertex labels the computational complexity is only quadratic in the number of nodes.

Many recent methods have been able to exploit the rich information inherent to graphs' structure and attributes, including *graph kernels* techniques [25] adapted to the graph matching problem [17, 42]. Even more recently, deep learning methods specialized in graph data called *graph neural networks* (GNN) have emerged and been successfully applied to many problems across all areas of science (see [7, 23] for reviews). Both approaches rely on an iterative message-passing core, where each vertex collects messages in the form of vector representations from its neighboring vertices, aggregates them into a single vector, and uses this vector to update its representation. Upon iteration, message-passing enables each vertex to build representations capturing larger patterns from their surrounding graph.

Here, we propose a new algorithm termed *Graph Attributes and Structure Matching* (GASM) that is not based on graph kernels or GNN but uses very similar concepts in a lightweight framework. GASM has been largely inspired by iterative methods and especially a series of three articles: Kleinberg's HITS algorithm [24] that projects any graph on the so-called "hubs and authorities" graph, the generalization by Blondel *et al.* [2] that adapts the same idea to calculate iteratively vertex similarity scores and match any pair of graphs, and finally the work of Zager and Verghese [52] who elegantly introduced an edge score matrix to fix convergence issues and the dependency on initial conditions. Interestingly, this latter approach is similar to the representation update of both vertices and edges used in directional message passing for GNN [51]. Zager and Verghese also tried to incorporate the handling of a categorical vertex attribute, but the resulting algorithm is not robust. The main contribution of GASM with respect to this trilogy is that constraints related to attributes are introduced *a priori*, which creates a coupling between the attributes and the structure during iterations and provides highly accurate matchings. An infinitesimal noise is also introduced to lift the degeneracies due to local symmetries and further improve the general quality of the solutions, a strategy that has also been employed for GNN [1, 39].

There are several benefits from using non-GNN methods for graph matching. There is no training set and no bias associated with training, a better interpretability of the process, and one

avoids poor performances in the low data regime which is particularly relevant in domains where datasets are scarce and where large GNNs can overfit easily. In this matter, pretraining graph-structured data has not been successful [29], likely due to limited transposability between graphs of different nature. Another limitation of GNN is oversmoothing: individual vertex features become nearly identical as the number of message-passing layers increases because each layer behaves as a graph-smoothing operator [38]. This phenomenon hinders the processing of very deep networks, and GNNs typically struggle to model large graph patterns, despite these being crucial in several applications [12]. The "message-passing" algorithm of iterative methods (Blondel-Zager) is not subject to oversmoothing as scores span over several orders of magnitude to account for all sizes of structural patterns. More generally, there is a profound methodological discrepancy in GNN models as the number of message-passing layers/iterations is fixed *a priori* while the relevant message-passing distance is the graph diameter, which is different for each graph. We therefore employ here a dynamical convergence criterion, intending to compute just the necessary amount of message-passing steps. Finally, it is particularly challenging to incorporate uncertainty in the GNN models and to understand how it propagates through layers and passed messages, despite some efforts have been made in this direction [41, 20]. Our algorithm exploits the knowledge of uncertainties for each attribute to directly tune how much the solutions should rely on the structure or on the attributes, and ultimately improve the matching result.

This paper is organized as follows. In Section 2 some general ideas, definitions, and notations are introduced, while Section 3 presents the Zager-Verghese (ZV) and GASM algorithms. Results are presented in Section 4, with a comparison with ZV (4.1) and benchmarks on isomorphic matching (4.2), QAPLIB (4.3), graph alteration (4.4) and speed (4.5). The paper finishes in Section 5 with a conclusion.

2 Preamble: Definitions, Notations and Matching Ranking

Let us consider the comparison of two graphs G_A and G_B, which may be both directed or both undirected, but that are not multigraphs. We index variables and matrices of the corresponding graphs with A and B, and with $*$ as a replacement symbol for quantities that are defined similarly in both graphs; for instance the number of vertices is n_*, meaning that it is n_A for G_A and n_B for G_B. Let us also note the number of edges m_*, among which there are μ_* self-loops, and the adjacency matrix Λ_*. Variables are also used without index when they are equal for both graphs: for instance, we may refer to the number of vertices n when $n = n_A = n_B$.

2.1 Attributes

The graphs considered here can have any number of vertex or edge attributes. Let us separate the *measurable* attributes from the non-measurable, or *categorical* ones. There are many cases where this distinction is obvious: in the case of neural networks for instance [32], the activation functions belong to a set of functions of different families and can thus be considered as a categorical attribute of vertices, which means it can only be compared for exact correspondence or not. In contrast, the edge weights and vertex biases have values in $\mathbb{R}$ and a distance can be defined, so they are measurable attributes; indeed, a weight of 0.1 is strictly different from 0 and 1, but in a matching context it is natural to consider that it is better matched with the former.

However, an attribute with numerical values does not automatically belong to the measurable class, since it can represent indexes or numeric identifiers. The measurability of any attribute is specific to the graphs of interest and has to be determined accordingly by the end user.

2.2 Accuracy

As one usually wants to associate the vertices, if $n_A \geq n_B$ there are $\frac{n_A!}{(n_A-n_B)!}$ possible association sets, a potentially prodigious amount among which a few may be meaningful matchings while the vast majority is composed of non-sensical matchups. A key factor for finding good matchings and benchmarking algorithms is the ability to compare candidate solutions.

A standard measure is the *accuracy* of a matching $\mathcal{M}$, defined as the proportion of vertices pairs corresponding to the ground truth and noted $\gamma(\mathcal{M})$. Throughout the rest of this article we will only use the mean accuracy, *i.e.* the value averaged over several matchings where the indices of at least one of the graphs are randomly shuffled, and refer to it as the accuracy γ.

Notably, the accuracy can go up to 1 for some pairs of isomorphic graphs, but not always, and when the graphs are non-isomorphic this is generally not true. In some cases, the maximum possible accuracy – *i.e.* the maximum average accuracy computed over the set of all possible matching algorithms – can be computed independently of the algorithm (see Section 4.2), and for graphs with many local symmetries it can be arbitrarily low. So this is a delicate quantity to manipulate as in the general case the maximum possible average value is unknown, and the values have no absolute meaning; one can only compare different accuracies relative to each other, without knowing up to what point these solutions can be further improved. There are also very counterintuitive cases, like pairs of graphs for which any matching algorithm returns solutions with the same average accuracy – see for instance the case of circular ladders in Section 4.2.

2.3 Matching quality

But probably the most obvious limitation of accuracy is that it necessitates to have the ground truth, which is by definition unknown in all real-world applications. Still, comparing the local properties of all the matched pairs to compute global quantities can always be done, and we shall refer to these global quantities as the matching *qualities*. Several definitions of qualities can be derived, depending on the assessed local property.

Let us define the *structural quality* q_S as a measure of the local structural similarity between pairs of vertices. For a given matching $\mathcal{M}$, let M be the binary matrix of size $n_A \times n_B$ such that:

$$[M]_{ij} = \begin{cases} 1, & \text{if node } i \text{ in } G_A \text{ is matched with node } j \text{ in } G_B \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Let us define:

$$Z = \Lambda_A M - M \Lambda_B \tag{2}$$

The structural quality q_S of the matching then reads:

$$q_S = \begin{cases} 0 & \text{if } m_A = m_B = 0, \text{ otherwise:} \\ 1 - \frac{tr\left(Z^\top Z\right)}{m_A + m_B} & \text{for directed graphs} \\ 1 - \frac{tr\left(Z^\top Z\right)}{2(m_A+m_B)-\mu_A-\mu_B} & \text{for undirected graphs} \end{cases} \tag{3}$$

The idea behind this definition is to count all the edge mismatches, defined as edges whose terminating vertices have matchups that are not themselves connected with an edge in the other

graph. The intermediary matrix Z has elements z_{ij} set to 0 when the pair of vertices (i, j), respectively from G_A and G_B, both have or both don't have a neighbor associated with the other vertex, and ± 1 otherwise. The trace of the product $Z^\top Z$ is a way to compute the grandsum of the squares of each element of Z, which is the number of discrepancies contained in Z. q_S is then a scalar bounded in $[0, 1]$, a higher value indicating a better overall matching of the local structure.

Of course, it would be impractical to compute all the possible matchings and sort them by their structural quality q_S, so this cannot constitute the core of a brute force matching algorithm. Yet, it is a useful quantity as it allows the raising of certain types of degeneracy introduced by score matrices, which is a pivotal element for any LAP-based approach. A very simple illustration is given in Figure 1, where a simple linear graph is matched with itself ($G_A = G_B$); let us call the vertices 1 and 4 of this example the *side* vertices while vertices 2 and 3 are the *inner* ones. Any algorithm exploiting the graph structure would give a score matrix with the form displayed in Figure 1-b: as the side vertices are indistinguishable and the inner vertices as well, there can be only 3 different values in the score matrix: a standing for the side-side scores, b for side-inner scores and c for inner-inner scores. In addition, $a + c$ should be greater than $2b$ if the local structural similarity is favored in the score determination. The point is that the score matrix cannot handle 4-point interactions and in this example there are 4 solutions with equal maximal scores $s = 2(a + c)$ (Figure 1-c), among which only 2 have a perfect structural quality q_S. So, without further processing, the user has a $1/2$ probability of ending up with a structurally unsound solution, because of the limitations of the score matrix and more generally that graph matching is treated as an LAP and not a QAP. We will see later in Section 4.1 how GASM can circumvent this limitation of LAP-based approaches.

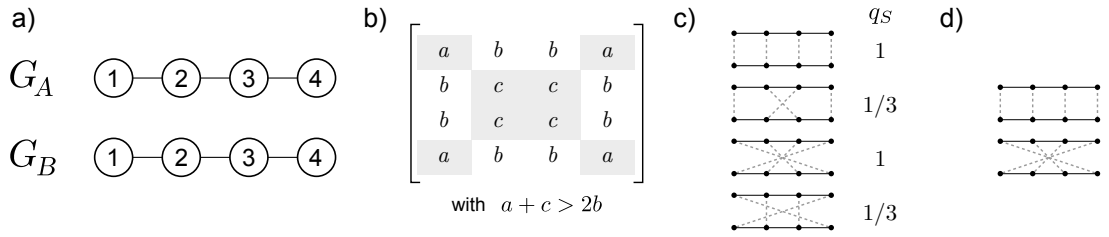

Figure 1: Example of matching degeneracy introduced by the score matrix. **a)** The graphs to match. **b)** Form of the score matrix returned by any algorithm exploiting the graph structure. The best matching solutions are composed of the grayed cells exclusively. **c)** The 4 matchings with a maximum total score of $2(a+c)$, along with their structural quality q_S. **d)** The only 2 matchings respecting structural correspondence ($q_S = 1$).

Similarly, it is possible to define a matching quality for any attribute, but we will not use it here so let us jump directly to the description of the GASM algorithm.

3 Algorithm

3.1 Zager and Verghese's algorithm (ZV)

GASM has been greatly inspired by the work of Zager and Verghese [52], so let us remind here briefly their matching algorithm for directed graph: pairs of vertices and edges from G_A and G_B

have scores that are iteratively updated, and converged score matrices are processed by a LAP solver.

At iteration k, let $x_{uv}(k)$ denote the vertex similarity score between vertex u in G_A and vertex v in G_B, and $y_{ij}(k)$ the edge similarity score between edge i in G_A and edge j in G_B. Initially, all the scores $x_{uv}(1)$ and $y_{ij}(1)$ are set to 1. Zager and Verghese have shown that both simultaneous and sequential updates are possible; in the case of sequential updates, the update equations are, for $k > 1$:

$$y_{ij}(k) = \frac{\hat{y}_{ij}}{\sum\limits_{k,l} \hat{y}_{kl}^2} \quad \text{with} \quad \hat{y}_{ij} = x_{s(i)s(j)}(k-1) + x_{t(i)t(j)}(k-1) \tag{4}$$

$$x_{uv}(k) = \frac{\hat{x}_{uv}}{\sum\limits_{p,q} \hat{x}_{pq}^2} \quad \text{with} \quad \hat{x}_{uv} = \sum_{\substack{s(i)=u \\ s(j)=v}} y_{ij}(k) + \sum_{\substack{t(i)=u \\ t(j)=v}} y_{ij}(k) \tag{5}$$

where, for any edge i, $s(i)$ and $t(i)$ are its source and target vertices, respectively.

The idea behind these equations is to set the score of a pair of edges as the sum of the scores of their pairs of vertices, and vice-versa. It follows that highly connected vertices have higher scores than poorly connected ones, and the maximal scores are bound to the pairs of major hubs. The normalization resets the values back to a reasonable range at each iteration, while keeping the contrast between high and low scores. At each iteration, the scores integrate information from one vertex further in the graph and spread over several orders of magnitude to account for all the details of the surrounding structure. This ranking simplifies the work of the subsequent LAP solver, which can essentially make pairs by recursively taking the highest score.

Zager and Verghese have further shown that these equations can be rewritten in a compact matrix form, as we will see in the sequel, and that convergence is ensured with arbitrary initial conditions. They also tried to address the case of "nodes with type labels" (termed here *vertex categorical attributes*) by modifying the converged vertex score matrix before using the LAP solver. Unfortunately, one can easily find cases where this approach fails – see Section 4.1.3 for instance.

Though the ZV algorithm provides very decent matchings out of the box, it has some flaws and can be optimized in a number of ways to provide better matchings. While the details of our implementation are presented in the next paragraphs, our main improvements can be summarized as follows:

- adding minute noise to lift the degeneracies caused by local symmetries
- defining a convergence criterion, to make a minimal amount of iterations while ensuring convergence
- properly incorporating the information of all graph attributes, along with parameters tuning each attribute's reliability.

We then analyze in a few case studies how the modifications in GASM improve over some of ZV limitations in Section 4.1, and include ZV in the subsequent benchmarks to provide a quantified comparison.

3.2 Attribute matrices

Let us start by covering how GASM properly handles graph attributes. We consider the general case where vertices and edges of G_A and G_B can have several attributes, some being measurable

and some categorical. However, if one vertex (resp. edge) has a value for a given attribute all the other vertices (resp. edges) of both graphs should also have a value for this attribute.

Let us consider an attribute $\mathscr{A}: \theta \mapsto a(\theta)$ where θ can be a vertex or an edge without loss of generality. If $\mathscr{A}$ is categorical, the comparison of two elements θ in G_A and θ' in G_B can only have a binary outcome (similar or dissimilar) and is naturally represented by a boolean *attribute similarity matrix* $\mathcal{A}$ of size $n_A \times n_B$ (vertex attribute) or $m_A \times m_B$ (edge attribute) defined by:

$$[\mathcal{A}]_{\theta\theta'} = \delta_{a(\theta)a(\theta')} = \begin{cases} 1, & \text{if } a(\theta) = a(\theta') \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where δ_{ij} is the Kronecker symbol.

However, in real-world applications, there may be measurement uncertainties on some attribute values. For each attribute $\mathscr{A}$ let us introduce a positive scalar ρ estimating the uncertainty over its values, a parameter that should be defined by the end user. If there is no stochasticity or other source of uncertainty in the process providing $a(\theta)$ and $a(\theta')$, *i.e.* in the limit where $\rho \to 0$, $\mathcal{A}$ can be directly computed with eq.(6). In the limit where $\rho \to \infty$, there is no way to decipher which attribute values are correct and all the values in $\mathcal{A}$ should be equal and non-zero. So, in the general case let us consider the following definition:

$$[\mathcal{A}]_{\theta\theta'} = \begin{cases} 1, & \text{if } a(\theta) = a(\theta') \\ e^{-\frac{1}{2\rho^2}}, & \text{otherwise} \end{cases} \tag{7}$$

where the uncertainty parameter ρ tunes the contrast in $\mathcal{A}$.

Now, if the attribute $\mathscr{A}$ is measurable the comparison between $a(\theta)$ and $a(\theta')$ can be a real scalar. To combine several distance matrices altogether it is preferable to keep them bound in $[0, 1]$, with 0 indicating dissemblance and 1 indicating similarity. Let us define the corresponding attribute distance matrix by:

$$[\mathcal{A}]_{\theta\theta'} = e^{-\frac{[a(\theta)-a(\theta')]^2}{2\rho^2}} \tag{8}$$

In the limit where $\rho \to 0$, $\mathscr{A}$ become similar to a categorical attribute with as many categories as values and $\mathcal{A}$ can be computed using (6) as well. Otherwise, it is important to keep ρ as close as possible to the real uncertainty to ensure the accuracy of the final solutions does not drop artificially. In case ρ cannot be estimated, a safe replacement is the standard deviation σ_a of the distances $a(\theta) - a(\theta')$ for measurable attributes and of $\delta_{a(\theta)a(\theta')}$ for categorical ones, computed over all possible pairs (θ, θ'). σ_a is indeed a higher bound for the estimation of uncertainty.

Equation (8), and by extension eq.(7), assumes that the uncertainty is akin to adding a Gaussian noise with standard deviation ρ. This may not be true in the general case, and other decreasing forms are possible depending on the details of each application.

Then, if there are ζ vertex attributes with associated distance matrices $(\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_\zeta)$ and ξ edge attributes with distance matrices $(\bar{\mathcal{A}}_1, \bar{\mathcal{A}}_2, ..., \bar{\mathcal{A}}_\xi)$, the global *vertex distance matrix* V and *edge distance matrix* E are defined as:

$$V = [\nu_{uv}] = J_n \odot \mathcal{A}_1 \odot \mathcal{A}_2 \odot ... \odot \mathcal{A}_\zeta \tag{9}$$

$$E = [\epsilon_{ij}] = J_m \odot \bar{\mathcal{A}}_1 \odot \bar{\mathcal{A}}_2 \odot ... \odot \bar{\mathcal{A}}_\xi \tag{10}$$

where J_n and J_m are unit matrices of respective sizes $n_A \times n_B$ and $m_A \times m_B$, $\odot$ stands for the Hadamard product and (u, v) and (i, j) are pairs of vertices and edges from G_A and G_B. As all elements of all distance matrices stand in the interval $[0, 1]$, all the ν_{uv} and ϵ_{ij} are also bounded in $[0, 1]$.

3.3 Scores

The matrices V and E are akin to score matrices that can readily be used to match either the vertices or the edges, without taking into account any structural information on the graphs. For instance, if at least one of the graphs has no edge, a matching based on vertices attributes can be obtained by simply feeding V in a LAP solver and searching for a maximum score matching.

However, it is generally desirable to account for the similarities in both the structure and the attributes. Several algorithms have been designed to output a score matrix based on structural similarities, again to be supplied to an LAP solver. GASM integrates both information by using the vertex and edge distance matrices as initial conditions for an iterative procedure inspired by ZV [52].

In the sequel, we separate the cases of undirected and directed graphs. Since undirected graphs can easily be converted to directed graphs without loss of information it may seem sufficient to cover only the directed case. However, directed versions of undirected graphs are a peculiar subset of directed graphs for which a slightly different formalism can be applied and that has specific properties. We will see in Section 4 that there are significant differences in matching accuracy and structural qualities, at least for all the algorithms considered here. In addition, for several algorithms including GASM, using directed versions of undirected graphs requires twice the number of operations without any gain in return, so there is also a performance boost in separating the cases.

3.4 Iterative procedure for undirected graphs

Let us first cover the case where both G_A and G_B are undirected graphs. For any vertex v, let $C_*(v)$ be the set of edges that are connected to v, and for any edge i let $D_*(i)$ be the set of vertices it connects. $D_*(i)$ contains at most 2 vertex indices and only one if i is a self-loop.

As an initial step, the vertex scores are defined as:

$$x_{uv}(1) = (\nu_{uv} + h_{uv}) \sum_{\substack{i \in C_A(u) \\ j \in C_B(v)}} \epsilon_{ij} \tag{11}$$

where h_{uv} are random values drawn from the continuous uniform distribution between 0 and a parameter $\eta \ll 1$. The role of this minute positive "noise" term is to help lift degeneracies due to local symmetries and is discussed in detail with the examples in Section 4.1.1. In short, the noise initially favors at random some pairs of vertices, which translates during the iteration procedure in slightly favoring some structurally symmetric sub-patterns, thus lifting the degeneracy.

Then, for each iteration step $k > 1$ the update equations are:

$$y_{ij}(k) = \frac{1}{f_y} \sum_{\substack{u \in D_A(i) \\ v \in D_B(j)}} x_{uv}(k-1) \tag{12}$$

$$x_{uv}(k) = \frac{1}{f_x} \sum_{\substack{i \in C_A(u) \\ j \in C_B(v)}} y_{ij}(k) \tag{13}$$

where f_x and f_y are normalization coefficients. These coefficients can be set to any positive finite value at any iteration without altering the outcome of the whole algorithm since only the relative values of the scores are important for the LAP. To simplify the formulas we set $f_x = f_y = 1$ in the sequel; however, it is worth noting that during numerical computation some normalization may be used to avoid floating point overflow. This is discussed in more detail in Section S1.2, where an approximated normalization factor is introduced for this practical purpose.

Let us now express these equations in a concise form using only elementwise matrix operations and matrix multiplication. Let the *unoriented incidence matrix* R_* be:

$$[R_*]_{ui} = \begin{cases} 1, & \text{if } i \in C_*(u) \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

Each column of R_* stands for an edge and has exactly two non-zero elements, except for self-loops which have a unique non-zero element. The initialization and update equations can then be written as:

$$X_1 = (V + H) \odot (R_A E R_B^\top) \tag{15}$$

$$Y_k = R_A^\top X_{k-1} R_B \tag{16}$$

$$X_k = R_A Y_k R_B^\top \tag{17}$$

where H is the matrix composed of the noise terms h_{uv}.

To reduce computation time, two orthogonal strategies can be employed: parallelization on GPU and using graph complements. Both a CPU version based on eq.(15-17) and a GPU version using eq.(11-13) have been implemented, and GPU provides in general the best speed (see Section 4.5).

To limit the number of operations with highly connected graphs, one can also exploit the fact that in modern linear algebra libraries like Numpy the matrix multiplication is faster as matrices are sparser. When the graphs are dense enough it is thus interesting to use their complements, and we define the *complement incidence matrix* $\bar{R}_*$ as the unoriented incidence matrix of the complement graph $\bar{G}_*$. However, the complements cannot be used for one graph and not the other, so the switching criterion has to be globally defined based on the densities of both graphs. Let the *incidence matrix* $\tilde{R}_*$ be:

$$\tilde{R}_* = \begin{cases} R_*, & \text{if } 4(m_A + m_B) \leq n_A(n_A+1) + n_B(n_B+1) \\ \bar{R}_*, & \text{otherwise} \end{cases} \tag{18}$$

The update equations can then be rewritten:

$$Y_k = \tilde{R}_A^\top X_{k-1} \tilde{R}_B \tag{19}$$

$$X_k = \tilde{R}_A Y_k \tilde{R}_B^\top \tag{20}$$

Note that edge attributes cannot be preserved with graph complements, which might look like a severe incompatibility with the present algorithm as it is precisely designed to account for all graph attributes. However, only the initialization equation (15) uses the edge distance matrix E, which contains all the information about the similarities of edge attributes. Interestingly, as long as this information is injected during the initialization step it propagates as well in the complements.

3.5 Iterative procedure for directed graphs

Let us now cover the case where both G_A and G_B are directed. Consistently with eq.(4-5), for any edge i, let $s(i)$ and $t(i)$ be its source and target vertices, respectively. As previously, at iteration k the vertex similarity score is $x_{uv}(k)$ and the edge similarity score is $y_{ij}(k)$. For the initial step, the vertex scores are defined as:

$$x_{uv}(1) = (\nu_{uv} + h_{uv}) \left(\sum_{\substack{s(i)=u \\ s(j)=v}} \epsilon_{ij} + \sum_{\substack{t(i)=u \\ t(j)=v}} \epsilon_{ij} \right) \tag{21}$$

And the update equations are, for $k > 1$:

$$y_{ij}(k) = \frac{1}{f_y} \left(x_{s(i)s(j)}(k-1) + x_{t(i)t(j)}(k-1) \right) \tag{22}$$

$$x_{uv}(k) = \frac{1}{f_x} \left(\sum_{\substack{s(i)=u \\ s(j)=v}} y_{ij}(k) + \sum_{\substack{t(i)=u \\ t(j)=v}} y_{ij}(k) \right) \tag{23}$$

where f_x and f_y are the normalization coefficients, set at 1 for further equations as previously. Note that in eq.(23) there is a difference as compared to the directed case and eq.(13) in the handling of self-loops: they are counted twice – once as a source and once as a target – while they are counted only once in the undirected case.

As introduced in [52], it is convenient to represent the adjacency structure of the graphs by pairs of matrices termed the *source-edge matrix* S_* and *terminus-edge matrix* T_*, which are akin to the incidence matrix R_* in the undirected case and defined as follows:

$$[S_*]_{ui} = \begin{cases} 1, & \text{if } s(i) = u \\ 0, & \text{otherwise} \end{cases} \tag{24}$$

$$[T_*]_{ui} = \begin{cases} 1, & \text{if } t(i) = u \\ 0, & \text{otherwise} \end{cases} \tag{25}$$

The adjacency matrix can then be recovered with $\Lambda_* = S_* T_*^\top$, and the incidence matrix of the corresponding undirected graph is simply $R_* = S_* \vee T_*$. As for undirected graphs, the graph complements may be used for the update equations, and we define:

$$(\tilde{S}_*, \tilde{T}_*) = \begin{cases} (S_*, T_*), & \text{if } 2(m_A + m_B) \leq n_A^2 + n_B^2 \\ (\bar{S}_*, \bar{T}_*), & \text{otherwise} \end{cases} \tag{26}$$

The scores initialization and update equations are then:

$$X_1 = (V + H) \odot (S_A E S_B^\top + T_A E T_B^\top) \tag{27}$$

$$Y_k = \tilde{S}_A^\top X_{k-1} \tilde{S}_B + \tilde{T}_A^\top X_{k-1} \tilde{T}_B \tag{28}$$

$$X_k = \tilde{S}_A Y_k \tilde{S}_B^\top + \tilde{T}_A Y_k \tilde{T}_B^\top \tag{29}$$

3.6 Convergence criterion

Convergence has been extensively discussed in Blondel *et al.* [2] and Zager and Verghese [52], and the demonstration of convergence for GASM is the same as for ZV. However, an important aspect that has not been investigated so far is the number of iterations before convergence. Since in general the iteration time strongly depends on the number of edges in the graphs, avoiding unnecessary iterations is critical for large and dense graphs.

Let us propose an estimated convergence criterion based on *ad hoc* properties of the graphs. Considering that each iteration propagates the structural and attributes information one vertex further, the minimal number of iterations before every pair of vertices receives some information from all the other pairs of vertices is:

$$\tilde{k} = min(\Delta_A, \Delta_B) \tag{30}$$

where Δ_* is the diameter of G_*, *i.e.* the maximum eccentricity among all vertices. For directed graphs, a safer definition is to use the diameter of the undirected versions of the graphs, but it seems that this is not necessary in practice, so this is not what is used here.

Interestingly, as the graph diameter decreases when the density of edges increases (*e.g.* Supp. Fig. S2), the dense graphs – which have a higher iteration time – benefit from a faster convergence and a reduced number of iterations. Convergence is achieved in just a few iterations with small-world graphs: the isomorphic matching of an Erdös-Rényi (ER) G_{np} graph with 100 vertices and no attributes requires less than $\tilde{k} = 4$ iterations, and it is reduced down to $\tilde{k} = 2$ when the average degree is above 3 (Supp. Fig. S3).

3.7 Matching

The final step of score determination is to handle isolated vertices, *i.e.* vertices disconnected from the rest of the graph. These vertices may have attributes to be matched on, but their scores in X_k are all set to zeros at each iteration. So, in order to take them into account in the matching, we restore their scores to their initial values in V divided by the appropriate normalization factor:

$$\forall u, v : x_{u,v}(\tilde{k}) \leftarrow \nu_{u,v} / f_x^{\tilde{k}-1} \quad \text{if } u \text{ is isolated or } v \text{ is isolated} \tag{31}$$

Finally, the matching is performed on the vertex scores matrix $X_{\tilde{k}}$ with a standard LAP algorithm searching for a maximum global score. It can also be performed on the edge score matrix $Y_{\tilde{k}}$, in case it is more relevant.

4 Results

4.1 Comparison with ZV

ZV computes very decent matchings in general, but there are still several situations where it allows too many matching solutions, some of them being of poor quality. In the next paragraphs, we focus on three typical cases where ZV generates too many solutions (local symmetries, non-propagating attribute information and attribute incoherence) and explain how GASM's algorithmic improvements are able to optimize the three cases.

4.1.1 Local symmetries

Many graphs have local symmetries, *i.e.* similar subgraphs attached to the rest of the graph with the same anchoring points. Local symmetries can take the form of branches, cycles, or more complex patterns. Unfortunately, these symmetries create intrinsic matching indetermination as vertices at the same relative position in the symmetric sub-patterns have the same exact surrounding structure, so with a structure-based scoring like ZV their pairs have the same scores and the LAP solver has no way to determine which vertex belongs to which sub-pattern. This situation is reminiscent of the score matrix limitation discussed in Section 2.3 and exemplified in Figure 1.

One way to circumvent the problem could be to compute all the possible solutions of the LAP and rank them by structural quality for instance. Finding all the solutions of a LAP is P-complete [47] and some algorithms are available for this task [18, 46], but they represent a consequent computational overhead and would make the matching of large ($n_* > 100$) graphs intractable in practice.

We have thus chosen a different approach, explained here with an example for the sake of clarity. Figure 2-a depicts a basic graph with two symmetric branches and for which self-matching with ZV produces several pairs of vertices with similar scores (Figure 2-b), leading to 4 possible solutions, 2 being structurally unsound.

Interestingly, the addition of a minute random noise to the initial scores – lying in the h_{uv} term in equations (11) and (21) – allows filtering out the solutions with low structural quality (Figure 2-d,e). It may seem counterintuitive that adding some noise to the inputs can actually improve the outcome of a deterministic algorithm, so let us clarify how this works: first, the noise used in practice is so small ($\eta = 10^{-10}$) that it does not mess up with the general score determination and, for graphs without attribute, the integer part of the GASM score matrices is similar to the one obtained with ZV. This is also the case in Figure 2-b,d even though the noise is much larger to ease visualization. Second, the noise cannot improve the matching accuracy: in the example of Figure 2, the average accuracy of the solutions is $\gamma = 0.6$ for both algorithms. However, it allows filtering out the solutions with low structural quality wherever there is a local symmetry. The mechanism at play is the following: initially the noise favors at random some pairs of vertices among the pool that would be otherwise degenerated. After some iterations corresponding to the symmetric sub-pattern size, the initial conditions have propagated, and it is now pairs of whole sub-patterns that are slightly favored. After convergence, all the degeneracies have been raised and there is only one solution given by the score matrix. However, different solutions can still emerge from run to run, depending on the initial noise.

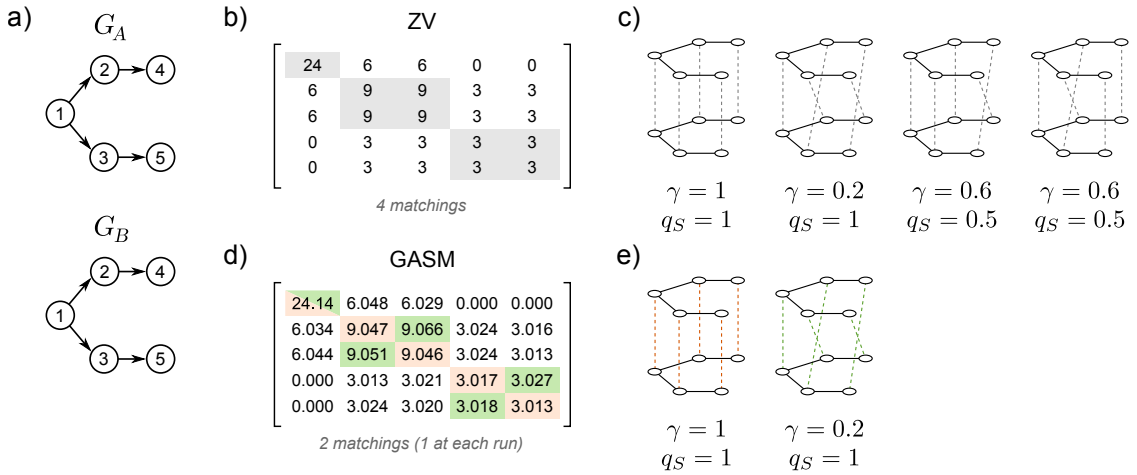

Figure 2: Managing symmetry with a minute noise. **a)** Self-matching ($G_A = G_B$) of a simple directed graph with a symmetry. **b)** Score matrix $X_{\tilde{k}=2}$ given by ZV [52], without normalization. The matching solutions comprise only the grayed cells. **c)** The 4 corresponding matchings with the best score (48), along with their accuracy γ and structural quality q_S. **d)** Example of score matrix $X_{\tilde{k}=2}$ produced by the GASM algorithm, without normalization and with a relatively large noise $\eta = 10^{-2}$ to ease visualization. Here the best matching solution lies in the green cells, but other initial random numbers could favor the orange cells. **e)** The 2 corresponding matching solutions.

4.1.2 Propagating attribute information

Attributes provide some information that is interesting to exploit. For categorical attributes, Zager and Verghese proposed to multiply term-wised the converged, purely structural vertex score matrix with a distance matrix made of -1 and +1 to adjust the score matrix, as shown in the example of Figure 3-b. Though this indeed raises the degeneracy for the concerned vertices, this approach suffers from the fact that neighboring vertices do not benefit from this information and can still be mismatched, as exemplified by the second solution in Figure 3-c.

GASM introduces the attribute information in the initial score matrix X_1 *via* the V and E matrices in eq.(15) and (27), so before the iterative procedure. This creates a coupling between the structure and the attributes during iterations which lets the scores be determined not only by the similarity of the local structure and the vertices/edges proper attributes but also by the similarity of the attributes of neighboring vertices and edges. In the example of Figure 3, the solution where vertices 4 and 5 are mixed up is filtered out by GASM, which increases both the average accuracy and structural quality of the matching. Actually, the mechanism of information propagation is similar to what has been described with noise in Section 4.1.1, except that the initial differences are based on the attributes and thus do not change from run to run.

4.1.3 Attribute incoherence

Finally, outside the case of isomorphic matching, there is by definition some incoherence between the graphs to compare. Though different structures are well-managed by both ZV and GASM, incoherent attributes can make ZV go really wrong, as exemplified in Figure 4: the vertices labelled

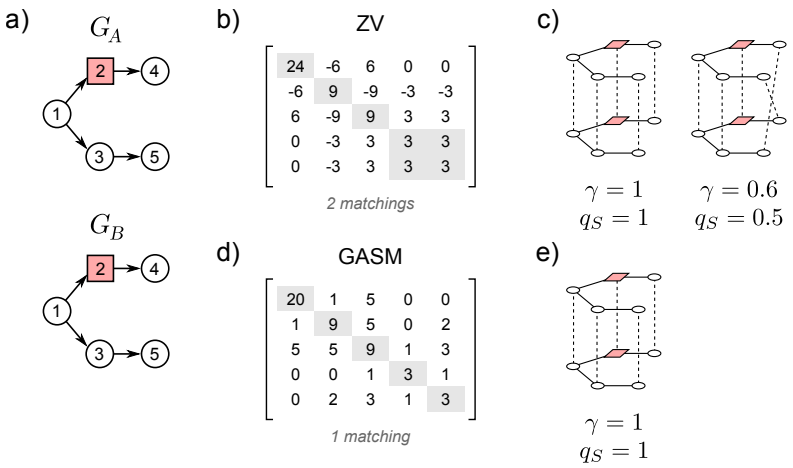

Figure 3: Propagation of attribute information through branches. **a)** Self-matching ($G_A = G_B$) of a simple directed branched graph with a categorical attribute on vertices. One vertex has a different value than the others, symbolized by a red square. **b)** Score matrix $X_{\tilde{k}=2}$ given by ZV, without normalization. The matching solutions comprise only the grayed cells. **c)** The 2 corresponding matchings solutions with the best score (48), along with their accuracy γ and structural quality q_S. **d)** Integer part of the score matrix $X_{\tilde{k}=2}$ produced by the GASM algorithm, without normalization. The decimal part, due to the artificial noise, is neglectable for the matching and is skipped to ease visualization. **e)** The corresponding matching solution.

2 have different categories in G_A and G_B and the $(3 \to 5)$ branch of G_B could be equally matched with the $(2 \to 4)$ and $(3 \to 5)$ branches of G_A. In this example, ZV leads to a large set of equally scored solutions (Figure 4-c) which all seem unacceptable: vertex 2 of G_B is always matched with either vertex 4 or vertex 5 in G_A, which goes against the structural similarity of the graphs.

As shown in Figure 4-d,e, GASM finds all and only the structurally sound solutions. Attribute incoherence modify the initial score matrix X_1 by lowering the scores of the corresponding pairs in a symmetric way that does not affect the building up of scores based on the structural information. Such inconsistencies leave a trace in the final score matrix – see for instance the differences in the integer parts of the scores in Figure 2-d and Figure 4-d – but it does not affect the emergence of the solutions based on structural cues. The noise plays the same determinant role as previously to filter out the matchings with low structural quality.

4.2 Isomorphic matching

Let us now delve deeper into the comparison of GASM with other algorithms and the quantification of performance. For this, we benchmarked 2opt, FAQ, ZV, and GASM on both the average accuracy and structural quality over the same sets of graphs. Let us start with isomorphic matching, *i.e.* the matching of two isomorphic graphs; in practice, Λ_B is a shuffled version of Λ_A. The results for 4 types of undirected graphs are compiled in Figure 5.

Balanced binary trees (Figure 5-a) are a good example of graphs having multiple local symmetries and for which the maximum possible average accuracy γ_{BT} can be determined analytically. Indeed, if there are r vertices on a row they all have a $1/r$ probability to be correctly assigned, so

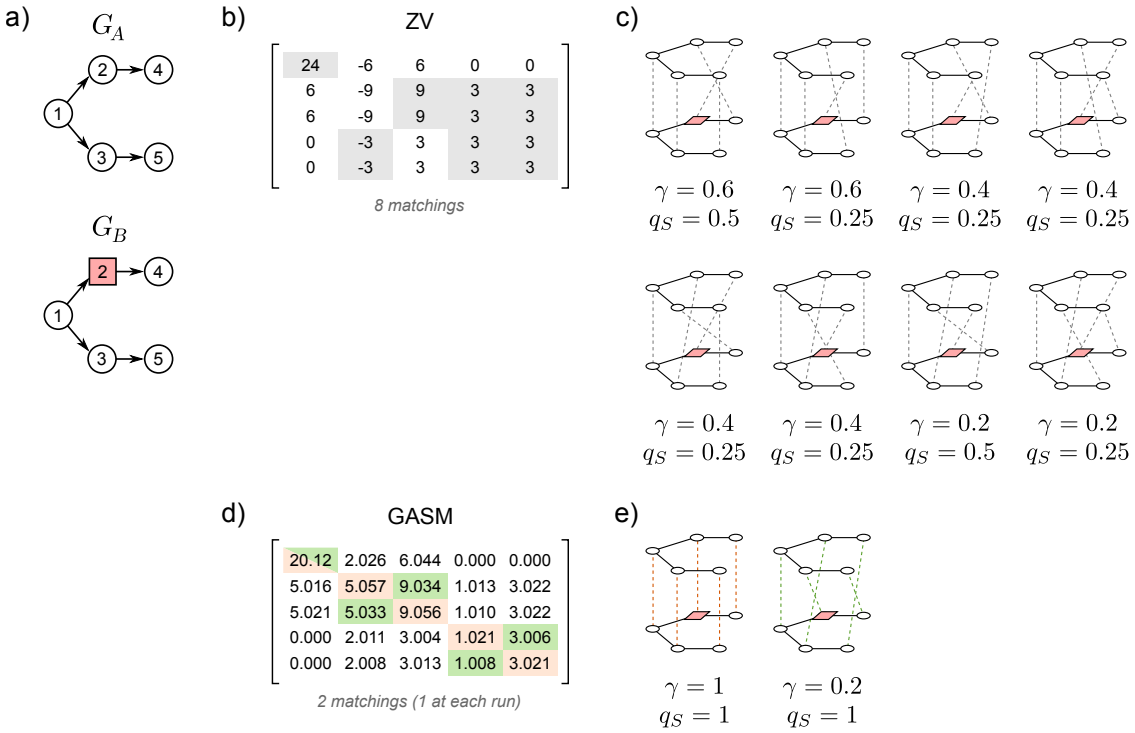

Figure 4: Managing intrinsic attribute incoherence. **a)** The two graphs share the same structure but one vertex categorical attribute differs, symbolized by a red square. **b)** Score matrix $X_{\tilde{k}=2}$ given by ZV, without normalization. The matching solutions comprise only the grayed cells. **c)** The 8 corresponding matchings solutions with the best score (36), along with their accuracy γ and structural quality q_S. **d)** Example of score matrix $X_{\tilde{k}=2}$ produced by the GASM algorithm, without normalization and with a relatively large noise $\eta = 10^{-2}$ to ease visualization. Here the best matching solution lies in the orange cells, but other initial random numbers could favor the green cells. **e)** The 2 corresponding matching solutions.

there is on average one vertex correctly assigned per row, and if h is the depth of the binary tree there are on average $h + 1$ vertices correctly matched in total. As the total number of vertices is $2^{h+1} - 1$, the maximum possible average accuracy reads:

$$\gamma_{BT} = \frac{h+1}{2^{h+1}-1} \tag{32}$$

While 2opt and FAQ have poor accuracy on such graphs (Figure 5-a middle), ZV and GASM stick to the theoretical maximum curve. To compare their solutions, one has to look at the structural quality q_S (Figure 5-a bottom), for which GASM has consistently higher values.

Star-branched graphs (Figure 5-b) also have a structure that allows us to determine easily the maximum possible accuracy γ_{SB}. Similarly, there is on average one vertex correctly assigned per row, and if β is the branch depth there are on average $\beta + 1$ vertices correctly matched. If there are k branches, the total number of vertices is $k\beta + 1$ and the maximum average accuracy reads:

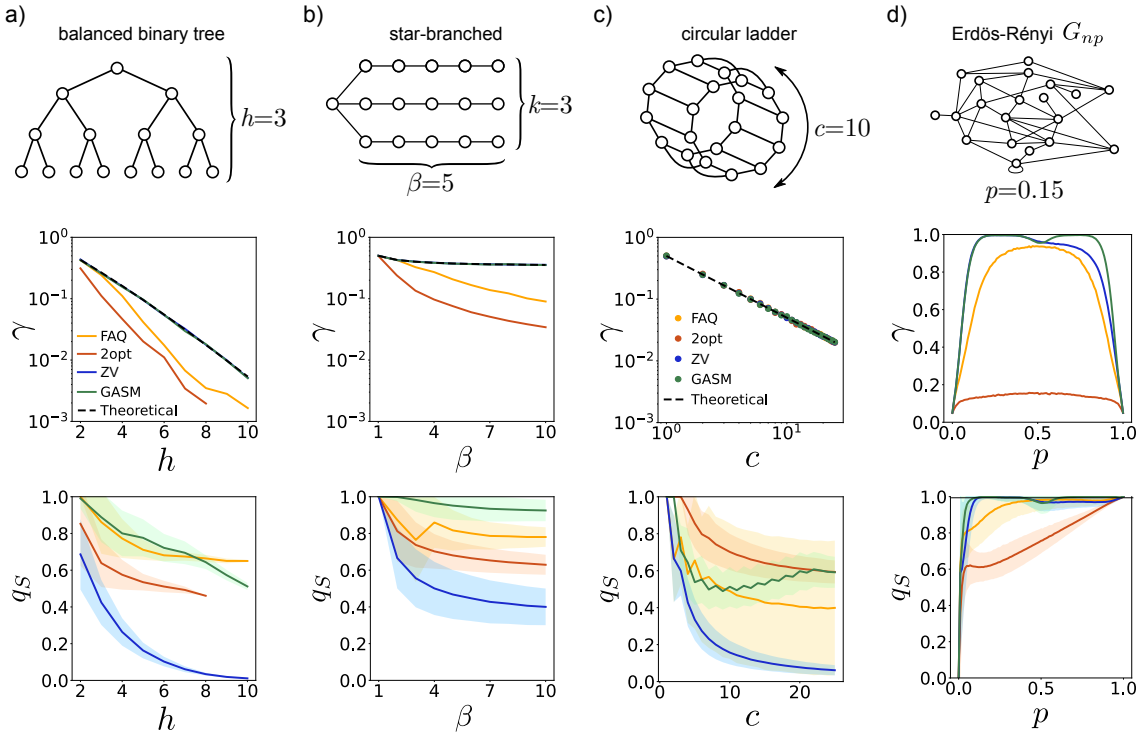

Figure 5: Isomorphic matching of different types of graphs: **a)** balanced binary trees with depth h, **b)** star-branched with $k = 3$ branches of length β, **c)** circular ladder with $2c$ vertices, and **d)** random Erdös-Rényi (ER) G_{np} graphs with $n_A = 20$ vertices and edge probability p. **Top**: Examples of each graph type. **Middle**: average accuracy γ as a function of the graph parameters. **Bottom**: average structural quality q_s computed over the same graphs. Colors are consistent in all panels. Each data point is averaged over 10^4 samples, except for the balanced binary tree where it is variable with h to keep a reasonable computation time; the data points for the 2opt algorithm are missing when $h > 8$ due to a prohibitive computation time.

$$\gamma_{SB} = \frac{\beta + 1}{k\beta + 1} \tag{33}$$

Again, the accuracy of 2opt and FAQ drop as branches grow, while ZV and GASM always sit on the theoretical maximum. We verified that this is true for virtually any values of k, and not just for $k = 3$ as displayed in Figure 5-b. The tie on accuracy is broken by looking at the structural quality, which GASM dominates in all the tested range of parameters.

Third, circular ladders (Figure 5-c) are a very special family of graphs in the context of graph matching. First, the maximum possible average accuracy γ_{CL} can also be determined analytically: since all vertices have the same surrounding structure any vertex can be matched with any vertex, and the best possible average accuracy simply reads:

$$\gamma_{CL} = \frac{1}{2c} \tag{34}$$

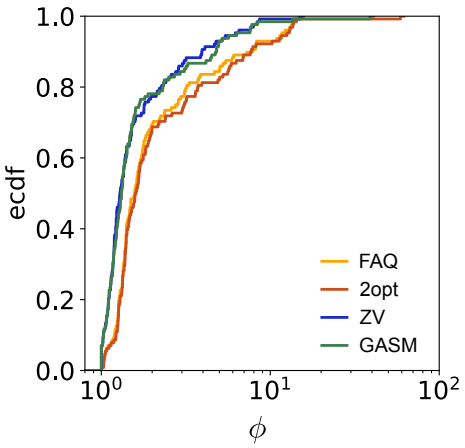

Figure 6: Benchmark on the QAPLIB database: empirical cumulative distributions over the QAPLIB instances of the score ratios ϕ, obtained for different algorithms.

where c is the number of vertices in a ring. Then, the minimal accuracy – corresponding to random pairings – is also equal to $1/2c$. It is thus expected that any algorithm would give solutions with the same accuracy, and indeed all four algorithms gave accuracies lying on the $1/2c$ curve. Again, the structural quality is useful in ranking them: 2opt is better, followed by GASM, FAQ, and finally ZV. When c is large ($c \geq 22$) GASM becomes as good as 2opt.

All these graphs have many symmetries by construction, which may explain why GASM is particularly efficient on these datasets. Let's now turn to the isomorphic matching of random Erdös-Rényi G_{np} graphs, which have much fewer symmetries. For such graph the maximum possible accuracy is difficult to derive analytically, but the general idea is that when p is close to 0 or 1 there are a lot of indeterminacies, *i.e.* many vertices have the same surrounding structure and can be mismatched, while for intermediate values of p there are much fewer indeterminacies and the maximum accuracy is close to 1. Computation reveals that the average accuracy of 2opt is very low and that FAQ is dominated by ZV, which is in turn below GASM. Apparently, ZV is working better on graphs with low p, and the better accuracy of GASM at high p is due to the complementing procedure described in Section 3.4, which is typically activated when $p \gtrsim 0.5$. Again, the structural quality of the solutions is mostly dominated by GASM.

Overall, for these 4 types of graphs, GASM features an excellent performance by consistently providing the best accuracy and displaying the best structural quality for all types but circular ladders, where it is second. In addition, when attributes are present and lift degeneracies GASM can exploit this information and break the theoretical limitations due to the structural local symmetries; in this case, provided there is a sufficient amount of attributes and/or uncertainties are low enough, there is no upper bound and perfect accuracy can, in principle, be achieved for any graph structure.

4.3 QAPLIB benchmark

Verifying the accuracy and structural quality for isomorphic matching is a good sanity check, but matching non-isomorphic graphs is a more realistic task. Let us simplify the problem by running benchmarks in two different contexts: the QAPLIB database, presented here, and graph alteration, which will be presented in the next section.

QAPLIB is a quadratic assignment problem library that has been widely used for benchmarking QAP algorithms [40, 53]. It features 128 problem instances ranging in size from 10 to 256 vertices along with the best known solutions [4]. Each problem comprises two matrices A and B (akin to Λ_A and Λ_B for graphs) and for each permutation P a score can be computed as $tr(APB^\top P^\top)$. The QAP here is thus formulated to search for the *minimal* score, while in the rest of this paper, the similarity scores in X and Y were maximized. We computed the score ratios ϕ as the scores obtained for an algorithm divided by the score of the best-known solution. One exception is the `esc16f` instance, whose best-known solution has a zero score. Since all algorithms found the minimal solution for this instance, the score ratios have been set to 1 for consistency.

The empirical cumulative distributions of ϕ for the 4 algorithms are shown in Figure 6-a. In this representation, the most leftwise curves have better solutions, and it is clear that ZV and GASM consistently found better solutions than 2opt and FAQ. As for the accuracy of isomorphic matching, the ZV and GASM algorithms provide very close scores. However, here the solutions cannot be ranked in terms of structural quality due to the nature of the dataset: many of the instances correspond to fully connected or other peculiar graphs, and the capacity of GASM to resolve local symmetries is largely irrelevant with QAPLIB.

4.4 Graph alteration

Let us now turn to an alteration benchmark, *i.e.* a matching task where one graph is an altered version of the other. There are many ways to alter a graph: vertex swapping, edge swapping or flipping, adding noise to the attributes, *etc.* but we just cover here two major cases where *i*) edges are removed or *ii*) vertices are removed along with the corresponding edges, the latter task being also known as *subgraph matching*. This covers a larger panel of alterations since edge (resp. vertex) removal is equivalent to edge (vertex) addition, by switching G_A and G_B. In this section we assume without loss of generality that G_B is an altered version of G_A, with $n_A \geq n_B$ and $m_A \geq m_B$. The indices of the vertices of G_B have also been systematically shuffled to avoid any accuracy bias. Note that the way elements are chosen for alteration is also determinant: it can be at random or in a given graph region, meaning that other regions are preserved. Here we will stick to the random case.

4.4.1 Edge removal

Let us start with random edge removal, a task controlled by the alteration parameter δ_e defined as the number of removed edges divided by the initial number of edges: $\delta_e = 1 - m_B/m_A$.

A comparison of the average accuracy of FAQ and GASM for the edge removal of G_{np} graphs is displayed in Figure 7, both for directed and undirected graphs. Without attribute, both algorithms see their accuracy drop steeply with δ_e for undirected graphs, while for directed graphs GASM has an exponential decay and FAQ features a very good tolerance to small alterations with an accuracy close to perfection up to $\delta_e = 0.2$, though a rapid subsequent drop in accuracy places it below GASM for severe alterations ($\delta_e > 0.5$).

The might of GASM becomes apparent when attributes can be exploited to improve the solutions. FAQ can also manage one measurable edge attribute but does not take into account the uncertainty over this attribute. The middle panels of Figure 7 compare FAQ and GASM accuracies in this situation ($\xi = 1$), and the result highly depends on the uncertainty ρ over the attribute: high uncertainties make GASM score poorly while low uncertainty turns it into an extremely alteration-tolerant algorithm. For instance, the average accuracy for directed graphs with $\rho = 0$ is still at $\gamma = 0.9997$ when $\delta_e = 0.5$, to be compared to $\gamma = 0.1121$ for FAQ. This is not surprising since in

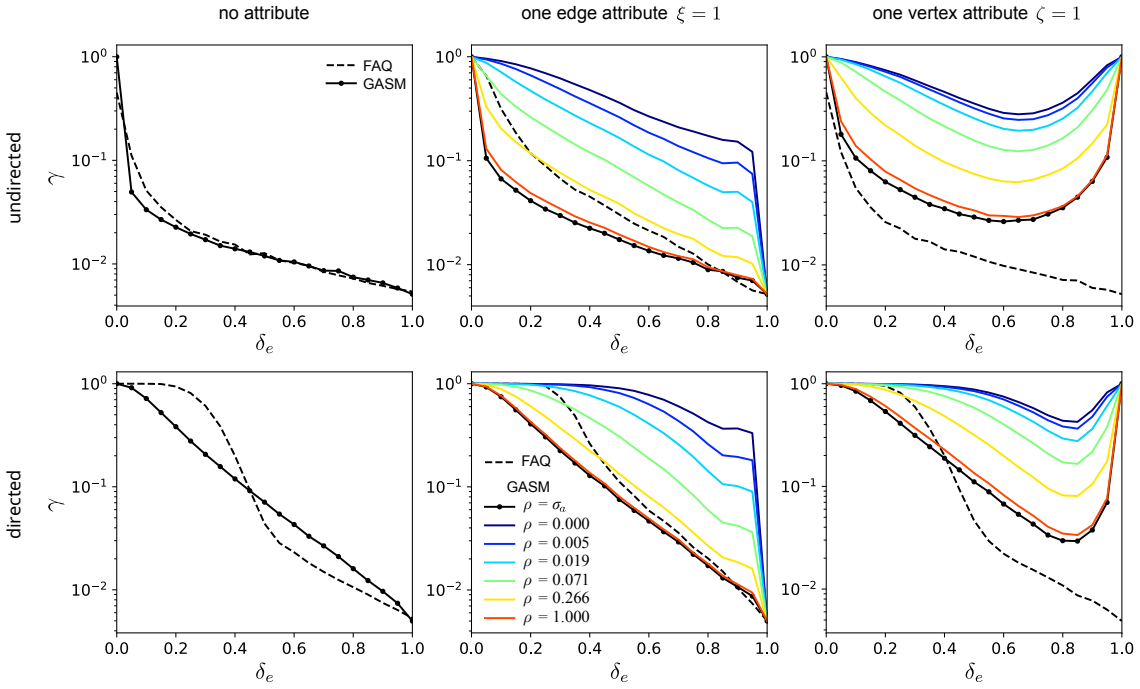

Figure 7: Edge alteration benchmark. Average accuracy γ of FAQ and GASM plotted as a function of the edge alteration ratio δ_e for various conditions. Averaging is performed over 1,000 ER G_{np} graphs with $n = 200$ vertices and $p_A = log(n)/n \simeq 0.01$. The top plots are for undirected graphs while the bottom plots are for directed graphs. **Left**: graphs have no attribute. **Middle**: graphs have one measurable edge attribute ($\xi = 1$) drawn from the standard normal distribution $\mathcal{N}(0, 1)$. **Right**: graphs have one measurable vertex attribute ($\zeta = 1$) drawn from the standard normal distribution. In this case, the FAQ curves correspond to no attribute and are displayed for reference. When an attribute is defined (middle, right), the default GASM attribute uncertainty (solid black) is the standard deviation of the difference of all attribute pairs, $\rho = \sigma_a$ and colored curves correspond to manually defined attribute uncertainty ρ.

that case, GASM bases the matching primarily on the attribute information and the structure is almost ignored.

A similar tendency is obtained when there is one measurable vertex attribute ($\zeta = 1$), as shown in the right panels of Figure 7. As FAQ cannot manage such an attribute, it is ignored and the average accuracy is the same as if there was no attribute.

4.4.2 Subgraph matching

Let us now turn to vertex removal or subgraph matching. In this task, the graph G_B is composed of a random subset of vertices of G_A, and only the edges of G_A whose both vertices are in G_B are kept. The subgraph task is thus parametrized by the properties of the initial graph G_A and the alteration ratio δ_v defined as the number of vertices removed divided by the initial number of vertices, which can be expressed as $\delta_v = 1 - n_B/n_A$. The solution accuracy γ is defined as the

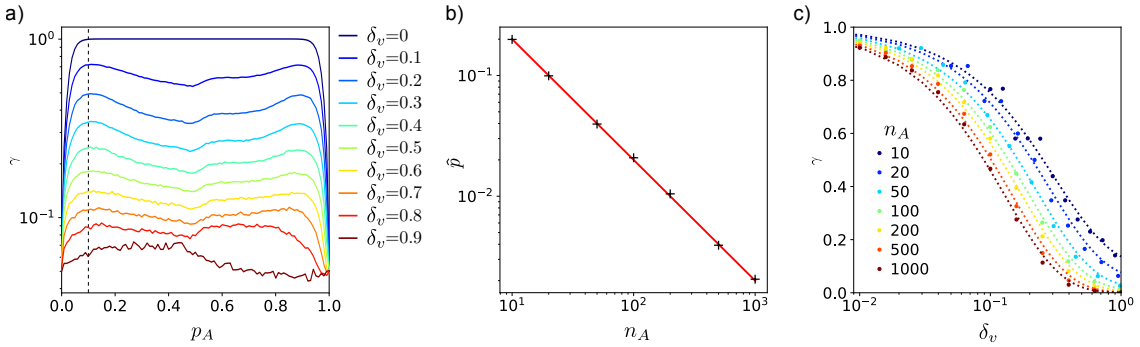

Figure 8: Accuracy peaks for directed ER graphs. **a)** Accuracy γ as a function of the initial probability of edge p_A, for different vertex alteration ratios δ_v and $n_A = 20$. All data points are averaged over 10^4 realizations. Note that the curves with $\delta_v \in [0.1, 0.8]$ all have a peak at the same value $\hat{p}$ (dashed line). A similar peak occurs at $1 - \hat{p}$ because of the complement procedure described in Section 3.4 – without it, the accuracy would monotonously decrease for $p_A > \hat{p}$. **b)** Scaling of $\hat{p}$ as a function of the initial number of vertices n_A (black), fitted by $\hat{p} = 2/n_A$ (red). **c)** Accuracy γ as a function of the alteration ratio δ_v for different initial network sizes n_A with $p_A = \hat{p}(n_A)$ (dots). Dotted curves correspond to fits given by eq.(36).

ratio of correctly matched vertices divided by the total number of vertices in the subgraph n_B.

For ER graphs, the initial graph G_A is parametrized by the number of vertices n_A and the edge ratio p_A. To reduce the number of parameters for the benchmark, we first tried to find the value of p_A for which the accuracy maximum. Indeed, Zager and Verghese [52] observed that, for a few values of p_A, the accuracy decayed when p_A increased. However, at both limits $p_A = 0$ (fully disconnected) and $p_A = 1$ (fully connected) no structural information can be inferred, and the accuracy has to drop to the minimal value of random matchings. So there has to be a maximum accuracy for some value of p_A in $]0, 1[$.

Figure 8-a shows the accuracy of GASM as a function of p_A for different alteration ratios, and it appears that the peak location is essentially independent of δ_v. Of course, when $\delta_v = 0$ the subgraph G_B is isomorphic to G_A and the maximum possible accuracy is equal to one, except close to the extreme values of p_A where the informational cut-off occurs. We then define $\hat{p}$ as the location of the accuracy peak in the range $\delta_v \in [0.1, 0.8]$, and computed it for different sizes of the initial graph n_A (Figure 8-b). The data points are nicely well-fitted by an inverse law, such that one can assess the empirical relationship:

$$\hat{p} = \frac{2}{n_A} \tag{35}$$

Since $p_A = d_A/n_A$, with $d_A = m_A/n_A$ being the average degree of G_A, this equation has a simple interpretation: the informational cut-off appears when $d_A \leq 2$, which corresponds to the threshold below which a graph is necessarily fragmented. Then, it is noticeable that the accuracy of GASM is well described by fits of the following form as a function of the alteration ratio δ_v:

$$\gamma(\delta_v) = \frac{1}{n_A} + \left(1 - \frac{1}{n_A}\right) e^{-\delta_v/\alpha} \tag{36}$$

where α is a fit parameter (Figure 5-c).

A comparison of the average accuracy of FAQ and GASM for the vertex alteration of ER G_{np} graphs is displayed in Figure 9, both for directed and undirected graphs. Without attribute, the general trend is very similar to what is observed for edge alteration in Figure 7-left, except for directed graphs where FAQ is less robust to small alterations and here the curves for both algorithms are much more similar.

Again, the ability of GASM to exploit attributes makes it more accurate than FAQ when the uncertainty is low enough, for both an edge attribute ($\xi = 1$, Figure 9-middle) and a vertex attribute ($\zeta = 1$, Figure 9-right). The dominance of GASM is particularly striking with one vertex attribute, since for all the tested uncertainties the resulting accuracy is always at least one order of magnitude above FAQ.

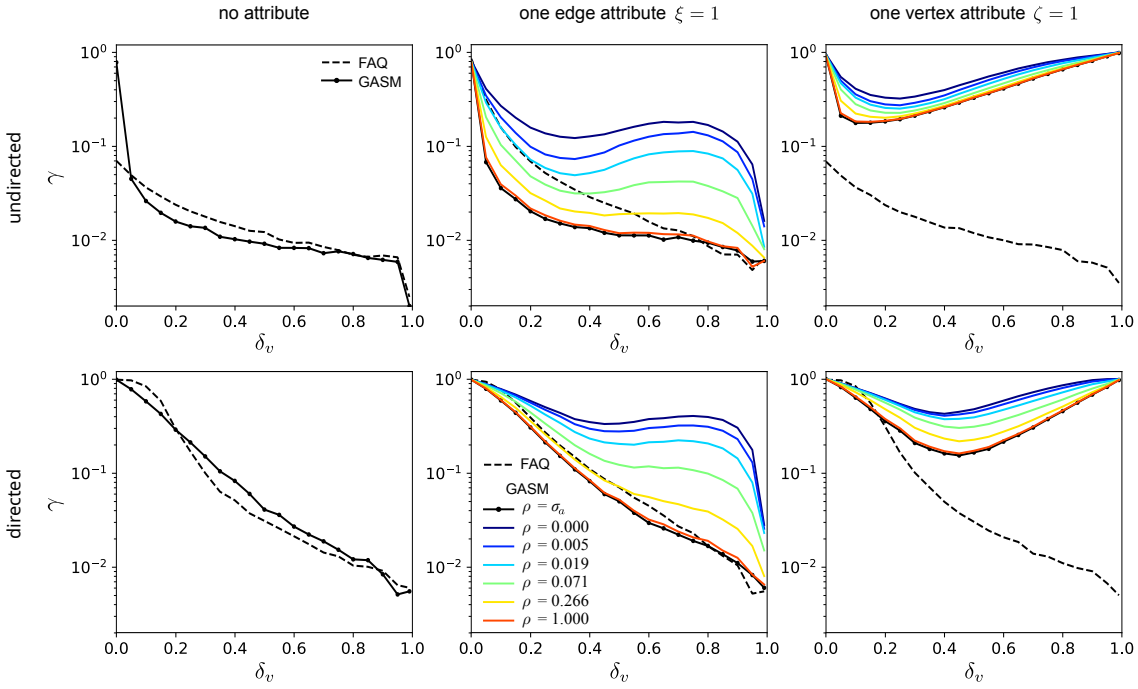

Figure 9: Vertex alteration benchmark. Average accuracy γ of FAQ and GASM plotted as a function of the vertex alteration ratio δ_v for various conditions. Averaging is performed over 10^3 ER G_{np} graphs with $n_A = 200$ vertices and $p_A = \hat{p}(n_A) = 0.005$. The top plots are for undirected graphs while the bottom plots are for directed graphs. **Left**: graphs have no attribute. **Middle**: graphs have one measurable edge attribute ($\xi = 1$) drawn from the standard normal distribution $\mathcal{N}(0, 1)$. **Right**: graphs have one measurable vertex attribute ($\zeta = 1$) drawn from the standard normal distribution. In this case, the FAQ curves correspond to no attribute and are displayed for reference. When an attribute is defined (middle, right), the default GASM uncertainty (solid black) is the standard deviation of the difference of all attribute pairs, $\rho = \sigma_a$ and colored curves correspond to manually defined uncertainties ρ.

4.5 Speed

As for many resource-demanding problems, the search for approximate matching solutions faces a trade-off between accuracy and efficiency, the latter referring to computation time and memory resources. As observed in [48], slow algorithms could probably achieve better accuracy given more time, and at the extreme, an exhaustive search could reach optimal solutions at the cost of utmost time and memory budget. Put differently, accuracy and efficiency define a space where the best algorithms sit on a Pareto front and proper benchmarks should take both aspects into account.

However, computation speeds are difficult to compare for several reasons. Implementations and hardware constantly improve, and there has been a speedup of 2 orders of magnitude from the first Matlab implementation of FAQ in 2015 [48] to our tests 10 years later with the current Scipy implementation running on a more recent computer. Even when one takes care to run a timing benchmark with algorithms written in the same language on the same computer, the high level of optimization of older algorithms makes the comparison with the first implementation of an emerging algorithm rather unfair. Finally, some algorithms are suitable for implementations on a GPU while others are not, and it is delicate to compare the timings when the hardware and technology are different.

With these limitations in mind, we conducted a timing benchmark for the isomorphic matching of ER G_{np} graphs whose results are summarized in Figure 10. Hardware specifics are detailed in Supplementary Section S1.1. Importantly, GASM is well-suited to GPU parallelization and, as explained in Section 3.4, we implemented both a CPU version and a CUDA version. As with all GPU algorithms, the data transfer between the host and the device has a cost, and for GASM it dominates the global running time for small graphs (typically below 10^2 vertices). This is probably not an issue for many applications since for such small graphs the total running time remains below 10ms.

The stakes are higher for larger graphs ($n > 10^2$). In this range, the benchmark shows that: *i*) 2opt is always slower by orders of magnitude, *ii*) the CPU version of GASM is as fast as ZV in all cases, and *iii*) slightly faster than FAQ for undirected graphs, but approximately 10 times slower

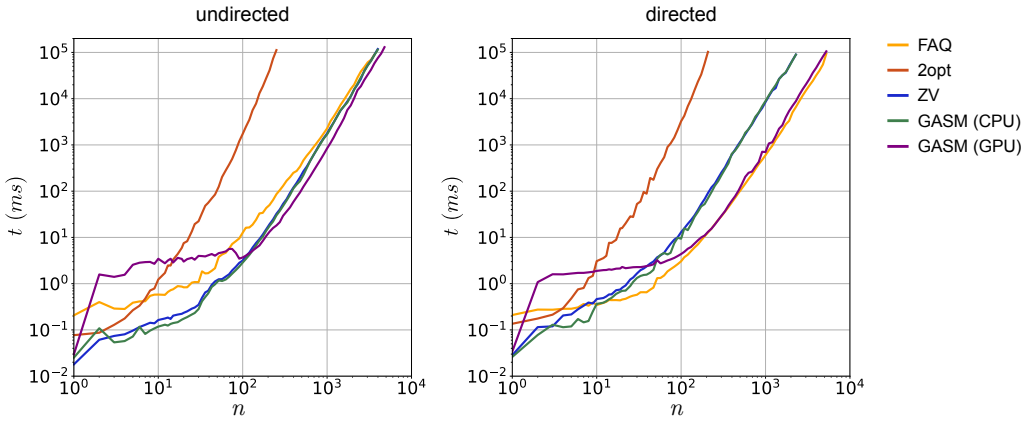

Figure 10: Timing benchmark on the isomorphic matching of ER G_{np} graphs with n vertices and $p = log(n)/n$. The benchmark has been performed on undirected (left) and directed (right) graphs, for 4 algorithms running on CPU and the implementation of GASM on GPU. All computing times are averaged over 10 realizations.

for directed graphs. However, the GPU version of GASM is faster than FAQ for undirected graphs – though the difference tends to vanish for large graphs – and as fast as FAQ for directed graphs.

The previous Sections indicated that GASM provides better solutions (in terms of accuracy, QAP scores and structural quality) than the other algorithms in most situations. The timing benchmark indicates that it is possible to have GASM running as fast or faster than the other algorithms, including FAQ, so altogether it seems to point out that GASM is more Pareto-optimal than all the other algorithms tested here.

5 Conclusion

This work presents the *Graph Attributes and Structure Matching* (GASM) algorithm, which takes root in the iterative methods for graph matching. Notably, it improves ZV in a number of ways:

- it uses a minute noise to lift the degeneracies due to local symmetries,
- it implements a complement procedure, to take advantage of the fact that solutions are more accurate when the graphs are sparse,
- it handles properly isolated vertices,
- an *ad hoc* convergence criterion is proposed,
- a GPU implementation has been implemented, which is particularly well-suited for this family of algorithms,
- the integration of the attributes is done before the iterative procedure, which improves the quality of solutions and makes the algorithm more robust to discrepancies in the attributes,
- the ability to handle any number and types of attributes makes it well-suited to tackle real-world problems.

Importantly, GASM also introduces the notion of uncertainty over the attributes, which tunes *in fine* how much the algorithm relies on the structure or the attributes: if the attribute values are highly discriminant then GASM exploits principally this information, while attributes estimated with a large uncertainty only influence the solution search marginally.

Beyond the GASM algorithm, this study also formalizes the difference between categorical and measurable attributes and proposes a common framework to incorporate all this information. It also sheds light on the importance of systematically benchmarking the undirected and directed cases separately, as we saw differences in all the measurements of our benchmarks, for all the tested algorithms. Moreover, we emphasize the importance of taking into account not only the accuracy /performance score of the solutions but also other measurements that are relevant in the context of graph matching, like the structural quality for instance.

For future work, several leads can be explored. First, sometimes a partial matching of the vertices is known *a priori*, and seeded graph matching has gained a lot of attention in recent years [30]. GASM can certainly be modified to leverage this information as well.

Then, there is room for further speed improvements. On the algorithmic side, a better convergence criterion could considerably speed up the process by avoiding unnecessary iterations. It is clear from Supp. Fig. S3 that there is still a lot of room for improvements on this matter; in addition, taking into account the attributes for the convergence criterion should be also beneficial.

Next, a fully on-GPU version is yet to be implemented, by porting on the device the computation of the initial score matrices and the LAP solver itself, as well as further optimization with libraries dedicated to sparse matrices. On the hardware side, the future use of PCIe 5.0 should in theory double the transfer rates and yet improve further the speed of GPU implementations.

Finally and most excitingly, the next scientific challenge is to apply GASM to real datasets, like for instance the comparison of protein-protein interaction networks, connectomes and artificial neural networks.

References

[1] R. Abboud, I. I. Ceylan, M. Grohe, and T. Lukasiewicz. The surprising power of graph neural networks with random node initialization. *IJCAI International Joint Conference on Artificial Intelligence*, 3:2112–2118, 8 2021. URL: https://www.ijcai.org/proceedings/2021/291, doi:10.24963/IJCAI.2021/291.

[2] V. D. Blondel, A. Gajardo, M. Heymans, P. Senellart, and P. V. Dooren. A measure of similarity between graph vertices: Applications to synonym extraction and web searching. *https://doi.org/10.1137/S0036144502415960*, 46:647–666, 8 2006. URL: https://epubs.siam.org/doi/10.1137/S0036144502415960, doi:10.1137/S0036144502415960.

[3] H. Bunke. Recent developments in graph matching. *Proceedings - International Conference on Pattern Recognition*, 2:117–124, 2000. doi:10.1109/ICPR.2000.906030.

[4] R. E. Burkard, S. Karisch, and F. Rendl. Qaplib-a quadratic assignment problem library. *European Journal of Operational Research*, 55:115–119, 11 1991. doi:10.1016/0377-2217(91)90197-4.

[5] D. Conte, P. Foggia, C. Sansone, and M. Vento. Graph matching applications in pattern recognition and image processing. *IEEE International Conference on Image Processing*, 2:21–24, 2003. doi:10.1109/ICIP.2003.1246606.

[6] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *https://doi.org/10.1142/S0218001404003228*, 18:265–298, 11 2011. URL: https://www.worldscientific.com/worldscinet/ijprai, doi:10.1142/S0218001404003228.

[7] G. Corso, H. Stark, S. Jegelka, T. Jaakkola, and R. Barzilay. Graph neural networks. *Nature Reviews Methods Primers*, 4:17, 3 2024. doi:10.1038/s43586-024-00294-7.

[8] G. A. Croes. A method for solving traveling-salesman problems. *Operations Research*, 6:791–812, 12 1958. doi:10.1287/OPRE.6.6.791.

[9] D. F. Crouse. On implementing 2d rectangular assignment algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 52:1679–1696, 8 2016. doi:10.1109/TAES.2016.140952.

[10] K. Devkota, A. Blumer, X. Hu, and L. Cowen. *Fast Approximate IsoRank for Scalable Global Alignment of Biological Networks*, volume 14758. Springer, 2024.

[11] P. J. Dickinson, H. Bunke, A. Dadej, and M. Kraetzl. Matching graphs with unique node labels. *Pattern Analysis and Applications*, 7:243–254, 9 2004. URL: https://dl.acm.org/doi/10.1007/s10044-004-0222-5, doi:10.1007/S10044-004-0222-5.

[12] V. P. Dwivedi, L. Rampášek, M. Galkin, A. Parviz, G. Wolf, A. T. Luu, and D. Beaini. Long range graph benchmark. *Advances in Neural Information Processing Systems*, 35:22326–22340, 12 2022. URL: https://github.com/vijaydwivedi75/lrgb.

[13] F. Emmert-Streib and M. Dehmer. Networks for systems biology: conceptual connection of data and function. *IET systems biology*, 5:185–207, 5 2011. URL: https://pubmed.ncbi.nlm.nih.gov/21639592/, doi:10.1049/IET-SYB.2010.0025.

[14] F. Emmert-Streib, M. Dehmer, and Y. Shi. Fifty years of graph matching, network alignment and network comparison. *Information Sciences*, 346-347:180–197, 6 2016. doi:10.1016/J.INS.2016.01.074.

[15] S. E. Fienberg. A brief history of statistical models for network analysis and open challenges. *Journal of Computational and Graphical Statistics*, 21:825–839, 2012. URL: https://www.tandfonline.com/doi/abs/10.1080/10618600.2012.738106, doi:10.1080/10618600.2012.738106.

[16] D. E. Fishkind, S. Adali, H. G. Patsolic, L. Meng, D. Singh, V. Lyzinski, and C. E. Priebe. Seeded graph matching. *Pattern Recognition*, 87:203–215, 3 2019. doi:10.1016/J.PATCOG.2018.09.014.

[17] H. Fröhlich, J. K. Wegner, F. Sieker, and A. Zell. Optimal assignment kernels for attributed molecular graphs. In *ICML 2005 - Proceedings of the 22nd International Conference on Machine Learning*, pages 225–232. Association for Computing Machinery (ACM), 2005. URL: https://dl.acm.org/doi/10.1145/1102351.1102380, doi:10.1145/1102351.1102380.

[18] K. Fukuda and T. Matsui. Finding all minimum-cost perfect matchings in bipartite graphs. *Networks*, 22:461–468, 8 1992. URL: https://onlinelibrary.wiley.com/doi/full/10.1002/net.3230220504https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230220504https://onlinelibrary.wiley.com/doi/10.1002/net.3230220504, doi:10.1002/NET.3230220504.

[19] S. M. Hsieh and C. C. Hsu. Graph-based representation for similarity retrieval of symbolic images. *Data and Knowledge Engineering*, 65:401–418, 6 2008. doi:10.1016/J.DATAK.2007.12.004.

[20] K. Huang, Y. Jin, E. Candès, and J. Leskovec. Uncertainty quantification over graph with conformalized graph neural networks. In *37th Conference on Neural Information Processing System, NeurIPS 2023*, 2023. URL: https://github.com/snap-stanford/.

[21] R. Jonker and A. Volgenant. Computing a shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38:325–340, 1987.

[22] B. H. Junker and F. Schreiber. Analysis of biological networks. *Analysis of Biological Networks*, pages 1–346, 8 2007. URL: https://onlinelibrary.wiley.com/doi/book/10.1002/9780470253489, doi:10.1002/9780470253489.

[23] B. Khemani, S. Patil, K. Kotecha, and S. Tanwar. A review of graph neural networks: concepts, architectures, techniques, challenges, datasets, applications, and future directions. *Journal of Big Data*, 11:1–43, 12 2024. URL: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-023-00876-4, doi:10.1186/S40537-023-00876-4/TABLES/13.

[24] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46:604–632, 9 1999. URL: https://dl.acm.org/doi/10.1145/324133.324140, doi:10.1145/324133.324140.

[25] N. M. Kriege, F. D. Johansson, and C. Morris. A survey on graph kernels. doi:10.1007/s41109-019-0195-3.

[26] C. O. Laura, S. Wesarg, and G. Sakas. Graph matching survey for medical imaging: On the way to deep learning. *Methods*, 202:3–13, 6 2022. doi:10.1016/J.YMETH.2021.06.008.

[27] Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli. Graph matching networks for learning the similarity of graph structured objects, 5 2019. URL: https://proceedings.mlr.press/v97/li19d.html.

[28] A. Lin, N. Dyubankova, T. I. Madzhidov, R. I. Nugmanov, J. Verhoeven, T. R. Gimadiev, V. A. Afonina, Z. Ibragimova, A. Rakhimbekova, P. Sidorov, A. Gedich, R. Suleymanov, R. Mukhametgaleev, J. Wegner, H. Ceulemans, and A. Varnek. Atom-to-atom mapping: A benchmarking study of popular mapping algorithms and consensus strategies. *Molecular Informatics*, 41:2100138, 4 2022. URL: https://onlinelibrary.wiley.com/doi/full/10.1002/minf.202100138https://onlinelibrary.wiley.com/doi/abs/10.1002/minf.202100138https://onlinelibrary.wiley.com/doi/10.1002/minf.202100138, doi:10.1002/MINF.202100138.

[29] Y. Liu, M. Jin, S. Pan, C. Zhou, Y. Zheng, F. Xia, and P. S. Yu. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35:5879–5900, 6 2023. URL: https://dl.acm.org/doi/10.1109/TKDE.2022.3172903, doi:10.1109/TKDE.2022.3172903.

[30] V. Lyzinski, D. E. Fishkind, and C. E. Priebe. Seeded graph matching for correlated erdos-renyi graphs. *Journal of Machine Learning Research*, 15:3693–3720, 2014. URL: http://jmlr.org/papers/v15/lyzinski14a.html.

[31] A. Mehler, A. Lücking, and P. Weiß. A network model of interpersonal alignment in dialog. *Entropy 2010, Vol. 12, Pages 1440-1483*, 12:1440–1483, 6 2010. URL: https://www.mdpi.com/1099-4300/12/6/1440/htmhttps://www.mdpi.com/1099-4300/12/6/1440, doi:10.3390/E12061440.

[32] A. Meyer-Baese and V. Schmid. Foundations of neural networks. *Pattern Recognition and Signal Analysis in Medical Imaging*, pages 197–243, 1 2014. doi:10.1016/B978-0-12-409545-8.00007-8.

[33] A. Mheich, F. Wendling, and M. Hassan. Brain network similarity: methods and applications. *Network Neuroscience*, 4:507–527, 7 2020. URL: https://dx.doi.org/10.1162/netn_a_00133, doi:10.1162/NETN_A_00133.

[34] M. Nikolić. Measuring similarity of graph nodes by neighbor matching. *Intelligent Data Analysis*, 16:865–878, 1 2012. doi:10.3233/IDA-2012-00556.

[35] E. Olivetti, N. Sharmin, and P. Avesani. Alignment of tractograms as graph matching. *Frontiers in Neuroscience*, 10:224488, 12 2016. URL: www.frontiersin.org, doi:10.3389/FNINS.2016.00554/BIBTEX.

[36] B. D. Pedigo, M. Winding, C. E. Priebe, and J. T. Vogelstein. Bisected graph matching improves automated pairing of bilaterally homologous neurons from connectomes. *Network Neuroscience*, 7:522–538, 6 2023. URL: https://dx.doi.org/10.1162/netn_a_00287, doi:10.1162/NETN_A_00287.

[37] M. Randic and C. L. Wilkins. Graph theoretical approach to recognition of structural similarity in molecules. *Journal of Chemical Information and Computer Sciences*, 19:31–37, 1979. URL: https://pubs.acs.org/doi/abs/10.1021/ci60017a009, doi:10.1021/CI60017A009/ASSET/CI60017A009.FP.PNG_V03.

[38] T. K. Rusch, E. Zurich, M. M. Bronstein, and S. Mishra. A survey on oversmoothing in graph neural networks. 3 2023. URL: https://arxiv.org/abs/2303.10993v1.

[39] R. Sato, M. Yamada, and H. Kashima. Random features strengthen graph neural networks. *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pages 333–341, 2021. URL: https://epubs.siam.org/doi/10.1137/1.9781611976700.38, doi:10.1137/1.9781611976700.38.

[40] C. Schellewald, S. Roth, and C. Schnörr. Evaluation of convex optimization techniques for the weighted graph-matching problem in computer vision. In *Proceedings of the 23rd DAGM-Symposium on Pattern Recognition*, pages 361–368. Springer-Verlag, 2001.

[41] M. Stadler, B. Charpentier, S. Geisler, D. Zügner, and S. Günnemann. Graph posterior network: Bayesian predictive uncertainty for node classification. In *35th Conference on Neural Information Processing System, NeurIPS 2021*, 2021. URL: https://www.daml.in.tum.de/graph-postnet.

[42] Y. Su, F. Han, R. E. Harang, and X. Yan. A fast kernel for attributed graphs. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 486–494. Society for Industrial and Applied Mathematics Publications, 2016. URL: https://epubs.siam.org/doi/10.1137/1.9781611974348.55, doi:10.1137/1.9781611974348.55.

[43] H. Sun, W. Zhou, and M. Fei. A survey on graph matching in computer vision. *Proceedings - 2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics, CISP-BMEI 2020*, pages 225–230, 10 2020. doi:10.1109/CISP-BMEI51763.2020.9263681.

[44] D. Szklarczyk, R. Kirsch, M. Koutrouli, K. Nastou, F. Mehryary, R. Hachilif, A. L. Gable, T. Fang, N. T. Doncheva, S. Pyysalo, P. Bork, L. J. Jensen, and C. V. Mering. The string database in 2023: protein–protein association networks and functional enrichment analyses for any sequenced genome of interest. *Nucleic Acids Research*, 51:D638, 1 2023. URL: /pmc/articles/PMC9825434//pmc/articles/PMC9825434/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC9825434/, doi:10.1093/NAR/GKAC1000.

[45] S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10:695–703, 1988. doi:10.1109/34.6778.

[46] T. Uno. Algorithms for enumerating all perfect, maximum and maximal matchings in bipartite graphs. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1350:93–101, 1997.

URL: https://link.springer.com/chapter/10.1007/3-540-63890-3_11, doi:10.1007/3-540-63890-3_11/COVER.

[47] L. G. Valiant. The complexity of enumeration and reliability problems. *https://doi.org/10.1137/0208032*, 8:410–421, 8 1979. URL: https://epubs.siam.org/doi/10.1137/0208032, doi:10.1137/0208032.

[48] J. T. Vogelstein, J. M. Conroy, V. Lyzinski, L. J. Podrazik, S. G. Kratzer, E. T. Harley, D. E. Fishkind, R. J. Vogelstein, and C. E. Priebe. Fast approximate quadratic programming for graph matching. *PLOS One*, 2015. URL: http://www.acq.osd.mil/rd/basic_, doi:10.1371/journal.pone.0121002.

[49] P. Willett. Chemoinformatics: a history. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 1:46–56, 1 2011. URL: https://onlinelibrary.wiley.com/doi/full/10.1002/wcms.1https://onlinelibrary.wiley.com/doi/abs/10.1002/wcms.1https://wires.onlinelibrary.wiley.com/doi/10.1002/wcms.1, doi:10.1002/WCMS.1.

[50] J. Yan, S. Yang, and E. Hancock. Learning for graph matching and related combinatorial optimization problems. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 2021-January:4988–4996, 2020. URL: https://cir.nii.ac.jp/crid/1360298759936590336, doi:10.24963/IJCAI.2020/694.

[51] K. Yang, K. Swanson, W. Jin, C. Coley, P. Eiden, H. Gao, A. Guzman-Perez, T. Hopper, B. Kelley, M. Mathea, A. Palmer, V. Settels, T. Jaakkola, K. Jensen, and R. Barzilay. Analyzing learned molecular representations for property prediction. *Journal of Chemical Information and Modeling*, 59:3370–3388, 8 2019. URL: https://pubs.acs.org/doi/full/10.1021/acs.jcim.9b00237, doi:10.1021/ACS.JCIM.9B00237/ASSET/IMAGES/LARGE/CI9B00237_0020.JPEG.

[52] L. A. Zager and G. C. Verghese. Graph similarity scoring and matching. *Applied Mathematics Letters*, 21:86–94, 1 2008. doi:10.1016/J.AML.2007.01.006.

[53] M. Zaslavskiy, F. Bach, and J.-P. Vert. A path following algorithm for the graph matching problem. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 31, 12 2009. doi:10.1109/TPAMI.2008.245.

Supplementary Materials

S1 Computational details

S1.1 Hardware and software

All the code used for this article has been written in Python and is available in the following repository:

`https://github.com/CandelierLab/GraphMatching.git`

Scipy's optimized routines have been used as much as possible, notably the LAP solver in `scipy.optimize.linear_sum_assignment()`, which implements the algorithm in [9], and the QAP solver `scipy.optimize.quadratic_assignment()` which implements both the FAQ [48] and 2opt [16] algorithms. ZV and GASM (CPU version) have been written to rely extensively on Numpy's optimization. The GPU version of GASM uses Numba to define the CUDA kernels. The timing benchmark was realized with the `perf_counter_ns()` function of Python's `time` module.

The timing benchmark has been performed on a single machine with the following specifications: Motherboard Asus ROG Maximus Z790 Formula, Intel Core i9-13900KS processor (24 cores, 32 threads) with 192Go of DDR5 RAM (5200 MHz, CL38), and a PNY Nvidia RTX A4500 graphics card (7,168 cores).

S1.2 Approximate normalization factor.

Equations (12-13) and (22-23) define the normalization coefficients f_x and f_y. From a formal point of view, these coefficients can be set to any strictly positive value at each step without altering the outcome of the algorithm, so they are removed in the subsequent equations for readability.

However, in practice, it can be dangerous to set $f_x = f_y = 1$ because the values of the score matrices increase exponentially with the iterations and may cause either a floating point overflow or precision issues related to the unit of least precision. For instance, when two graphs with an average degree of 500 are matched, the scores increase by a factor of the order of 10^6 at each iteration. To reduce transfer and computation times the GPU version of GASM uses single-precision floats, which overflow at approximately $2^{128} \simeq 3.10^{38}$, so the overflow would occur in just 7 iterations. Also, the first integer that is not exactly representable is $2^{24} + 1 \simeq 1.7 \times 10^7$, so precision issues may start within the first iterations.

One way to avoid these issues is to normalize the score matrices by their mean value (or any other norm) at each iteration as in [52], but determining the mean scores is a computational overhead that can be avoided. First, only one normalization per iteration is enough, and we can safely ignore the normalization of the edge scores Y_k, set $f_y = 1$, and normalize only the scores in X_k. Then, only a rough estimate of the normalization coefficient is necessary for keeping the scores in a reasonable range, and an *ad hoc* estimation based on the graphs' average degree can be formulated as follows, both for directed and undirected graphs:

$$f_x = max(4d_A d_B, 1) \tag{S1}$$

where d_* is the average degree of graph G_* (*outdegree* for directed graphs). It can then be slightly improved into the following form:

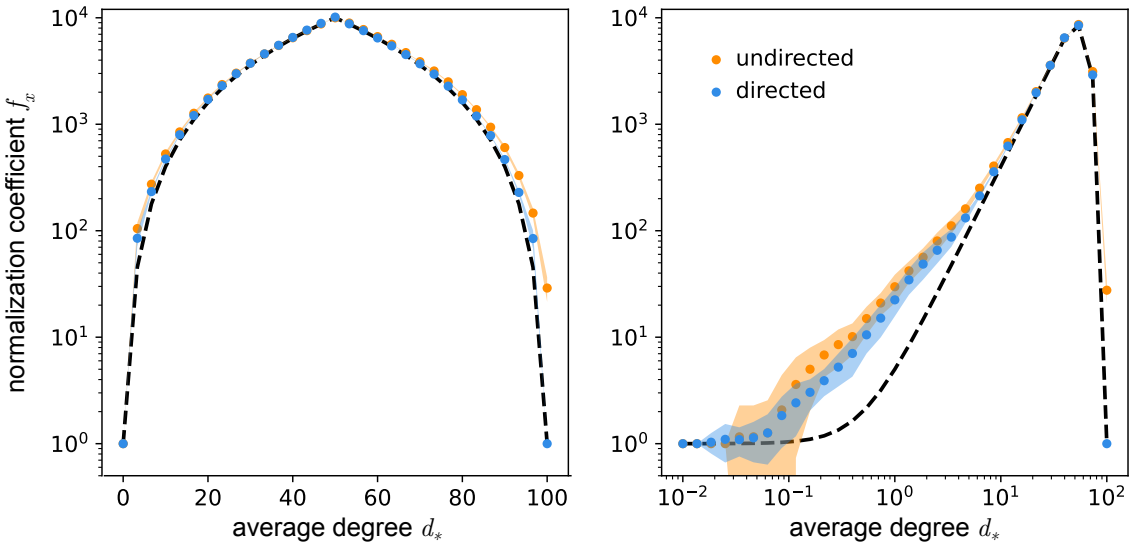

Figure S1: Estimated and approximated normalization coefficients f_x for ER graphs with $n_* = 100$ vertices in the isomorphic matching task, as a function of the average degree of the graphs d_* with linear (left) and logarithmic (right) scales. The estimation of f_x (dots) is defined as the ratio between the mean values of the vertex score matrix X during the last 2 iterations before convergence, *i.e.* $\langle X_{\tilde{k}} \rangle / \langle X_{\tilde{k}-1} \rangle$, averaged over 100 independent runs for directed (blue) and undirected graphs (orange). Shaded areas represent standard deviations. The dashed curve is the approximated normalization coefficient proposed in eq.(S2). The graph is symmetric due to the complement procedure described in Section 3.4.

$$f_x = 4 d_A d_B + 1 \tag{S2}$$

A comparison with estimated coefficients for ER G_{np} graphs is provided in Supp. Fig. S1, showing that the error remains below a factor 10. All the results presented in this article have been computed with the normalization coefficient provided by eq.(S2).

S2 Graphs

S2.1 Random graphs

The random graphs used in the paper are Erdös-Rényi-Gilbert G_{np} graphs, which are constructed by defining a set of n vertices and including every possible edge with probability p, independently of every other edge [15].

S2.2 QAPLIB

The instances and solutions of the QAPLIB library have been downloaded from https://coral.ise.lehigh.edu.

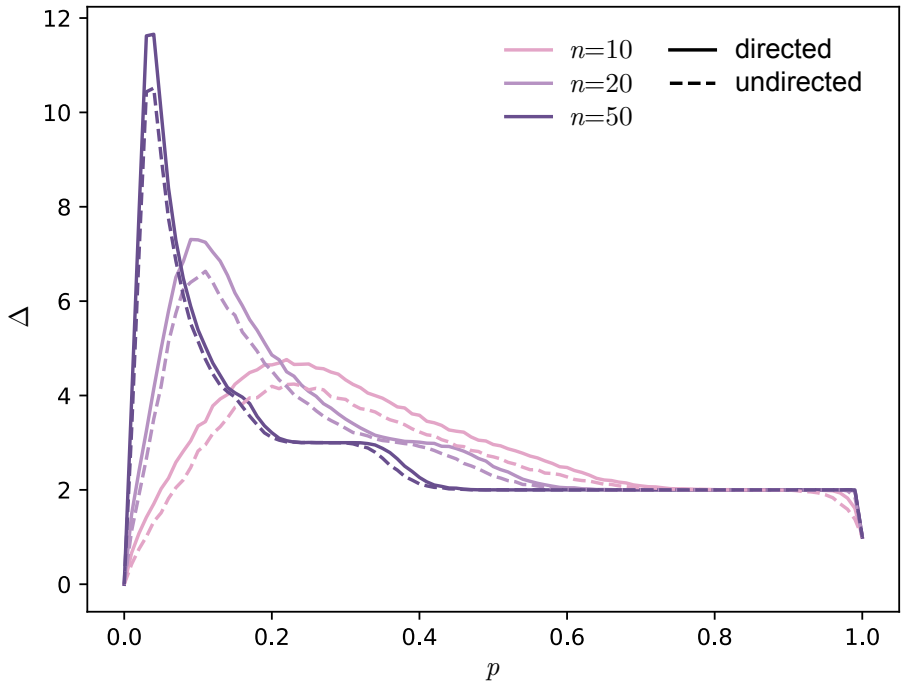

Figure S2: Average diameter Δ of directed and undirected Erdös-Rényi G_{np} graphs as a function of p, for different values of n. Each point is averaged over 1,000 graphs.

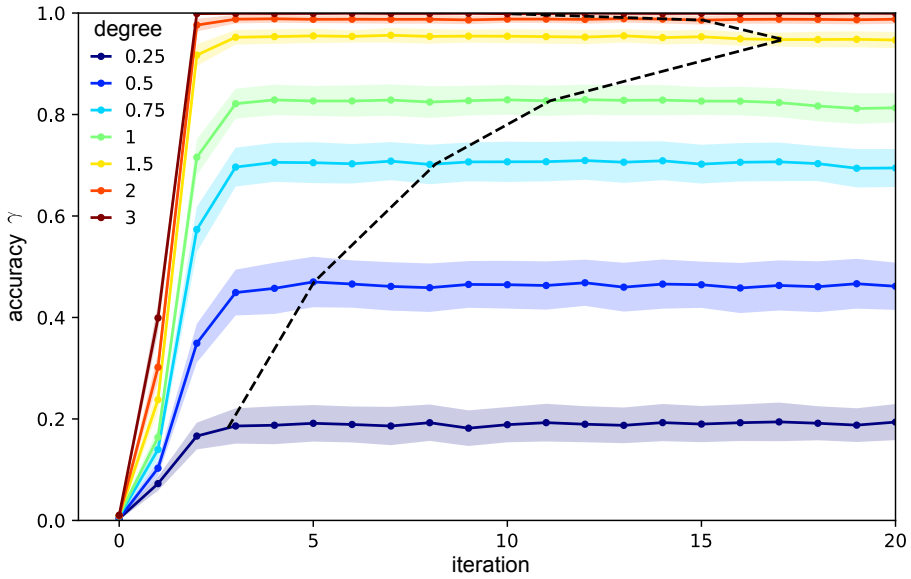

Figure S3: Convergence of GASM. Accuracy γ during isomorphic matching of ER graphs with $n_* = 100$ vertices and no attribute, for different average degrees. Accuracies are averaged over 100 runs, and the standard deviations are represented by the shaded areas. Iteration 0 represents uniform scores (random matching), and iteration $k \geqslant 1$ relies on the score matrix X_k. The dashed black curve indicates the average $\tilde{k}$ for the different degrees.