Graph GOSPA metric: a metric to measure the discrepancy between graphs of different sizes

Jinhao Gu, Ángel F. García-Fernández, Robert E. Firth, Lennart Svensson

***Abstract*—This paper proposes a metric to measure the dissimilarity between graphs that may have a different number of nodes. The proposed metric extends the generalised optimal subpattern assignment (GOSPA) metric, which is a metric for sets, to graphs. The proposed graph GOSPA metric includes costs associated with node attribute errors for properly assigned nodes, missed and false nodes and edge mismatches between graphs. The computation of this metric is based on finding the optimal assignments between nodes in the two graphs, with the possibility of leaving some of the nodes unassigned. We also propose a lower bound for the metric, which is also a metric for graphs and is computable in polynomial time using linear programming. The metric is first derived for undirected unweighted graphs and it is then extended to directed and weighted graphs. The properties of the metric are demonstrated via simulated and empirical datasets.**

***Index Terms*—Metrics, graph matching, generalised optimal sub-pattern assignment metric.**

I. Introduction

Graphs are a type of structured data that can be defined as collections of objects, where certain pairs of objects have a relationship of some kind represented by edges [1]. As a flexible and rich representation, graphs are prevalent in real-world applications in multiple research fields, such as social networks [2], physical systems [3], [4], and chemical reactions [5]. Additionally, graphs can also be applied in machine learning [6] and signal processing tasks [7]–[9].

It is important to be able to measure the discrepancy between graphs using a distance or a loss function [10]. One option to measure the discrepancy between graphs is to first represent each graph as a graphon (also known as a graph limit) [11], which is a bounded symmetric measurable function defined in $[0,1] \times [0,1]$. One can then use a metric in the graphon space, such as the L^p metric or the cut metric, to measure the difference between graphs [11]. However, this approach does not consider node attributes and the resulting metrics on the graphon space cannot be easily interpreted in terms of edge errors, or difference in the number of nodes. In addition, there are many design variables to convert a graph into a graphon.

Another option, used in [12], to measure the difference between graphs of the same sizes is to obtain their single linkage matrices, and then use the Gromov-Haussdorf metric [13], also used in shape analysis [14], to measure the difference between theses single linkage matrices. An alternative is to obtain the Laplacian matrices of the graphs and compute the Wasserstein distance of their induced Gaussian distributions, as in [15]. This distance can measure structural properties of the graphs, but is limited to graphs with the same number of nodes, without node attributes.

A desirable characteristic of a distance function is that it operates directly on the space of interest (in our case graphs) and that it is a mathematically principled metric, as metrics are consistent with an intuitive notion of error. Metrics are functions that meet the properties of non-negativity, identity, symmetry and triangle inequality [16]. Distances that satisfy all the properties of a metric, but not the identity property, are commonly referred to as pseudometrics [17]. Pseudometrics play an important role in measuring the discrepancy between graphs that do not have node attributes. In this case, nodes are represented by arbitrary labels and these pseudometrics only consider the discrepancies in the edge connections so the identity property of metrics is not of relevance. Pseudometrics for graphs can nevertheless be used to define a metric for the equivalence class of isomorphic graphs [17].

Graph distances can be divided into several types based on their properties: 1) distances for graphs with or without node attributes; 2) distances that are metrics for graphs of the same size (same number of nodes); 3) distances that are also metrics for graphs of different sizes; 4) distances that are pseudometrics for graphs of the same size; 5) distances that are pseudometric for graphs of different sizes. In the following, we review some commonly used graph distances.

A classic way to compute the dissimilarity between graphs is through graph edit distances (GEDs) [18], [19]. Based on costs for insertions, deletions, and substitutions of both nodes and edges, the GEDs compute the path of modifications to transform one graph into the other with minimum cost. A GED defines a metric if the individual cost functions for each edit operation are metrics [20]. GEDs are generally NP-hard to compute but there are approximate strategies to compute them in polynomial time, e.g., linear programming relaxation [20], bipartite graph matching [21], and greedy algorithms [22], [23]. However, these approximations do not necessarily preserve the metric properties.

It is also possible to define graph distances by computing

J. Gu and A. F. García-Fernández are with Department of Electrical Engineering and Electronics, University of Liverpool, Liverpool L69 3GJ (emails: {jinhgu, angel.garcia-fernandez}@liverpool.ac.uk). A. F. García-Fernández is also with the ARIES research centre, Universidad Antonio de Nebrija, Madrid, Spain. R. E. Firth is with the STFC Hartree Centre, WA4 4AD Daresbury, UK (email: robert.firth@stfc.ac.uk). L. Svensson is with the Department of Electrical Engineering, Chalmers University of Technology, SE-412 96 Gothenburg, Sweden. This work was supported by the EPSRC Centre for Doctoral Training in Distributed Algorithms EP/S023445/1.

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org., provided by the author. The material includes the appendices. This material is 418 KB in size.

the maximum common subgraph (MCS) between two graphs. For example, a GED can be defined based on the MCS [24]. In addition, it is possible to define a pseudometric based on the MCS [25], in which distances between isomorphic graphs are zero. However, finding the MCS of two graphs is NP-hard, making it computationally challenging, especially for large graphs.

The chemical distance is a pseudo-metric that does not consider information on node attributes [5]. It is obtained by minimising the edge discrepancy between two graphs of the same size, whose mapping is represented via a permutation matrix [5]. Finding this optimal mapping between two graphs is computationally expensive, and the computationally efficient approximations to estimate the optimal permutations [26], [27] break the triangle inequality property. The Chartrand-Kubiki-Shultz (CKS) distance [28] for graphs of the same size consists of applying the chemical distance to the weighted graphs whose adjacency matrices contain pairwise shortest path distances between the nodes.

A family of graph distances for graphs of the same size that includes the chemical distance and the CKS distance was defined in [29]. We refer to these distances as generalised chemical distances (GCD). In its main form, the GCDs only consider edge mismatches and therefore define pseudometrics. The GCDs have the benefit that being computationally tractable. For example, an implementation based on the alternating direction method of multipliers (ADMM) for large graphs was proposed in [30]. Node attribute errors can be incorporated into the GCDs such that GCDs define metrics for graphs of the same size [29]. If graphs are of different sizes, we can apply the GCD by adding dummy nodes such that both graphs are of the same size, as done in the graph matching algorithm in [31] and in the integer programming GED metric in [20]. However, special care should be taken in how these dummy nodes are penalised to preserve the GCD metric properties. For example, in [30], the dummy nodes have zero as an attribute and do not have edges with other nodes. This use of dummy nodes implies that the triangle inequality property is not met. For instance, if we compare two graphs without edges, we compare two sets of nodes, and defining a metric for sets of nodes requires more considerations [32].

In this paper, we propose the graph generalised optimal subpattern assignment (GOSPA) metric to measure the discrepancy between graphs of different sizes with node attributes. The graph GOSPA metric is an extension of the GOSPA metric [32], which is a metric for sets, to graphs. The graph GOSPA metric computes an optimal assignment between nodes by penalising attribute errors for assigned nodes and also penalising nodes that are left unassigned. Apart from penalties for nodes, the graph GOSPA metric includes penalties for edge mismatches in the graphs. As the graph GOSPA metric can leave nodes in both graphs without assignment, it does not require the introduction of dummy nodes to deal with graphs of different sizes. This is computationally beneficial as it lowers the dimensionality of the optimisation problem. In addition, it includes a hyperparameter p to adapt the cost of outliers. We also propose a lower bound on the graph GOSPA metric that is also a metric and is computable in polynomial time using linear programming (LP). We first present the graph GOSPA metric for unweighted, undirected graphs, and then we extend it to weighted and also directed graphs. We also define an associated pseudometric for graphs that do not have node attributes.

The outline of the paper is as follows. In Section II, we formulate the problem of designing a metric and provide the background. Section III presents the proposed metric based on optimal assignments. In Section IV, we present the LP metric and its decomposition in terms of localisation costs for properly assigned nodes, costs for missed nodes, false nodes, and edge mismatch. In Section V, we extend the proposed metric to directed and weighted graphs. In Section VI, we analyse the proposed metric implementations via simulations and real datasets. Finally, conclusions are drawn in Section VII.

II. Problem formulation and background

In this section, we formulate the problem of designing a metric for undirected and unweighted graphs and review the GOSPA metric.

A. A metric for graphs

An undirected, unweighted graph (V, E) consists of a pair of sets: the vertex set V and the edge set E. The vertex set $V = \{x_1, ..., x_{n_X}\}$ is a set of vertices (also called nodes) with the i-th node denoted by $x_i \in \mathbb{X}$, where $\mathbb{X}$ is the single-node attribute space, which has an associated base metric $d(\cdot, \cdot)$. For example, if the attribute is a real vector, we can set $\mathbb{X} = \mathbb{R}^N$. Therefore, $V \in \mathcal{F}(\mathbb{X})$, where $\mathcal{F}(\mathbb{X})$ denotes the set of finite subsets of $\mathbb{X}$. Given the set of nodes, the edge set $E(V) \subseteq \mathbb{E} = \{\{x, y\} : x, y \in V, x \neq y\}$, where we consider graphs without self-edges. The space of these graphs is denoted by Ω.

Our objective is to design a metric $d(\cdot, \cdot)$ to measure the difference between two graphs $X = (V_X, E_X)$ and $Y = (V_Y, E_Y)$. Without loss of generality, we assume that X is the ground truth graph and Y is the estimated graph, provided by some algorithm. Therefore, a metric enables us to rank different graph estimates according to their closeness w.r.t. the ground truth graph.

Suppose $d(\cdot, \cdot)$ is a function such that $d : \Omega \times \Omega \to \mathbb{R}_+ = [0, +\infty)$. The function $d(\cdot, \cdot)$ is a metric on Ω if it meets the following properties for all $X, Y, Z \in \Omega$ [16]:

1) (identity) $d(X, Y) = 0$ if and only if $X = Y$;
2) (symmetry) $d(X, Y) = d(Y, X)$;
3) (triangle inequality) $d(X, Y) \leq d(X, Z) + d(Z, Y)$.

The proposed metric will be based on the GOSPA metric [32], which is a metric for sets, and we proceed to review it in the next subsection.

B. GOSPA metric

In this section, we review the GOSPA metric between two sets of nodes. Suppose that $V_X = \{x_1 \dots, x_{n_X}\}$ is the ground truth of set of nodes and $V_Y = \{y_1, \dots, y_{n_Y}\}$ is the estimate of the corresponding set of nodes.

We first define an assignment vector $\pi = \{\pi_1, \ldots, \pi_{n_X}\}$ between sets $\{1, \ldots, n_X\}$ and $\{1, \ldots, n_Y\}$. The assignment vector meets $\pi_i \in \{0, \ldots, n_Y\}$ and if $\pi_i = \pi_{i'} = j > 0$, then this implies that $i = i'$. Here, $\pi_i = j$ implies that x_i is assigned to y_j, and $\pi_i = 0$ implies that x_i is left unassigned. The set of all possible assignment vectors between V_X and V_Y is denoted Π_{V_X, V_Y}.

Definition 1. Given a base metric $d(\cdot, \cdot)$ in $\mathbb{X}$, a scalar $c > 0$, and a scalar p with $1 \leq p \leq \infty$, the GOSPA metric ($\alpha = 2$) [32] between sets V_X and V_Y is

$$d^{(c)}(V_X, V_Y) = \min_{\pi \in \Pi_{V_X, V_Y}} \left(\sum_{i=1}^{n_X} d(x_i, y_{\pi_i})^p + \frac{c^p}{2} \left(n_X + n_Y - 2 \sum_{i=1}^{n_X} \mathbb{1}_{\pi_i > 0}(\pi_i) \right) \right)^{1/p} \tag{1}$$

where $\mathbb{1}_A(\cdot)$ is an indicator function of the set A. That is, $\mathbb{1}_A(\cdot) = 1$ if $x \in A$ or 0 otherwise.

The first term in (1) represents the distance between node attributes for assigned nodes to the power of p. The second term represents the cost of unassigned nodes, to the power of p. Two assigned nodes x_i and y_{π_i} contribute to the overall error with $d(x_i, y_{\pi_i})^p$. An unassigned node in V_X or V_Y incurs a penalty of $c^p/2$. In multi-object tracking, assigned nodes would represent an object and its estimate with a localisation error $d(x_i, y_{\pi_i})$. An unassigned node in V_X would represent a missed object, and an unassigned node in V_Y would represent a false object.

The GOSPA metric is a metric for sets, and is not a metric for graphs as it does not take into account the corresponding set of edges. This implies that GOSPA for sets does not meet the identity property when we consider graphs. Nevertheless, the metric for graphs we propose will be based on extending the GOSPA metric to additionally consider the set of edges, so we refer to the proposed metric as graph GOSPA metric. That is, the graph GOSPA metric will make an assignment between nodes in V_X and V_Y to penalise localisation errors for properly assigned nodes and the number of missed and false nodes. Additionally, the graph GOSPA metric will include an edge mismatch penalty depending on these assignments in a manner that the metric properties in Section II-A are preserved.

III. Graph metric based on optimal assignments

This section is organised as follows. Section III-A presents the graph GOSPA metric. Section III-B provides some examples to illustrate how the metric works.

A. Graph GOSPA metric

The graphs $X = (V_X, E_X)$ and $Y = (V_Y, E_Y)$ have sets of nodes $V_X = \{x_1, ..., x_{n_X}\}$ and $V_Y = \{y_1, ..., y_{n_Y}\}$. The corresponding adjacency matrices to represent the sets of edges for the node orderings $x_{1:n_X}, y_{1:n_Y}$ are A_X and A_Y, respectively [8]. We use $A_X(i, j)$ to represent the (i, j) element of matrix A_X. Specifically $A_X(i, j) = A_X(j, i) = 1$ if there is an edge between node i and node j of X, and $A_X(i, j) = A_X(j, i) = 0$ if there is no edge.

Definition 2. For $1 \leq p < \infty$, a scalar $c > 0$, edge mismatch penalty $\epsilon > 0$ and base metric $d(\cdot, \cdot)$ in the single node space $\mathbb{X}$, the graph GOSPA metric $d^{(c,\epsilon)}(X, Y)$ between two graphs X and Y is

$$d^{(c,\epsilon)}(X, Y) = \min_{\pi \in \Pi_{V_X, V_Y}} \left(\sum_{i=1}^{n_X} d(x_i, y_{\pi_i})^p + \frac{c^p}{2} \left(n_X + n_Y - 2 \sum_{i=1}^{n_X} \mathbb{1}_{\pi_i > 0}(\pi_i) \right) + e_{X,Y}(\pi)^p \right)^{1/p} \tag{2}$$

where the edge mismatch cost is given by

$$\begin{aligned} e_{X,Y}(\pi)^p = & \frac{\epsilon^p}{2} \sum_{i=1:\pi_i>0}^{n_X} \sum_{j=1:\pi_j>0}^{n_X} |A_X(i, j) - A_Y(\pi_i, \pi_j)| \\ & + \frac{\epsilon^p}{2} \sum_{i=1:\pi_i=0}^{n_X} \sum_{j=1:\pi_j>0}^{n_X} |A_X(i, j)| \\ & + \frac{\epsilon^p}{2} \sum_{i=1:\nexists l:\pi_l=i}^{n_Y} \sum_{j=1:\exists l:\pi_l=j}^{n_Y} |A_Y(i, j)|. \end{aligned} \tag{3}$$

We can see that, similarly to the GOSPA metric in (1), (2) is an optimisation problem over all assignment vectors. The function to optimise is the same as in (1), including node attribute errors (localisation errors) and costs for missed and false nodes, but adding an extra penalty for the edge mismatch $e_{X,Y}(\pi)$. There are two types of edge mismatch costs:

- An edge cost created by two pairs of properly assigned nodes (two nodes in X and two nodes in Y). If the edge exists in one graph but it does not exist in the other graph, this contributes to an edge mismatch penalty. The corresponding cost is ϵ^p and the sum over all these costs is given by the first line in (3).
- For both graphs, each edge connecting an assigned node and an unassigned node contributes to a half-edge mismatch penalty. The corresponding cost is $\epsilon^p/2$. The sum over all such costs for edges in X is given by the second line of (3) and the corresponding sum for edges in Y is given by the third line of (3). As this type of edge mismatch cost involving an unassigned node is half the edge cost for a pair of assigned nodes, we refer to it as half-edge mismatch cost.

In the graph GOSPA metric, c, ϵ and p are hyperparameters. We can change c to control the penalty for node unassignments, and the sensitivity to node attribute error. We can also adjust the value of ϵ to change the sensitivity to the changes in connectivity between nodes. Hyperparameter p controls how much outliers are penalised. These parameters should be chosen according to the application.

As we will see in Section IV, the introduction of the half-edge mismatch cost allows us to express the metric using binary assignment matrices. This representation is computationally convenient as it gives rise to the linear programming relaxation of the metric, computable in polynomial time.

The proof that $d^{(c,\epsilon)}(\cdot, \cdot)$ is a metric is provided as a particular case of the proof of the linear programming relaxation metric in Proposition 2, in Section IV.

B. Examples

We use Figure 1 to illustrate how the graph GOSPA metric works. This figure shows a ground truth graph X and four different graph estimates Y. For simplicity, we consider the condition $p = 1$, $\epsilon \ll c$ and $\Delta \ll c$, with Δ shown in Figure 1.

We first consider Figure 1(a), where we have a graph X with nodes x_1, x_2, x_3 (left to right) and graph Y with nodes y_1, y_2, y_3. The optimal assignments for the nodes in X are $(\pi_1, \pi_2, \pi_3) = (1, 2, 3)$, which means that each node in X has been assigned to a node in Y based on the distance between them. In Figure 1(a), every edge between nodes is correctly detected, therefore, there is only localisation error in Figure 1(a) and the metric value is, $d^{(c,\epsilon)}(X, Y) = 3\Delta$, with each pair of assigned nodes contributing with Δ.

In Figure 1(b), the assignment vector for X is also $(1, 2, 3)$. In this case, apart from the three localisation errors, there is also a missed edge between nodes y_1 and y_3 within Y, such that the edge mismatch error ϵ. The metric value for Figure 1(b) is $d^{(c,\epsilon)}(X, Y) = 3\Delta + \epsilon$.

In Figure 1(c), we compare two graphs with different sizes. The optimal assignments for nodes in X are $(1, 2, 0)$. That is, the last node x_3 is left unassigned, which contributes to the error of a missed node, $c/2$. In this case, the edge between the two assigned nodes in Y is properly detected. However, the third node with edges connected the other two nodes in X does not exist in Y. Therefore, we penalise each missed edge connected to the missed node with half edge mismatch penalty $\epsilon/2$. Thus, the total edge mismatch cost in Figure 1(c) is $\epsilon/2 + \epsilon/2 = \epsilon$. As there are two pairs of assigned nodes, each with a localisation error of Δ, the metric value is $d^{(c,\epsilon)}(X, Y) = \Delta + \Delta + c/2 + \epsilon = 2\Delta + c/2 + \epsilon$.

In Figure 1(d), we have one unassigned node in graph X, and another node in Y is also unassigned, as the distance δ between x_3 and y_3 is much larger than c and $\epsilon \ll c$. Therefore, x_3 a missed node and y_3 is a false node. The assignments are $(1, 2, 0)$. Each edge between the unassigned nodes and the assigned nodes contributes with a half-edge mismatch cost. As there are 3 of these edges, the total edge mismatch cost in Figure 1(d) is $\epsilon/2 + \epsilon/2 + \epsilon/2 = 3\epsilon/2$. Then, as there are two pairs of assigned nodes, each with a localisation error of Δ, the metric value is $d^{(c,\epsilon)}(X, Y) = \Delta + \Delta + c/2 + c/2 + 3\epsilon/2 = 2\Delta + c + 3\epsilon/2$. According to the metric, the best estimate of X is in Figure 1(a), which only has localisation errors.

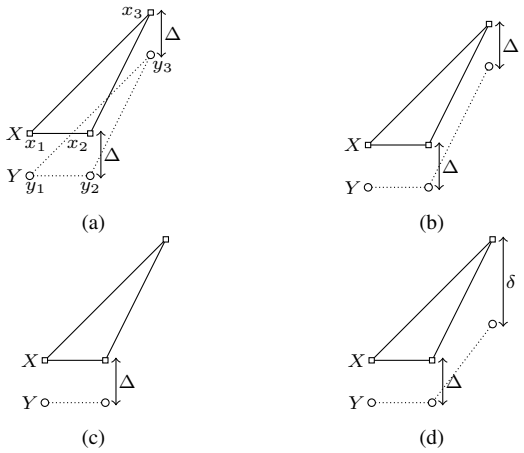


Fig. 1: Example to illustrate the node and edge mismatch costs for the same ground truth graph X, and different estimated graphs Y ($\Delta \ll c$): (a) three properly assigned nodes and no edge mismatch; (b) three properly assigned nodes and one missed edge; (c) two properly assigned nodes, one missed node and two half-edge mismatch penalties; (d) Two properly assigned nodes and two unassigned nodes ($\delta \gg c$), three half-edge mismatch penalties.

IV. LP Graph GOSPA metric

In this section, we show that the metric in (2) can be reformulated as an integer linear programming problem in Section IV-A. In Section IV-B, we also show that the resulting distances are metrics for graphs. In Section IV-C, we explain the metric decomposition.

A. Integer linear programming formulation

We proceed to write the graph GOSPA metric in terms of binary matrices instead of the assignment vectors in Section IV-A. We use $\mathcal{W}_{X,Y}$ to denote the set of all binary matrices of dimension $(n_X + 1) \times (n_Y + 1)$, representing assignments between V_X and V_Y. A matrix $W \in \mathcal{W}_{X,Y}$ satisfies the following properties [33]:

$$\sum_{i=1}^{n_X+1} W(i, j) = 1, \ j = 1, \ldots, n_Y \tag{4}$$

$$\sum_{j=1}^{n_Y+1} W(i, j) = 1, \ i = 1, \ldots, n_X \tag{5}$$

$$W(n_X + 1, n_Y + 1) = 0, \tag{6}$$

$$W(i, j) \in \{0, 1\}, \forall \ i, j \tag{7}$$

where $W(i, j)$ is the element in row i and column j of matrix W. The element $W(i, j) = 1$ if x_i is assigned to y_j. If x_i remains unassigned, $W(i, n_Y + 1) = 1$, and if y_j remains unassigned then $W(n_X + 1, j) = 1$.

There is a bijection between the sets of binary matrices $\mathcal{W}_{X,Y}$ meeting the conditions in (4)–(7) and the sets of assignment vectors Π_{V_X,V_Y}, such that for $\pi \in \Pi_{X,Y}$, $W \in \mathcal{W}_{X,Y}$, $i = 1, \ldots, n_X$ and $j = 1, \ldots, n_Y$:

$$\pi_i = j \neq 0 \quad \Longleftrightarrow W(i, j) = 1 \tag{8}$$

$$\pi_i = 0 \quad \Longleftrightarrow W(i, n_Y + 1) = 1 \tag{9}$$

$$\nexists i \in \{1, \ldots, n_X\}, \pi_i = j \neq 0 \quad \Longleftrightarrow W(n_X + 1, j) = 1. \tag{10}$$

We also introduce the following matrix that contains all possible costs (to the p-th power) between pairs of nodes (including unassigned nodes).

Matrix $D_{X,Y}$ is an $(n_X + 1) \times (n_Y + 1)$ matrix whose (i, j) element is:

$$D_{X,Y}(i, j) = \begin{cases} d(x_i, y_j)^p & i \leq n_X, j \leq n_Y, \\ \frac{c^p}{2} & i = n_X + 1, j \leq n_Y, \\ \frac{c^p}{2} & i \leq n_X, j = n_Y + 1, \\ 0 & i = n_X + 1, j = n_Y + 1. \end{cases} \tag{11}$$

The entries for $i \leq n_X$ and $j \leq n_Y$ represent localisation costs (to the p-th power) for assigned nodes. The entries for $i = n_X + 1, j \leq n_Y$ and $j = n_Y + 1, i \leq n_X$ represent

the costs (to the p-th power) for unassigned nodes (miss/false detected nodes).

We also denote the component-wise 1-norm of a matrix $A \in \mathbb{R}^{n_X \times n_Y}$ as [34]:

$$||A|| = \sum_{i=1}^{n_X} \sum_{j=1}^{n_Y} |A(i,j)|. \tag{12}$$

Then we can introduce Lemma 1.

Lemma 1. *For $1 < p < \infty$, a scalar $c > 0$, edge mismatch penalty $\epsilon > 0$ and base metric $d(\cdot,\cdot)$, the graph GOSPA metric $d^{(c,\epsilon)}(\cdot,\cdot)$ in* (2) *between two graphs X and Y can be written as*

$$d^{(c,\epsilon)}(X,Y) = \min_{W \in \mathcal{W}_{X,Y}} \left(\mathrm{tr}\big[D_{X,Y}^{\top} W\big] + e_{X,Y}(W)^p\right)^{1/p} \tag{13}$$

where

$$e_{X,Y}(W)^p = \frac{\epsilon^p}{2} ||A_X W_{1:n_X,1:n_Y} - W_{1:n_X,1:n_Y} A_Y|| \tag{14}$$

and $W_{1:n_X,1:n_Y}$ is the matrix formed by the first n_X rows and the first n_Y columns of matrix W (e.g., removing the last row and column of W).

It should be noted that

$$\mathrm{tr}\big[D_{X,Y}^{\top} W\big] = \sum_{i=1}^{n_X+1} \sum_{j=1}^{n_Y+1} D_{X,Y}(i,j) W(i,j) \tag{15}$$

and

$$\begin{aligned} &||A_X W_{1:n_X,1:n_Y} - W_{1:n_X,1:n_Y} A_Y|| \\ &= \sum_{i=1}^{n_X} \sum_{j=1}^{n_Y} \left| \sum_{k=1}^{n_X} A_X(i,k) W(k,j) - \sum_{k=1}^{n_Y} W(i,k) A_Y(k,j) \right|. \end{aligned} \tag{16}$$

The proof of Lemma 1 is given in Appendix A. Thus, we can write the optimisation problem based on the optimisation over assignment vectors $\pi \in \Pi_{X,Y}$ in (2) in terms of an optimisation problem over assignment matrices $W \in \mathcal{W}_{X,Y}$ in (13).

If we set $\epsilon = 0$ in (13), we obtain $d^{(c,0)}(\cdot,\cdot)$, which corresponds to the GOSPA metric (1) applied to the set of nodes and can be efficiently computed by solving a 2-D assignment problem [32], [35]. This actually provides a lower bound of $d^{(c,\epsilon)}(\cdot,\cdot)$. However, $d^{(c,0)}(\cdot,\cdot)$ is not a metric for graphs as edge information is discarded and therefore the identity property of metrics is not met.

B. LP relaxation

In this section, we relax the constraints of matrix W in Lemma 1 for polynomial computation time with linear programming, and we also show that the result is a metric.

Let $\overline{\mathcal{W}}_{X,Y}$ be the set of matrices of dimension $(n_X+1) \times (n_Y+1)$ with relaxation of the constraints in (7) such that $W \in \overline{\mathcal{W}}_{X,Y}$ satisfies (4), (5), (6) and

$$W(i,j) \geq 0, \, \forall i,j. \tag{17}$$

With the relaxation of the constraints in (7), we can write the metric function $d^{(c,\epsilon)}(X,Y)$ in (13) as an alternative metric function $\bar{d}^{(c,\epsilon)}(X,Y)$, where the optimisation is over W in $\overline{\mathcal{W}}_{X,Y}$ rather than $\mathcal{W}_{X,Y}$. Therefore it follows immediately that $\bar{d}^{(c,\epsilon)}(X,Y) \leq d^{(c,\epsilon)}(X,Y)$.

Proposition 2. *For $1 \leq p \leq \infty$, a scalar $c > 0$, edge mismatch penalty $\epsilon > 0$ and base metric $d(\cdot,\cdot)$, the LP relaxation of metric $d^{(c,\epsilon)(X,Y)}$ in* (13) *is also a metric given by*

$$\bar{d}^{(c,\epsilon)}(X,Y) = \min_{W \in \overline{\mathcal{W}}_{X,Y}} \left(\mathrm{tr}\big[D_{X,Y}^{\top} W\big] + e_{X,Y}(W)^p\right)^{1/p} \tag{18}$$

where $D_{X,Y}$ is given by (11) *and $e_{X,Y}(W)^p$ is given by* (14)*.*

The proof of Proposition 2 is given in Appendix B-A. This proof also implies that the metric in (13) is a metric. In Appendix B-C, we also prove that the metric can be computed using LP. This implies that there are optimisation algorithms that are ensured to converge and that the metric can be computed in polynomial time [36].

For Equation (18) in Proposition 2, set $\overline{\mathcal{W}}_{X,Y}$ is obtained by relaxing the binary constraints of set $\mathcal{W}_{X,Y}$ to $W(i,j) \geq 0, \forall i,j$. With the three other constraints in equation (4), (5), (6) in Section IV-A, the set $\overline{\mathcal{W}}_{X,Y}$ is closed and bounded. Since the metric function in equation (18) is continuous over $\overline{\mathcal{W}}_{X,Y}$, the minimum is achievable [37, Thm. 4.6].

C. Metric decomposition

In this section, we explain how the graph GOSPA metric, as shown in Lemma 1, and the LP graph GOSPA metric decompose into costs for properly assigned nodes, missed and false nodes, and edge mismatch costs.

As explained after Definition 2, the node assignments are determined by the assignment vector π defined in Section II-B. Missed nodes are those nodes in V_X that are not assigned to a node in V_Y according to π. False nodes are those nodes in V_Y that are not assigned to a node in V_X according to π. Then, $D_{X,Y}(i,j)$ represents the following costs:

- A localisation error for assigned nodes if $i \leq n_X$, $j \leq n_Y$.
- A missed node if $i \leq n_x$, $j = n_y + 1$.
- A false node if $i = n_x + 1$, $j \leq n_y$.

We denote the sets of indices (i,j) that belong to each of the previous categories as S_1, S_2, S_3. Then, we have

$$\begin{aligned} d_p^{(c,\epsilon)}(X,Y) = \min_{W \in \mathcal{W}_{X,Y}} \Big(&\mathrm{l}(X,Y,W)^p + \mathrm{m}(X,Y,W)^p \\ &+ \mathrm{f}(X,Y,W)^p + e_{X,Y}(W)^p \Big)^{1/p} \end{aligned} \tag{19}$$

where

$$\mathrm{l}(X,Y,W)^p = \sum_{(i,j) \in S_1} D_{X,Y}(i,j) W(i,j)$$

$$\mathrm{m}(X,Y,W)^p = \frac{c^p}{2} \sum_{(i,j) \in S_2} W(i,j)$$

$$\mathrm{f}(X,Y,W)^p = \frac{c^p}{2} \sum_{(i,j)\in S_3} W(i,j) \tag{20}$$

represent the costs (to the p-th power) for assigned, missed and false nodes given the assignment W. Therefore, we can decompose the metric in terms of these costs (and the edge mismatch cost) once we obtain the optimal assignment W. For the LP metric, we have the same decomposition but with soft assignments.

V. Extensions to other types of graphs

In the previous sections, we have presented the graph GOSPA metric for undirected, unweighted graphs, as the counting procedure to evaluate the graph edge mismatches is based on this type of graph. In Section V-A, we explain that the metric in terms of binary matrices, see Lemma 1 and Proposition 2, can be directly extended to weighted graphs by using the right forms of the adjacency matrices. In Section V-B, we show how the graph GOSPA metric can be extended to directed graphs, weighted and unweighted. In Section V-C, we explain how the graph GOSPA metric can also be applied to graphs without node attributes, resulting in a pseudo-metric.

A. Graph GOSPA metric for weighted undirected graphs

In a weighted undirected graph, the adjacency matrix is symmetric but is not constrained to have elements that are either 0 or 1. Each element now contains a non-zero scalar associated with its edge. For example, we can use the weight between edges to represent the type of bond between two atoms in a molecule. Note that, for the edge mismatch cost in (14), a lack of edge would be equivalent to an edge with zero weight. Therefore, we assume that the edge weights are different from zero so that the identity property of metrics holds for weighted graphs. For weighted graphs (with possible self-loops), we can establish the following proposition.

Proposition 3. *The metrics $d^{(c,\epsilon)}(\cdot,\cdot)$ and $\bar{d}^{(c,\epsilon)}(\cdot,\cdot)$ in* (2) *and* (18)*, respectively, are also metrics for weighted undirected graphs (with possible self-loops).*

The proof is the same as the one in Appendix B, as it is also valid for weighted and undirected graphs. The main difference with undirected, unweighted graphs is that now the adjacency matrix elements are not constrained to be 0 or 1.

The interpretation of the graph GOSPA metric for undirected, weighted graphs is the same as for undirected, unweighted graphs in terms of node attribute errors, missed and false nodes, but differs in the edge mismatch penalty, which now considers the differences in the edge weights. The computational complexity of the LP implementation of the graph GOSPA metric for weighted graphs is similar to the case of unweighted graphs, as there are the same number of variables and constraints.

We use Figure 2 to illustrate how the graph GOSPA metric works on weighted graphs. Figure 2 considers weighted versions of the graphs in Figure 1. As in the examples in Figure 1, we consider $p = 1$, $\epsilon \ll c$, and $\Delta \ll c$, and also that

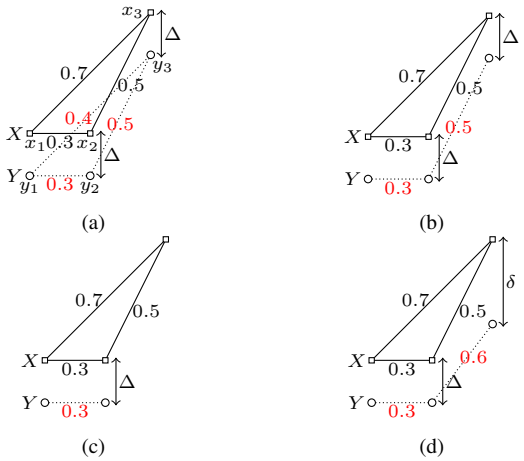


Fig. 2: Example to illustrate the node and edge mismatch costs for weighted graphs, ground truth graph X, and different estimated graphs Y: (a) three properly assigned nodes, 0.3 edge-mismatch penalty; (b) three properly assigned nodes ($\Delta \ll c$) and one missed edge with 0.7 edge-mismatch penalty; (c) two properly assigned nodes, one missed node and 0.6 edge mismatch penalties; (d) Two properly assigned nodes ($\Delta \ll c$) and two unassigned nodes ($\delta \gg c$), 0.9 edge mismatch penalties.

Δ is sufficiently small such that the assignment matrices with respect to Figure 1 do not change. In this case, the localisation errors remain unchanged and we only proceed to calculate the edge mismatch costs.

In Figure 2(a), there is no actual edge mismatch, but there is a weight difference between graph X and graph Y. Therefore, the edge mismatch cost is $e_{X,Y} = \epsilon \times (0.7 - 0.4) = 0.3\epsilon$, and the metric value is $d^{(c,\epsilon)}(X,Y) = 3\Delta + 0.3\epsilon$.

In Figure 2(b), we also have a missed edge between nodes y_1 and y_2 in Y, and the weight of the corresponding edge in X is 0.7, so the edge mismatch error for this case is 0.7ϵ. The metric value for Figure 2(b) is $d^{(c,\epsilon)}(X,Y) = 3\Delta + 0.7\epsilon$.

In Figure 2(c), as we have two half-edge mismatch penalties, and the weights for the missed edges are 0.7 and 0.5. The edge mismatch error is then $e_{X,Y} = \epsilon/2 \times (0.7 + 0.5) = 0.6\epsilon$. The metric value is $d^{(c,\epsilon)}(X,Y) = 2\Delta + c/2 + 0.6\epsilon$.

In Figure 2(d), similarly, there are three half-edge mismatch penalties, and the weight for each edge is 0.7, 0.5, 0.6 respectively. The edge mismatch error is $e_{X,Y} = \epsilon/2 \times (0.7 + 0.5 + 0.6) = 0.9\epsilon$. The metric value is $d^{(c,\epsilon)}(X,Y) = 2\Delta + c + 0.9\epsilon$.

B. Graph GOSPA metric for directed graphs

In a directed unweighted graph, the adjacency matrix has elements that are either 0 or 1, but it is not symmetric. A possible application of directed graphs is modelling the one-way connectivities between users in social networks. In directed weighted graphs, the adjacency matrix has elements that can be any real number and it is not symmetric. Due to the non-symmetric adjacency matrices in directed graphs, the metric for directed graphs must be adapted, as (13) only meets the symmetry property for symmetric A_X and A_Y. The extension of this metric for directed graphs (both weighted and unweighted) is given in the following lemma.

Lemma 4. *For $1 \leq p \leq \infty$, a scalar $c > 0$, edge mismatch penalty $\epsilon > 0$ and base metric $d(\cdot,\cdot)$, the graph GOSPA metric for directed graphs X and Y is*

$$d^{(c,\epsilon)}(X,Y) = \min_{W \in \mathcal{W}_{X,Y}} \left(\mathrm{tr}\left[D_{X,Y}^{\top} W\right] + e_{X,Y}(W)^p\right)^{1/p} \tag{21}$$

where $D_{X,Y}$ is given by (11) *and the edge cost is*

$$e_{X,Y}(W) = \frac{\epsilon^p}{4}\Big(||A_X W_{1:n_X,1:n_Y} - W_{1:n_X,1:n_Y} A_Y|| + ||A_Y W^T_{1:n_X,1:n_Y} - W^T_{1:n_X,1:n_Y} A_X||\Big) \tag{22}$$

The LP lower bound of (21), $\bar{d}^{(c,\epsilon)}(\cdot,\cdot)$ is also a metric and is obtained by minimising over $W \in \bar{W}_{X,Y}$.

Both $d^{(c,\epsilon)}(\cdot,\cdot)$ and $\bar{d}^{(c,\epsilon)}(\cdot,\cdot)$ for directed graphs are symmetric and meet the identity property. The proof of the triangle inequality is given in Appendix B-B. In Appendix B-C we prove that $\bar{d}^{(c,\epsilon)}(\cdot,\cdot)$ for directed graphs is also computable in polynomial time using LP. Note that if A_X and A_Y are symmetric (as in undirected graphs), the edge mismatch cost (22) can be written as in (14). Therefore, (22) is a general edge mismatch cost that can be used to cover directed and undirected graphs, and for undirected graphs, (21) coincides with (13). The computational complexity of the LP implementation of the graph GOSPA metric for directed graphs is higher than the implementation for undirected graphs as there is a higher number of constraints, as explained in Appendix B-C [38].

The decomposition of the graph GOSPA metric for directed graphs into node attribute errors, missed and false nodes are the same as for undirected graphs. The edge mismatch cost is obtained as follows:

- For every two pairs of properly assigned nodes (two nodes in X and two nodes in Y), if a (directed) edge exists in one graph but it does not exist in the other graph, the associated edge mismatch penalty is $\epsilon^p/2$.
- Each (directed) edge in either graph connecting an assigned node and an unassigned node contributes to an edge mismatch penalty $\epsilon^p/4$.

We proceed to use the examples in Figure 3 to explain how the graph GOSPA metric works on directed graphs. We also use the same hyper-parameters $p = 1$ and $\Delta \ll c$.

In Figure 3(a), the edge between nodes x_1 and x_3 in X has the opposite direction to the edge between nodes y_1 and y_3 in Y, which yields one edge mismatch penalty, ϵ. The metric value in this case is $\hat{d}^{(c,\epsilon)}(X,Y) = 3\Delta + \epsilon$.

In Figure 3(b), there is a missed edge between nodes y_1 and y_3 in Y, and this contributes to a half-edge mismatch penalty, $\epsilon/2$. The total metric value is $\hat{d}^{(c,\epsilon)}(X,Y) = 3\Delta + \epsilon/2$.

In Figure 3(c), there is a missed node and two missed edges, and each gives a quarter-edge mismatch penalty, $e_{X,Y} = \epsilon/2$. The metric value for Figure 3(c) is $\hat{d}^{(c,\epsilon)}(X,Y) = 2\Delta + c/2 + \epsilon/2$.

In Figure 3(d), there are two unassigned nodes, two missed edges and one extra edge, and in this case, the edge error is ϵ. The total metric value is $\hat{d}^{(c,\epsilon)}(X,Y) = 2\Delta + c + 3\epsilon/4$.

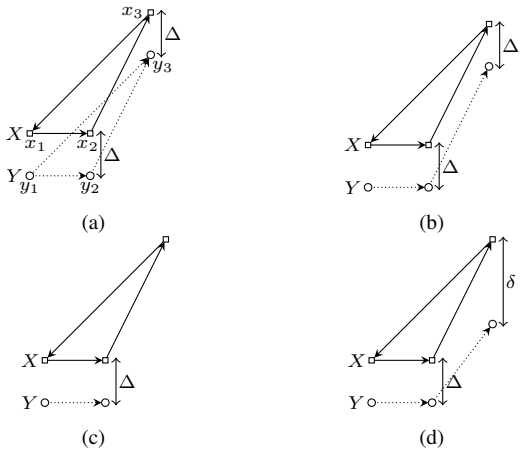


Fig. 3: Example to illustrate the node and edge mismatch costs for directed graphs, ground truth: X and estimated: Y: (a) three properly assigned nodes and one edge mismatch penalty; (b) three properly assigned nodes ($\Delta \ll c$) and half edge mismatch penalty; (c) two properly assigned nodes, one missed node and half edge mismatch penalty; (d) Two properly assigned nodes ($\Delta \ll c$) and two unassigned nodes ($\delta \gg c$), one edge mismatch penalty.

C. Graphs without node attributes

There are applications in which graphs do not have node attributes with physical meaning, or these attributes are not available. In this case, one can denote the set of nodes with integers, such that $V_X = \{1,...,n_X\}$ and $V_Y = \{1,...,n_Y\}$. Therefore, the localisation part of the graph GOSPA metric does not have a physical meaning either. In this case, the base metric can be set to zero $d(\cdot,\cdot) = 0$ and the following lemma holds.

Lemma 5. *If the base distance is $d(\cdot,\cdot) = 0$, the function in* (18) *becomes*

$$\tilde{d}^{(c,\epsilon)}(X,Y) = \min_{W \in \overline{\mathcal{W}}_{X,Y}} \left(\frac{c^p}{2}\left(\sum_{i=1}^{n_X} W(i,n_Y+1) + \sum_{j=1}^{n_Y} W(n_X+1,j)\right) + e_{X,Y}(W)^p\right)^{1/p} \tag{23}$$

where W satisfies (4), (5), (6) *and* (17)*, and $e_{X,Y}$ is given in* (14) *for undirected graphs and* (22) *for directed graphs.*

The function $\tilde{d}^{(c,\epsilon)}(\cdot,\cdot)$ is a pseudometric for graphs [17], which implies:

1) *$\tilde{d}^{(c,\epsilon)}(X,X) = 0$;*
2) *(symmetry) $\tilde{d}^{(c,\epsilon)}(X,Y) = \tilde{d}^{(c,\epsilon)}(Y,X)$;*
3) *(triangle inequality) $\tilde{d}^{(c,\epsilon)}(X,Y) \leq \tilde{d}^{(c,\epsilon)}(X,Z) + \tilde{d}^{(c,\epsilon)}(Z,Y)$.*

The proof of Lemma 5 is given in Appendix C and also holds for the non-relaxed version of the graph GOSPA metric. Additionally, it is direct to check that $\tilde{d}^{(c,\epsilon)}(X,Y) = 0$ if and only if X and Y are isomorphic. That is, graphs X and Y are isomorphic if and only if they are of the same size and there is a permutation matrix P such that $A_X P = P A_Y$ [29].

The pseudometric $\tilde{d}^{(c,\epsilon)}(\cdot,\cdot)$ then defines an equivalence relation in Ω such that $X \sim Y$ if $\tilde{d}^{(c,\epsilon)}(X,Y) = 0$ [17]. Then, the equivalence class of an element (graph) X, denoted

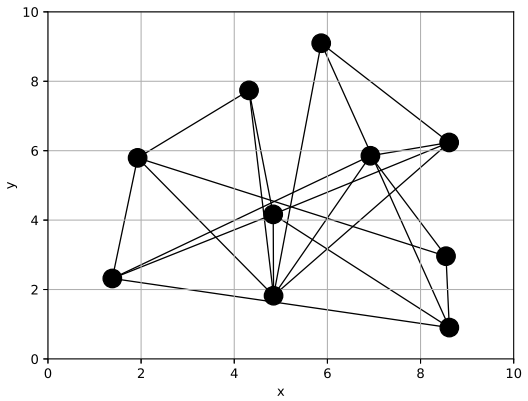


Fig. 4: Randomly generated ground truth graph X.

by $[X]$, is the set of elements in Ω equivalent to X such that $[X] = \{Y \in \Omega : Y \sim X\}$, which is the set of graphs Y that are isomorphic to X. The set of equivalence classes of isomorphic graphs is referred to as quotient space $\Omega \setminus \sim$, such that $[X] \in \Omega \setminus \sim$ [17].

Lemma 6. *The distance $\tilde{\rho}^{(c,\epsilon)}(\cdot,\cdot)$ on the quotient space $\Omega \setminus \sim$ (set of equivalence classes of isomorphic graphs) defined as*

$$\tilde{\rho}^{(c,\epsilon)}([X],[Y]) = \tilde{d}^{(c,\epsilon)}(X,Y) \tag{24}$$

where X and Y are two elements of $[X]$ and $[Y]$, respectively, is a metric on $\Omega \setminus \sim$ [17].

We now proceed to write the pseudo-metric $\tilde{d}^{(c,\epsilon)}(\cdot,\cdot)$ values for the graphs in Figure 1, by removing their node attributes. In Figure 1(a), the two graphs are isomorphic so $\tilde{d}^{(c,\epsilon)}(X,Y) = 0$. In Figure 1(b), there is only one edge mismatch cost ϵ, therefore, $\tilde{d}^{(c,\epsilon)}(X,Y) = \epsilon$. In Figure 1(c), the graphs have different sizes, which contributes to a cardinality error $c/2$, and $\tilde{d}^{(c,\epsilon)}(X,Y) = c/2 + \epsilon$. In Figure 1(d), as there are no node attributes in both graphs, the two graphs X and Y are the same as the graphs in Figure 1(b), therefore, they have the same metric value, $\tilde{d}^{(c,\epsilon)}(X,Y) = \epsilon$.

VI. Experimental Results

This section demonstrates the properties of the graph GOSPA metric by evaluating it on simulated undirected graphs (Section VI-A) and simulated directed graphs (Section VI-B). We also demonstrate the application of the graph GOSPA metric on a real dataset with molecules (Section VI-C). We compare the proposed metric with the MCS distance [24, Thm. 1], GED [19] and GCD [30, Eq. (2)]. The MCS distance between two graphs X and Y is $n_X + n_Y - 2n_c$, where n_c represents the number of nodes in the largest common subgraph. In this section, we use the LP implementation of the graph GOSPA metric[1] and refer to it simply as graph GOSPA metric and the solver we use for LP is the interior-point method [36].

[1]Matlab and Python implementations of the graph GOSPA metric are available via Github: https://github.com/JinhaoGu/The-graph-GOSPA-metric.

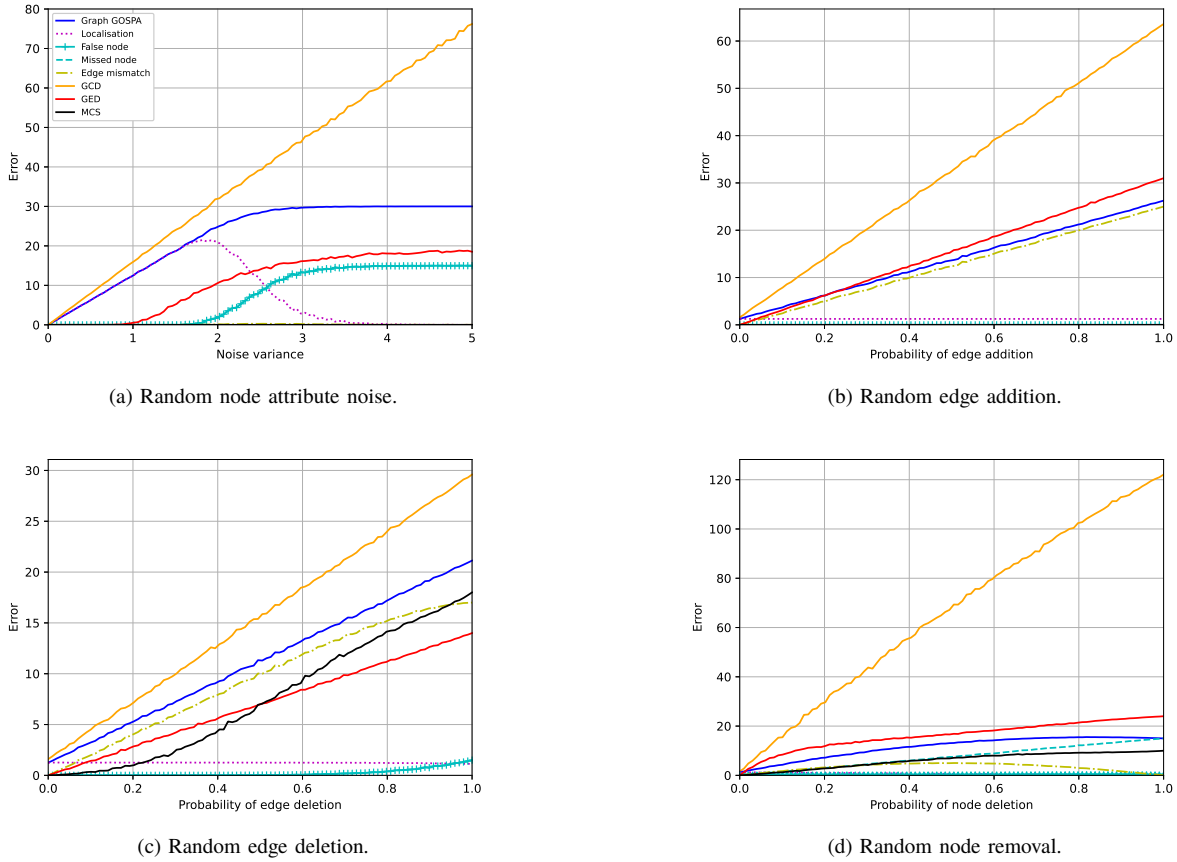
To use the GCD with graphs of different sizes, we use the procedure in [29] by adding dummy nodes to the smallest graph so that graphs have the same size. The dummy nodes have zero attribute value and are well-separated from any other nodes in the graphs. We have followed the publicly available code in [30] to write our own version of the GCD distance [30, eq. (2)], directly optimising over the doubly stochastic matrix that represents the node assignments. The MCS distance is computed by using the built-in function in RDkit to compute the MCS [39]. For the MCS distance, we do not use node attributes in the simulations, while in the dataset with molecules, we use the atom type as the node attribute. The GED is computed with the built-in function in NetworkX [40]. For the GED, we use a node match function such that, if the distance between two nodes is smaller than c, which is set to 3, they are treated as matched nodes and node deletion and insertion cost are set to 1. The edge substitution cost is set to 0 and the edge deletion and insertion cost are set to 1 as well. We set the hyper-parameters $p = 1$, $\lambda = 1$ for the GCD, and $p = 1$, $c = 3$, $\epsilon = 1$ for the graph GOSPA metric. We also use the Euclidean distance as the base metric for both GCD and graph GOSPA metric.

A. Simulation results for undirected graphs

In this section, we consider random graphs generated from a $G(n,p)$ Erdos-Renyi model [41], where n represents the number of nodes in the graph, and p represents the probability of an edge existing between two nodes. The nodes of the graphs have 2-D node states. These graphs can represent the locations of some users and the connectivities with other users.

We generate a ground truth graph X using the Erdos-Renyi model by setting $n = 10$ and $p = 0.4$ with the GraSP toolbox [42], as shown in Figure 4, and the location elements of X are generated by using the neato layout engine with parameter $maxiter = 5000$ and $start = 7$ in Graphviz toolkit [43]. We then generate a modified version of this graph, denoted as Y, which varies across different scenarios. Each of these modified graphs Y represents a different type of change or perturbation in the original graph X, such as adding noise to node attributes (Scenario 1), adding or removing edges (Scenario 2 and 3), or removing nodes (Scenario 4). By comparing X and Y, we can assess how well the algorithm performs in detecting and characterizing these changes in the graph structure.

We compute the average distances in each of the previous scenarios via Monte Carlo simulation with 1000 runs. In each Monte Carlo run for Scenario 1, we draw a graph Y by adding independent zero-mean Gaussian random variables with covariance matrix $\sigma^2 I_2$ to the attributes of X. The average errors for different values of σ^2 are shown in Figure 5(a). With increasing noise variance in node attributes, the GCD value keeps growing, which is caused by the growth of localisation error in GCD. In the graph GOSPA metric, when the localisation error is too large to assign two nodes to each other, the missed node error and false node error will increase until all nodes are missed/false nodes. Furthermore, the contribution of missed/false error is only determined by the quantity of missed/false nodes, therefore the graph



(a) Random node attribute noise.

(b) Random edge addition.

(c) Random edge deletion.

(d) Random node removal.

Fig. 5: Plots of average graph GOSPA metric, GCD, GED and MCS comparing the ground truth graph with (a) graphs with random Gaussian noise with increasing noise variance in node attributes; (b) graphs with random edge addition in adjacency matrix and Gaussian noise in node attributes; (c) graphs with random edge removal in adjacency matrix and Gaussian noise in node attributes; (d) graphs with different sizes and Gaussian noise in node attributes.

GOSPA saturates, when there is a large noise variance in node attributes. For the GED, as the noise level increases, the error grows until it reaches its maximum, which is determined by the number of nodes in the corresponding graphs, which explains the saturation area in Figure 5(a).

In Scenario 2, we obtain graph Y by adding edges in the adjacency matrix of ground truth X with probability q and we also add random noise with covariance matrix $0.1I_2$ to the node attributes. The average errors for different q are shown in Figure 5(b). In Figure 5(b), the values of GED, GCD and graph GOSPA increase, where the error grows slowly as the probability of adding extra edges increases. We can also notice that the edge mismatch cost increases linearly with the increasing probability of adding edges, which plays a predominant role in determining the metric value of the graph using the GOSPA metric. For the MCS distance, as we are adding edges to the ground truth graph in Scenario 2, the maximum common subgraph between the target graph and the ground truth graph is actually the ground truth itself, therefore we get zero MCS distance error.

The graph Y in Scenario 3 is generated by randomly deleting edges in ground truth X with probability q, and we also add random noise with covariance matrix $0.1I_2$ to the node attributes. In Figure 5(c), we notice that the errors for all the considered methods are similar to the values in Figure 5(b), except for the MCS distance. This is due to the fact that the size of maximum common subgraphs is getting smaller with more edges being deleted from the prototype graph X. Thus, the error increases in the MCS distance as the probability of edge deletion grows gradually.

In Scenario 4, we draw graph Y by randomly removing the nodes in the ground truth graph X with a certain probability of node deletion. We also add Gaussian noise with covariance matrix $0.1I_2$ to node attributes. In Figure 5(d), we see that with the increasing probability of removing nodes, the number of unassigned nodes in graph GOSPA rises, missed nodes specifically, and the localisation error for assigned nodes decreases. The edge mismatch cost in this example grows first until the probability of node deletion $q = 0.5$, then the edge mismatch cost decreases. This is because the graph GOSPA metric does not consider the edges connected to unassigned nodes. Thus, as there are more unassigned nodes with a higher probability of node deletion, the edge mismatch cost decreases. For the other graph distances, GCD, GED and the MCS distance, their error also increases with the probability of node deletion. For GCD, the error grows linearly, while for the other two distances the error grows similarly as in the graph GOSPA metric.

Computational complexity evaluation: In this section, we generate random undirected graphs using the same settings

TABLE I: Computational time over undirected graphs (CPU time in seconds).

(n_X, n_Y)	Graph GOSPA	GCD	GED	MCS
(10,10)	0.02	0.03	58.26	5.81
(20,20)	0.06	0.20	>3000	>3000
(40,40)	0.97	8.78	>3000	-
(60,60)	6.71	102.70	>3000	-
(80,80)	45.29	654.69	>3000	-
(100,100)	131.40	2550.10	>3000	-

TABLE II: Memory usage over undirected graphs (in MBytes).

(n_X, n_Y)	Graph GOSPA	GCD	GED	MCS
(10,10)	0.49	0.77	0.86	0.90
(20,20)	4.81	2.21	7.11	4968.46
(40,40)	71.07	14.78	153.28	–
(60,60)	353.34	46.36	1084.15	–
(80,80)	1111.18	108.27	4674.12	–
(100,100)	2660.09	244.50	14272.59	–

as in Section VI-A, but we change the number of nodes in the generated graphs to $n_X = n_Y \in \{10, 20, 40, 60, 80, 100\}$. Then, we calculate the time and memory usage of the implementations of the graph GOSPA metric for undirected graphs, GCD, GED and the MCS distance.

In Table I, we can see that the graph GOSPA metric is the one with the lowest computational time due to the LP implementation. The second fastest is the GCD. The GED takes longer than 3000 seconds (used as a time limit) to compute even when we consider graphs with 20 nodes. The MCS also takes more than 3000 seconds to compute graphs with 20 nodes. The considered MCS implementation did not work for graphs with more than 40 nodes. The memory usage for each metric has been calculated using the *tracemalloc* library in Python and is shown in Table II. The graph GOSPA metric requires more memory than GCD to store the constraints, but less memory than GED.

B. Simulation results for directed graphs

In this section, we compare the graph GOSPA metric with the GCD and the GED over directed graphs. In this comparison, we use the symmetrised GCD for directed graphs in [29, Thm. 2], which requires solving two optimisation problems. We consider the same graphs used in Scenario 1 in Section VI-A, but setting zeros in the lower triangle of the adjacency matrix to make the graphs directed. The metric values for increasing noise variance σ^2 in the attributes are shown in Figure 6.

The graph GOSPA metric error in Figure 6 is similar to the undirected case, only grows slower than the graph GOSPA error in Figure 5(a), since the edge penalties for edges connected to unassigned nodes are halved. The GCD in Figure 6, is also similar but twice the value due to how the extension to directed graphs is done in [29, Thm. 2]. The GED remains the same as the undirected case, although we removed one direction in the compared graphs.

Computational complexity evaluation: In this section, we use the same graphs used for the computational analysis in Section VI-A, but we replace the lower triangle of the adjacency matrices with zeros to obtain directed graphs. We also compute the time and memory usage for directed graphs. The simulation results are summarised in Table III and IV.

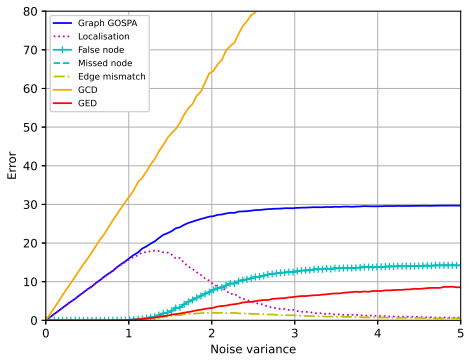


Fig. 6: Plots of average graph GOSPA metric, GCD and GED comparing the directed ground truth graph with directed graphs with increasing random noise variance in node attributes.

TABLE III: Computational time over directed graphs (CPU time in seconds).

(n_X, n_Y)	Graph GOSPA	GCD	GED
(10,10)	0.01	0.5	58.34
(20,20)	0.05	0.37	>3000
(40,40)	0.93	16.44	>3000
(60,60)	6.37	203.28	>3000
(80,80)	26.41	1269.95	>3000
(100,100)	84.24	5119.86	>3000

As with undirected graphs, the graph GOSPA metric is the one with the lowest computational complexity, though it has higher memory usage than the GCD.

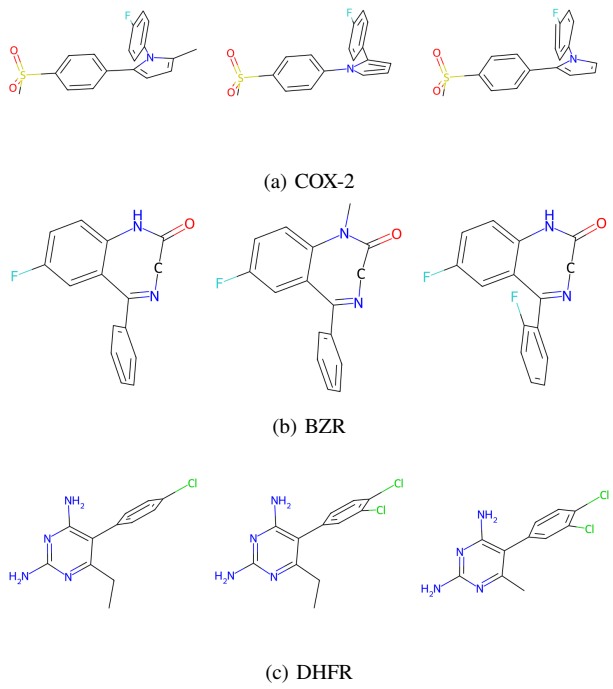
C. Experiments on a molecular dataset

In this section, we illustrate the use of the graph GOSPA metric on the molecular dataset in the supplementary material of [44]. This molecular dataset contains five collections of different types of compounds and we use three collections of them: cyclooxygenase-2 inhibitors (COX-2, 467 graphs), benzodiazepine receptor ligands (BZR, 405 graphs), dihydrofolate reductase inhibitors (DHFR, 756 graphs). All collections have molecules with more than 20 heavy atoms (C, N, O, Cl, etc.). For simplicity, we choose three molecules from each collection, and nine molecules in total. Within each collection, the molecular graphs have similar structures as those shown in Figure 7.

In this experiment, we use the atomic number of each element (or node) in the molecular graphs as the node attributes. In addition, as there are different bonds within molecules, the molecular graphs are weighted graphs, in which the entries of the adjacency matrices can be 0, 1, 2, to represent different types of bonds. That is, weight 0 means there is no bond, weight 1 represents a single bond, and weight 2 represents a

TABLE IV: Memory usage over directed graphs (in MBytes).

(n_X, n_Y)	Graph GOSPA	GCD	GED
(10,10)	0.79	0.77	0.83
(20,20)	10.38	2.26	11.44
(40,40)	158.77	14.83	166.14
(60,60)	787.80	46.40	1107.38
(80,80)	2473.88	108.31	4703.80
(100,100)	6008.09	244.53	14315.83

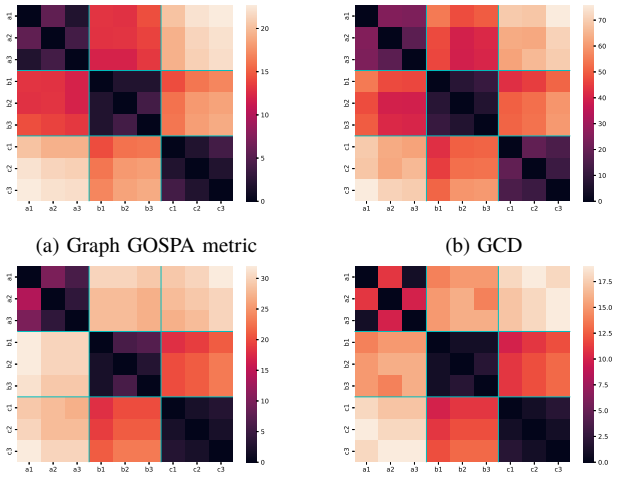


(a) COX-2

(b) BZR

(c) DHFR

Fig. 7: Plots of three molecular graph samples from the following collections: (a) COX-2; (b) BZR; (c) DHFR.



(a) Graph GOSPA metric

(b) GCD

(c) GED

(d) MCS distance

Fig. 8: Illustration of the distance matrices computed using (a) graph GOSPA, (b) GED, (c) GED and (d) GCD for molecules from three different collections. Each collection's metric value is separated by a solid line. Each entry of the x and y axes is named as 'ai', 'bi' or 'ci' where i is either 1, 2 or 3, and represents the i-th molecule in the 'a', 'b' or 'c' family in Figure 7.

double bond. We set the hyperparameters $p = 1$, $\lambda = 1$ for the GCD distance, $p = 1$, $c = 3$, $\epsilon = 1$ for the graph GOSPA metric. Both the GCD and the graph GOSPA metric use the Euclidean distance as the base metric.

In Figure 8, we compare four different graph distances on molecules from three different types of collections. The results of graph GOSPA, GCD, GED and MCS distance are shown in Figure 8(a), 8(b), 8(c) and 8(d) respectively. In all four distance matrices, different types of molecules are separated by a solid line. This implies that, in the diagonal blocks, we are comparing the molecules from the same family, whose distances should in principle be lower than the distances of molecules between different families. All distances manage to keep the property, though the MCS distance has more problems with the COX-2 family.

TABLE V: Average distance between molecule COX-2,1-1 and 100 molecules from each family.

	COX-2	BZR	DHFR
Graph GOSPA	11.78	19.15	23.49
GCD	51.58	52.17	70.27
GED	21.42	32.57	37.65
MCS	15.54	17.41	23.15

We now proceed to compare the molecule COX-2,1-1, considered as the ground truth graph X, with molecules from other collections. We use 100 molecules from each collection and compute the average distance value for every collection. In Table V, we show that, in graph GOSPA, the average distance between COX-2,1-1 and molecules in COX-2, is lower than the average distances with molecules in BZR and DHFR, as the structure becomes more diverse to the reference molecule COX-2,1-1. In this experiment, all considered distances work well, though the GCD performs a bit worse for the molecules in family COX-2.

A feature of the graph GOSPA metric is that we can analyse the results using the metric decomposition. To this end, in Table VI, we show the decomposition of the graph GOSPA in the previous experiment that considered molecule COX-2,1-1 as the ground truth graph X. In the experiment, we use the atom number as the node attribute, the localisation error can indicate the difference between atom types of two molecules. For example, suppose two molecules have similar structures, and almost all nodes within two molecular graphs have been assigned. In that case, the localisation error will show the difference in the atom types between two assigned nodes. The false node error shows the part that exists in the compared molecule but does not exist in the ground truth molecule. Similarly, the miss node error tells us some substructures exist in the ground truth but do not exist in the compared molecule. As for the edge mismatch cost, we can see the difference in molecule structures. The difference in bonds types, whether there is a bond between two assigned atoms (nodes) or if the bond type is different, they all contribute to the edge mismatch cost.

In COX-2 collection, the false node error and edge mismatch error are predominant in the metric value, which indicates that the other molecules in this family have substructures that do not exist in the ground truth molecule COX-2,1-1. In BZR and DHFR, the metric value is mostly affected by the missed node and edge mismatch cost, which suggests that there are differences in the chemical structure.

VII. Conclusion

In this paper, we have proposed the graph GOSPA metric, which is based on obtaining the optimal assignment matrix between nodes, to measure the discrepancy between two graphs of different sizes. The graph GOSPA metric meets the

TABLE VI: Decomposition of the average graph GOSPA error between molecule COX-2,1-1 and 100 molecules from each family.

	COX-2	BZR	DHFR
Localisation	2.04	3.36	4.75
False node	4.94	3.45	6.74
Miss node	0.24	4.16	3.70
Edge mismatch	4.56	8.19	8.30

metric properties (identity, symmetry and triangle inequality) for graphs with attributes and different sizes. The proposed metric has an intuitive decomposition that captures the node attribute errors for properly assigned nodes, missed and false node costs, and edge mismatch costs.

For metric computation, we have proposed a lower bound to the graph GOSPA metric that is also a metric and can be computed using linear programming with polynomial time complexity. We have also extended this metric to weighted and directed graphs.

A possible line of future work is the distributed implementation of the metric using for example the alternating direction of multipliers methods (ADMM) [30], [45] or other distributed optimisation methods [46], [47]. We are also going to explore the use of the graph GOSPA metric in different applications such as network science and chemistry in future work.

References

[1] R. J. Trudeau, *Introduction to Graph Theory*. Dover Publications, Inc., 1993.

[2] Y. Wu, D. Lian, Y. Xu, L. Wu, and E. Chen, "Graph convolutional networks with Markov random field reasoning for social spammer detection," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 01, 2020, pp. 1054–1061.

[3] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia, "Graph networks as learnable physics engines for inference and control," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4470–4479.

[4] P. Battaglia, R. Pascanu, M. Lai, D. Jimenez Rezende *et al.*, "Interaction networks for learning about objects, relations and physics," *Advances in neural information processing systems*, vol. 29, 2016.

[5] V. Kvasnička, J. Pospíchal, and V. Baláž, "Reaction and chemical distances and reaction graphs," *Theoretica chimica acta*, vol. 79, no. 1, pp. 65–79, 1991.

[6] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro, "Convolutional neural network architectures for signals supported on graphs," *IEEE Transactions on Signal Processing*, vol. 67, no. 4, pp. 1034–1049, 2019.

[7] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, "Connecting the dots: Identifying network structure via graph signal processing," *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 16–43, 2019.

[8] A. Ortega, *Introduction to Graph Signal Processing*. Cambridge University Press, 2022.

[9] J. Domingos and J. M. Moura, "Graph Fourier transform: A stable approximation," *IEEE Transactions on Signal Processing*, vol. 68, pp. 4422–4437, 2020.

[10] S. Shit, R. Koner, B. Wittmann, J. Paetzold, I. Ezhov, H. Li, J. Pan, S. Sharifzadeh, G. Kaissis, V. Tresp *et al.*, "Relationformer: A unified framework for image-to-graph generation," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXVII*. Springer, 2022, pp. 422–439.

[11] L. Lovász, *Large networks and graph limits*. American Mathematical Soc., 2012, vol. 60.

[12] H. Lee, M. K. Chung, H. Kang, B.-N. Kim, and D. S. Lee, "Computing the shape of brain networks using graph filtration and gromov-hausdorff metric," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2011: 14th International Conference, Toronto, Canada, September 18-22, 2011, Proceedings, Part II 14*. Springer, 2011, pp. 302–309.

[13] D. Burago, Y. Burago, S. Ivanov *et al.*, *A course in metric geometry*. American Mathematical Society Providence, 2001, vol. 33.

[14] F. Mémoli, "On the use of Gromov-Hausdorff Distances for Shape Comparison," in *Eurographics Symposium on Point-Based Graphics*. The Eurographics Association, 2007.

[15] H. Petric Maretic, M. El Gheche, G. Chierchia, and P. Frossard, "GOT: An optimal transport framework for graph comparison," in *Advances in Neural Information Processing Systems*, 2019.

[16] T. M. Apostol, *Mathematical Analysis*. Addison Wesley, 1974.

[17] N. R. Howes, *Modern analysis and topology*. Springer Science & Business Media, 2012.

[18] A. Sanfeliu and K.-S. Fu, "A distance measure between attributed relational graphs for pattern recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 3, pp. 353–362, 1983.

[19] A. Fischer, C. Y. Suen, V. Frinken, K. Riesen, and H. Bunke, "Approximation of graph edit distance based on Hausdorff matching," *Pattern Recognition*, vol. 48, no. 2, pp. 331–343, 2015.

[20] D. Justice and A. Hero, "A binary linear programming formulation of the graph edit distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 8, pp. 1200–1214, 2006.

[21] K. Riesen and H. Bunke, "Approximate graph edit distance computation by means of bipartite graph matching," *Image and Vision Computing*, vol. 27, no. 7, pp. 950–959, 2009.

[22] K. Riesen, M. Ferrer, R. Dornberger, and H. Bunke, "Greedy graph edit distance," in *Proceedings of the 11th International Conference on Machine Learning and Data Mining in Pattern Recognition*, 2015, pp. 3–16.

[23] S. Saadah, M. F. E. Saputro, and K. R. S. Wiharja, "Graph edit distance optimized with greedy algorithm for similarity search in business process graphs," in *2020 International Conference on Data Science and Its Applications (ICoDSA)*, 2020, pp. 1–7.

[24] H. Bunke, "On a relation between graph edit distance and maximum common subgraph," *Pattern recognition letters*, vol. 18, no. 8, pp. 689–694, 1997.

[25] H. Bunke and K. Shearer, "A graph distance metric based on the maximal common subgraph," *Pattern recognition letters*, vol. 19, no. 3-4, pp. 255–259, 1998.

[26] M. El-Kebir, J. Heringa, and G. W. Klau, "Natalie 2.0: Sparse global network alignment as a special case of quadratic assignment," *Algorithms*, vol. 8, no. 4, pp. 1035–1051, 2015.

[27] V. Lyzinski, D. E. Fishkind, M. Fiori, J. T. Vogelstein, C. E. Priebe, and G. Sapiro, "Graph matching: Relax at your own risk," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 1, pp. 60–73, 2016.

[28] G. Chartrand, G. Kubicki, and M. Schultz, "Graph similarity and distance in graphs," *Aequationes Mathematicae*, vol. 55, no. 1, pp. 129–145, 1998.

[29] J. Bento and S. Ioannidis, "A family of tractable graph metrics," *Applied Network Science*, vol. 4, no. 107, 2019.

[30] A. Moharrer, J. Gao, S. Wang, J. Bento, and S. Ioannidis, "Massively distributed graph distances," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 6, pp. 667–683, 2020.

[31] J. Yan, J. Wang, H. Zha, X. Yang, and S. Chu, "Consistency-driven alternating optimization for multigraph matching: A unified approach," *IEEE Transactions on Image Processing*, vol. 24, no. 3, pp. 994–1009, 2015.

[32] A. S. Rahmathullah, A. F. García-Fernández, and L. Svensson, "Generalized optimal sub-pattern assignment metric," in *20th International Conference on Information Fusion*, 2017, pp. 1–8.

[33] A. F. García-Fernández, A. S. Rahmathullah, and L. Svensson, "A metric on the space of finite sets of trajectories for evaluation of multi-target tracking algorithms," *IEEE Transactions on Signal Processing*, vol. 68, pp. 3917–3928, 2020.

[34] X.-D. Zhang, *Matrix analysis and applications*. Cambridge University Press, 2017.

[35] D. F. Crouse, "On implementing 2D rectangular assignment algorithms," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 4, pp. 1679–1696, August 2016.

[36] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming*. Springer Publishing Company, Incorporated, 2015.

[37] W. Rudin, *Principles of mathematical analysis*. New York: McGraw Hill, 1976.

[38] L. Khachiyan, "Polynomial algorithms in linear programming," *USSR Computational Mathematics and Mathematical Physics*, vol. 20, no. 1, pp. 53–72, 1980.

[39] "RDKit: Open-source cheminformatics." 2010. [Online]. Available: https://www.rdkit.org/

[40] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11–15.

[41] P. Erdős, A. Rényi *et al.*, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, vol. 5, no. 1, pp. 17–60, 1960.

[42] B. Girault, S. S. Narayanan, A. Ortega, P. Gonçalves, and E. Fleury, "Grasp: A Matlab toolbox for graph signal processing," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 6574–6575.

[43] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull, "Graphviz— open source graph drawing tools," in *Graph Drawing*, P. Mutzel, M. Jünger, and S. Leipert, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 483–484.

[44] J. J. Sutherland, L. A. O'Brien, and D. F. Weaver, "Spline-fitting with a genetic algorithm: A method for developing classification structure-activity relationships," *Journal of Chemical Information and Computer Sciences*, vol. 43, no. 6, pp. 1906–1915, 2003.

[45] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[46] Z. Jiang, K. Mukherjee, and S. Sarkar, "Convergence and noise effect analysis for generalized gossip-based distributed optimization," in *2017 American Control Conference (ACC)*, 2017, pp. 4353–4358.

[47] V. P. Chellapandi, A. Upadhyay, A. Hashemi, and S. H. Zak, "On the convergence of decentralized federated learning under imperfect information sharing," *IEEE Control Systems Letters*, vol. 7, pp. 2982–2987, 2023.

[48] J. Bento and J. J. Zhu, "A metric for sets of trajectories that is practical and mathematically consistent," 2018. [Online]. Available: https://arxiv.org/abs/1601.03094

Supplementary material of "Graph GOSPA metric: a metric to measure the discrepancy between graphs of different sizes"

Appendix A

In this appendix, we prove that (2) can be written as in (13) in Lemma 1. We know from [32] that the localisation error, missed and false node costs can be written in matrix form as $\mathrm{tr}\big[D_{X,Y}^{\top}W\big]$. We proceed to prove the edge mismatch penalty (14). We know that the assignment vector can be written as a binary matrix meeting (4)-(7). We note that the following sum in (16) can be written as

$$\sum_{k=1}^{n_X} A_X(i,k)W(k,j) = \begin{cases} A_X(i,k_x) & \exists k_x : W(k_x,j)=1, \\ 0 & W(n_X+1,j)=1 \end{cases} \tag{25}$$

where the first line represents that node k_x in X is assigned to node j in Y, and the second line means node j in Y is unassigned. Equivalently, the following sum in (16) can be written as

$$\sum_{k=1}^{n_Y} W(i,k)A_Y(k,j) = \begin{cases} A_Y(k_y,j) & \exists k_y : W(i,k_y)=1, \\ 0 & W(i,n_Y+1)=1 \end{cases} \tag{26}$$

where the first line represents that node k_y in Y is assigned to node i in X, and the second line means node i in X is unassigned.

In (16) we then go through each node i in X and each node j in Y. For the node j in Y, we find the node k_x in X assigned to it (if it exists), using (25). For the node i in X, we find the node k_y in Y assigned to it (if it exists), using (26). If both i and j are properly assigned, there is an edge penalty if one of the graphs has the corresponding edge but the other graph does not. That is, either there is an edge $\{i,k_x\}$ in X or $\{j,k_y\}$ in Y, but not both simultaneously. These types of edges are penalised twice in the above sum, as we will obtain the same result when $i=k_x$ and $j=k_y$ i.e., we are evaluating the node on the other side of the edge. This corresponds to the first line of (3).

If node j in Y is unassigned, there is an edge penalty if node i is assigned to k_y and there is an edge linking k_y to j in Y. That is, each edge in which one of the ends is assigned but the other end is unassigned, contributes to a half-edge error (as these errors are not repeated in the above sum). This corresponds to the third line of (3).

For the case that node i in X is unassigned, it is identical to the previous case that node j in Y is unassigned, since (16) is symmetric. This corresponds to the second line in (3). Therefore, we have proved that (2) can be written in the form of (13).

Appendix B

In this appendix, we prove the triangle inequality property of the graph metric in (18) for both undirected and directed graphs, as the identity and symmetry properties of the metric are immediate from the definition. The proof in this section follows the proof in [33], and we use the same notation as well.

A. Proof of the triangle inequality property for undirected graphs

We denote the objective function in (18), $\bar{d}^{(c,\epsilon)}(X,Y,W)$, as a function of the assignment matrix W. The proof of the triangle inequality is as follows.

Let $W^*_{X,Y} \in \overline{\mathcal{W}}_{X,Y}$, $W^*_{X,Z} \in \overline{\mathcal{W}}_{X,Z}$, $W^*_{Z,Y} \in \overline{\mathcal{W}}_{Z,Y}$ be the weight matrices which minimise $\bar{d}^{(c,\epsilon)}(X,Y,W_{X,Y})$, $\bar{d}^{(c,\epsilon)}(X,Z,W_{X,Z})$ and $\bar{d}^{(c,\epsilon)}(Z,Y,W_{Z,Y})$, respectively. Given the matrices $W^*_{X,Z}$ and $W^*_{Z,Y}$, we define a matrix $W_{X,Y}$ that meets

$$W_{X,Y}(i,j) = \begin{cases} 1-\sum_{j=1}^{n_Y} W_{X,Y}(i,j) & j=n_Y+1, i=1,\dots,n_X \\ 1-\sum_{i=1}^{n_X} W_{X,Y}(i,j) & i=n_X+1, j=1,\dots,n_Y \\ 0 & i=n_X+1, j=n_Y+1 \\ \sum_{l=1}^{n_Z} W^*_{X,Z}(i,l)W^*_{Z,Y}(l,j) & \text{otherwise.} \end{cases} \tag{27}$$

Then we show that

$$\bar{d}^{(c,\epsilon)}(X,Y,W_{X,Y}) \le \bar{d}^{(c,\epsilon)}(X,Z) + \bar{d}^{(c,\epsilon)}(Z,Y). \tag{28}$$

It is met that $\bar{d}^{(c,\epsilon)}(X,Y) \le \bar{d}^{(c,\epsilon)}(X,Y,W_{X,Y})$, therefore, (28) implies that the triangle inequality holds

$$\bar{d}^{(c,\epsilon)}(X,Y) \le \bar{d}^{(c,\epsilon)}(X,Z) + \bar{d}^{(c,\epsilon)}(Z,Y) \tag{29}$$

More generally, we show that for any $W_{X,Y} \in \overline{\mathcal{W}}_{X,Y}$, $W_{X,Z} \in \overline{\mathcal{W}}_{X,Z}$, $W_{Z,Y} \in \overline{\mathcal{W}}_{Z,Y}$ generated from (27), the following inequality holds:

$$\bar{d}^{(c,\epsilon)}(X,Y,W_{X,Y}) \le \bar{d}^{(c,\epsilon)}(X,Z,W_{X,Z}) + \bar{d}^{(c,\epsilon)}(Z,Y,W_{Z,Y}). \tag{30}$$

To show that (30) holds, we first obtain inequalities for the terms corresponding to $\mathrm{tr}\big[D_{X,Y}^{\top}W\big]$ and to the edge mismatch costs separately. The inequality for $\mathrm{tr}\big[D_{X,Y}^{\top}W\big]$ is given by (46) in [33]. In the next subsection, we will prove the inequality of the edge mismatch: $e_{X,Y}(W_{X,Y})^p \le e_{X,Z}(W_{X,Z})^p + e_{Z,Y}(W_{Z,Y})^p$.

Based on these two inequalities, the triangle inequality proof is finalised by using the Minkowski inequality. The steps are similar to those from Eq. (48) to (52) in [33] and are not included here for brevity. Therefore, we just need to prove the following edge mismatch cost inequality.

1) Edge mismatch cost inequality: For the edge mismatch cost $e_{X,Y}(W_{X,Y})$, we show that

$$\begin{aligned} e^{(\epsilon=2,p=1)}_{X,Y}(W_{X,Y}) &= \sum_{i=1}^{n_X}\sum_{j=1}^{n_Y}\left|\sum_{k=1}^{n_X} A_X(i,k)W_{X,Y}(k,j) - \sum_{k=1}^{n_Y} W_{X,Y}(i,k)A_Y(k,j)\right| \\ &\le \sum_{i=1}^{n_X}\sum_{l=1}^{n_Z}\left|\sum_{k=1}^{n_X} A_X(i,k)W_{X,Z}(k,l)\right. \end{aligned}$$

$$- \sum_{k=1}^{n_Z} W_{X,Z}(i,k) A_Z(k,l) \Bigg| + \sum_{j=1}^{n_Y} \sum_{l=1}^{n_Z} \Bigg| \sum_{k=1}^{n_Z} A_Z(l,k) W_{Z,Y}(k,j) - \sum_{k=1}^{n_Y} W_{Z,Y}(l,k) A_Y(k,j) \Bigg|. \quad (31)$$

where we have used $\epsilon = 2$ and $p = 1$ without loss of generality, as ϵ and p are just part of the multiplication constant in (16).

From (27), we know that $W_{X,Y}(i,j) = \sum_{l=1}^{n_Z} W_{X,Z}(i,l) W_{Z,Y}(l,j)$, when $i \leq n_X, j \leq n_Y$, thus, the left hand side of (31) becomes

$$\sum_{i=1}^{n_X} \sum_{j=1}^{n_Y} \Bigg| \sum_{k=1}^{n_X} \sum_{l=1}^{n_Z} A_X(i,k) W_{X,Z}(k,l) W_{Z,Y}(l,j) - \sum_{k=1}^{n_Y} \sum_{l=1}^{n_Z} W_{X,Z}(i,l) W_{Z,Y}(l,k) A_Y(k,j) \Bigg|. \quad (32)$$

Here, we add and subtract the same term to the previous equation, and also drop the dependency of $e_{X,Y}$ on $W_{X,Y}$ for notational clarity

$$e_{X,Y}^{(2,1)} = \sum_{i=1}^{n_X} \sum_{j=1}^{n_Y} \Bigg| \sum_{k=1}^{n_X} \sum_{l=1}^{n_Z} A_X(i,k) W_{X,Z}(k,l) W_{Z,Y}(l,j) - \sum_{l=1}^{n_Z} \sum_{k=1}^{n_Z} W_{X,Z}(i,k) A_Z(k,l) W_{Z,Y}(l,j) + \sum_{l=1}^{n_Z} \sum_{k=1}^{n_Z} W_{X,Z}(i,k) A_Z(k,l) W_{Z,Y}(l,j) - \sum_{k=1}^{n_Y} \sum_{l=1}^{n_Z} W_{X,Z}(i,l) W_{Z,Y}(l,k) A_Y(k,j) \Bigg|, \quad (33)$$

$$= \sum_{i=1}^{n_X} \sum_{j=1}^{n_Y} \Bigg| \sum_{l=1}^{n_Z} \left(\sum_{k=1}^{n_X} A_X(i,k) W_{X,Z}(k,l) - \sum_{k=1}^{n_Z} W_{X,Z}(i,k) A_Z(k,l) \right) W_{Z,Y}(l,j) + \sum_{l=1}^{n_Z} W_{X,Z}(i,l) \left(\sum_{k=1}^{n_Z} A_Z(l,k) W_{Z,Y}(k,j) - \sum_{k=1}^{n_Y} W_{Z,Y}(l,k) A_Y(k,j) \right) \Bigg|. \quad (34)$$

Note that the range of k and l in (33) above are the same, so they can be swapped in (34). Then, we apply the inequality $|\sum_l a_l| \leq \sum_l |a_l|$ to the equation above, yields

$$e_{X,Y}^{(2,1)} \leq \sum_{i=1}^{n_X} \sum_{j=1}^{n_Y} \sum_{l=1}^{n_Z} \Bigg| \left(\sum_{k=1}^{n_X} A_X(i,k) W_{X,Z}(k,l) - \sum_{k=1}^{n_Z} W_{X,Z}(i,k) A_Z(k,l) \right) W_{Z,Y}(l,j) + W_{X,Z}(i,l) \left(\sum_{k=1}^{n_Z} A_Z(l,k) W_{Z,Y}(k,j) - \sum_{k=1}^{n_Y} W_{Z,Y}(l,k) A_Y(k,j) \right) \Bigg| \quad (35)$$

$$\leq \sum_{i=1}^{n_X} \sum_{l=1}^{n_Z} \sum_{j=1}^{n_Y} \Bigg| \left(\sum_{k=1}^{n_X} A_X(i,k) W_{X,Z}(k,l) - \sum_{k=1}^{n_Z} W_{X,Z}(i,k) A_Z(k,l) \right) W_{Z,Y}(l,j) \Bigg| + \sum_{j=1}^{n_Y} \sum_{l=1}^{n_Z} \sum_{i=1}^{n_X} \Bigg| W_{X,Z}(i,l) \left(\sum_{k=1}^{n_Z} A_Z(l,k) W_{Z,Y}(k,j) - \sum_{k=1}^{n_Y} W_{Z,Y}(l,k) A_Y(k,j) \right) \Bigg|. \quad (36)$$

We now use the absolute value of $|ab| = |a||b|$ for two real numbers a and b. From (17), we know that the weight matrices W are non-negative. Note that the sum over j only affects weight matrix $W_{Z,Y}$ and i only affects weight matrix $W_{X,Z}$, therefore, based on (4), we can write (36) as

$$\sum_{i=1}^{n_X} \sum_{l=1}^{n_Z} \sum_{j=1}^{n_Y} \Bigg| \sum_{k=1}^{n_X} A_X(i,k) W_{X,Z}(k,l) - \sum_{k=1}^{n_Z} W_{X,Z}(i,k) A_Z(k,l) \Bigg| W_{Z,Y}(l,j) + \sum_{j=1}^{n_Y} \sum_{l=1}^{n_Z} \sum_{i=1}^{n_X} W_{X,Z}(i,l) \Bigg| \sum_{k=1}^{n_Z} A_Z(l,k) W_{Z,Y}(k,j) - \sum_{k=1}^{n_Y} W_{Z,Y}(l,k) A_Y(k,j) \Bigg|. \quad (37)$$

Thus, we can write the equation above as

$$e_{X,Y}^{(2,1)}(W_{X,Y}) \leq \sum_{i=1}^{n_X} \sum_{l=1}^{n_Z} \Bigg| \sum_{k=1}^{n_X} A_X(i,k) W_{X,Z}(k,l) - \sum_{k=1}^{n_Z} W_{X,Z}(i,k) A_Z(k,l) \Bigg| + \sum_{j=1}^{n_Y} \sum_{l=1}^{n_Z} \Bigg| \sum_{k=1}^{n_Z} A_Z(l,k) W_{Z,Y}(k,j) - \sum_{k=1}^{n_Y} W_{Z,Y}(l,k) A_Y(k,j) \Bigg| \quad (38)$$

$$= e_{X,Z}^{(2,1)}(W_{X,Z}) + e_{Z,Y}^{(2,1)}(W_{Z,Y}) \quad (39)$$

and this proves (31). As explained before, this result, together with the derivations in [33], proves (30), which in turn proves Proposition 2. This also proves that (2) is a metric, as the proof remains unchanged if we use the constraint (7).

B. Proof of the triangle inequality property for directed graphs

In this section, we prove the triangle inequality of the metric for directed graphs, see Lemma 4. The proof of the triangle inequality is analogous to the proof for undirected graphs, which was provided in Appendix B-A, except for the inequality in the edge mismatch cost. Therefore, in this appendix, we prove the required inequality for the edge mismatch cost.

The edge mismatch cost for directed graphs in (22) can be divided into two parts,

$$e_{X,Y}(W) = \frac{1}{2}\left(\tilde{e}_{X,Y}(W) + \tilde{e}^{(2)}_{Y,X}(W)\right) \tag{40}$$

where

$$\tilde{e}_{X,Y} = \frac{\epsilon^p}{2}||A_X W_{1:n_X,1:n_Y} - W_{1:n_X,1:n_Y} A_Y|| \tag{41}$$

and

$$\tilde{e}^{(2)}_{Y,X} = \frac{\epsilon^p}{2}||A_Y W^T_{1:n_X,1:n_Y} - W^T_{1:n_X,1:n_Y} A_X||. \tag{42}$$

We have already proved in Appendix B-A that, for any graphs X, Y, Z, $\tilde{e}_{X,Y} \leq \tilde{e}_{X,Z} + \tilde{e}_{X,Y}$. Importantly, to obtain this result, the adjacency matrices can have a general form (they are not required to be symmetric). Therefore, we can analogously derive the inequality $\tilde{e}^{(2)}_{Y,X} \leq \tilde{e}^{(2)}_{Z,X} + \tilde{e}^{(2)}_{Y,Z}$. Therefore, we have

$$\begin{aligned} e_{X,Y} &= \frac{1}{2}\left(\tilde{e}_{X,Y} + \tilde{e}^{(2)}_{Y,X}\right) \\ &\leq \frac{1}{2}\left(\tilde{e}_{X,Z} + \tilde{e}_{Z,Y}) + \tilde{e}^{(2)}_{Z,X} + \tilde{e}^{(2)}_{Y,Z}\right) \\ &= \frac{1}{2}(\tilde{e}_{X,Z} + \tilde{e}^{(2)}_{Z,X}) + \frac{1}{2}(\tilde{e}_{Z,Y} + \tilde{e}^{(2)}_{Y,Z}) \\ &= e_{X,Z} + e_{Z,Y}. \end{aligned} \tag{43}$$

Then, we can continue the proof as in the undirected case.

C. Proof for computability using LP

The proof for the computability of the metric in (18) using LP follows the proof in [33], [48]. We note that the computation of the metric in (18) requires solving the following optimisation problem:

$$\underset{W \in \overline{\mathcal{W}}_{X,Y}}{\arg\min} \left(\mathrm{tr}\big[D^\top_{X,Y} W\big] + e_{X,Y}(W)^p\right) \tag{44}$$

where $e_{X,Y}(W)$ is given in (14).

1) Proof for undirected graphs: The objective function for undirected graphs can be written as

$$\underset{W \in \overline{\mathcal{W}}_{X,Y},\, \hat{e} \in \mathbb{R}}{\arg\min} \left(\mathrm{tr}\big[D^\top_{X,Y} W\big] + \frac{\epsilon^p}{2}\hat{e}\right) \tag{45}$$

where we have introduced an extra variable $\hat{e} \in \mathbb{R}$ to the optimisation problem with constraint

$$\hat{e} \geq \sum_{i=1}^{n_X}\sum_{j=1}^{n_Y}\left|\sum_{k=1}^{n_X} A_X(i,k)W(k,j) - \sum_{k=1}^{n_Y} W(i,k)A_Y(k,j)\right|, \tag{46}$$

All the constraints in (4),(5), (6) and (17) are linear except the constraints in (46). To make the optimisation problem linear, we can introduce additional variables $H(i,j) \in \mathbb{R}$ with the constraints:

$$\hat{e} \geq \sum_{i=1}^{n_X}\sum_{j=1}^{n_Y} H(i,j), \tag{47}$$

$$H(i,j) \geq \sum_{k=1}^{n_X} A_X(i,k)W(k,j) - \sum_{k=1}^{n_Y} W(i,k)A_Y(k,j), \tag{48}$$

$$H(i,j) \geq \sum_{k=1}^{n_Y} W(i,k)A_Y(k,j) - \sum_{k=1}^{n_X} A_X(i,k)W(k,j). \tag{49}$$

Now, the optimisation problem (45) is linear with linear constraints (4), (5), (6), (7), (17), (47), (48) and (49) so it is a linear programming task, which is computable in polynomial time [38].

2) Proof for directed graphs: The objective function for directed graphs can be written as

$$\underset{W \in \overline{\mathcal{W}}_{X,Y},\, \hat{e} \in \mathbb{R},\, \hat{e}_2 \in \mathbb{R}}{\arg\min} \left(\mathrm{tr}\big[D^\top_{X,Y} W\big] + \frac{\epsilon^p}{4}\hat{e} + \frac{\epsilon^p}{4}\hat{e}_2\right) \tag{50}$$

where we have introduced two extra variables $\hat{e}$, $\hat{e}_2 \in \mathbb{R}$. For $\hat{e}$, the constraints are exactly the same as the case for undirected graphs, see (46). For $\hat{e}_2$, the constraints are

$$\hat{e}_2 \geq \sum_{i=1}^{n_X}\sum_{j=1}^{n_Y}\left|\sum_{l=1}^{n_Y} A_Y(i,l)W(j,l) - \sum_{l=1}^{n_X} W(l,i)A_X(l,j)\right|, \tag{51}$$

To write the metric in LP form, we need to introduce the additional variables $H(i,j)$, see (47). In addition, similar to $H(i,j)$, we introduce additional variables $H_2(i,j) \in \mathbb{R}$ with constraints:

$$\hat{e}_2 \geq \sum_{i=1}^{n_X}\sum_{j=1}^{n_Y} H_2(i,j), \tag{52}$$

$$H_2(i,j) \geq \sum_{l=1}^{n_Y} A_Y(i,l)W(j,l) - \sum_{l=1}^{n_X} W(l,i)A_X(l,j), \tag{53}$$

$$H_2(i,j) \geq \sum_{l=1}^{n_X} W(l,i)A_X(l,j) - \sum_{l=1}^{n_Y} A_Y(i,l)W(j,l). \tag{54}$$

Now, the metric computation can be written as an optimisation problem (50) with linear constraints (4), (5), (6), (7), (17), (47)–(49), and (52)–(54), therefore, it is also computable in polynomial time.

Appendix C

In this appendix, we prove Lemma 5. If the base metric $d(\cdot,\cdot)$ is set to zero, from (11), we have

$$D_{X,Y} = \begin{pmatrix} 0_{n_X,n_Y} & \frac{c^p}{2}1_{n_X,1} \\ \frac{c^p}{2}1_{1,n_Y} & 0 \end{pmatrix} \tag{55}$$

$$D^\top_{X,Y} = \begin{pmatrix} 0_{n_Y,n_X} & \frac{c^p}{2}1_{n_Y,1} \\ \frac{c^p}{2}1_{1,n_X} & 0 \end{pmatrix}. \tag{56}$$

. Then, we directly obtain

$$\operatorname{tr}\left[D_{X,Y}^{\top} W\right] = \frac{c^2}{2}\left(\sum_{i=1}^{n_X} W\left(i, n_Y+1\right) + \sum_{j=1}^{n_Y} W\left(n_X+1, j\right)\right) \tag{57}$$

This proves Eq. (23). The symmetry property is direct to check. The triangle inequality proof in Appendix B is valid for $d\left(\cdot,\cdot\right)=0$. In addition, it is direct to show that $\tilde{d}^{(c,\epsilon)}\left(X,X\right)=0$, which finishes the proof that $\tilde{d}^{(c,\epsilon)}\left(\cdot,\cdot\right)$ is a pseudometric.