A Differentiable Approximation of the Graph Edit Distance

Julia Wallnig[1][0009−0006−7676−2820], Luc Brun[2][0000−0002−1658−0527], Benoit Gaüzère[3][0000−0001−9980−2641], Sébastien Bougleux[2][0000−0002−4581−7570], Florian Yger[4][0000−0002−7182−8062], and David B. Blumenthal[1][0000−0001−8651−750X]

[1] Biomedical Network Science Lab, Department Artificial Intelligence in Biomedical Engineering, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany
julia.wallnig@fau.de, david.b.blumenthal@fau.de

[2] Normandie Univ, ENSICAEN, CNRS, UNICAEN, GREYC, Caen, France
sebastien.bougleux@unicaen.fr, luc.brun@ensicaen.fr

[3] Normandie Univ, INSA Rouen Normandie, LITIS, Rouen, France
benoit.gauzere@insa-rouen.fr

[4] PSL-Université Paris-Dauphine, CNRS, LAMSADE, Paris, France
florian.yger@lamsade.dauphine.fr

Abstract. To determine the similarity of labeled graphs, the graph edit distance (GED) is widely used due to its metric properties on the graph space and its interpretability. It is defined as the minimal cost of a sequence of edit operations transforming one graph into another one, with the cost of each edit operation being a parameter of the distance. Although calculating GED is NP-hard, various heuristics exist which, in practice, typically yield tight upper or lower bounds. Since determining appropriate edit operation costs for a given dataset or application can be challenging, it is attractive to learn these costs from the data, e. g., using metric learning architectures. However, for this approach to be feasible, a differentiable algorithm to approximate the GED is required. In this work, we present such an algorithm and show via an empirical evaluation on three datasets that the obtained distances closely match the distances computed by a state-of-the-art combinatorial GED heuristic.

Keywords: Graph Edit Distance · Sinkhorn Algorithm · Differentiable Solver.

1 Introduction

The *graph edit distance* (GED) [10] is a classical approach to quantify the dissimilarity of two labeled graphs G and H. It involves transforming a graph G into a graph isomorphic to H through a sequence of six elementary edit operations (node or edge insertion, deletion, or substitution), which all come with associated *edit costs*. A sequence of edit operations transforming G into H is called *edit path*, and its cost is defined as the sum of the individual operation costs. The GED from G to H is defined as the minimal cost of an edit path from

G to H. Computing GED is NP-hard [19], but various well-performing heuristics exist which provide upper or lower bounds [2].

In most GED heuristics, the edit operation costs are treated as parameters to be provided by the user. Since defining sensible costs for a given dataset can be challenging, is has been suggested to learn the edit costs from the data [11,14]. However, in existing approaches, a ground-truth mapping from the nodes of G to the nodes of H is required which is often not available. Moreover, even if such a mapping exists, it may not be unique, and *a priori* fixing one mapping may induce a bias in the learning process. In view of recent advances in the field of metric learning [18], it is hence attractive to learn the edit costs in an end-to-end fashion. For this, however, a differentiable operator to (approximately) compute GED is required which currently does not exist. Here, we close this gap.

Our differentiable GED operator relies on two building blocks. The first one is the widely used GED heuristic BRANCH [15], which computes both upper and lower bounds for GED [3]. BRANCH transforms the problem of computing GED to a *linear sum assignment problem with error-correction* (LSAPE), which is then solved using variants of the Hungarian algorithm [8,7]. The other building block is a recent work [9] which shows that the (differentiable) Sinkhorn algorithm [16] can be adapted such that it yields a differentiable approximation of LSAPE. We combine these building blocks into a differentiable operator to compute GED by replacing the Hungarian algorithm in BRANCH with the adapted Sinkhorn algorithm. Tests on three datasets show that our differentiable GED approximator achieves equally good results as BRANCH. Our results hence pave the way for incorporating GED into differentiable metric learning frameworks, thereby facilitating future advances in end-to-end learning of the edit costs.

2 Preliminaries

Basic Notations. We consider undirected, possibly labeled graphs $G = (V^G, E^G)$ and $H = (V^H, E^H)$, set $n := |V^G|$ and $m := |V^H|$, and use the notations $[n] := \{1, \dots, n\}$ and $[m] := \{1, \dots, m\}$ to denote the index sets for the nodes of G and H, respectively. In addition, for any matrix X of size $n \times m$, $x_{i,j}$ with $i \in [n]$ and $j \in [m]$ will always denote the entries of the matrix.

Graph Edit Distance and Node Maps. While GED is classically defined in terms of edit paths as explained above, most algorithmic approaches rely on an alternative definition in terms of node maps. A *node map* $\pi \subseteq (V^G \cup \{\epsilon\}) \times (V^H \cup \{\epsilon\})$ specifies for all nodes and, derived from this, for all edges if they are substituted, inserted, or deleted. Applying π's induced edit operations yields an *induced edit path* P_π, with $c(\pi)$ denoting the sum of the costs of all its edit operations. Under reasonable constraints that hold in almost all applications, minimizing $c(\pi)$ over all node maps π from G to H yields the GED between these graphs [6,4].

The Linear Sum Assignment Problem with Error Correction. LSAPE is an extension of the well-known bipartite matching a. k. a. linear sum assignment problem (LSAP), where two additional dummy nodes $n+1$ and $m+1$ are inserted to model insertions and deletions. Given a cost matrix C of size $(n+1)\times(m+1)$ with $c_{n+1,m+1}=0$, LSAPE asks to solve the discrete minimization problem

$$\min_X \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} c_{i,j}x_{i,j} =: C(X), \tag{1}$$

where $X \in \{0,1\}^{(n+1)\times(m+1)}$ is an *error-correcting permutation matrix* (a. k. a. ϵ-permutation matrix), i. e., a binary matrix with $\sum_{i=1}^{n+1} x_{i,j} = 1$ for all $j \in [m]$, and $\sum_{j=1}^{m+1} x_{i,j} = 1$ for all $i \in [n]$ [7] (note that there are no constraints on $i = n+1$ and $j = m+1$). Each ϵ-permutation matrix X induces an *error-correcting matching* (also called ϵ-matching)

$$\pi_X \coloneqq \{(i,j) \in [n+1]\times[m+1] \mid x_{i,j}=1\}, \tag{2}$$

and we use the notation $C(\pi_X) \coloneqq C(X)$ to denote the matching cost of π_X. The Hungarian algorithm [12] is a widely used method for optimally solving LSAP and can also be extended to optimally solve LSAPE [8,7].

The Sinkhorn Algorithm. The Sinkhorn-Knopp theorem states that, for each positive matrix S of size $n \times n$, there are positive diagonal matrices D_1 and D_2 such that $X \coloneqq D_1SD_2 \in \mathbb{R}^{n\times n}$ is bistochastic (i. e., all rows and columns of X sum up to 1). The Sinkhorn algorithm produces a series of matrices that converge to X via alternate scaling of S's rows and columns to unit sum. If applied on an exponentiated cost matrix

$$S \coloneqq \exp(-\varepsilon^{-1}C), \tag{3}$$

the Sinkhorn algorithm converges to the optimal solution of the *optimal transport problem with entropic regularization* ($\varepsilon > 0$ is the regularization constant) [13]. This problem, in turn, can be seen as a continuous relaxation of LSAP.

While the classical Sinkhorn algorithm is applicable only to matrices S of size $n \times n$, it has recently been shown that that it can be generalized to positive rectangular matrices of size $(n+1)\times(m+1)$ [9] and then yields an *ϵ-bistochastic matrix* $X \in \mathbb{R}^{n+1\times m+1}$, where the firsts n row sums, the first m column sums, and the entry $x_{n+1,m+1}$ equals 1. Since ϵ-bistochastic matrices are continuous relaxations of ϵ-matchings, running the adapted Sinkhorn algorithm on an exponentiated cost matrix yields a differentiable approximation of LSAPE [9].

Heuristics for Graph Edit Distance Computation Based on the Linear Sum Assignment with Error Correction. Various GED heuristics exist that are based on LSAPE [2]. These algorithms decompose the input graphs G and H into local structures rooted at their nodes and then construct an instance C of LSAPE, where the last row $C_{n+1,\bullet}$ and the last column $C_{\bullet,m+1}$ contain, respectively, local structure insertion and deletion costs and the remaining cells $C_{i,j}$ contain local structure substitution costs. C is then solved optimally, e. g., with a variant of the Hungarian algorithm, which yields an ϵ-matching $\pi_X \subseteq [n+1]\times[m+1]$.

Since π can be interpreted as a node map from V^G to V^H, its induced edit cost $c(\pi_X)$ is an upper bound for GED. Moreover, for some LSAPE-based GED heuristics, the matching cost $C(\pi_X)$ is a lower bound for GED. One such approach is the widely used algorithm BRANCH [15]. In this algorithm, the local structures are defined as branches (i. e., root nodes together with their adjacent edges) and the LSAPE instance C is defined in terms of the elementary node and edge edit costs required to transform a branch rooted at node $u \in V^G$ into a branch rooted at node $v \in V^H$ (see [15,3,2] for details). For this article, we tested our differentiable GED approximator with the LSAPE instance C as constructed by BRANCH but the underlying approach is compatible with any LSAPE-based GED heuristic. Moreover, for our algorithm to be differentiable, we assume that C is provided as input. This assumption can be lifted by using the Sinkhorn algorithm not only as an approximate solver for C but also for the approximate computation of the branch edit costs during the construction of C.

3 A Differentiable Graph Edit Distance Approximator

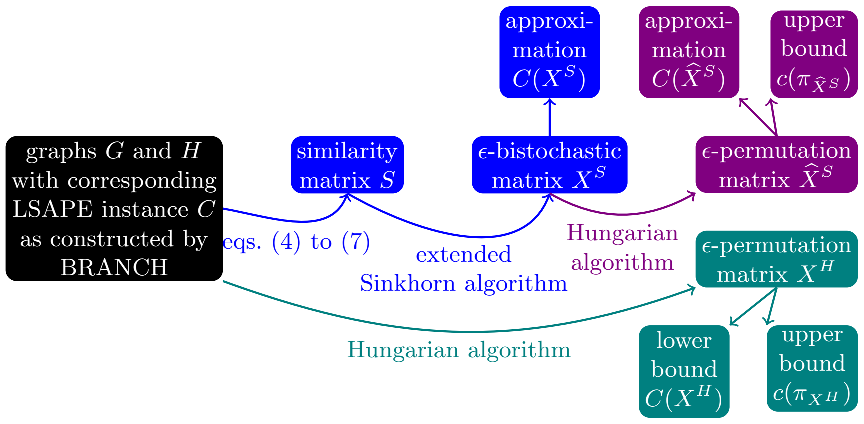
approxi-mation $C(X^S)$
approxi-mation $C(\widehat{X}^S)$
upper bound $c(\pi_{\widehat{X}^S})$
graphs G and H with corresponding LSAPE instance C as constructed by BRANCH
similarity matrix S
ϵ-bistochastic matrix X^S
ϵ-permutation matrix $\widehat{X}^S$
eqs. (4) to (7)
extended Sinkhorn algorithm
Hungarian algorithm
ϵ-permutation matrix X^H
Hungarian algorithm
lower bound $C(X^H)$
upper bound $c(\pi_{X^H})$

Fig. 1. Overview of our differentiable GED approximator (blue) with discrete post-processing (violet) in comparison to BRANCH (turquoise).

Figure 1 provides an overview of our continuous GED approximator. Given two graphs G and H and a corresponding LSAPE instance C as constructed by BRANCH, we start by transforming C into a similarity matrix S as follows: We first compute a transformed cost matrix C' by setting

$$c'_{i,j} := \begin{cases} c_{i,j} - \min\{c_{i,j'} \mid j' \in [m+1]\} & i \in [n] \\ c_{i,j} & i = n+1 \end{cases} \tag{4}$$

for all $(i,j) \in [n+1] \times [m+1]$, and further transform C' into C'' by defining

$$c''_{i,j} \coloneqq \begin{cases} c'_{i,j} - \min\{c'_{i',j} \mid i' \in [n+1]\} & j \in [m] \\ c'_{i,j} & j = m+1 \end{cases} \tag{5}$$

for all $(i,j) \in [n+1] \times [m+1]$. Let us note that eqs. (4) and (5) correspond to the initialization steps of the LSAPE algorithm [8,7]. The matrices C'' and C correspond to equivalent LSAPE problems. Within our context, these simplifications of the cost matrix ensure that the minimum of each row and each column is set to 0. Using the construction scheme described below (eq. (7)), the corresponding similarity values is set to 1 independently of the entropic regularization.

Next, we compute a scaling factor

$$T \coloneqq \min\left\{\alpha, \beta \cdot \log(\alpha) \cdot \max(C'')^{-1}\right\}, \tag{6}$$

where $\alpha, \beta > 0$ are scaling hyper-parameters. Constant T can be interpreted as the inverse of the regularization constant ε in eq. (3). Equipped with C'' and T, we then compute a similarity matrix as follows:

$$S \coloneqq \exp(-T \cdot C'') \tag{7}$$

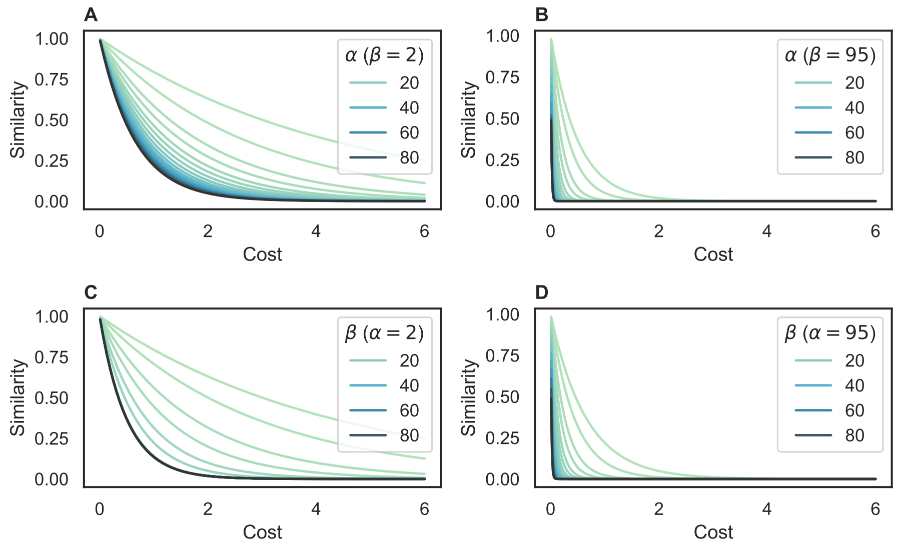

Fig. 2. Shape of the cost-to-similarity transformation function for different values of the parameters α and β. (A, B) Shape for varying α and fixed values of β. (C, D) Shape for varying β and fixed values of α.

Let us note that the division by $\max(C'')$ allows to normalize all cost values between 0 and 1. The resulting cost-to-similarity function in dependence on β

and α is visualized in Figure 2: For small values of α and β, the function has a near-linear behaviour. For larger β and α, it assumes the shape of a step function.

Once S has been constructed, we feed it into the extended Sinkhorn algorithm proposed in [9]. This yields an ϵ-bistochastic matrix X^S, whose matching cost $C(X^S)$ (see eq. (1)) constitutes our differentiably computable approximation of the GED from G to H. $C(X^S)$ is the analogue to the matching cost $C(X^H)$ of the ϵ-permutation matrix X^H computed by BRANCH, but — unlike $C(X^H)$ — is not guaranteed to constitute a lower bound for GED.

To enable comparison also w. r. t. BRANCH's upper bound $c(\pi_{X^H})$, we further project X^S to an ϵ-permutation matrix $\widehat{X}^S$, using the Hungarian algorithm (violet boxes and arrows in Figure 1). The node map $\pi_{\widehat{X}^S}$ induced by $\widehat{X}^S$ then yields an upper bound $c(\pi_{\widehat{X}^S})$ for GED, and the matching cost $C(\widehat{X}^S)$ gives us another approximation for GED. Note that since the computation of $\widehat{X}^S$ relies on the non-differentiable Hungarian algorithm, neither $c(\pi_{\widehat{X}^S})$ nor $C(\widehat{X}^S)$ should be viewed as direct output of our differentiable GED approximator.

4 Empirical Evaluation

Datasets and Edit Costs. We used the datasets Acyclic and MAO from GREYC's Chemistry Dataset (https://brunl01.users.greyc.fr/CHEMISTRY/). The graphs contained in these datasets represent molecular compounds. The Acyclic dataset contains 183 acyclic graphs, while the MAO dataset contains 68 graphs including cycles. Both datasets contain node labels, and edge labels assumed to be uniform. To also consider variability in edge labels, we used a dataset presented in [17], containing 72 graphs with non-uniform node and edge labels, representing pseudoknotted RNA secondary structures. We used a preprocessed version of the data provided by the authors of [5] (https://github.com/bionetslab/edge-preservation-similarity/). As elementary edit costs, we choose $(c^s_V, c^d_V, c^i_V, c^s_E, c^d_E, c^i_E) = (2, 4, 4, 1, 1, 1)$, as proposed in [1].

Evaluation Metrics. As evaluation metrics, we calculated the relative diversions of the matching costs $C(X^S)$ and $C(\widehat{X}^S)$ w. r. t. the matching cost (lower bound for GED) $C(X^H)$ of the ϵ-permutation matrix computed by BRANCH:

$$div(X^S) := \frac{C(X^S) - C(X^H)}{C(X^H) + 1} \quad div(\widehat{X}^S) := \frac{C(\widehat{X}^S) - C(X^H)}{C(X^H) + 1} \tag{8}$$

Since neither $C(X^S)$ nor $C(\widehat{X}^S)$ are provable lower or upper bounds for GED, it is unclear whether higher or lower values $C(X^S)$ and $C(\widehat{X^S})$ are closer to GED. This is why we use the term "diversion" in the context of comparison of the matching costs. If $div(X^S)$ and $div(\widehat{X}^S)$ are close to 0, $C(X^S)$ and $C(\widehat{X}^S)$ constitute approximations of GED which are close-to-equivalent to the lower bound $C(X^H)$ computed by BRANCH. To quantify the quality of the upper bound $c(\pi_{\widehat{X}^S})$, we computed the relative error:

$$error(\widehat{X}^S) := \frac{c(\pi_{\widehat{X}^S}) - c(\pi_{X^H})}{c(\pi_{X^H}) + 1} \tag{9}$$

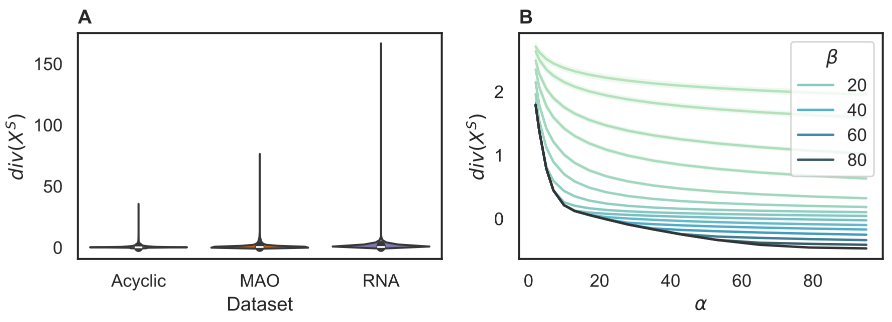

Fig. 3. Relative diversions of the matching costs of the ϵ-bistochastic matrices X^S from the matching costs of the ϵ-permutation matrices X^H computed by BRANCH. A: Violin plots showing distributions of relative diversions $div(X^S)$ split by datasets. B: Relative diversions $div(X^S)$ in dependence of α and β.

Here, we use the term "error" because $c(\pi_{\widehat{X}^S})$ is an upper bound for GED and so a lower value is always preferable. If $error(\widehat{X}^S)$ is negative, the upper bound $c(\pi_{\widehat{X}^S})$ is tighter than the upper bound provided by BRANCH.

Implementation, Availability, and Hardware Specification. We implemented our algorithm in Python and relied on PyTorch to implement the adapted Sinkhorn algorithm suggested in [9]. The source code is available on GitHub (https://github.com/juliawal/continuous_ged_approximation). Tests were run on a MacBook with an Apple M1 processor and 8GB of RAM.

Quality of the Differentiable Approximation. Figure 3 shows the diversions of the matching costs of the ϵ-bistochastic matrices X^S, split across datasets (Figure 3A) and for varying hyper-parameters α and β (Figure 3B). For all three datasets, the median is around zero, with individual outliers that are largest for the RNA dataset and smallest for Acyclic. Larger values of α and β result in a smaller diversions. Overall, this means that our differentiable GED approximation is close to the lower bound for GED computed by BRANCH.

Quality of the Projection. Figure 4 shows the diversions of the matching costs of the projections $\widehat{X}^S$, split across datasets (Figure 4A) and for varying hyper-parameters α and β (Figure 4B). While we observe a minor increase in diversion for increasing α and β, diversions are overall tiny and even smaller than for the ϵ-bistochastic matrices.

The quality of the projections' induced edit costs $c(\pi_{\widehat{X}^S})$ (upper bound for GED) is visualized in Figure 5. For all three datasets, the median relative error is around 0 (Figure 5A). While there are individual outliers for the RNA dataset, for larger values of α and β, the upper bound $c(\pi_{\widehat{X}^S})$ is actually tighter than the upper bound computed by BRANCH for around 70 to 75 % of all instances (Figure 5B). Errors decrease with increasing values of α and β (Figure 5C).

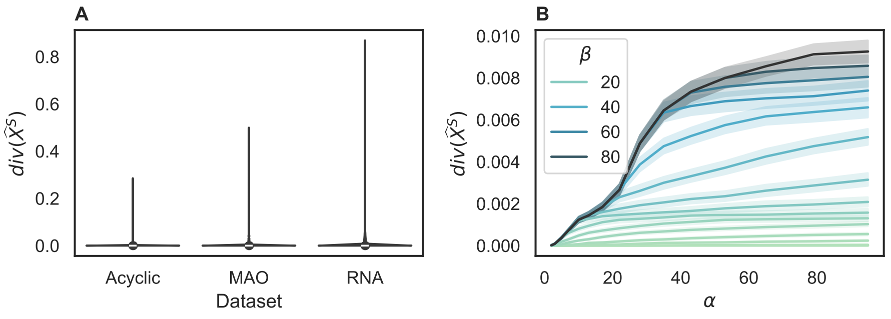

Fig. 4. Relative diversions of the matching costs of the projections $\widehat{X}^S$ of the ϵ-bistochastic matrices X^S from the matching costs of the ϵ-permutation matrices X^H computed by BRANCH. A: Violin plots showing distributions of relative diversions $div(\widehat{X}^S)$ split by datasets. B: Relative diversions $div(\widehat{X}^S)$ in dependence of α and β.

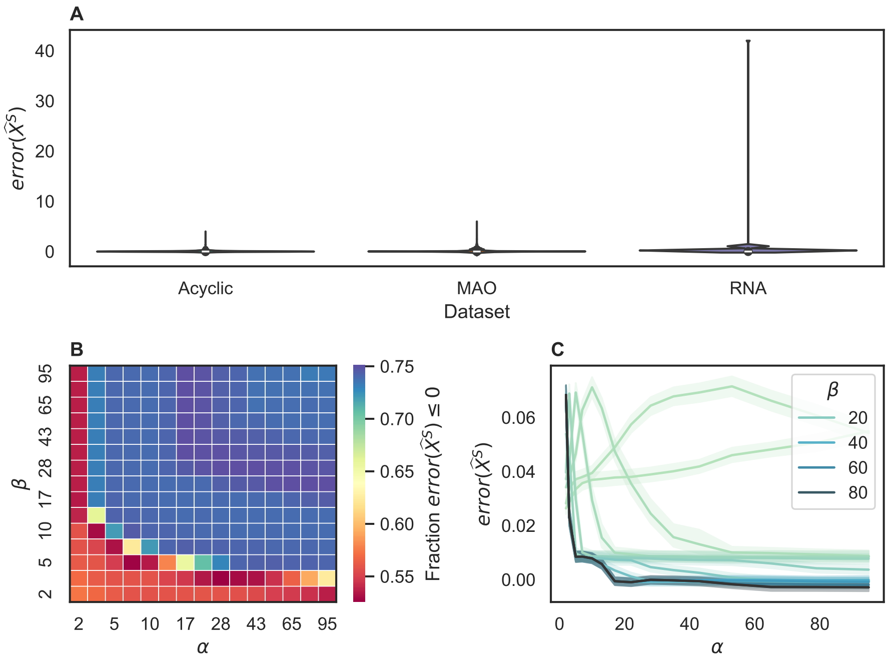

Fig. 5. Relative errors of the induced edit costs of the projections $\widehat{X}^S$ of the ϵ-bistochastic matrices X^S from the induced edit costs of the ϵ-permutation matrices X^H computed by BRANCH. A: Violin plots showing distributions of relative errors split by datasets. B: Fractions of instances with negative or zero relative errors. C: Relative errors in dependence of α and β.

Runtime. Figure 6 shows the runtime of the Sinkhorn algorithm when fed with similarity matrices constructed according to eq. (7) in dependence of α and β. Larger values of α and β lead to longer runtimes, independently of the dataset.

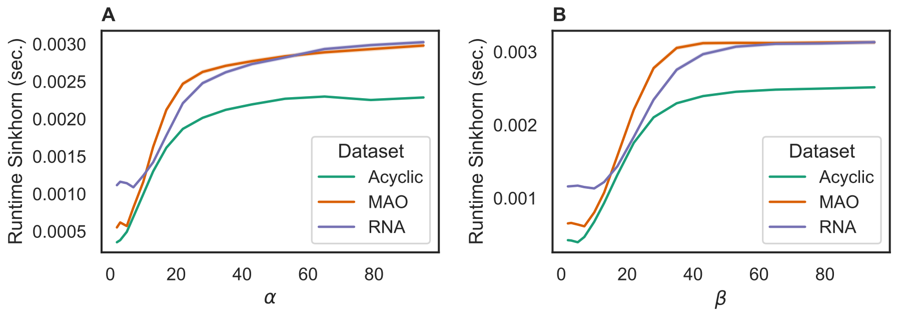

Fig. 6. Runtime of the Sinkhorn algorithm step within our continuous GED approximator for varying hyper-parameters α (A) and β (B).

5 Summary and Outloook

In this article, we presented a differentiable approximator for GED based on a reduction to LSAPE and an extended version of the Sinkhorn algorithm. Having access to such an approximator is desirable because it allows to incorporate the approximation of GED into deep metric learning frameworks, which may pave the way for a data-driven inference of elementary edit costs. Tests on three datasets showed that our approach performs similar to the widely used GED heuristic BRANCH. Our work opens up various avenues for future work: For instance, it would be interesting to assess the effect of replacing the Hungarian algorithm by the Sinkhorn algorithm also in the construction of the LSAPE instance C, which we here treated as part of the input. It may also be possible to formulate an error-correcting version of the optimal transport problem with entropic regularization, which may provide a strong theoretical justification for using the Sinkhorn algorithm as an approximate solver for LSAPE.

References

1. Abu-Aisheh, Z., Gaüzère, B., Bougleux, S., Ramel, J., Brun, L., Raveaux, R., Héroux, P., Adam, S.: Graph edit distance contest: Results and future challenges. Pattern Recognit. Lett. **100**, 96–103 (2017). https://doi.org/10.1016/J.PATREC.2017.10.007
2. Blumenthal, D.B., Boria, N., Gamper, J., Bougleux, S., Brun, L.: Comparing heuristics for graph edit distance computation. VLDB J. **29**(1), 419–458 (2020). https://doi.org/10.1007/s00778-019-00544-1

3. Blumenthal, D.B., Gamper, J.: Improved lower bounds for graph edit distance. IEEE Trans. Knowl. Data Eng. **30**(3), 503–516 (2018). https://doi.org/10.1109/TKDE.2017.2772243
4. Blumenthal, D.B., Gamper, J.: On the exact computation of the graph edit distance. Pattern Recognit. Lett. **134**, 46–57 (2020). https://doi.org/10.1016/j.patrec.2018.05.002
5. Boria, N., Kiederle, J., Yger, F., Blumenthal, D.B.: The edge-preservation similarity for comparing rooted, unordered, node-labeled trees. Pattern Recognit. Lett. **167**, 189–195 (2023). https://doi.org/10.1016/j.patrec.2023.02.017
6. Bougleux, S., Brun, L., Carletti, V., Foggia, P., Gaüzère, B., Vento, M.: Graph edit distance as a quadratic assignment problem. Pattern Recognit. Lett. **87**, 38–46 (2017). https://doi.org/10.1016/j.patrec.2016.10.001
7. Bougleux, S., Gaüzère, B., Blumenthal, D.B., Brun, L.: Fast linear sum assignment with error-correction and no cost constraints. Pattern Recognit. Lett. **134**, 37–45 (2020). https://doi.org/10.1016/j.patrec.2018.03.032
8. Bougleux, S., Gaüzère, B., Brun, L.: A Hungarian algorithm for error-correcting graph matching. In: Foggia, P., Liu, C., Vento, M. (eds.) GbRPR 2017. LNCS, vol. 10310, pp. 118–127 (2017). https://doi.org/10.1007/978-3-319-58961-9_11
9. Brun, L., Gaüzère, B., Renton, G., Bougleux, S., Yger, F.: A differentiable approximation for the linear sum assignment problem with edition. In: ICPR 2022. pp. 3822–3828. IEEE (2022). https://doi.org/10.1109/ICPR56361.2022.9956203
10. Bunke, H., Allermann, G.: Inexact graph matching for structural pattern recognition. Pattern Recognit. Lett. **1**(4), 245–253 (1983). https://doi.org/10.1016/0167-8655(83)90033-8
11. Cortés, X., Conte, D., Cardot, H.: Learning edit cost estimation models for graph edit distance. Pattern Recognit. Lett. **125**, 256–263 (2019). https://doi.org/10.1016/j.patrec.2019.05.001
12. Kuhn, H.W.: The Hungarian method for the assignment problem. Nav. Res. Logist. Q. **2**(1-2), 83–97 (1955). https://doi.org/10.1002/nav.3800020109
13. Peyré, G., Cuturi, M.: Computational optimal transport: With applications to data science. Foundations and Trends® in Machine Learning **11**(5-6), 355–607 (2019). https://doi.org/10.1561/2200000073
14. Rica, E., Álvarez, S., Serratosa, F.: On-line learning the graph edit distance costs. Pattern Recognit. Lett. **146**, 55–62 (2021). https://doi.org/10.1016/j.patrec.2021.02.019
15. Riesen, K., Bunke, H.: Approximate graph edit distance computation by means of bipartite graph matching. Image Vis. Comput. **27**(7), 950–959 (2009). https://doi.org/10.1016/j.imavis.2008.04.004
16. Sinkhorn, R., Knopp, P.: Concerning nonnegative matrices and doubly stochastic matrices. Pacific J. Math. **21**(2), 343–348 (1967). https://doi.org/10.2140/pjm.1967.21.343
17. Wang, F., Akutsu, T., Mori, T.: Comparison of pseudoknotted RNA secondary structures by topological centroid identification and tree edit distance. J. Comput. Biol. **27**(9), 1443–1451 (2020). https://doi.org/10.1089/CMB.2019.0512
18. Yoshida, T., Takeuchi, I., Karasuyama, M.: Distance metric learning for graph structured data. Mach. Learn. **110**(7), 1765–1811 (2021). https://doi.org/10.1007/s10994-021-06009-3
19. Zeng, Z., Tung, A.K.H., Wang, J., Feng, J., Zhou, L.: Comparing stars: On approximating graph edit distance. Proc. VLDB Endow. **2**(1), 25–36 (2009). https://doi.org/10.14778/1687627.1687631