# Structure- and Function-Aware Substitution Matrices via Differentiable Graph Matching

**Paolo Pellizzoni** [1] **Carlos Oliver** [1] **Karsten Borgwardt** [1]

## Abstract

Substitution matrices, which are crafted to quantify the functional impact of substitutions or deletions in biomolecules, are central component of remote homology detection, functional element discovery, and structure prediction algorithms. However, they are often limited to sequence data and the conditioning on external priors can only be given implicitly through the curation of the ground-truth alignments they are crafted on. Here we propose an algorithmic framework, based on regularized optimal transport, for learning graph-based substitution matrices from data, conditioned on any functional knowledge. In particular, our graph-neural-network-based model learns to produce substitution matrices and graph matchings such that the resulting metric correlates with the function at hand. Our method shows promising performance in functional similarity classification and shows potential for interpreting the functional importance of molecular substructures.

## 1. Introduction

Alignment algorithms, a crucial tool in bioinformatics, use substitution matrices to measure the impact of changes in the sequence and structure of biomolecules, with alignment quality depending on the choice of these matrices [1].

Substitution matrices, like BLOSUM62 and PAM for amino acids [2], [3], BLASTN for DNA [4], and specific matrices for RNA and chemicals [5], [6], are based on the frequency of modifications in homologous molecules [7]. Low-frequency changes indicate functional significance, while high-frequency changes suggest neutrality.

Conservation of structural data led to structural alphabets for alignments, such as Blast3D and Foldseek's 3Di [8], [9]. These matrices focus on structural fragments and target specific protein families [10], organisms [11], and phylogenetic knowledge [12]. Alignment-free methods predict the function of biomolecules using neural networks, as seen in DeepFRI, RNAmigos, and ChemBERT [13]–[16], but they lack interpretability compared to substitution matrices. Ideally, substitution matrices would use prior knowledge without needing pre-curated alignments. The DEDAL model learns, in a data-driven fashion, substitution costs using Pfam annotations, but relies on sequence data [17].

The graph edit distance [18], [19] models the similarity between graphs based on an optimal matching between their nodes, generalizing sequence alignment. The problem is NP-hard, so several attempts have been made towards developing heuristics, including machine-learning-based approaches [20]–[25].

In a recent paper [26], the problem of automatically obtaining task-specific substitution matrices for biological *structures* has been tackled taking inspiration from the graph edit distance literature. The substitution matrices are learnt in a metric learning framework based on a functional prior, and they provide an interpretable explanation of which biochemical substructure drive the function at hand.

The optimal transport problem [27], [28], which we use in this paper, is in general concerned with finding the minimum-cost transport plan between two distributions, and it has been used in tasks like domain adaptation [29], graph kernels [30], and generative adversarial networks [31].

### 1.1. Contributions

This work extends our recent paper: *Structure- and Function-Aware Substitution Matrices via Learnable Graph Matching. In: RECOMB 2024* [26]. In particular, we:

1. generalize the GMSM (Graph Matching Substitution Matrices) model to work with entropy-regularized optimal transport. This allows the model to have lower time complexity and to be fully differentiable.
2. provide experimental evidence that the entropy regularization helps in finding better substitution matrices.

[1]Department of Machine Learning and Systems Biology, Max Planck Institute of Biochemistry, Germany.. Correspondence to: Paolo Pellizzoni <pellizzoni@biochem.mpg.de>.

*Published at the $2^{nd}$ Differentiable Almost Everything Workshop at the $41^{st}$ International Conference on Machine Learning*, Vienna, Austria. July 2024. 

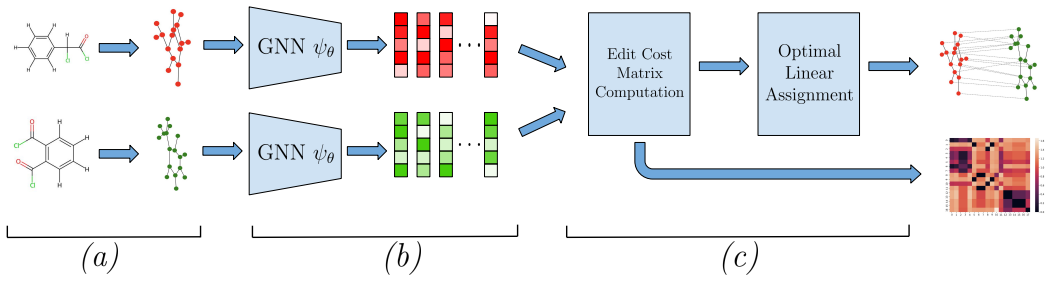


*Figure 1.* Architecture of GMSM. *(a)* Biochemical structures are transformed into graphs. *(b)* For each graph, its nodes are represented as a structure-aware embeddings using the same GNN. *(c)* The model computes the substitution matrix from node embeddings and obtains the graph alignment with respect to the learned substitution matrix.

## 2. Preliminaries

In what follows, we denote as a graph a tuple $G = (V, E)$, with $V$ the set of nodes and $E \subseteq \binom{V}{2}$ the set of undirected edges. Both nodes and edges can have discrete labels.

### 2.1. Graph Neural Networks

Message passing graph neural networks (GNNs), given a graph $G$, iteratively produce for each node $v \in V_G$, at each level $k = 1, \ldots, K$, the embeddings $h_v^k \in \mathbb{R}^{d_k}$ as $h_v^k = f_{\text{upd}}\left(h_v^{k-1}, f_{\text{agg}}\left(\{h_u^{k-1} : u \in \mathcal{N}(v)\}\right)\right)$, where $f_{\text{agg}}$ and $f_{\text{upd}}$ are the aggregate and the update operations, respectively. The first layer of the GNN is fed with the initial node embeddings $h_v^0$, e.g. one-hot encodings of the node labels. Finally, one can get a graph-level readout $h_G$ by aggregating the last-level embeddings via a function $f_{\text{out}}$.

### 2.2. Graph Edit Distance

The *graph edit distance* [18], [19] is a distance that assesses the similarity between two graphs. In particular, it is computed as the minimum cumulative cost of the edit operations required to transform one graph into another via node and edge insertion, deletion, and substitution, each with an associated cost. When the edit costs form a metric [32], the graph edit distance can be equivalently defined as a graph matching problem:

**Definition 2.1** (Graph Edit Distance [33])**.** Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be the source and the target graphs respectively. Let $V_1^+ = V_1 \cup \{\varepsilon_1, \ldots, \varepsilon_{|V_2|}\}$ be the vertex set of $G_1$ enriched with $|V_2|$ dummy nodes $\varepsilon$ to allow for insertion and deletions. Let the same hold for $V_2^+$, with $|V_1|$ dummy nodes. The *graph edit distance* (GED) between $G_1$ and $G_2$ is defined by

$$GED(G_1, G_2) = \min_{\pi \in \Pi} \sum_{v_i \in V_1^+} c_v(v_i, \pi(v_i)) + \sum_{v_i, v_j \in V_1} c_e(v_i, v_j, \pi(v_i), \pi(v_j)),$$

where $\Pi$ denotes the set of bijections from $V_1^+$ to $V_2^+$, and $c_v$ denotes the cost for node edits and $c_e$ for edge edits.

The graph edit distance, which is closely related to the Fused Gromov-Wasserstein distance [34], is known to be NP-hard [18]. The *bipartite graph matching* heuristic [32] to the graph edit distance is a heuristic technique to get approximate GEDs in polynomial time. The main idea is to transform the problem into a linear assignment problem, which is well-known to be polynomial-time solvable (e.g. with the Hungarian algorithm), by disregarding edge edit operations. Let $n = |V_1| + |V_2|$. Let $\Pi$ be the set of matrices $\pi \in \mathbb{R}_+^{n \times n}$ such that $\pi \mathbf{1}_n = \mathbf{1}_n$, $\pi^T \mathbf{1}_n = \mathbf{1}_n$. Then, the linear assignment problem to be solved is $d_C(G_1, G_2) := \min_{\pi \in \Pi} \langle C, \pi \rangle$ with $\langle \cdot, \cdot \rangle$ being the Frobenius inner product, and with node-edit cost matrix $C$. Many approximation algorithms to the GED modify the node assignment costs in order to account for edge edit operations [35]. The common technique is to represent nodes as rooted substructures, such as neighborhoods [36] or subgraphs [37], and to compute the assignment costs accordingly. In fact, in this work we generalize this approach by adding an entropic regularization to the linear assignment problem in order to make it differentiable.

## 3. Methods

### 3.1. GMSM Architecture

We describe the architecture of the method GMSM, which is designed to learn expressive and interpretable substitution matrices for biological structures and, at the same time, when given two such graph-represented structures, to output an interpretable alignment of the two graphs based on such substitution matrices.

Taking inspiration from the bipartite graph matching heuristic for graph edit distance, we represent our graphs as bag of *learnable* node features, and compute the edit distance between graphs by computing the optimal assignment between such features. More formally, given a graph $G$ we repre-

sent each node $v \in V$ as a parametric function $\psi_\theta(v, G)$, which is implemented by a graph neural network (GNN) parametrized by $\theta$. For a pair of graphs $G_1$, $G_2$, the set of node features is computed using the same parametric function, a *siamese network* [38], as shown in Figure 1.

Then, the cost of substituting $v$ with $u$ is given by $c_{u,v} = \|\psi_\theta(v, G_1) - \psi_\theta(u, G_2)\|_2$. Node insertion and deletion costs are obtained by computing the distance to a learnable embedding for a dummy isolated node $\varepsilon$. In particular, given two graphs $G_1, G_2$, the model computes the corresponding edit cost matrix $C_{G_1,G_2}(\psi_\theta) = (c_{u,v})_{u \in V_1^+, v \in V_2^+}$, which is a submatrix of a global matrix $C(\psi_\theta)$.

Finally, the matching between the node features is computed by solving an optimal transport problem [27], [28], regularized with the entropic term $\Omega(\pi) = \sum_{ij} \pi_{ij} \log \pi_{ij}$:

$$\pi^* = \operatorname{argmin}_{\pi \in \Pi} \langle C_{G_1,G_2}(\psi_\theta), \pi \rangle + \varepsilon \Omega(\pi),$$
$$d_{\psi_\theta,\varepsilon}(G_1, G_2) := \langle C_{G_1,G_2}(\psi_\theta), \pi^* \rangle.$$

As shown in [26], the function $d_{\psi_\theta,0}(\cdot,\cdot)$ is symmetric and satisfies the triangle inequality.

Unlike the original version of the model, which was limited to $\varepsilon = 0$ and had $O\left(\max(n_1, n_2)^3\right)$ time complexity, the regularized version of GMSM can run in $O\left(\max(n_1, n_2)^2\right)$ time [28], allowing for faster training.

Note that the function $d_{\psi_\theta,\varepsilon}$ depends on the size of the graphs that are being compared. To avoid biasing the model into considering smaller graphs as more similar, we normalize the distance by the sum of the number of nodes of the two graphs [22] and use the *graph dissimilarity* $\hat{d}(G_1, G_2) = (|V_1| + |V_2|)^{-1} d_{\psi_\theta,\varepsilon}(G_1, G_2)$ as the output of the GMSM model, rather than the unnormalized distance.

### 3.2. Training GMSM

Informally, we want that the graph dissimilarity induced by the learned substitution matrix $C(\psi_\theta)$ correlates with the functional labels at hand. In particular, we would like for graphs belonging to the same class to have low distance, and for graphs belonging to different classes to have higher distance, as common in the *metric learning* setting [39]. In particular, we use the *margin loss* proposed in [40].

Crucially, since one wants the cost matrix $C := C_{G_1,G_2}(\psi_\theta)$ to be *learnt* based on the data at hand, the graph dissimilarity should have a well-defined gradient with respect to $C$, to allow for the optimization of the loss function. The original model GMSM, which sets $\varepsilon = 0$, relied on the fact that when the solution to the optimal transport problem is unique, the dissimilarity is differentiable with respect to $C$ by Danskin's theorem. In fact, by adding a non-zero entropic regularization, the assignment problem becomes convex and everywhere differentiable [28], [41].

**Proposition 3.1.** *Let* $\pi^* = \operatorname{argmin}_{\pi \in \Pi} \langle C, \pi \rangle + \varepsilon \Omega(\pi)$ *and* $d(C) = \langle C, \pi^* \rangle$. *Then* $d(C)$ *is differentiable with gradient*

$$\nabla_C d(C) = \operatorname{argmin}_{\pi \in \Pi} \langle C, \pi \rangle + \varepsilon \Omega(\pi).$$

When $\varepsilon = 0$, we define $\pi_A^*(C)$ the solution returned by the assignment algorithm $A$ (e.g. Hungarian algorithm). We define $\tilde{\nabla}_C d(C) = \pi_A^*(C)$ and apply gradient descent to GMSM with this re-defined gradient. When $\varepsilon > 0$, we instead use the true gradient.

## 4. Experimental Evaluation

In this section, we provide experimental evidence that the substitution matrices learned by our method indeed distill useful information about the conditioning priors on which it was trained, and in particular that the entropic regularization can help the learning task. The code is publicly available[1].

### 4.1. Experimental setup

We tackle the similarity-based classification task. Given two graphs, the task is to predict whether or not they belong to the same class, solely as a function of their learned distance. In particular, we evaluate the triplet accuracy and the pair AUROC, smilarly to [23]. The goal of this task is to evaluate whether the learned dissimilarity between graphs correlates, at both the short range and long range scale, with the conditioning priors on which it was trained on. We focus here on datasets of small molecules (see Section C.2), but the method can be applied also to, e.g., proteins and RNA [26].

We ask how the substitution matrices learned by GMSM compare to alternative architectures, isolating the effects of graph-matching, structure-awareness and prior information. To this end, we report the result of three baseline architectures: WL kernel, which is an approach that only captures structural information, Siamese-GNN, which has the same GNN architecture, but replaces the graph matching step by computing a graph-level pooled embedding for each graph and by using the distance between such embeddings, and GMSM with fixed uniform costs between all nodes. Moreover, we report the results for GMSM both with and without entropic regularization during training. Since the goal is to obtain interpretable cost matrices between simple substructures, we fix the number of message passing layers in GMSM to 2. For the sake of a fair comparison, we do the same for the baselines. In order to allow the model to output one-to-one graph matchings, we set $\varepsilon = 0$ at inference time.

Note that this experiment is meant to isolate the effect of different algorithmic components on the task rather than to maximize molecule property prediction performance.

[1] github.com/BorgwardtLab/GraphMatchingSubstitutionMatrices

*Table 1.* Similarity-based classification on small molecules

| Method | Mutagenicity | | NCI1 | | AIDS | |
|---|---|---|---|---|---|---|
| | Trip. acc. | AUROC | Trip. acc. | AUROC | Trip. acc. | AUROC |
| WL kernel | 0.535 $\pm 0.004$ | 0.531 $\pm 0.003$ | 0.519 $\pm 0.005$ | 0.516 $\pm 0.004$ | 0.411 $\pm 0.008$ | 0.427 $\pm 0.005$ |
| Siamese-GNN | 0.720 $\pm 0.005$ | 0.729 $\pm 0.005$ | 0.693 $\pm 0.005$ | 0.708 $\pm 0.004$ | 0.981 $\pm 0.001$ | 0.993 $\pm 0.000$ |
| GMSM (uniform costs) | 0.518 $\pm 0.004$ | 0.520 $\pm 0.004$ | 0.550 $\pm 0.004$ | 0.543 $\pm 0.004$ | 0.944 $\pm 0.003$ | 0.933 $\pm 0.002$ |
| GMSM ($\varepsilon = 0$) | 0.655 $\pm 0.005$ | 0.652 $\pm 0.005$ | 0.677 $\pm 0.003$ | 0.675 $\pm 0.004$ | 0.980 $\pm 0.001$ | 0.988 $\pm 0.000$ |
| GMSM ($\varepsilon \geq 0$) | 0.663 $\pm 0.005$ | 0.659 $\pm 0.005$ | 0.689 $\pm 0.003$ | 0.686 $\pm 0.004$ | 0.981 $\pm 0.002$ | 0.986 $\pm 0.000$ |

### 4.2. Similarity-based classification

We see in Table 1 that GMSM without entropic regularization ($\varepsilon = 0.0$) outperforms the simple WL kernel and uniform-cost baselines, showing that training on class labels yields indeed task-specific edit costs, as expected. Moreover, the Siamese-GNN baseline, which is not forced to assign similarities based on a graph matching, is more expressive and yields better results, at the expense of not providing the interpretable edit costs, which are the focus of this work.

Allowing for entropic regularization during training provides a performance boost, suggesting that the smoothness of the objective leads to better convergence.

### 4.3. Regularization at inference time

If one is willing to sacrifice getting one-to-one graph matchings in output, the model can be run with entropic regularization also at inference time. As shown in Section D.1, this can yield better classification performance.

### 4.4. Retrieval

We also investigate, in Section D.2, the retrieval task, where the model returns the graphs in a database that are most similar to a query graph, testing the quality of the learned graph dissimilarity at short scales. Results show that entropic regularization can yield better results, although the improvement is not as marked as in the classification task.

## 5. Analysis of the edit cost matrices

Figure 2 shows an example of two chemicals from the Mutagenicity dataset that have a very different graph topology, and therefore would be classified as dissimilar by method such as the WL kernel, but that have the same label. In this case, they are both known to be mutagenic. In particular, GMSM (trained with $\varepsilon = 1.0$) learns to assign a low graph dissimilarity between the two chemicals.

This example showcases the explainability of GMSM. Indeed, we can observe that the substructures containing halogens (Br and Cl) have very low edit costs. We hypothesize

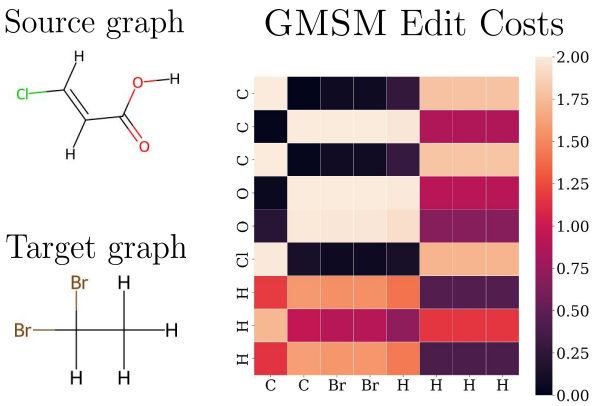


*Figure 2.* Two sample chemicals from the Mutagenicity dataset, shown on the left, and the edit cost matrix calculated by GMSM.

this is due to the known mutagenic effect of halogens [42] and thus substituting halogens would not be likely to affect the class label.

In general, these interpretable edit cost matrices offer insights into the substructures driving the function at hand, showcasing the potential of GMSM as tool for hypothesis generation in biochemistry.

At the same time, we notice that these matrices reflect only the way the particular instance of the model classifies pairs of graphs, and is therefore subject to change depending on the model architecture, the training data, and the initial parameters of the model.

## 6. Conclusion

In this paper, we have extended the GMSM model, which allows to obtain task-specific substitution matrices for biochemical structures, to use entropy-regularized optimal transport in its graph-matching module. We show that this modification allows to lower the time complexity of the model and at the same time to make its output differentiable with respect to the model parameters. This in turn allows the model to converge more easily to good solutions.

## References


[1] O. Gotoh, "Multiple sequence alignment: Algorithms and applications," *Advances in biophysics*, vol. 36, pp. 159–206, 1999.

[2] D. Song, J. Chen, G. Chen, *et al.*, "Parameterized blosum matrices for protein alignment," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 12, no. 3, pp. 686–694, 2014.

[3] W. J. Wilbur, "On the PAM matrix model of protein evolution.," *Molecular biology and evolution*, vol. 2, no. 5, pp. 434–447, 1985.

[4] A. C. Eklund, P. Friis, R. Wernersson, and Z. Szallasi, "Optimization of the blastn substitution matrix for prediction of non-specific DNA microarray hybridization," *Nucleic acids research*, vol. 38, no. 4, e27–e27, 2010.

[5] I. L. Hofacker, S. H. Bernhart, and P. F. Stadler, "Alignment of RNA base pairing probability matrices," *Bioinformatics*, vol. 20, no. 14, pp. 2222–2227, 2004.

[6] R. P. Sheridan, "The most common chemical replacements in drug-like compounds," *Journal of chemical information and computer sciences*, vol. 42, no. 1, pp. 103–108, 2002.

[7] S. Henikoff and J. G. Henikoff, "Amino acid substitution matrices," *Advances in protein chemistry*, vol. 54, pp. 73–98, 2000.

[8] C.-H. Tung, J.-W. Huang, and J.-M. Yang, "Kappa-alpha plot derived structural alphabet and blosum-like substitution matrix for rapid search of protein structure database," *Genome biology*, vol. 8, no. 3, pp. 1–16, 2007.

[9] M. van Kempen, S. S. Kim, C. Tumescheit, *et al.*, "Fast and accurate protein structure search with Foldseek," *Nature Biotechnology*, pp. 1–4, 2023.

[10] F. Keul, M. Hess, M. Goesele, and K. Hamacher, "Pfasum: A substitution matrix from Pfam structural alignments," *BMC bioinformatics*, vol. 18, pp. 1–14, 2017.

[11] R. A. Sutormin, A. B. Rakhmaninova, and M. S. Gelfand, "Batmas30: Amino acid substitution matrix for alignment of bacterial transporters," *Proteins: Structure, Function, and Bioinformatics*, vol. 51, no. 1, pp. 85–95, 2003.

[12] J. M. Koshi and R. A. Goldstein, "Context-dependent optimal substitution matrices," *Protein Engineering, Design and Selection*, vol. 8, no. 7, pp. 641–645, 1995.

[13] V. Gligorijević, P. D. Renfrew, T. Kosciolek, *et al.*, "Structure-based protein function prediction using graph convolutional networks," *Nature communications*, vol. 12, no. 1, p. 3168, 2021.

[14] M. Kulmanov and R. Hoehndorf, "DeepGOPlus: Improved protein function prediction from sequence," *Bioinformatics*, vol. 36, no. 2, pp. 422–429, 2020.

[15] C. Oliver, V. Mallet, R. S. Gendron, *et al.*, "Augmented base pairing networks encode RNA-small molecule binding preferences," *Nucleic acids research*, vol. 48, no. 14, pp. 7690–7699, 2020.

[16] S. Chithrananda, G. Grand, and B. Ramsundar, "Chemberta: Large-scale self-supervised pretraining for molecular property prediction," *arXiv preprint arXiv:2010.09885*, 2020.

[17] F. Llinares-López, Q. Berthet, M. Blondel, O. Teboul, and J.-P. Vert, "Deep embedding and alignment of protein sequences," *Nature Methods*, vol. 20, no. 1, pp. 104–111, 2023.

[18] H. Bunke and K. Riesen, "Graph edit distance–optimal and suboptimal algorithms with applications," *Analysis of Complex Networks: From Biology to Linguistics*, pp. 113–143, 2009.

[19] M. Neuhaus and H. Bunke, "A probabilistic approach to learning costs for graph edit distance," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, IEEE, vol. 3, 2004, pp. 389–393.

[20] Y. Bai, H. Ding, S. Bian, T. Chen, Y. Sun, and W. Wang, "Simgnn: A neural network approach to fast graph similarity computation," in *Proceedings of the twelfth ACM international conference on web search and data mining*, 2019, pp. 384–392.

[21] R. Ranjan, S. Grover, S. Medya, V. Chakaravarthy, Y. Sabharwal, and S. Ranu, "Greed: A neural framework for learning graph distance functions," *Advances in Neural Information Processing Systems*, vol. 35, pp. 22 518–22 530, 2022.

[22] K. D. Doan, S. Manchanda, S. Mahapatra, and C. K. Reddy, "Interpretable graph similarity computation via differentiable optimal alignment of node embeddings," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 665–674.

[23] Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli, "Graph matching networks for learning the similarity of graph structured objects," in *International conference on machine learning*, PMLR, 2019, pp. 3835–3845.

[24] P. Riba, A. Fischer, J. Lladós, and A. Fornés, "Learning graph edit distance by graph neural networks," *Pattern Recognition*, vol. 120, p. 108 132, 2021.

[25] J. Heo, S. Lee, S. Ahn, and D. Kim, "EPIC: Graph augmentation with edit path interpolation via learnable cost," *arXiv preprint arXiv:2306.01310*, 2023.

[26] P. Pellizzoni, C. Oliver, and K. Borgwardt, "Structure- and function-aware substitution matrices via learnable graph matching," in *Research in Computational Molecular Biology*, 2024.

[27] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport," in *Advances in Neural Information Processing Systems*, vol. 26, 2013.

[28] G. Peyré and M. Cuturi, *Computational optimal transport*, 2020. arXiv: 1803.00567 [stat.ML].

[29] N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy, "Optimal transport for domain adaptation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 9, pp. 1853–1865, 2017.

[30] M. Togninalli, E. Ghisu, F. Llinares-López, B. Rieck, and K. Borgwardt, "Wasserstein weisfeiler-lehman graph kernels," in *Advances in Neural Information Processing Systems*, 2019.

[31] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning*, D. Precup and Y. W. Teh, Eds., ser. Proceedings of Machine Learning Research, vol. 70, PMLR, Jun. 2017, pp. 214–223.

[32] S. Bougleux, L. Brun, V. Carletti, P. Foggia, B. Gaüzère, and M. Vento, "Graph edit distance as a quadratic assignment problem," *Pattern Recognition Letters*, vol. 87, pp. 38–46, 2017, Advances in Graph-based Pattern Recognition, ISSN: 0167-8655.

[33] H. Bunke and K. Riesen, "Graph classification based on dissimilarity space embedding," in *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshop, SSPR & SPR 2008, Orlando, USA, December 4-6, 2008. Proceedings*, Springer, 2008, pp. 996–1007.

[34] V. Titouan, N. Courty, R. Tavenard, C. Laetitia, and R. Flamary, "Optimal transport for structured data with application on graphs," in *Proceedings of the 36th International Conference on Machine Learning*.

[35] D. B. Blumenthal, N. Boria, J. Gamper, S. Bougleux, and L. Brun, "Comparing heuristics for graph edit distance computation," *The VLDB journal*, vol. 29, no. 1, pp. 419–458, 2020.

[36] D. B. Blumenthal and J. Gamper, "Improved lower bounds for graph edit distance," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 3, pp. 503–516, 2017.

[37] V. Carletti, B. Gaüzere, L. Brun, and M. Vento, "Approximate graph edit distance computation combining bipartite matching and exact neighborhood substructure distance," in *Graph-Based Representations in Pattern Recognition: 10th IAPR-TC-15 International Workshop, GbRPR 2015, Beijing, China, May 13-15, 2015. Proceedings 10*, Springer, 2015, pp. 188–197.

[38] D. Chicco, "Siamese neural networks: An overview," *Artificial neural networks*, pp. 73–94, 2021.

[39] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, IEEE, vol. 2, 2006, pp. 1735–1742.

[40] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krahenbuhl, "Sampling matters in deep embedding learning," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2840–2848.

[41] M. Cuturi, O. Teboul, and J.-P. Vert, "Differentiable ranking and sorting using optimal transport," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[42] H. Brem, A. B. Stein, and H. S. Rosenkranz, "The mutagenicity and dna-modifying effect of haloalkanes," *Cancer research*, vol. 34, no. 10, pp. 2576–2579, 1974.

[43] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" In *International Conference on Learning Representations*, 2018.

[44] C. Morris, M. Ritzert, M. Fey, *et al.*, "Weisfeiler and leman go neural: Higher-order graph neural networks," 2019.

[45] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[46] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[47] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann, "Tudataset: A collection of benchmark datasets for learning with graphs," *arXiv preprint arXiv:2007.08663*, 2020.

[48] J. Kazius, R. McGuire, and R. Bursi, "Derivation and validation of toxicophores for mutagenicity prediction," *Journal of medicinal chemistry*, vol. 48 1, pp. 312–20, 2005.

[49] K. Riesen and H. Bunke, "IAM graph database repository for graph based pattern recognition and machine learning," in *Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshop, SSPR & SPR 2008, Orlando, USA, December 4-6, 2008. Proceedings*, N. da Vitoria Lobo, T. Kasparis, F. Roli, *et al.*, Eds., ser. Lecture Notes in Computer Science, vol. 5342, Springer, 2008, pp. 287–297.

[50] N. Wale and G. Karypis, "Comparison of descriptor spaces for chemical compound retrieval and classification," in *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM)*, IEEE Computer Society, 2006, pp. 678–689.

# A. Additional background

## A.1. Optimal transport

In the discrete version of optimal transport, we consider two finite sets of with associated weights. Let $\{a_i\}_{i=1}^m$ and $\{b_j\}_{j=1}^n$ be two such sets with corresponding weight vectors, or probability distributions, $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{v} \in \mathbb{R}^n$ such that $u^\top \mathbf{1}_n = 1$ and $v^\top \mathbf{1}_m = 1$. Here, $\mathbf{1}_n$ and $\mathbf{1}_m$ are vectors of ones of appropriate dimensions. The goal is to find a transportation plan $\pi \in \mathbb{R}_+^{m \times n}$ that transfers the mass from $\{a_i\}$ to $\{b_j\}$ while minimizing the total transportation cost. This can be formulated as the linear program:

$$d_C(\mathbf{u}, \mathbf{v}) = \min_{\pi \in \mathbb{R}_+^{m \times n}} \langle \pi, C \rangle,$$

subject to the constraints $\pi \mathbf{1}_n = \mathbf{u}$ and $\pi^\top \mathbf{1}_m = \mathbf{v}$, where $c_{ij}$ represents the cost of transporting mass from $a_i$ to $b_j$ and $C = (c_{ij})_{i=1,\dots,m;j=1,\dots,n}$. In [27] it was proposed an entropy-regularized version of the optimal transport problem, which allows the resulting distance to be differentiable with respect to $C$ and to solve the problem in quadratic time using the Sinkhorn algorithm.

A closely related problem is the linear assignment problem, where the sets have the same size, i.e. $n = m$, and the weight vectors are defined as $u = v = \mathbf{1}_n$. Then, the feasible assignments are the set $\Pi$ of matrices $\pi \in \mathbb{R}_+^{n \times n}$ such that $\pi \mathbf{1}_n = \mathbf{1}_n$, $\pi^T \mathbf{1}_n = \mathbf{1}_n$. The task is then to find $\min_{\pi \in \Pi} \langle \pi, C \rangle$. The problem can be solved in cubic time, e.g., with the Hungarian algorithm.

## A.2. Graph neural networks

Message passing graph neural networks (GNNs), given a graph $G$, iteratively produce for each node $v \in V_G$, at each level $k = 1, \dots, K$, the embeddings $h_v^k \in \mathbb{R}^{d_k}$ as $h_v^k = f_{\text{upd}}\left(h_v^{k-1}, f_{\text{agg}}\left(\{h_u^{k-1} : u \in \mathcal{N}(v)\}\right)\right)$, where $f_{\text{agg}}$ and $f_{\text{upd}}$ are the aggregate and the update operations, respectively. The first layer of the GNN is fed with the initial node embeddings $h_v^0$, e.g. one-hot encodings of the node labels.

In [43] it was shown that there exist injective functions $f_{\text{agg}}$, $f_{\text{upd}}$ yielding GNNs that are provably as expressive as color refinement.

An example of such functions that leads to models that are provably as expressive as color refinement [44], denoting $\|$ as concatenation, is

$$h_v^k = \text{mlp}\Big(h_v^{k-1} \Big\| \sum_{u \in \mathcal{N}(v)} h_u^{k-1}\Big) \in \mathbb{R}^{d_k}.$$

In fact, provided that different node labels are encoded to linearly independent $h_v^0$'s, even the following simpler architecture, denoting with $\sigma$ a nonlinear function such as ReLU, is as expressive as color refinement [44]:

$$h_v^k = \sigma\Big(W_1^k h_v^{k-1} + W_2^k \sum_{u \in \mathcal{N}(v)} h_u^{k-1}\Big) \in \mathbb{R}^{d_k}. \tag{1}$$

We then let the final node embedding be a function of the per-layer node embeddings as $\psi(v, G) = f(\{h_v^k : k = 0, \dots, K\})$. We denote $(v, G_1) =_\psi (u, G_2)$ if $\psi(v, G_1) = \psi(u, G_2)$, and denote by $\mathcal{V}_{\psi_\theta}$ the set of equivalence classes induced by $=_\psi$. Moreover, we will denote, with abuse of notation, $G_1 =_\psi G_2$ if $\{\{\psi(v, G_1 = (V_1, E_1)) : v \in V_1\}\} = \{\{\psi(v, G_2 = (V_2, E_2)) : v \in V_2\}\}$, and denote for brevity with $\mathcal{G}_\psi$ the set $\mathcal{G} \setminus =_\psi$ of equivalence classes induced by $=_\psi$ on graphs.

## A.3. Heuristics for the graph edit distance

The *bipartite graph matching* heuristic [32] to the graph edit distance is an heuristic technique to get approximate GEDs in polynomial time. The main idea is to transform the problem into a linear assignment problem, which is well-known to be polynomial-time solvable (e.g. with the Hungarian algorithm), by disregarding edge edit operations and only considering node edit ones.

Then, the linear assignment problem to be solved is

$$d_C(G_1, G_2) := \min_{\pi \in \Pi} \langle C, \pi \rangle = \min_{\pi \in \Pi} \sum_{v_i \in V_1^+} c_{v_i, \pi(v_i)},$$

with $\langle \cdot, \cdot \rangle$ being the Frobenius inner product, and with cost matrix $C$ defined as

$$C = \left[\begin{array}{ccc|ccc} c_{1,1} & \cdots & c_{1,n_2} & c_{1,\varepsilon} & \cdots & \infty \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ c_{n_1,1} & \cdots & c_{n_1,n_2} & \infty & \cdots & c_{n_1,\varepsilon} \\ \hline c_{\varepsilon,1} & \cdots & \infty & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \infty & \cdots & c_{\varepsilon,n_2} & 0 & \cdots & 0 \end{array}\right]$$

where $c_{i,j}$ denotes the cost of a node substitution $v_i \to v_j$, $c_{i,\varepsilon}$ denotes the cost of a node deletion $u_i \to \varepsilon$, and $c_{\varepsilon,j}$ denotes the costs of a node insertion $\varepsilon \to v_j$ with $v_i \in V_1$ and $v_j \in V_2$; and where $\Pi$ is the set of permutation matrices of size $|V_1| + |V_2|$, representing bijections from $V_1^+$ to $V_2^+$.

If one chooses $c_{i,j} = c_v(v_i, v_j)$, i.e. setting the cost of assigning node $v_i$ to node $v_j$ as the node edit cost in the original problem, then $d_C(G_1, G_2)$ lower bounds $GED(G_1, G_2)$.

Many approximation algorithms to the GED modify the node assignment costs in order to account for edge edit operations [35]. The common technique is to represent the nodes as some local substructure around them, such as neighborhoods [36] or subgraphs [37], and to compute the assignment costs accordingly. In many practical applications, representing the graph as such a bag of local structures is enough to compute a good approximation to the true edit distance [35].

In fact, in this work we generalize this approach by representing nodes as their GNN embeddings and by adding an entropic regularization to the linear assignment problem in order to make it differentiable.

### A.4. Metric learning

Common losses for metric learning are the contrastive loss [39] and the triplet loss [45], which act on pairs and triplets of graphs, respectively. We use the *margin loss* proposed in [40], which reportedly yields better results than the contrastive one. Moreover, it can work with randomly sampled pairs, while the triplet loss usually requires hard or semi-hard sample mining to work properly [40].

In particular, for a pair of graphs $G_1, G_2$, the loss is defined as

$$\ell_{margin}(G_1, G_2) = \max\left(0, \alpha + y(\hat{d}_C(G_1, G_2) - \beta)\right),$$

with $y = 1$ for positive pairs, i.e. both graphs belonging to the same class, and $y = -1$ for negative pairs. This loss strives to push the graph dissimilarities of graphs belonging to the same class to be less than $\beta - \alpha$ and the dissimilarities of graphs belonging to different classes to be more than $\beta + \alpha$. In our experiments, we set $\beta = 0.5$ and $\alpha = 0.1$.

## B. Architecture of GMSM

Our model, GMSM, is implemented as follows. The GNN message passing layers are realized by Equation 1, with the nonlinearity realized by a ReLU. We fix the number of layers to $K = 2$. The final node embeddings $\psi(v, G)$ are obtained by concatenating the embeddings for each layer and by normalizing them, i.e. $h_v = \Big\|_{k=0}^{K} h_v^k$ and $\psi(v, G) = h_v / \|h_v\|_2$.

Then, the cost of substituting the structure rooted at $v$ with the one rooted at $u$ is given by $c_{u,v} = \|\psi_\theta(v, G_1) - \psi_\theta(u, G_2)\|_2$. Node insertion and deletion costs are obtained by computing the distance to a learnable embedding for a dummy isolated node $\varepsilon$, that is $c_{v,\varepsilon} = c_{\varepsilon,v} = \|\psi_\theta(v, G_1) - \psi_\theta(\varepsilon)\|_2$. Note that the normalization of the embeddings makes it so that the edit costs are bounded in $[0, 2]$.

This yields a global learnable substitution matrix $C(\psi_\theta) = (c_{u,v})_{u,v \in \mathcal{V}_{\psi_\theta}}$ of costs between elements of $\mathcal{V}_{\psi_\theta} \cup \{\varepsilon\}$, which correspond to nodes and their rooted subgraph explored by the GNN. In particular, given two graphs $G_1, G_2$, the model computes the corresponding edit cost matrix $C_{G_1,G_2}(\psi_\theta) = (c_{u,v})_{u \in V_1^+, v \in V_2^+}$, which is a submatrix of $C(\psi_\theta)$.

Finally, the matching between the node features is computed by solving an optimal transport problem [27], [28], regularized with the entropic term $\Omega(\pi) = \sum_{ij} \pi_{ij} \log \pi_{ij}$:

$$\pi^* = \text{argmin}_{\pi \in \Pi} \langle C_{G_1,G_2}(\psi_\theta), \pi \rangle + \varepsilon \Omega(\pi),$$

$$d_{\psi_\theta,\varepsilon}(G_1, G_2) := \langle C_{G_1,G_2}(\psi_\theta), \pi^* \rangle.$$

Finally, the model outputs the graph dissimilarity $\hat{d}(G_1, G_2) = (|V_1| + |V_2|)^{-1} d_{\psi_\theta,\varepsilon}(G_1, G_2)$, as well as the edit cost matrix and the transport plan.

With respect ot the original work [26], we substituted the GNN layers from GAT layers [46] to layers given by Equation 1 [44] and we concatenate the embedding of all the layers. This yields a provably more expressive model [44].

## C. Experimental setup

The model is trained taking the same number of random positive and negative pairs from the training set. In particular, for each anchor graph we sample positive pairs by selecting 10 random graphs with the same label, and for each each of such positive pairs we sample negative pairs by selecting 10 random graphs with different label. The model parameters are optimized with respect to the margin loss, using $\beta = 0.5$ and $\alpha = 0.1$, with the Adam optimizer. We set the learning rate to $0.001$. The best model is selected using the validation set pair AUROC metric on the classification task, using early stopping. For the GMSM with regularization, we choose $\varepsilon \in \{0.1, 1.0\}$ based on the validation set pair AUROC metric. In particular, for the `Mutagenicity` and `NCI1` datsets, the highest validation AUROC is obtained for $\varepsilon = 1.0$, while for the `AIDS` datset it is obtained for $\varepsilon = 0.1$. These models are then used at inference time on both the similarity-based classification task and the retrieval task. The models are tested using the same sampling strategy for random positive and negative pairs from the validation and test set.

### C.1. Tasks

The first task we tackle is the similarity-based classification task. Given two graphs, the task is to predict whether or not they belong to the same class, solely as a function of their learned distance. In particular, we evaluate two metrics. The first one is the triplet accuracy. Namely, given an anchor graph, one graph from the same class of the anchor (positive pair) and one graph from another class (negative pair), the triplet of graphs is considered a successful prediction if the distance of the positive pair is lower than the one of the negative pair. The second is pair AUROC, the area under the ROC curve for classifying pairs of graphs as similar or not based on a distance threshold. The goal of this task is to evaluate whether the learned dissimilarity between graphs correlates, at both the short range and long range scale, with the conditioning priors on which it was trained on. We report the means and standard deviations over 5 different random samples from the training set.

The second task we evaluate is the retrieval task. In particular, given a query graph, the task is to return a set of graphs that are the most similar to the query one. The goal of this task is to evaluate the quality of the learned graph similarity at very short scales. Indeed, for the retrieval task it does not matter if some positive pairs are at a high distance as long as there are enough positive pairs at a very short distance, which will be returned as hits by the retrieval procedure. In particular, we take as queries the graphs of the test set, and search for the hits in the training set, which serves then as the searchable database. We report the precision@k (P@k), for $k \in \{10, 50\}$. In particular, we report the mean and the standard deviation over all queries.

### C.2. Datasets

In our experimental evaluation we three small molecule datasets. All datasets are split into training, validation and test sets at random and with ratios $\{0.8, 0.1, 0.1\}$.

The datasets are obtained from the TUDataset [47] and contain small molecules annotated with a label on mutagen activity (`Mutagenicity` [48], [49]), on activity against non-small cell lung cancer (`NCI1` [50]) and on evidence of anti-HIV activity (`AIDS` [49]).

# D. Additional experimental results

## D.1. Sensitivity to entropic regularization

Table 2 report the test set metrics for GMSM models that are trained and run at inference time with different levels of entropic regularization.

As shown by the results, adding an entropic regularization at inference time can boost the classification performance. In particular, for models trained with high regularization ($\varepsilon = 1.0$), using regularization also at inference time provides a significant increase in the metrics. This could be due to the fact that when one removes the regularization at inference time, there is shift in the distributions of the predicted graph dissimilarities. Maintaining the regularization, this shift is removed. For models trained with smaller or no regularization, this clear behavior does not appear. Adding regularization at inference time can nonetheless prove to be an effective hyperparameter to select with a validation set.

*Table 2.* Effect of entropic regularization at inference time

| Training | Inference | Mutagenicity | | NCI1 | | AIDS | |
|---|---|---|---|---|---|---|---|
| | | Trip. acc. | AUROC | Trip. acc. | AUROC | Trip. acc. | AUROC |
| GMSM ($\varepsilon = 0.0$) | $\varepsilon = 0.0$ | $0.655_{\pm 0.005}$ | $0.652_{\pm 0.005}$ | $0.677_{\pm 0.003}$ | $0.675_{\pm 0.004}$ | $0.980_{\pm 0.001}$ | $0.988_{\pm 0.000}$ |
| | $\varepsilon = 0.1$ | $0.663_{\pm 0.003}$ | $0.661_{\pm 0.004}$ | $0.664_{\pm 0.004}$ | $0.661_{\pm 0.004}$ | $0.977_{\pm 0.001}$ | $0.985_{\pm 0.000}$ |
| | $\varepsilon = 1.0$ | $0.651_{\pm 0.004}$ | $0.648_{\pm 0.005}$ | $0.660_{\pm 0.004}$ | $0.658_{\pm 0.005}$ | $0.984_{\pm 0.001}$ | $0.988_{\pm 0.000}$ |
| GMSM ($\varepsilon = 0.1$) | $\varepsilon = 0.0$ | $0.651_{\pm 0.005}$ | $0.648_{\pm 0.005}$ | $0.679_{\pm 0.003}$ | $0.676_{\pm 0.004}$ | $0.981_{\pm 0.002}$ | $0.986_{\pm 0.000}$ |
| | $\varepsilon = 0.1$ | $0.660_{\pm 0.003}$ | $0.659_{\pm 0.004}$ | $0.676_{\pm 0.002}$ | $0.674_{\pm 0.004}$ | $0.979_{\pm 0.001}$ | $0.985_{\pm 0.000}$ |
| | $\varepsilon = 1.0$ | $0.646_{\pm 0.005}$ | $0.643_{\pm 0.005}$ | $0.662_{\pm 0.003}$ | $0.659_{\pm 0.005}$ | $0.984_{\pm 0.002}$ | $0.987_{\pm 0.000}$ |
| GMSM ($\varepsilon = 1.0$) | $\varepsilon = 0.0$ | $0.663_{\pm 0.005}$ | $0.659_{\pm 0.005}$ | $0.689_{\pm 0.003}$ | $0.686_{\pm 0.004}$ | $0.973_{\pm 0.001}$ | $0.980_{\pm 0.000}$ |
| | $\varepsilon = 0.1$ | $0.677_{\pm 0.004}$ | $0.673_{\pm 0.004}$ | $0.652_{\pm 0.003}$ | $0.643_{\pm 0.004}$ | $0.972_{\pm 0.001}$ | $0.980_{\pm 0.000}$ |
| | $\varepsilon = 1.0$ | $0.680_{\pm 0.004}$ | $0.682_{\pm 0.005}$ | $0.690_{\pm 0.003}$ | $0.692_{\pm 0.004}$ | $0.980_{\pm 0.001}$ | $0.984_{\pm 0.000}$ |

## D.2. Retrieval

In the retrieval task, we observe as in the classification task, that GMSM models are in general more expressive that the WL kernel and the baseline with uniform costs, although the latter performs surprisingly well on `AIDS`. Moreover, the Siamese-GNN model in general outperforms GMSM. Interestingly, the GMSM model trained with entropic regularization and selected based on the validation performance on the classification task, as described in Section C, preform very similarly to GMSM trained with no entropic regularization.

*Table 3.* Retrieval performance on small molecules

| Method | Mutagenicity | | NCI1 | | AIDS | |
|---|---|---|---|---|---|---|
| | P@10 | P@50 | P@10 | P@50 | P@10 | P@50 |
| WL kernel | $0.700_{\pm 0.276}$ | $0.643_{\pm 0.248}$ | $0.701_{\pm 0.256}$ | $0.627_{\pm 0.203}$ | $0.892_{\pm 0.180}$ | $0.802_{\pm 0.193}$ |
| Siamese-GNN | $0.785_{\pm 0.297}$ | $0.779_{\pm 0.293}$ | $0.777_{\pm 0.338}$ | $0.771_{\pm 0.332}$ | $0.988_{\pm 0.105}$ | $0.985_{\pm 0.116}$ |
| GMSM (uniform costs) | $0.646_{\pm 0.262}$ | $0.613_{\pm 0.215}$ | $0.597_{\pm 0.229}$ | $0.575_{\pm 0.179}$ | $0.998_{\pm 0.021}$ | $0.991_{\pm 0.073}$ |
| GMSM ($\varepsilon = 0$) | $0.773_{\pm 0.294}$ | $0.751_{\pm 0.271}$ | $0.773_{\pm 0.292}$ | $0.744_{\pm 0.278}$ | $0.991_{\pm 0.076}$ | $0.988_{\pm 0.090}$ |
| GMSM ($\varepsilon \geq 0$) | $0.774_{\pm 0.284}$ | $0.757_{\pm 0.268}$ | $0.772_{\pm 0.289}$ | $0.747_{\pm 0.268}$ | $0.988_{\pm 0.094}$ | $0.985_{\pm 0.111}$ |

## D.3. Stability of the learnt substitution matrices

The learnt substitution matrices depend on the model parameters that are learnt from the training data at hand, and therefore correspond in local minima in the loss function landscape. Because of this, they can depend on several factors, including the specific model architecture, the training data and the initialization of the parameters. For example, we observe that switching from the GAT-based architecture in the original GMSM paper [26] to the more expressive architecture we use here leads to different edit costs, e.g., for the graphs of Figure 2.