A Wasserstein Graph Kernel based on Substructure Isomorphism Problem of Shortest Paths

Jianming Huang[1,a) Zhongxi Fang[2,b) Hiroyuki Kasai[2,c)

Abstract: For graph classification tasks, many methods use a common strategy to aggregate information of vertex neighbors. Although this strategy provides an efficient means of extracting graph topological features, it brings excessive amounts of information that might greatly reduce its accuracy when dealing with large-scale neighborhoods. Learning graphs using paths or walks will not suffer from this difficulty, but many have low utilization of each path or walk, which might engender information loss and high computational costs. To solve this, we propose a graph kernel using a longest common subsequence (LCS kernel) to compute more comprehensive similarity between paths and walks, which resolves substructure isomorphism difficulties. We also combine it with optimal transport theory to extract more in-depth features of graphs.

1. Introduction

Graph-structured data have been used widely in various fields, such as chemoinformatics, bioinformatics, social networks, and computer vision [1], [2]. In graph-structured data classification tasks, a key difficulty is finding a criterion with which to compare graphs. However, no matter what method is used for comparison, graph isomorphism is a common difficulty that must be confronted and overcome: Do two graphs have identical topology? The degree to which two graphs mutually fit is usually regarded as similarity between two graphs. However, graph isomorphism has not been proved as NP complete. To date, no polynomial-time solution has been reported [3].

Various strategies are available to assess graph isomorphism. Some studies specifically find a relation between vertices and their neighbors. This type of strategy is a *breadth-first strategy*, which tends to construct a neighbor network surrounding a certain vertex. However, such breadth-first strategies entail several shortcomings. (1) Because breadth-first strategies are based on aggregating information of all neighbors, the embedding of a vertex usually involves excessive information, especially in a dense graph. (2) As the depth of a neighbor network increases, it grows geometrically, as does the quantity of information, especially when we want to assess the relationships of vertices that are mutually distant.

In other avenues of development, some studies have specifically examined paths and walks of graphs. In contrast to breadth-first strategies, they learn graph topology by comparing paths or walks. We consider these as *depth-first strategies*, which use a simple form of subgraphs to learn the subgraph structure. It becomes easier to solve problems presented by breadth-first strategies. Nevertheless, path-based (or walk-based) methods are adversely affected by the common truth that they entail high computational costs because sampled paths (or walks) are usually numerous and have high redundancy.

Because of computational improvements in solving optimal transport problems in recent years, optimal transport theory [4], [5] has been used in many machine-learning domains. Many studies have examined graph classification based on optimal transport theory. Our proposed method also uses optimal transport schemes to compute kernel values. More specifically, we computed optimal transport over two sets of paths.

To overcome shortcomings of breadth-first strategies, and to reduce high computational costs of current path-based methods, we propose path-comparison methods using the longest common subsequences (LCS) and optimal transport theory. We also apply this method to graph classification tasks, and present the LCS graph kernel. Generally, our contributions can be summarized as described below.

- We present a method to compare labeled graphs using the LCS of path sequences and optimal transport theory, which transforms the graph isomorphism problem to a sequence comparison problem.

[1] WASEDA University, Graduate School of Fundamental Science and Engineering
[2] WASEDA University, School of Fundamental Science and Engineering, Dept. of Communications and Computer Engineering
a) koukenmei@toki.waseda.jp
b) fzx@akane.waseda.jp
c) hiroyuki.kasai@waseda.jp

- We proposed a new strategy of comparing graphs based on the LCS metric space.

2. Related Work

For graph classification tasks, graph kernel methods have been used widely for several decades. They are also developing rapidly in recent years. Graph kernels are kernel functions that compute similarity between two graphs. For now, graph kernel methods can be generally divided into two categories as methods of (1) Traditional graph kernel and (2) OT-based graph kernel. The first of those classifications of methods includes traditional graph kernels, most of which are based on $\mathcal{R}$-convolution framework. To compute similarity between graphs in various data mining tasks, random walk kernel [6] has been developed and used widely as an important tool for graph classification. However, it also faces a difficulty of high computational cost. Subsequent work on Weisfeiler–Lehman graph kernel [7] has brought great success. They improved the original Weisfeiler–Lehman test using a form of multiple iteration, where neighbor patterns are aggregated. Although this method yields attractive performance, it still presents difficulties of breadth-first strategies and $\mathcal{R}$-convolution methods. The second class includes graph kernels combined with optimal transport theory. Recent research by [8], presents a Wasserstein-based Weisfeiler–Lehman graph kernel (WWL), which maps node embedding of a Weisfeiler–Lehman pattern to a feature space, and which computes kernel values using the Wasserstein distance of two point clouds in the feature space. They received better results than those yielded by the original Weisfeiler–Lehman kernel. GOT [9] uses optimal transport differently to compute the Wasserstein distance between two normal distributions derived by graph Laplacian matrices, instead of generating walks or comparing vertex neighbors in graphs. Another attractive work by [10] proposes the Fused Gromov–Wasserstein (FGW) distance which combines both the Wasserstein distance and the Gromov–Wasserstein distance.

3. Preliminaries

This section first introduces some notation and preliminary points of graphs. Hereinafter, we represent scalars as lower-case letters $(a, b, \ldots)$, vectors as bold typeface lower-case letters $(\boldsymbol{a}, \boldsymbol{b}, \ldots)$, and matrices as bold typeface capitals $(\mathbf{A}, \mathbf{B}, \ldots)$. An element at (i, j) of a matrix $\mathbf{A}$ is represented as $\mathbf{A}(i, j)$. $\mathbf{1}_n = (1, 1, \ldots, 1)^T \in \mathbb{R}^n$. We write $\mathbb{R}^n_+$ to denote non-negative n-dimensional vector. Also, $\mathbb{R}^{n\times m}_+$ represents a nonnegative matrix of size $n \times m$. The unit-simplex, simply called *simplex*, is denoted by Δ_n, which is the subset of $\mathbb{R}^n$ comprising all nonnegative vectors for which sums are 1. In addition, δ_x is the Dirac function at x.

3.1 Preliminaries of graphs

A graph is a pair $G = (\mathcal{V}, \mathcal{E})$ consisting of a set of n vertices (or nodes) $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ and a set of m edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. G is an undirected graph if a graph G includes only edges with no direction. The numbers of vertices and edges are, respectively, $|\mathcal{V}|$ and $|\mathcal{E}|$. If two vertices, say $v_i, v_j \in \mathcal{V}$, are connected by an edge e, then this edge is denoted as e_{ij}. These two vertices are said to be adjacent or neighbors. We consider only undirected graphs with no self-loop.

Given an undirected graph $G = (\mathcal{V}, \mathcal{E})$ and a vertex $v_i \in \mathcal{V}$, the degree of v_i in G, denoted as σ_i, is the number of edges incident to v_i. It is defined as $\sigma_i = |\{v_j : e_{ij} \in \mathcal{E}\}| = |\mathcal{N}(v_i)|$, where $\mathcal{N}(v_i)$ represents the neighborhood set of v_i.

A walk in a graph $G = (\mathcal{V}, \mathcal{E})$ is a sequence of vertices $v_1, v_2, ..., v_{k+1}$, where $v_i \in \mathcal{V}$ for all $1 \leq i \leq k+1$ and $v_i, v_{i+1} \in \mathcal{V}$ for all $1 \leq i \leq k$. The walk length is equal to the number of edges in the sequence, i.e., k in the case above. A walk in which $v_i \neq v_j \Leftrightarrow i \neq j$ is called a path. The path between the two adjacent vertices is equivalent to the edge. A set of paths between two non-adjacent vertices is denoted as $\mathcal{P}_{i,j}$. One path of $\mathcal{P}_{i,j}$ is denoted as $p_{i,j}$. Moreover, the length of $p_{i,j}$ is denoted as $|p_{i,j}|$. A shortest path, denoted as $p^\star_{i,j}$, from vertex v_i to vertex v_j of a graph G, is a path from v_i to v_j such that no other path exists between these two vertices with smaller length.

3.2 Longest common subsequence

A sequence is defined mathematically as an enumerated collection of objects of a certain order, where repetition of elements is allowed. The length of a sequence denotes the number of its elements. Given a sequence with length n, it can be written as $\boldsymbol{x} = (x_i)^n_{n=1}$, where x_i denotes the i-th element. A subsequence of a given sequence is derived by removing some elements of the original sequence under the premise that the order of the remaining elements is retained as the original sequence. For example, given a sequence of $(1, 2, 3, 4, 5)$, both $(2, 3, 4)$ and $(1, 3, 5)$ are subsequences of the original sequence. The definition of the longest common subsequence problem can be presented as shown below [11], [12].

Definition 1 (Longest Common Subsequence Problem). *The goal of a longest common subsequence problem is to find a sequence $\boldsymbol{x}^\star$ that satisfies the following conditions: (1) $\boldsymbol{x}^\star$ is a subsequence of each sequence in a set of sequences $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2 ..., \boldsymbol{x}_n\}$. (2) $\boldsymbol{x}^\star$ is the longest among all subsequences that satisfy condition (1). Then we consider $\boldsymbol{x}^\star$ as the longest common subsequence in $\mathcal{X}$.*

3.3 Optimal transport (OT)

Optimal transport provides a means of comparing probability distributions, which are histograms in the finite dimensional case [4], [5]. This quantity is defined over the *same ground space* or multiple pre-registered ground spaces. This comparison is interpreted as a *mass movement problem*, which seeks an optimum plan to move the mass from one distribution to the other at minimal cost. OT is rapidly gaining popularity in a multitude of machine learning problems, ranging from low-rank approximation, to dictionary learning, domain adaptation, clustering, and semi-supervised learning.

We define two simplexes of histograms with n_1 and n_2 on the same metric space, which are defined, respectively, as $\Delta_{n_1} = \{\boldsymbol{p} \in \mathbb{R}_+^{n_1}; \sum_i p_i = 1\}$, and $\Delta_{n_2} = \{\boldsymbol{q} \in \mathbb{R}_+^{n_2}; \sum_j q_j = 1\}$, where in *mass movement problem*, $\boldsymbol{p}, \boldsymbol{q}$ are usually regarded as mass vectors of each histogram, with elements denoting the mass of each bin. Subsequently, we define two probability measures $\mu = \sum_{i=1}^{n_1} p_i \delta_{x_i}$, and $\nu = \sum_{j=1}^{n_2} q_j \delta_{y_j}$, where $x_i \neq x_j$ for $i \neq j$ is assumed without loss of generality. We also consider the *ground cost* matrix $\mathbf{C} \in \mathbb{R}_+^{n_1 \times n_2}$, where $\mathbf{C}(i,j)$ represents the transportation cost between the i-th and j-th element. The optimal transport problem between these two histograms is defined as

$$\mathbf{T}^*(\mathbf{C}, \boldsymbol{p}, \boldsymbol{q}) = \underset{\mathbf{T} \in \mathcal{U}_{n_1 n_2}}{\arg\min} \langle \mathbf{T}, \mathbf{C} \rangle,$$

where $\mathcal{U}_{n_1 n_2}$ is defined as

$$\mathcal{U}_{n_1 n_2} := \left\{ \mathbf{T} \in \mathbb{R}_+^{n_1 \times n_2} : \mathbf{T}\mathbf{1}_{n_2} = \boldsymbol{p},\ \mathbf{T}^T \mathbf{1}_{n_1} = \boldsymbol{q} \right\},$$

and $\mathcal{U}_{n_1 n_2}$ represents the polytope of $n_1 \times n_2$ nonnegative matrices such that their row and column marginals are respectively equal to p_i and q_j. Then, the Wasserstein distance between the two measures, denoted as $\mathcal{W}(\mu, \nu)$, is equal to the total distance traversed by the mass under the optimal transport plan $\mathbf{T}^*$. Furthermore, by adding an entropic regularizer $H(\mathbf{T}) = -\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \mathbf{T}(i,j)(\log(\mathbf{T}(i,j)) - 1)$, a solution of the entropically regularized optimal transport problem is solvable efficiently using Sinkhorn's fixed-point iterations [13], [14], [15]. If no prior information is known about a space, then we set $\boldsymbol{p} = \frac{1}{n_1}\mathbf{1}_{n_1}$ and $\boldsymbol{q} = \frac{1}{n_2}\mathbf{1}_{n_2}$.

4. Longest Common Subsequence (LCS) Kernel

This section proposes a metric between two labeled graphs through comparison of all the shortest path sequences, and presents elaboration of the proposed longest common subsequence (LCS) kernel.

4.1 Basic concepts

This subsection first introduces several basic concepts in our LCS kernel. They show the operation of transforming the path into a manageable form of data, and a formula of path sequence similarity used in graph comparison.

4.1.1 Shortest path serialization

Given an undirected and connected graph $G = (\mathcal{V}, \mathcal{E})$ with a set of vertices $\mathcal{V} = \{v_i\}_{i=1}^N$ and a set of edges $\mathcal{E} = \{e_{ij}\}$, then both vertices and edges in G are assigned a categorical label. To describe the subgraph structure of G, we choose the shortest path set which contains all shortest paths in G.

Let $\mathcal{P}$ denote the shortest path set of G, $\mathcal{P}$ defined as

$$\mathcal{P} \triangleq \left\{ p_{i,j}^{\star} | \forall v_i, v_j \in \mathcal{V} \right\},$$

where $p_{i,j}^{\star}$ represents the shortest path from vertex v_i to v_j. Assuming that two vertices v_k, v_l are in the shortest path $p_{i,j}^{\star}$, except for the start vertex v_i and the end vertex v_j, then $p_{i,j}^{\star}$ can be expressed as

$$p_{i,j}^{\star} : v_i \xrightarrow{e_{i,k}} v_k \xrightarrow{e_{k,l}} v_l \xrightarrow{e_{l,j}} v_j \qquad (1)$$

Because of the difficulty in comparing paths directly, we perform a kind of *shortest path serialization* before path comparison, for which we serialize these paths as *label sequences*. We define the operation of this serialization as

Definition 2 (Shortest Path Serialization)**.** *Let $l : \mathcal{V} \to \Sigma$ denote a function that maps a vertex object v to its categorical node label assigned from a finite label alphabet Σ. Furthermore, let $w : \mathcal{E} \to \Sigma$ be the edge label mapping function. The operation of shortest path serialization is definable as (take $p_{i,j}^{\star}$ in Eq. (1) for instance):*

$$\boldsymbol{x} = (l(v_i), -w(e_{i,k}), l(v_k), -w(e_{k,l}), l(v_l), -w(e_{l,j}), l(v_j)),$$

where $\boldsymbol{x}$ is a shortest path sequence derived from $p_{i,j}^{\star}$. In the special condition in which the graph has no edge label, $p_{i,j}^{\star}$ is serialized as $\boldsymbol{x} = (l(v_i), l(v_k), l(v_l), l(v_j))$.

It is noteworthy that we take the negative value of edge label as $-w(e_{i,j})$ so that edge labels are distinguished from node labels during path sequence comparison.

4.1.2 Path sequence similarity

Using shortest path serialization, we transform the path comparison problem to a sequence comparison problem. As described ealier, to make use of a path sequence maximally, we solve the substructure isomorphism problem of paths instead of simply judging whether or not these two paths are completely identical.

As sequence comparison strategies, many methods might be chosen, such as longest common substring (LCStr) and longest common subsequence (LCSeq). The difference between them is that the LCStr demands continuity of sequence above LCSeq. We prefer to use LCSeq rather than LCStr because we discover LCSeq as typically more robust and stable than LCStr. Using the length of LCSeq, we propose our formula of path sequence similarity as

Definition 3 (Path Sequence Similarity)**.** *Given two path sequences $\boldsymbol{x}_1, \boldsymbol{x}_2$, their similarity is defined as*

$$F_{\text{sim}}(\boldsymbol{x}_1, \boldsymbol{x}_2) := \frac{F_{\text{lcs}}(\boldsymbol{x}_1, \boldsymbol{x}_2)}{\max(\text{len}(\boldsymbol{x}_1), \text{len}(\boldsymbol{x}_2))}, \qquad (2)$$

where $F_{\text{lcs}}(\boldsymbol{x}_1, \boldsymbol{x}_2)$ denotes the function returning the length of the LCS of $\boldsymbol{x}_1, \boldsymbol{x}_2$, and where $\text{len}(\cdot)$ *denotes the function returning the length of a sequence.*

We use the maximum length of the objective path sequences as a denominator to limit its value in $[0, 1]$. When two path sequences have a longer common subsequence, the value of their similarity will be larger. From the perspective of the graph, large similarity of path sequences shows large similarity of subgraph structures.

4.2 LCS Kernel in graph comparing

4.2.1 Generating graph representation

Given two undirected and connected graphs $G_1(\mathcal{V}_1, \mathcal{E}_1)$ and $G_2(\mathcal{V}_2, \mathcal{E}_2)$ with nodes and edges labeled, then using a classical algorithm like Floyd–Warshall or the Dijkstra algorithm, the shortest path sets $\mathcal{P}_1$ and $\mathcal{P}_2$ of G_1 and G_2 can be obtained, respectively. Through shortest path serialization, we first transform all paths in $\mathcal{P}_1$ and $\mathcal{P}_2$ as shown below.

$$\mathcal{X}_1 = \left\{ \boldsymbol{x}_{i,j}^{(1)} | \boldsymbol{x}_{i,j}^{(1)} = F_{\text{serialize}}(p_{i,j}^\star), p_{i,j}^\star \in \mathcal{P}_1 \right\}$$
$$\mathcal{X}_2 = \left\{ \boldsymbol{x}_{i,j}^{(2)} | \boldsymbol{x}_{i,j}^{(2)} = F_{\text{serialize}}(p_{i,j}^\star), p_{i,j}^\star \in \mathcal{P}_2 \right\}$$

Therein, $F_{\text{serialize}}$ denotes the operation of shortest path serialization described earlier. Actually, $\mathcal{X}_1$ and $\mathcal{X}_2$ respectively represent two path sequence sets derived from $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively describing the subgraph structures of G_1 and G_2. For every path sequence $\boldsymbol{x}_{i,j}^{(1)}$ and $\boldsymbol{x}_{i,j}^{(2)}$, the superscript denotes the graph to which they belong.

It is noteworthy that some path sequences that are exactly the same as those in the path sequence set. To avoid the increase of the identical sequences, we do not add them to the sequence set $\mathcal{X}$ and store the number of identical sequences in a mass vector $\boldsymbol{m}$. For example, the i-th element in $\boldsymbol{m}$ denotes the number of sequences which are identical to the i-th path sequence in $\mathcal{X}$. In doing so, we obtain two mass vectors $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ respectively belonging to $\mathcal{X}_1$ and $\mathcal{X}_2$.

4.2.2 Wasserstein Distance in LCS metric space

To compare path sequence sets $\mathcal{X}_1$ and $\mathcal{X}_2$, we propose a metric space of path sequence over the union of $\mathcal{X}_1$ and $\mathcal{X}_2$, where we use path sequence similarity to define the metric.

Definition 4 (LCS Metric Space)**.** *Given two path sequence sets $\mathcal{X}_1, \mathcal{X}_2$, and $\mathcal{X} := \mathcal{X}_1 \cup \mathcal{X}_2$ is their union. The LCS Metric Space of $\mathcal{X}_1$ and $\mathcal{X}_2$ is written as $S(\mathcal{X}, d)$, where the metric $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is defined as below.*

$$d(\boldsymbol{x}_1, \boldsymbol{x}_2) = 1 - F_{\text{sim}}(\boldsymbol{x}_i^{(1)}, \boldsymbol{x}_j^{(2)}) \tag{3}$$

Actually, $\mathcal{X}_1$ and $\mathcal{X}_2$ can be regarded as two distributions in their LCS Metric Space, where the path sequence is a discrete point of these distributions. Finally, we propose to leverage the Wasserstein distance between these two distributions to define our graph distance as

Definition 5 (LCS Graph Distance)**.** *Given two undirected and connected graphs $G_1(\mathcal{V}_1, \mathcal{E}_1)$ and $G_2(\mathcal{V}_2, \mathcal{E}_2)$ and their respectivepath sequence sets $\mathcal{X}_1$ and $\mathcal{X}_2$ and mass vectors $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$, then the LCS Graph Distance between G_1 and G_2 is defined as*

$$d_G(G_1, G_2) = \mathcal{W}_1(\mathcal{X}_1, \mathcal{X}_2) = \langle \mathbf{T}^\star(\mathbf{D}, \frac{\boldsymbol{m}_1}{|\boldsymbol{m}_1|_1}, \frac{\boldsymbol{m}_2}{|\boldsymbol{m}_2|_1}), \mathbf{D} \rangle$$

where $\mathcal{W}_1(\mathcal{X}_1, \mathcal{X}_2)$ represents the 1-Wasserstein distance between $\mathcal{X}_1$ and $\mathcal{X}_2$, and where $\mathbf{D}$ denotes the ground distance matrix including the distances $d(\boldsymbol{x}_1, \boldsymbol{x}_2)$ between $\boldsymbol{x}_1 \in \mathcal{X}_1$ and $\boldsymbol{x}_2 \in \mathcal{X}_2$ in the LCS Metric Space $S(\mathcal{X}_1 \cup \mathcal{X}_2, d)$. Here, $\mathbf{T}^\star(\mathbf{D}, \frac{\boldsymbol{m}_1}{|\boldsymbol{m}_1|_1}, \frac{\boldsymbol{m}_2}{|\boldsymbol{m}_2|_1})$ denotes the optimal transport plan w.r.t. mass vectors $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ of $\mathcal{X}_1$ and $\mathcal{X}_2$, respectively, and $\mathbf{D}$, where $|\cdot|_1$ denotes the l_1 norm.

We use LCS graph distance to compute a similarity measure between two graphs. The measure will be used in machine learning algorithm as a kernel value. To guarantee the positive definiteness of our proposed kernel, we combine the LCS graph distance with the Laplacian kernel function to construct a graph kernel. The LCS graph kernel is then defined as presented below.

Definition 6 (LCS Graph Kernel)**.** *Given a set of graphs $\mathcal{G}$, then G_i and G_j are two arbitrary graphs in $\mathcal{G}$. The LCS graph kernel is defined as*

$$k_{\text{LCS}}(G_i, G_j) = e^{-\lambda d_G(G_i, G_j)}, \tag{4}$$

where d_G denotes the LCS graph distance, and where λ is within the range of $(0, +\infty]$.

5. Conclusion

We propose an LCS graph kernel of computing similarity between graphs using the Wasserstein distance in LCS metric space, which successfully transforms graph classification problem to sequence comparing problem. For results of numerical experiments, we will show them in the presentation.

References

[1] Vishwanathan, S.V.N., Schraudolph N.N.. and et al.: Graph kernels, *The Journal of Machine Learning Research*, Vol.11, pp.1201–1242, 2010.

[2] Kriege, N.M., Johansson F.D. and et al.: A survey on graph kernels, *Applied Network Science*, Vol.5, No.1, pp.1–42, 2020.

[3] Gary, M.R. and Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness, *New York* (1979).

[4] Villani, C.: Optimal transport: Old and new, Springer Science & Business Media (2008).

[5] Peyré, G. and Cuturi, M.: Computational Optimal Transport, *Foundations and Trends in Machine Learning*, Vol.11, No.5–6, pp.355–607, 2019.

[6] Gärtner, T., Flach, P. and et al.: On graph kernels: Hardness results and efficient alternatives, *Learning theory and kernel machines* Springer, pp.129–143 (2003).

[7] Shervashidze, N., Schweitzer, P. and et al.: Weisfeiler-Lehman Graph Kernels, *The Journal of Machine Learning Research*, Vol.12, pp.2539–2561, 2011.

[8] Togninalli, M., Ghisu, E. and et al.: Wasserstein weisfeiler-lehman graph kernels, *Advances in Neural Information Processing Systems* pp.6439–6449, 2019.

[9] Maretic, H.P., El Gheche, M. and et al.: GOT: an optimal transport framework for graph comparison, *Advances in Neural Information Processing Systems*, pp.13876–13887, 2019.

[10] Titouan, V., Courty, N. and et al.: Optimal Transport for structured data with application on graphs, *International Conference on Machine Learning*, pp.6275–6284, 2019.

[11] Wagner, R.A. and Fischer, M.J.: The string-to-string correction problem, *Journal of the ACM*(JACM), Vol.21, No.1, pp.168–173 (1974).

[12] Cormen, T.H., Leiserson, C.E. and et al.: Introduction to algorithms, MIT press (2009).

[13] Sinkhorn, R. and Knopp, P.: Concerning nonnegative matrices and doubly stochastic matrices, *Pacific Journal of Mathematics*, Vol.21, No.2, pp.343–348 (1967).

[14] Cuturi, M.: Sinkhorn distances: Lightspeed computation of optimal transport, *Advances in Neural Information Processing Systems* (NIPS), 2013.

[15] Benamou, J. D., Carlier, G. and et al.: Iterative Bregman projections for regularized transportation problems, *SIAM Journal on Scientific Computing*, Vol.37, No.2, pp.1111–A1138, 2015.