Near-Linear Time Generalized Sinkhorn Algorithms for Bounded Genus Graphs

Krzysztof Choromanski[1, 2 * †], **Derek Long**[1 *], **Ananya Parashar**[1 *], **Dwaipayan Saha**[1 *]
[1]Columbia University, [2]Google DeepMind

Abstract

We present *GenusSink*, a new class of approximate generalized Sinkhorn algorithms with shortest-path-distance costs for bounded genus (e.g. planar) graphs, providing near-linear time: (1) pre-processing, (2) iteration step, (3) final transport plan matrix querying and near-linear memory. Graphs handled by GenusSink include in particular planar graphs and bounded-genus meshes approximating 3D objects. GenusSink addresses total quadratic time complexity of its brute-force counterpart by leveraging separator-based decomposition of graphs, computational geometry techniques, and new results on fast matrix-vector multiplications with generalized distance matrices, using, in particular, Fourier analysis and low displacement rank theory. It is inspired by recent breakthroughs in graph theory on approximating bounded genus metrics with small treewidth metrics [16]. The graph-centric approach enables us to target optimal transport problem with the corresponding distributions defined on the manifolds approximated by weighted graphs and with cost functions given by geodesic distances. We conduct rigorous theoretical analysis of GenusSink, provide practical implementations, leveraging newly introduced in this paper *separation graph field integrators* (S-GFIs) data structures and present empirical verification. GenusSink provides orders of magnitude more accurate computations than other efficient Sinkhorn algorithms, while still guaranteeing significant computational improvements, as compared to the baseline. As a by-product of the developed methods, we show that GenusSink is **numerically equivalent** to the brute-force geodesic Sinkhorn algorithm on n-vertex graphs with treewidth $O(\log\log(n))$ (e.g. on trees).

1 Introduction & Related Work

Consider two spaces X, Y, with the corresponding measures μ and ν and a function: $c : X \times Y \rightarrow [0, +\infty]$, quantifying the cost of transporting a "unit of mass" from locations in X to those in Y. The celebrated *Optimal Transport Problem* (OTP) [29, 27, 31, 30, 3, 18, 15, 28, 4] can be depicted as the task of "transporting" μ to ν, by minimizing total cost. More formally, consider a measure $\pi \in \mathcal{P}(X \times Y)$, such that the projection of π into X and Y results in μ and ν respectively, i.e.:

$$\pi(\mathcal{A} \times Y) = \mu(\mathcal{A}),\ \pi(X \times \mathcal{B}) = \nu(\mathcal{B}) \text{ for all measurable } \mathcal{A} \in X, \mathcal{B} \in Y. \tag{1}$$

We denote the set of such measures π as $\Pi(\mu, \nu)$. The famous Kantorovich formulation [23] casts the OTP as the following optimization problem:

$$\min_{\pi \in \Pi(\mu,\nu)} \int_{X \times Y} c(x, y) d\pi(x, y) \tag{2}$$

Optimal Transport Problem found applications in several fields of machine learning (ML), ranging from representation and metric learning [17, 24], graph modeling [32, 41, 42], through computer

*equal contribution
†senior lead

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

vision [25, 33] to generative modeling [19, 35, 5], differentiable combinatorial computations [12] and (more recently) even Transformer architectures [38, 22].

Without loss of generality, we can assume that μ and ν correspond to the probabilistic distributions. After the discretization, the optimization problem from Eq. 2 takes the following form:

$$\min_{\mathbf{P}\in\mathbb{R}^{n\times n}} \mathbf{P}\odot\mathbf{C} \text{ st. } \mathbf{P}\mathbf{1}_n=\mathbf{a},\ \mathbf{P}^\top\mathbf{1}_n=\mathbf{b},\ \mathbf{P}\geq 0, \tag{3}$$

where $\mathbf{a}\in\mathbb{R}^n_{\geq 0}, \mathbf{b}\in\mathbb{R}^n_{\geq 0}$ encode discrete probabilistic distributions ($\sum_{i=1}^n a_i=1, \sum_{j=1}^n b_j=1$). In Eq. 3, $\odot$ denotes Hadamard (element-wise) product, $\mathbf{C}=[c_{i,j}]\in\mathbb{R}^{n\times n}$ is called the *cost matrix* and matrix $\mathbf{P}$, discretizing the joint distribution $\pi(X\times Y)$, is often referred to as a *transport plan*.

Matrix $\mathbf{P}^*$, encoding the optimal transport plan, already takes space $O(n^2)$. However, in several applications, one only needs to efficiently query $\mathbf{P}$, e.g. compute efficiently $\mathbf{P}\mathbf{v}_{\mathcal{S}}$ for indicator vectors $\mathbf{v}_{\mathcal{S}}$, corresponding to subsets $S\subseteq\{1,...,n\}$ (providing mass amounts taken to a particular subset of target locations from initial locations). The brute-force, linear-program approach to solve the optimization problem from Eq. 2 (either finding $\mathbf{P}^*$ or efficient querying) requires cubic time complexity.

Sinkhorn (entropic) re-formulation [11, 2] of the OTP relaxes it by adding entropy regularization term. This leads to the following optimization problem, where ϵ quantifies the importance of the regularizer:

$$\min_{\mathbf{P}\in\mathbb{R}^{n\times n}} \mathbf{P}\odot\mathbf{C}+\epsilon\sum_{i,j}P_{i,j}(\log(P_{i,j})-1). \tag{4}$$

In the *generalized h-Sinkhorn formulation*, $\log$ from Eq. 4 is replaced by h for some $h:\mathbb{R}\rightarrow\mathbb{R}$. Adding an extra term to the regular OTP formulation makes the problem convex and leads to this closed-form formula for $\mathbf{P}^*$, where $f=h^{-1}$ stands for h-inverse (for regular Sinkhorn, $f=\exp$):

$$\mathbf{P}^*=\mathrm{diag}(\mathbf{u})\cdot\mathbf{K}^\epsilon_f\cdot\mathrm{diag}(\mathbf{v}),\ \mathbf{K}^\epsilon_f=f(-\frac{\mathbf{C}}{\epsilon}), \tag{5}$$

with vectors $\mathbf{u},\mathbf{v}\in\mathbb{R}^n$ obtained via the following iterative procedure: $\mathbf{v}^0=\mathbf{1}_n$, $\mathbf{u}\leftarrow\frac{\mathbf{a}}{\mathbf{K}^\epsilon_f\mathbf{v}}$, $\mathbf{v}=\frac{\mathbf{b}}{(\mathbf{K}^\epsilon_f)^\top\mathbf{u}}$. The exponentiation and division operations are conducted element-wise. Since in practice, the number of the steps of that iterative procedure (*number of Sinkhorn iterations*) is constant, the overall time complexity of all Sinkhorn steps is quadratic, rather than cubic, in n. This algorithm still needs to materialize the cost matrix $\mathbf{C}\in\mathbb{R}^{n\times n}$, which takes quadratic space and at least quadratic time. Several computational improvements of the Sinkhorn algorithms were proposed [2, 1, 40, 21, 26]. For instance, the *Greenkhorn* algorithm [2], improves the $O(n^2)$ time complexity of the single iteration to $O(n)$, via efficient updates involving only one column/row of $\mathbf{K}$ per step. Those methods still require quadratic memory to store $\mathbf{C}$ and quadratic time to query $\mathbf{P}^*$ for accurate approximation. They also do not provide exact computations for low treewidth graph domains.

Our approach is closely related to the separator-based graph field integrators of Choromanski et al. [8]. Their exact analysis for unweighted graphs uses bounded-size connected separators to bound the number of possible distance signatures. S-GFI instead uses geometric range computations to handle small separators without requiring them to be connected. Combined with low-treewidth metric embeddings, this supports efficient Sinkhorn computations on bounded-genus graphs.

We also note a relation to Euclidean kernel implementations, specifically, KeOps [7] provides symbolic kernel reductions without materializing the full kernel matrix and can exploit GPU parallelism for large point sets. For Euclidean distances, approximate linear-time Sinkhorn algorithms can also leverage linearization of Gaussian kernels with random features [36]. In this paper, our focus is rather on exploiting graph structure to accelerate computations with kernels defined through shortest-path distances, which encode the intrinsic geometry of graph-supported domains.

Consider the setting where $Y=X$. Without loss of generality, we assume that after the discretization, we have n locations supporting both distributions. A natural class of cost functions for the OTPs in the non-Euclidean input spaces X are shortest-path distances in the weighted undirected graphs approximating them. The above methods can be directly applied there, but at some point require the materialization of the so-called *generalized distance matrix* $\mathbf{D}^\epsilon_f\in\mathbb{R}^{n\times n}$ defined as: $\mathbf{D}^\epsilon_f=[f_\epsilon(d_{i,j})]_{i,j=1,..,n}$, for $f_\epsilon:\mathbb{R}\rightarrow\mathbb{R}$ given as: $f_\epsilon(x)=f(-\frac{x}{\epsilon})$, and where: $d(i,j)$ stands for the shortest path between nodes: i and j in the given n-vertex graph G_X that approximates X.

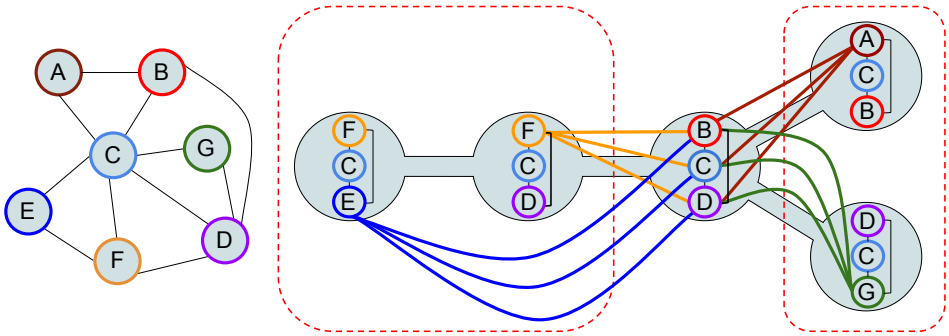

Figure 1: **Left**: Seven-vertex planar graph. **Right:** A pictorial description of its *treewidth* decomposition, with the treewidth parameter upper-bounding the sizes of the clusters of nodes above (that we will refer to as *bags*, see: Sec. 2.1). Two graph pieces, obtained with the separator $\{B, C, D\}$, are highlighted via red dashed boxes. The treewidth decomposition provides a gateway for designing efficient algorithms for various graph problems.

For general graphs G_X, it is hopeless to get more computationally-efficient methods, but one can ask whether we can do better under additional assumptions on G_X, present in several practical ML applications. In this paper, we provide a strong affirmative answer to this question.

We present *GenusSink*, a new class of approximate generalized Sinkhorn algorithms with shortest-path-distance costs for **bounded genus graphs** (e.g. planar graphs), providing near-linear time: (1) pre-processing, (2) iteration step, (3) final transport plan matrix querying and near-linear memory. Graphs handled by GenusSink include in particular **planar graphs** and **bounded genus** meshes approximating 3D objects. GenusSink addresses total quadratic time complexity of its brute-force counterpart by leveraging separator-based decomposition of graphs, computational geometry techniques, and new results on fast matrix-vector multiplications with generalized distance matrices, using in particular Fourier analysis and low displacement rank theory. It is inspired by recent breakthroughs in graph theory on approximating bounded genus metrics with small-treewidth metrics [16]. The graph-centric approach enables us to target optimal transport problem with the corresponding distributions defined on the manifolds approximated by weighted graphs and with cost functions given by geodesic distances. We conduct rigorous theoretical analysis of GenusSink, provide practical implementations, leveraging *separation graph field integrators* (S-GFIs) data structures, which are newly introduced in this paper, and present empirical verification. GenusSink provides orders of magnitude more accurate computations than other efficient Sinkhorn algorithms, while still guaranteeing significant computational improvements, as compared to the baseline. As a by-product of the developed methods, we show that GenusSink is **numerically equivalent** to the brute-force geodesic Sinkhorn algorithm on the n-vertex graphs with treewidth $O(\log\log(n))$ (e.g. on trees).

This paper is organized as follows (additional details are provided in the Appendix).

In Sec. 2, we introduce basic graph theory concepts, used throughout this paper. Those include in particular the notion of the *treewidth* (Sec. 2.1, see also: Fig. 1), a parameter describing how "close" a graph is to a tree (acyclic connected graph) and playing an important role in our algorithmic approach. In Sec. 3.1 we present the separation graph field integration (S-GFI) data structure and show how it can be used to efficiently compute the actions of the generalized distance matrices for graphs under consideration. We then show how those translate to efficient Sinkhorn algorithms. In Sec. 3.2, we finalize the GenusSink algorithm by lifting those techniques to bounded genus graphs, leveraging recent results on approximating metrics induced by bounded genus graphs with those of low treewidth graphs [16]. In Sec. 3.3, we propose the practical implementation of GenusSink. Its empirical verification, including the comparison with **six** other efficient Sinkhorn methods and the brute-force, is conducted in Sec. 4. We conclude in Sec. 5. Additional theoretical details regarding handling general functions f are provided in the Appendix.

2 Preliminaries

In this paper, we consider undirected weighted n-vertex graphs $G = (V, E, W)$ with the set of vertices $V = \{0, \ldots, n-1\}$, the set of edges E and the set of the corresponding positive weights W.

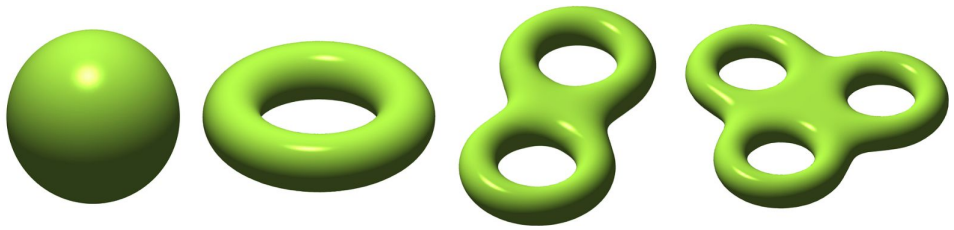

Figure 2: Left to right: 2D-surfaces of 3D objects with genus values: 0, 1, 2 and 3. In this paper, we are especially interested in families of graphs with bounded genus that can be used in particular to provide discretized mesh representations of the above surfaces.

For a given subset $T \subseteq V$, we denote by $G[T]$ an *induced subgraph* of G, obtained by taking vertices T and the weighted edges between those vertices in G. We call a triplet (A, S, B), for three pairwise disjoint subsets: $A, S, B \subseteq V$, satisfying: $A \cup S \cup B = V$, a *separation configuration* if there are no edges between A and B. We call S a **separator** (vertices from A and B can be connected via paths of length > 1 in G, but each such path needs to use a vertex from S). S is called **balanced** if $|A|, |B| \geq cn$ for some fixed constant $c > 0$, and where $|\cdot|$ stands for the set size.

A graph is called **planar** if it can be drawn on the plane with non-intersecting edges. A surface of genus g is a surface with g "handles" (holes). The **genus of the graph** G is the smallest nonnegative integer g such that G can be drawn on an orientable surface of genus g without edge crossing. We will be particularly interested in graphs of bounded genus (planar graphs provide a special sub-class here with zero-genus; see Fig. 2).

2.1 Treewidth & Treewidth Decomposition

The notion of the *treewidth* provides a way to quantify how "close" the graph is to a tree. Since several problems that are challenging on general graphs can be efficiently solved for trees, this parameter effectively provides an upper bound on the "algorithmic tractability" of the graph (see also: Fig. 1).

Definition 2.1 (tree decomposition)**.** If $G = (V, E, W)$ is a weighted undirected graph, then its *tree decomposition* is a pair $(T, \{X_t\}_{t \in V(T)})$, where T is a tree with vertex-set $V(T)$ and sets $X_t \subseteq V$ are called *bags* and satisfy the following properties:

- $\bigcup_{t \in V(T)} X_t = V$ (the bags cover all the vertices),
- for each edge $(u, v) \in E$, there exists a bag X_t such that: $\{u, v\} \subseteq X_t$,
- for every vertex $v \in V$, the set $\{t \in V(T) : v \in X_t\}$ is a sub-tree of T.

The *width* of a given tree decomposition $(T, \{X_t\}_{t \in V(T)})$ is defined as $\max_{t \in V(T)} |X_t| - 1$, where **treewidth** $tw(G)$ of G is the minimum width over all possible tree decompositions of G. Directly from Def. 2.1, we conclude that bags of the treewidth decomposition of G act as separators in G. Not only are bags valid separators, but a bag that is a balanced separator can be found efficiently in time $O(n + |V(T)|)$, using *centroid argument* for trees [13]. Furthermore, the following holds [14, 6]:

Theorem 2.2. *For a given graph G with $tw(G) = k$, there exists an algorithm of time complexity $2^{O(k)}n$ that outputs a tree decomposition $\mathcal{T}$ of width $\leq 2k + 1$. Furthermore, any tree decomposition of width w can be converted into a tree decomposition of the same width and $O(w \cdot n)$ bags in linear time. Thus it can be assumed that $\mathcal{T}$ has $O((2k + 1)n)$ bags.*

Even though finding an optimal tree decomposition can be challenging, the above theorem shows that if we accept a width within a small multiplicative constant from optimal, this can be done efficiently. This assumes that the treewidth is very small, as compared to n. That will be indeed the case in our analysis, where we will work with graphs of treewidth $O(\log \log n)$, whenever we apply those techniques. We can also assume that such a decomposition does not have too many bags. This will enable us to efficiently find those bags of the decomposition that are balanced separators.

3 GenusSink Algorithm

3.1 Separation Graph Field Integrators (S-GFI) for Sinkhorn

The separation graph field integrator (S-GFI) data structure provides a way to efficiently compute matrix-vector multiplications with generalized distance matrices $\mathbf{D}_f^\epsilon = [f_\epsilon(d_{i,j})]_{i,j=1,\ldots,n}$ for several classes of functions $f_\epsilon : \mathbb{R} \rightarrow \mathbb{R}$, and graphs with small treewidth, where $d_{i,j}$ stands for the shortest path distance between nodes i, j in a given graph G. This plays a critical role in designing GenusSink.

3.1.1 An Overview

The name: *graph field integrator* (GFI) corresponds to the fact that the action $\mathbf{D}_f^\epsilon \mathbf{x}$ of a matrix $\mathbf{D}_f^\epsilon$ on a given vector $\mathbf{x} \in \mathbb{R}^n$ can be thought of as a process of integrating a scalar field on the vertices of G and given by $\mathbf{x}$. The integration coefficients depend on the distances between the nodes in the graph (shortest path distance) metric. On a high-level, S-GFI is a tree with nodes corresponding to balanced separators (to apply divide-and-conquer strategy) and additional metadata supporting fast integration.

In the first phase, S-GFI is built in the **one-time** process per graph. That construction critically relies on the ability to efficiently find small balanced separators. It also leverages Dijkstra's shortest-path algorithm [10] to construct meta-data useful for the efficient calculations of those contributions to the overall integration that arise from sub-matrices $\mathbf{D}_{f_\epsilon}^{A,B} = [f_\epsilon(d_{i,j})]_{i \in A, j \in B} \in \mathbb{R}^{|A| \times |B|}$, for sets A, B corresponding to the separation (A, S, B) (nodes of the S-GFI tree correspond to separators S).

In the second phase, when S-GFI is ready, the integration is conducted for any $\mathbf{x} \in \mathbb{R}^n$. The key component of that procedure is a method computing efficiently actions of the matrices $\mathbf{D}_{f_\epsilon}^{X,Y} = [f_\epsilon(d_{i,j})]_{i \in X, j \in Y} \in \mathbb{R}^{|X| \times |Y|}$ for $X = A, Y = B$ and $X = B, Y = A$. Here we apply computational geometry methods. For generalized distance matrices, we also leverage Fourier analysis and low displacement rank theory (see: Appendix, Sec. A), but in the regular Sinkhorn setting with $f_\epsilon : x \rightarrow \exp(-\frac{x}{\epsilon})$, this is not needed. See Fig. 3 (left) for the visual scheme of S-GFI.

3.1.2 Deeper dive into S-GFI data structure

For a given graph G and subsets $X, Y \subseteq V(G)$ with $X \subseteq Y$, we denote by $G^X[Y]$ a subgraph of G induced by the subset Y, but with the edges between the nodes of X modified as follows. For any two different $i, j \in X$ a weight of an edge between i and j is assigned as the shortest path distance between i and j in G (thus if necessary, new edges are added; if i and j are not connected via any path in G, no edge is added). S-GFI is a binary tree with the following node-attributes (and separation denoted as (A, S, B)):

- **sep_dist_matrix**: matrix of shape $|S| \times |S|$ of shortest path distances between the nodes of the small balanced separator S of the graph represented by a sub-tree rooted at that node,
- **left_distances**: a matrix of shape $|A| \times |S|$, containing shortest path distances between vertices of A and vertices of S,
- **right_distances**: a matrix of shape $|B| \times |S|$, containing shortest path distances between vertices of B and vertices of S,
- **left/sep/right_ids**: the ids of the vertices from $A/S/B$,
- **explicit_graph**: an explicit graph that the sub-tree rooted at that node corresponds to (this is None, unless a node is a leaf).
- **left/right_sgfi**: left/right child of the node, corresponding to graph $G^S[A \cup S]$ / $G^S[B \cup S]$.

S-GFI supports efficient multiplication $\mathbf{D}_f^\epsilon \mathbf{x}$ via the **integrate** function, given in Algorithm 1. The critical component of the algorithm is a procedure: **cross_compute** that given two matrices of distances: $\mathbf{AD}$ = **left_distances** $\in \mathbb{R}^{|A| \times |S|}$ and $\mathbf{BD}$ = **right_distances** $\in \mathbb{R}^{|B| \times |S|}$, as well as two vectors: $\mathbf{u} \in \mathbb{R}^{|A|}$ and $\mathbf{v} \in \mathbb{R}^{|B|}$, computes: $\mathbf{M}_{f_\epsilon}^{A,B}\mathbf{v}$ and $(\mathbf{M}_{f_\epsilon}^{A,B})^\top \mathbf{u}$ in near-linear time, where $\mathbf{M}_{f_\epsilon}^{A,B}$ is defined as follows:

$$\mathbf{M}_{f_\epsilon}^{A,B} = \left[f_\epsilon \left(\min_{k=1,\ldots,|S|} (\mathbf{AD}[i][k] + \mathbf{BD}[j][k]) \right) \right]_{i=1,\ldots,|A|}^{j=1,\ldots,|B|} \in \mathbb{R}^{|A| \times |B|} \tag{6}$$

Algorithm 1 **Integrate method:** Calculate efficiently $\mathbf{y} = \mathbf{D}_f^\epsilon \mathbf{x}$ for a given $\mathbf{x} \in \mathbb{R}^n$, using S-GFI

Input: separation graph field integrator SGFI, input vector $\mathbf{x} \in \mathbb{R}^n$
Output: Exact or approximate $\mathbf{y} = \mathbf{D}_f^\epsilon \mathbf{x}$

1: initialize: $y = \mathbf{0} \in \mathbb{R}^n$
2: **if** SGFI.explicit_graph is None **then**
3: AD $\leftarrow$ SGFI.left_distances, AID $\leftarrow$ SGFI.left_ids
4: BD $\leftarrow$ SGFI.right_distances, BID $\leftarrow$ SGFI.right_ids
5: $\mathbf{L}, \mathbf{R} \leftarrow$ SGFI.cross_compute(AD, BD, $\mathbf{x}$[AID], $\mathbf{x}$[BID])
6: $\mathbf{y}$[AID]+=$\mathbf{L}$, $\mathbf{y}$[BID]+=$\mathbf{R}$, SID $\leftarrow$ SGFI.sep_ids
7: ASID $\leftarrow$ sort(concat(AID, SID)), BSID $\leftarrow$ sort(concat(BID, SID))
8: ASY = integrate(SGFI.left_sgfi, $\mathbf{x}$[ASID]), BSY = integrate(SGFI.right_sgfi, $\mathbf{x}$[BSID])
9: $\mathbf{y}$[ASID]+=ASY, $\mathbf{y}$[BSID]+=BSY, FS $\leftarrow f_\epsilon$(SGFI.sep_dist_matrix)
10: SY = FS $\cdot$ $\mathbf{x}$[SID], $\mathbf{y}$[SID]-=SY
11: **else**
12: compute $\mathbf{D}_f^\epsilon \mathbf{x}$ brute-force using SGFI.explicit_graph
13: **end if**
14: return $\mathbf{y}$

Since all paths between vertices of A and B use a vertex from S (by the definition of the separator), we conclude that $\mathbf{M}_{f_\epsilon}^{A,B} = \mathbf{D}_{f_\epsilon}^{A,B}$. Thus vectors $\mathbf{L}, \mathbf{R}$ from l.5 of Algorithm 1 encode contributions to $\mathbf{y} = \mathbf{D}_f^\epsilon \mathbf{x}$ from its two sub-matrices: $\mathbf{D}_{f_\epsilon}^{A,B}$ and $\mathbf{D}_{f_\epsilon}^{B,A} = (\mathbf{D}_{f_\epsilon}^{A,B})^\top$.

Assuming that this computation can be done efficiently, the remaining part of Alg. 1 is a relatively straightforward application of the divide-and-conquer strategy. To make the calculations clearer, we can write $\mathbf{K} = \mathbf{D}_f^\epsilon$ as the following nine-block matrix:

$$\mathbf{K} = \begin{pmatrix} \mathbf{D}_{f_\epsilon}^{A,A} & \mathbf{D}_{f_\epsilon}^{A,S} & \mathbf{D}_{f_\epsilon}^{A,B} \\ \mathbf{D}_{f_\epsilon}^{S,A} & \mathbf{D}_{f_\epsilon}^{S,S} & \mathbf{D}_{f_\epsilon}^{S,B} \\ \mathbf{D}_{f_\epsilon}^{B,A} & \mathbf{D}_{f_\epsilon}^{B,S} & \mathbf{D}_{f_\epsilon}^{B,B} \end{pmatrix}. \tag{7}$$

Let E_{AS} and E_{BS} extend vectors on $A \cup S$ and $B \cup S$, respectively, to the full vertex set by filling the remaining entries with zeros. Let $\mathbf{x}_U$ denote the restriction of $\mathbf{x}$ to a subset U. The corresponding decomposition of the matrix-vector product is

$$\mathbf{K}\mathbf{x} = E_{AS}\mathbf{D}_{f_\epsilon}^{A\cup S, A\cup S}\mathbf{x}_{A\cup S} + E_{BS}\mathbf{D}_{f_\epsilon}^{B\cup S, B\cup S}\mathbf{x}_{B\cup S} + (\mathbf{D}_{f_\epsilon}^{A,B}\mathbf{x}_B; -\mathbf{D}_{f_\epsilon}^{S,S}\mathbf{x}_S; \mathbf{D}_{f_\epsilon}^{B,A}\mathbf{x}_A). \tag{8}$$

The first two terms are computed recursively, the cross terms are computed by **cross_compute**, and the separator contribution is subtracted because it was counted twice. If the node of S-GFI contains an explicit graph it corresponds to, the computation is done brute-force (l.12). Otherwise, the contributions to $\mathbf{y}$ from $\mathbf{D}_{f_\epsilon}^{A,B}$ and $\mathbf{D}_{f_\epsilon}^{B,A}$ are added in l.6, as discussed before. This is followed by adding contributions from $\mathbf{D}_{f_\epsilon}^{A\cup S, A\cup S}$ and $\mathbf{D}_{f_\epsilon}^{B\cup S, B\cup S}$ via the recursive invocation of the procedure (l.9). Finally, in l.10 we subtract the contribution from $\mathbf{D}_{f_\epsilon}^{S,S}$ (computed brute-force), since it was counted twice in the previous calculations.

The correctness of the algorithm follows from the fact that in the graph corresponding to the left and right child of the given SGFI node, edge-weights between vertices of S are equal to the lengths of the shortest paths between them in G, if they exist (thus the lengths of the shortest paths between nodes $i, j \in A$ or $i, j \in B$ can be retrieved from the left or right child only, even if those paths intersect with both A and B). We present an efficient implementation of **cross_compute** next.

Procedure cross_compute: Let the rows of $\mathbf{AD} \in \mathbb{R}^{|A|\times|S|}$ and $\mathbf{BD} \in \mathbb{R}^{|B|\times|S|}$ be: $\mathbf{a}_1, ..., \mathbf{a}_{|A|} \in \mathbb{R}^{|S|}$ and $\mathbf{b}_1, ..., \mathbf{b}_{|B|} \in \mathbb{R}^{|S|}$ respectively. We will show how to compute efficiently $\mathbf{M}_{f_\epsilon}^{A,B}\mathbf{v}$. The computation of $\mathbf{M}_{f_\epsilon}^{B,A}\mathbf{u}$ can be conducted analogously. Note that the following holds:

$$(\mathbf{M}_{f_\epsilon}^{A,B}\mathbf{v})[i] = \sum_{k=1}^{|S|}\sum_{j=1}^{|B|} f_\epsilon(\mathbf{a}_i[k] + \mathbf{b}_j[k])\mathbf{I}[\mathbf{w}_i^k \prec \mathbf{z}_j^k]\mathbf{v}_j, \tag{9}$$

where $\prec$ denotes element-wise $<$-inequality and $\mathbf{w}_i^k, \mathbf{z}_j^k$ are defined as follows:

$$\mathbf{w}_i^k = (\mathbf{a}_i[k] - \mathbf{a}_i[0], ..., \mathbf{a}_i[k] - \mathbf{a}_i[k-1], \mathbf{a}_i[k] - \mathbf{a}_i[k+1], ..., \mathbf{a}_i[k] - \mathbf{a}_i[|S|]) \in \mathbb{R}^{|S|-1}$$
$$\mathbf{z}_j^k = (\mathbf{b}_j[0] - \mathbf{b}_j[k], ..., \mathbf{b}_j[k-1] - \mathbf{b}_j[k], \mathbf{b}_j[k+1] - \mathbf{b}_j[k], ..., \mathbf{b}_j[|S|] - \mathbf{b}_j[k]) \in \mathbb{R}^{|S|-1} \tag{10}$$

We will show how to efficiently compute the expressions $\xi_i = \sum_{j=1}^{|B|} f_\epsilon(\mathbf{a}_i[k] + \mathbf{b}_j[k])\mathbf{I}[\mathbf{w}_i^k \preceq \mathbf{z}_j^k]\mathbf{v}_j$ for a fixed k and all i. We then repeat the process $|S|$ times to consider all k. We start with the case $|S| = 1$. If $f = \exp$ (regular Sinkhorn objective) then:

$$(\xi_1, ..., \xi_{|A|})^\top = \left(\exp(-\frac{\mathbf{a}_1[k]}{\epsilon}), ..., \exp(-\frac{\mathbf{a}_{|A|}[k]}{\epsilon})\right)^\top \left(\exp(-\frac{\mathbf{b}_1[k]}{\epsilon}), ..., \exp(-\frac{\mathbf{b}_{|B|}[k]}{\epsilon})\right)\mathbf{v} \tag{11}$$

and thus all ξ_i can be clearly computed in time $O(|A| + |B|)$ (with the one time pre-computation of the dot-product $(\exp(-\frac{\mathbf{b}_1[k]}{\epsilon}), ..., \exp(-\frac{\mathbf{b}_{|B|}[k]}{\epsilon}))\mathbf{v}$. Interestingly, near-linear (approximate or exact) algorithms exist for a much broader class of functions f. We provide detailed corresponding analysis in the Appendix, applying several additional techniques: Fourier analysis, random features and the theory of low displacement rank matrices. That enables us to extend the results presented in the main body of the paper to a generalized h-Sinkhorn setting. Now let us assume that $|S| > 1$. Then we sort all vectors: $\mathbf{w}_1^k, ..., \mathbf{w}_{|A|}^k, \mathbf{z}_1^k, ..., \mathbf{z}_{|B|}^k$ in the ascending order of their first coordinates (see: Fig. 3 (right)). We denote the corresponding sorted sequence as: $(\mathbf{t}_1, ..., \mathbf{t}_{|A|+|B|})$. We have: $\mathbf{t}_1[0] \leq ... \leq \mathbf{t}_{|A|+|B|}[0]$. We compute the median of the first coordinates of the $\mathbf{w}$-vectors from $(\mathbf{t}_1, ..., \mathbf{t}_{|A|+|B|})$. Let us denote by $\mathcal{L}^A$ the set of indices of those $\mathbf{w}$-vectors with the first coordinate upper-bounded by this median. Denote by $\mathcal{R}^A$ the set of indices of those $\mathbf{w}$-vectors with the first coordinate larger than this median. We define $\mathcal{L}^B$ and $\mathcal{R}^B$ analogously (using $\mathbf{z}$-vectors, but the median already computed for $\mathbf{w}$-vectors). The task of computing all ξ_i for $i = 1, ..., |A|$ reduces to this of computing all ξ_i with $i \in \mathcal{L}^A$ and all ξ_i with $i \in \mathcal{R}^A$. We get the following for $i \in \mathcal{L}^A$:

$$\xi_i = \sum_{j \in \mathcal{L}^B} f_\epsilon(\mathbf{a}_i[k] + \mathbf{b}_j[k])\mathbf{I}[\mathbf{w}_i^k \prec \mathbf{z}_j^k]\mathbf{v}_j + \sum_{j \in \mathcal{R}^B} f_\epsilon(\mathbf{a}_i[k] + \mathbf{b}_j[k])\mathbf{I}[\mathbf{w}_i^k[1:] \prec \mathbf{z}_j^k[1:]]\mathbf{v}_j \tag{12}$$

Furthermore, for $i \in \mathcal{R}^A$, we obtain:

$$\xi_i = \sum_{j \in \mathcal{R}^B} f_\epsilon(\mathbf{a}_i[k] + \mathbf{b}_j[k])\mathbf{I}[\mathbf{w}_i^k \prec \mathbf{z}_j^k]\mathbf{v}_j \tag{13}$$

We can then continue the calculations recursively. The key observation is that each of the sums from Eq. 12, 13 needs to be computed for only half of the number of entries i that the original sums needed to be computed for. Furthermore, second sum in Eq. 12 uses sub-vectors $\mathbf{w}$ and $\mathbf{z}$ obtained by skipping their first dimensions. By solving standard recursive equation for the time complexity of the **cross_compute** procedure, we conclude that it is: $O(n \cdot \text{poly}(|S|) \cdot \log^{\text{poly}(|S|)}(n))$. Using this fact and solving another standard recursive time complexity equation for Algorithm 1, we conclude that:

Remark 3.1. The overall time complexity of the procedure **integrate** from Algorithm 1 is $O(n \cdot \text{poly}(|S|) \cdot \log^{\text{poly}(|S|)}(n))$ for a polynomial function poly, separator-size $|S|$ and number of vertices n.

It remains to build the S-GFI data structure. Note that given a small balanced separator S, all other steps for the construction of the particular node of the S-GFI tree can be conducted straightforwardly with the classical Dijkstra's shortest path algorithm that runs in time $O(n + m)\log(n)$, where $m = O(n \cdot tw(G))$ is the number of edges of the input bounded genus graph (we leverage well-known upper bound on m, using $tw(G)$). Thus if calculating S can be done in $O(n \cdot \text{poly}(|S|) \cdot \log^{\text{poly}(|S|)}(n))$ time, the overall time complexity of the construction is also: $O(n \cdot tw(G) \cdot \text{poly}(|S|) \cdot \log^{\text{poly}(|S|)}(n))$. We can also leverage the tree decomposition algorithm, discussed in Sec. 2.1. For the given input graph G with treewidth $tw(G) = k$, we can first construct its tree decomposition $\mathcal{T}$, as in Theorem 2.2 in time $2^{O(k)}n$. We can then find a bag corresponding to a small balanced separator of size at most $2k + 2$ in time $O(nk)$, as discussed in Sec. 2.1. Note that importantly, we do not need to re-calculate $\mathcal{T}$ as we proceed with the calculations of the children of the given S-GFI node, but simply use the corresponding subtrees of $\mathcal{T}$. Indeed, adding extra edges between the vertices of the bags (we do it for the balanced separator), **does not destroy** tree decomposition structure. We conclude that:

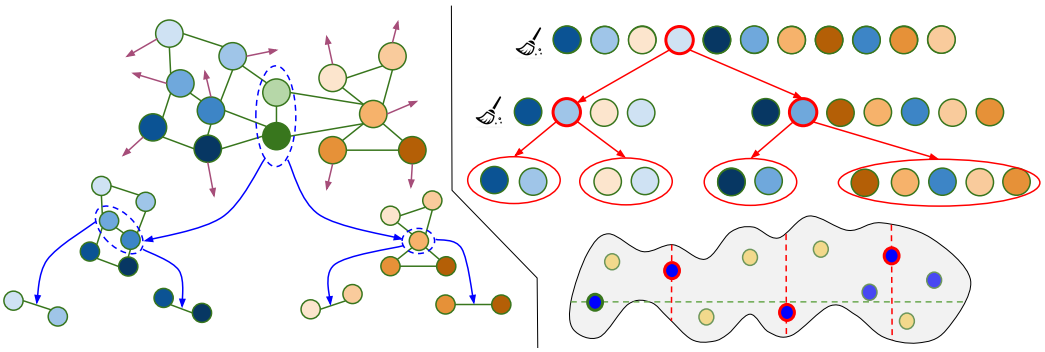

Figure 3: **Left:** The pictorial representation of the S-GFI data structure. In that example, the corresponding tree has a root with two children and four grandchildren. **Right:** The visualization of the **cross_compute** function with the medians depicted in red/green and their corresponding splits highlighted as dashed lines.

Remark 3.2. For graphs G with $tw(G) = O(\log\log(n))$, building S-GFI can be conducted in near-linear time $O(n \cdot \text{poly}(\log\log(n)) \cdot \log^{\text{poly}(\log\log(n))}(n))$ (in particular $o(n^{1+\epsilon})$ for any $\epsilon > 0$). The space complexity is also near-linear in n.

3.1.3 From distance matrices to efficient Sinkhorn

Given S-GFI that can be efficiently built and applied to multiply with generalized distance matrices, the path to the efficient Sinkhorn-algorithm with the following features: (1) efficient pre-processing, (2) improved iteration step and (3) improved querying of the final transport plan matrix (given a fixed number of Sinkhorn iteration steps), as well as: (4) efficient memory is straightforward. One only needs to build the S-GFI instance to support efficient multiplication with matrices $\mathbf{K}_f^\epsilon$ from Eq. 5, that are exactly the generalized distance matrices we considered here. The numerical equivalence to the brute-force geodesic Sinkhorn algorithm for graphs with $tw(G) = O(\log\log n)$ is stated explicitly in Corollary A.5 in Appendix A.3. The statement applies to the full-separator construction with exact cross computations. We next address this treewidth assumption to finalize the algorithm for bounded-genus graphs.

3.2 From Small Treewidth Graphs to Bounded Genus Graphs

Bounded genus n-vertex graphs do not necessarily have treewidth of order $O(\log\log(n))$. For example, planar graphs have treewidth $O(\sqrt{n})$, but not necessarily smaller. However shortest-path metrics on bounded genus graphs can be accurately approximated by the $O(\log\log(n))$-treewidth proxies and furthermore, those proxies can be efficiently computed. The following holds:

Theorem 3.3 ([16])**.** *Given an n-vertex graph $G = (V, E, W)$ of diameter D from the family of bounded genus graphs and a parameter $\eta \in (0, 1)$, there is a deterministic embedding $f : V(G) \to H$ into a graph H of treewidth $O(\eta^{-1}(\log\log(n))^2)$ and additive distortion $+\eta D$. Furthermore, f can be deterministically constructed in $O(n\frac{\log^3(n)}{\eta^2})$ time.*

The original statement given in [16] is for planar graphs, but the Authors explicitly show in the proof how the statement can be extended to bounded genus graphs.

With Theorem 3.3, we can finalize GenusSink algorithm. For a given graph G from the bounded genus family, we first construct its embedding into H, as given in Theorem 3.3. We then run our version of the Sinkhorn algorithm, applying S-GFI data structure on H. Here, we use η for the embedding tolerance: for example, taking $\eta = 1/\log\log n$ gives proxy treewidth $O((\log\log n)^3)$, embedding time $O(n\log^3 n(\log\log n)^2)$, and additive metric distortion $D/\log\log n$. When graph size increases through mesh refinement of a fixed object, rather than rescaling the object, the physical diameter D can remain bounded. For the exponential kernel, an additive distance error δ changes the kernel by a relative factor controlled by $\exp(\delta/\epsilon)$, so the relevant accuracy scale is δ/ϵ.

3.3 Practical implementations

We observed that in practical implementations, we do not even need to construct an embedding, as in Theorem 3.3. Instead we can run well-known linear time algorithms for constructing $O(\sqrt{n})$-size balanced separators for graphs G from the bounded genus family under consideration. We can then sub-sample a $\log\log(n)$ random points from each separator and distribute the remaining points of the separator randomly to both children of the given node of the S-GFI structure. As we show in Sec. 4, that version is still very accurate, in particular orders of magnitude more accurate than other efficient Sinkhorn methods used on a regular basis. Appendix A.2 makes this dependence explicit for a single split, while Appendix B.4 reports the empirical depth and separator-retention tradeoff. Finally, we do not need to construct all the levels of the S-GFI tree. The depth h of the tree is effectively a hyperparameter, and we can terminate at any time, simply by storing explicit graphs in the corresponding nodes (via **explicit_graph** fields). Hyperparameter h provides a convenient tradeoff between asymptotic computational superiority of the methods presented in this paper and highly optimized brute-force matrix-vector multiplications (for graphs small enough regular Sinkhorn algorithm leveraging standard matrix-vector multiplication suffices). In practice, we applied small values of h.

4 Experiments

4.1 Surfaces and Meshes

We evaluate GenusSink on a family of boundaryless sphere-topology "pseudo-genus" surface graphs that remain planar (see Fig. 9) and compare GenusSink against a comprehensive set of baseline approximation methods [1, 2, 37, 36, 20]—Appendix B.1 provides further details on these methods and the experiment set-up. Across all pseudo-genus families, GenusSink matches the geodesic full-kernel baseline to numerical precision while becoming significantly faster at larger scales (see Fig. 4). In contrast, most approximate baselines exhibit a clear accuracy gap.

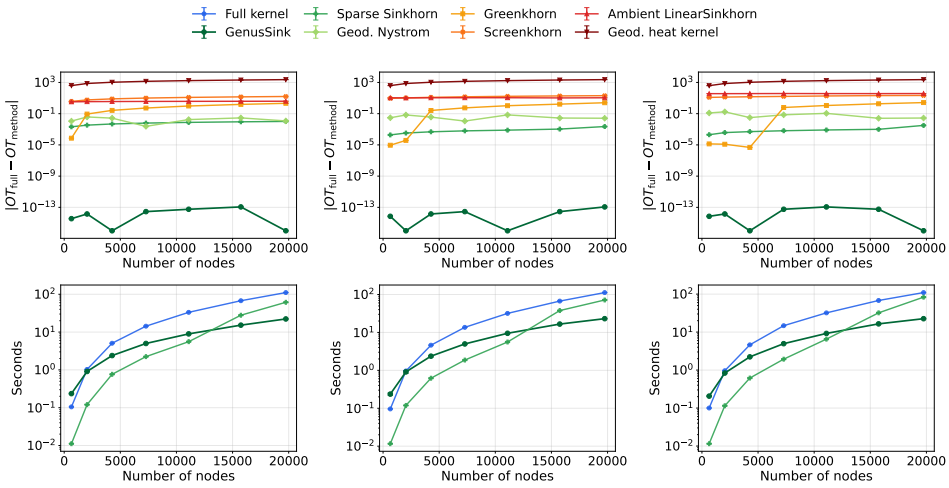

Figure 4: Synthetic pseudo-genus surface experiments for pseudo-genus orders 1, 2, and 3 (left to right). **Top:** absolute Sinkhorn-cost error relative to the dense geodesic full-kernel baseline. **Bottom:** total runtime for the Brute Force (full kernel) and the two most accurate efficient methods from previous studies: GenusSink and Sparse Sinkhorn. GenusSink stays numerically exact while providing significant computational gains.

Extending to more challenging settings, we evaluate GenusSink on real meshes from the **Thingi10K** dataset [43] with up to **13k** vertices. We convert mesh edges to weighted graph edges using Euclidean lengths, and generate smooth source and target measures as geodesic Gaussian mixtures around deterministic PCA-based mesh anchors. Again, accuracy compared with other approximation methods shows that GenusSink remains essentially exact while avoiding full kernel materialization on these real mesh graphs (see Fig. 5). Further details and related experiments can be found in Appendix B.2.

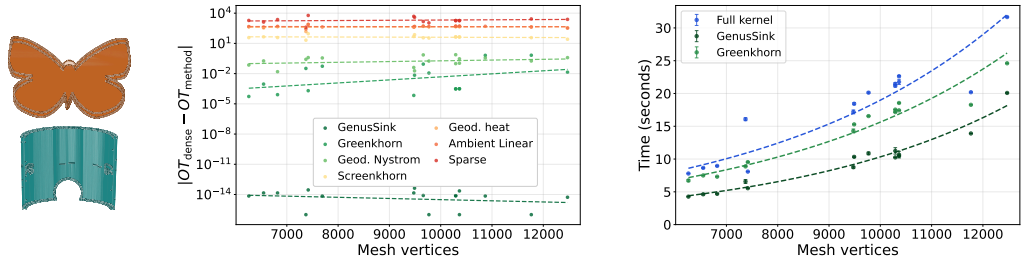

Figure 5: **Left:** examples of Thingi10K meshes. **Middle:** absolute OT-cost error relative to dense geodesic Sinkhorn for several efficient Sinkhorn methods. As before, GenusSink stays numerically exact while providing significant computational gains. **Right:** runtime scaling for dense geodesic kernel construction versus two most accurate efficient methods: GenusSink and Greenkhorn (see: left), with error bars over repeated runs.

4.2 Enhancing Ambulance Deployment in New York City

Emergency Medical Services (EMS) is a high-stakes application where every minute can materially affect patient outcomes. New York City operates one of the largest EMS systems in the world, serving more than 1M incidents annually. We model EMS deployment as the problem of matching ambulance supply to emergency-call demand on a Bronx road graph with 33,363 nodes, where nodes represent possible ambulance locations and the demand distribution is induced by historical incident patterns from publicly available dispatch data. The goal is to choose ambulance locations that reduce mean response and upper-tail response time, two operationally important EMS metrics.

We evaluate this task using a well-established EMS simulator called JEMSS [34] and add a new configuration for the Bronx, a borough of particular operational concern for EMS decision-makers. In our experiment, given the empirical demand distribution, we deploy 60 ambulances and compare GenusSink against other approximation baselines. Note that computing and storing the full graph kernel on this network would require all-pairs road distances over tens of thousands of nodes, which is intractable for frequent ambulance redeployment and motivates the need for fast, accurate Sinkhorn approximations. Hence, we did not include baselines that require materializing the full geodesic kernel. We see that GenusSink consistently produces better deployments: for high-severity calls (i.e., segments 1–3), it achieves the lowest mean response time, 12.5 minutes, compared with 13.4 to 14.5 minutes for the approximation baselines (see Fig. 6). The advantage is especially pronounced in the upper tail, where the baseline methods become increasingly slower than GenusSink as the response-time quantile increases beyond the 95th percentile.

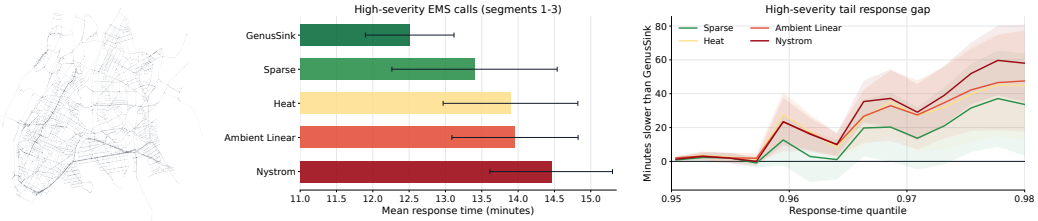

Figure 6: Bronx road graph (left), mean response times across different Sinkhorn methods (middle), and tail response time gaps relative to GenusSink at quantiles from 0.95 onwards (right).

5 Conclusion

We proposed in this paper *GenusSink*, a new class of efficient Sinkhorn algorithms operating on graph metric spaces for bounded genus graphs and characterized by near-linear time (1) pre-processing, (2) iteration step, (3) final transport plan matrix querying and (4) near-linear memory. GenusSink leverages several techniques from graph theory and computational geometry. We provided detailed theoretical analysis of the algorithm and complemented it with thorough empirical evaluation.

Acknowledgements

We did not receive any third party funding or third party support to help us complete this work. Additionally, to the best of our knowledge, there are no competing interests (financial activities outside the submitted work).

References

[1] Mokhtar Z. Alaya, Maxime Bérar, Gilles Gasso, and Alain Rakotomamonjy. Screening sinkhorn algorithm for regularized optimal transport. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[2] Jason M. Altschuler, Jonathan Weed, and Philippe Rigollet. Near-linear time approximation algorithms for optimal transport via sinkhorn iteration. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 1964–1974, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/491442df5f88c6aa018e86dac21d3606-Abstract.html.

[3] Shun-ichi Amari, Ryo Karakida, Masafumi Oizumi, and Marco Cuturi. Information geometry for regularized optimal transport and barycenters of patterns. *Neural Comput.*, 31(5):827–848, 2019. doi: 10.1162/NECO_A_01178. URL https://doi.org/10.1162/neco_a_01178.

[4] Luigi Ambrosio, Elia Brué, and Daniele Semola. Lectures on optimal transport. *UNITEXT*, 2021. URL https://api.semanticscholar.org/CorpusID:244769124.

[5] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, Proceedings of Machine Learning Research, pages 214–223. PMLR, 2017. URL http://proceedings.mlr.press/v70/arjovsky17a.html.

[6] Hans L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *Proceedings of the twenty-fifth annual ACM symposium on Theory of Computing*, 1993. URL https://api.semanticscholar.org/CorpusID:2028181.

[7] Benjamin Charlier, Jean Feydy, Joan Alexis Glaunès, François-David Collin, and Ghislain Durif. Kernel operations on the GPU, with autodiff, without memory overflows. *Journal of Machine Learning Research*, 22(74):1–6, 2021. URL https://jmlr.org/papers/v22/20-275.html.

[8] Krzysztof Marcin Choromanski, Arijit Sehanobish, Han Lin, Yunfan Zhao, Eli Berger, Tetiana Parshakova, Alvin Pan, David Watkins, Tianyi Zhang, Valerii Likhosherstov, Somnath Basu Roy Chowdhury, Kumar Avinava Dubey, Deepali Jain, Tamas Sarlos, Snigdha Chaturvedi, and Adrian Weller. Efficient graph field integrators meet point clouds. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 5978–6004. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/choromanski23b.html.

[9] Krzysztof Marcin Choromanski, Arijit Sehanobish, Somnath Basu Roy Chowdhury, Han Lin, Kumar Avinava Dubey, Tamás Sarlós, and Snigdha Chaturvedi. Fast tree-field integrators: From low displacement rank to topological transformers. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/41aa1c9f57ea83d7c41f0d3e98ed3dd4-Abstract-Conference.html.

[10] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009. ISBN 978-0-262-03384-8. URL http://mitpress.mit.edu/books/introduction-algorithms.

[11] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2292–2300, 2013. URL https://proceedings.neurips.cc/paper/2013/hash/af21d0c97db2e27e13572cbf59eb343d-Abstract.html.

[12] Marco Cuturi, Olivier Teboul, and Jean-Philippe Vert. Differentiable ranking and sorting using optimal transport. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 6858–6868, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/d8c24ca8f23c562a5600876ca2a550ce-Abstract.html.

[13] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. ISBN 978-3-319-21274-6. doi: 10.1007/978-3-319-21275-3. URL https://doi.org/10.1007/978-3-319-21275-3.

[14] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012. ISBN 978-3-642-14278-9.

[15] Alessio Figalli. The optimal partial transport problem. *Archive for Rational Mechanics and Analysis*, 195:533–560, 2010. URL https://api.semanticscholar.org/CorpusID:15899562.

[16] Arnold Filtser and Hung Le. Low treewidth embeddings of planar and minor-free metrics. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 1081–1092. IEEE, 2022. doi: 10.1109/FOCS54457.2022.00105. URL https://doi.org/10.1109/FOCS54457.2022.00105.

[17] Charlie Frogner, Chiyuan Zhang, Hossein Mobahi, Mauricio Araya-Polo, and Tomaso A. Poggio. Learning with a wasserstein loss. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2053–2061, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/a9eb812238f753132652ae09963a05e9-Abstract.html.

[18] Aude Genevay, Marco Cuturi, Gabriel Peyré, and Francis R. Bach. Stochastic optimization for large-scale optimal transport. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3432–3440, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/2a27b8144ac02f67687f76782a3b5d8f-Abstract.html.

[19] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5767–5777, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/892c3b1c6dccd52936e27cbd0ff683d6-Abstract.html.

[20] Guillaume Huguet, Alexander Tong, María Ramos Zapatero, Christopher J. Tape, Guy Wolf, and Smita Krishnaswamy. Geodesic sinkhorn for fast and accurate optimal transport on manifolds. In *2023 IEEE 33rd International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2023.

[21] Tomasz Kacprzak, Francois Kamper, Michael W. Weiss, Gianluca Janka, Ann M. Dillner, and Satoshi Takahama. Scalable approximate algorithms for optimal transport linear models. *ArXiv*, abs/2504.04609, 2025. URL https://api.semanticscholar.org/CorpusID:277621015.

[22] Kelvin Kan, Xingjian Li, and Stanley J. Osher. Ot-transformer: A continuous-time transformer architecture with optimal transport regularization. *CoRR*, abs/2501.18793, 2025. doi: 10.48550/ARXIV.2501.18793. URL https://doi.org/10.48550/arXiv.2501.18793.

[23] L. Kantorovitch. On the translocation of masses. *Management Science*, 5:1–4, 1958. URL https://api.semanticscholar.org/CorpusID:214798034.

[24] Tanguy Kerdoncuff, Rémi Emonet, and Marc Sebban. Metric learning in optimal transport for domain adaptation. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 2162–2168. ijcai.org, 2020. doi: 10.24963/IJCAI.2020/299. URL https://doi.org/10.24963/ijcai.2020/299.

[25] Nicholas I. Kolkin, Jason Salavon, and Gregory Shakhnarovich. Style transfer by relaxed optimal transport and self-similarity. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 10051–10060. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.01029.

[26] Mengyu Li, Jun Yu, Tao Li, and Cheng Meng. Importance sparsification for sinkhorn algorithm. *J. Mach. Learn. Res.*, 24:247:1–247:44, 2023. URL https://jmlr.org/papers/v24/22-1311.html.

[27] Eduardo Fernandes Montesuma, Fred Ngolè Mboula, and Antoine Souloumiac. Recent advances in optimal transport for machine learning, 2024. URL https://arxiv.org/abs/2306.16156.

[28] Yann Ollivier, Herve Pajot, and Cédric Villani, editors. *Optimal Transport - Theory and Applications*, volume 413 of *London Mathematical Society lecture note series*. Cambridge University Press, 2014. ISBN 978-1-10-768949-7. URL http://www.cambridge.org/de/academic/subjects/mathematics/geometry-and-topology/optimal-transport-theory-and-applications.

[29] Luiz Manella Pereira and M. Hadi Amini. A survey on optimal transport for machine learning: Theory and applications. *IEEE Access*, 13:26506–26526, 2025. doi: 10.1109/ACCESS.2025.3539926. URL https://doi.org/10.1109/ACCESS.2025.3539926.

[30] Gabriel Peyré. Optimal transport for machine learners. *CoRR*, abs/2505.06589, 2025. doi: 10.48550/ARXIV.2505.06589. URL https://doi.org/10.48550/arXiv.2505.06589.

[31] Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Found. Trends Mach. Learn.*, 11(5-6):355–607, 2019. doi: 10.1561/2200000073. URL https://doi.org/10.1561/2200000073.

[32] Gabriel Peyré, Marco Cuturi, and Justin Solomon. Gromov-wasserstein averaging of kernel and distance matrices. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, JMLR Workshop and Conference Proceedings, pages 2664–2672. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/peyre16.html.

[33] Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. Wasserstein barycenter and its application to texture mixing. In Alfred M. Bruckstein, Bart M. ter Haar Romeny, Alexander M. Bronstein, and Michael M. Bronstein, editors, *Scale Space and Variational Methods in Computer Vision - Third International Conference, SSVM 2011, Ein-Gedi, Israel, May 29 - June 2, 2011, Revised Selected Papers*, Lecture Notes in Computer Science, pages 435–446. Springer, 2011. doi: 10.1007/978-3-642-24785-9_37. URL https://doi.org/10.1007/978-3-642-24785-9_37.

[34] Samuel Ridler, Andrew J. Mason, and Andrea Raith. A simulation and optimisation package for emergency medical services. *European Journal of Operational Research*, 298(3):1101–1113, 2022. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2021.07.038. URL https://www.sciencedirect.com/science/article/pii/S0377221721006330.

[35] Tim Salimans, Han Zhang, Alec Radford, and Dimitris N. Metaxas. Improving gans using optimal transport. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rkQkBnJAb.

[36] Meyer Scetbon and Marco Cuturi. Linear time sinkhorn divergences using positive features. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 13468–13480. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/9bde76f262285bb1eaeb7b40c758b53e-Paper.pdf.

[37] Bernhard Schmitzer. Stabilized sparse scaling algorithms for entropy regularized transport problems. *SIAM Journal on Scientific Computing*, 41(3):A1443–A1481, 2019. doi: 10.1137/16M1106018.

[38] Ashkan Shahbazi, Elaheh Akbari, Darian Salehi, Xinran Liu, Navid Naderializadeh, and Soheil Kolouri. Espformer: Doubly-stochastic attention with expected sliced transport plans. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/shahbazi25a.html.

[39] Vikas Sindhwani, Tara N. Sainath, and Sanjiv Kumar. Structured transforms for small-footprint deep learning. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3088–3096, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/851300ee84c2b80ed40f51ed26d866fc-Abstract.html.

[40] Xun Tang, Michael Shavlovsky, Holakou Rahmanian, Elisa Tardini, Kiran Koshy Thekumparampil, Tesi Xiao, and Lexing Ying. Accelerating sinkhorn algorithm with sparse newton iterations. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=Kuj5gVp5GQ.

[41] Titouan Vayer, Nicolas Courty, Romain Tavenard, Laetitia Chapel, and Rémi Flamary. Optimal transport for structured data with application on graphs. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, Proceedings of Machine Learning Research, pages 6275–6284. PMLR, 2019. URL http://proceedings.mlr.press/v97/titouan19a.html.

[42] Hongteng Xu, Dixin Luo, Hongyuan Zha, and Lawrence Carin. Gromov-wasserstein learning for graph matching and node embedding. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, Proceedings of Machine Learning Research, pages 6932–6941. PMLR, 2019. URL http://proceedings.mlr.press/v97/xu19b.html.

[43] Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016.

A Additional Theoretical and Algorithmic Details

Efficient computation of cross_compute for $|S| = 1$ and general f. The problem of computing efficiently **cross_compute** is equivalent to the problem of the efficient (near-linear) multiplication with

matrices $\mathbf{M} = [f(x_i + y_j)]_{i=1,...,|A|}^{j=1,...,|B|}$ for given sequences of scalars: $(x_1, ..., x_{|A|})$ and $(y_1, ..., y_{|B|})$. In this section, we will define efficient (approximate or exact) algorithms to achieve this task. We start with the observation that the so-called *cordial functions*, defined in [9], by definition, support near-linear multiplication:

Definition A.1 (cordial functions; [9])**.** A function $f : \mathbb{R} \rightarrow \mathbb{R}$ is d-cordial (or: cordial if d is not specified) if there exists $d \in \mathbb{N}$ such that matrix-vector multiplication with a matrix $\mathbf{M} = [f(x_i + y_j)]_{i=1,...,a}^{j=1,...,b}$ can be **exactly** conducted in time $O((a+b)\log^d(a+b))$ for any two sequences of scalars $(x_1, ..., x_a)$ and $(y_1, ..., y_b)$.

As explained in [9], several general classes of functions are cordial:

- Rational functions f are $(2+\epsilon)$-cordial for any $\epsilon > 0$. Efficient multiplication can be conducted with the use of Fast Fourier Transform (FFT).
- Bounded-degree polynomial functions are 0-cordial. Thus for polynomial functions (special classes of rational functions) the above bound can be improved.
- Exponential functions are 0-cordial. This is literally the result that we presented in the main body of the paper, since it corresponds to the regular Sinkhorn algorithm with the entropy-based regularizer.
- Functions of the form: $f(x) = \frac{\exp(\lambda x)}{x+c}$ for hyperparameters λ, c are 3-cordial. This follows from the fact that in this case the corresponding matrix is the so-called *Cauchy-type low displacement rank matrix* and the theory of low-displacement rank matrices [39] (or to be more specific, *Cauchy Transform*) can be applied to produce efficient multiplication algorithm of time complexity $O((a+b)\log^3(a+b))$.

We can conclude the following:

Remark A.2. For all the classes of functions f defined above, *GenusSink* for the generalized Sinkhorn problem with the corresponding function f and leveraging the above methods remains asymptotic computational profile (space and time complexity wise) of its regular counterpart (for $f = \exp$).

A.1 Fourier analysis and random features for cross_compute

Assume that we can re-write function f as:

$$f(\mathbf{x} + \mathbf{y}) \approx \eta_1^\top(\mathbf{x})\eta_2(\mathbf{y}) \tag{14}$$

for some randomized $\eta_1, \eta_2 : \mathbb{R}^d \rightarrow \mathbb{R}^m$. Then it is easy to see that we can compute cross_compute for the case $|S| = 1$ in time $O(m(|A| + |B|))$, using matrix-associativity property. Thus for $m = \text{polylog}(|A| + |B|)$, the computations will be done efficiently. We will show now a systematic method to construct η_1, η_2. Note that we have:

$$f(\mathbf{z}) = \int_{\mathbb{R}^d} \exp(-2\pi k \omega^\top \mathbf{z})\tau(\omega)d\omega, \tag{15}$$

where τ is the inverse Fourier Transform of f:

$$\tau(\omega) = \int_{\mathbb{R}^d} \exp(2\pi k \mathbf{x}^\top \omega) f(\mathbf{x}) d\mathbf{x} \tag{16}$$

($k^2 = -1$). Thus, taking: $\mathbf{z} = \mathbf{x} + \mathbf{y}$, we can rewrite:

$$f(\mathbf{x} + \mathbf{y}) = C \cdot \mathbb{E}_{p(\omega)}[\exp(2\pi k \omega^\top \mathbf{x}) \exp(2\pi k \omega^\top \mathbf{y})], \tag{17}$$

where $p(\omega)$ is the probabilistic distribution with density proportional to $\tau(\omega)$ and $C = \int_{\mathbb{R}^d} \tau(\omega)d\omega$ (here we assume that the latter integral is well-defined!). Thus, if we can efficiently (potentially approximately) sample from $p(\omega)$, then we can leverage the following:

$$f(\mathbf{x} + \mathbf{y}) \stackrel{\mathbb{E}}{=} \eta_1^\top(\mathbf{x})(\eta_2(\mathbf{y})), \tag{18}$$

where for $\omega_1, ..., \omega_m \sim p(\omega)$ and for some $r \in \mathbb{N}$ (number of random features) the following holds:

$$\eta_1(\mathbf{v}) = \eta_2(\mathbf{v}) = \sqrt{\frac{C}{r}} \left(\exp(2\pi k \omega_1^\top \mathbf{v}), ..., \exp(2\pi k \omega_r^\top \mathbf{v})\right). \tag{19}$$

Thus the only thing we required from f is to have a well defined Inverse Fourier Transform τ and also in principle for τ to satisfy: $\tau \geq 0$. But the last assumption is actually not needed since we can always partition τ into its positive and negative part and conduct calculations separately for each of them, only to combine them at the end.

We can conclude the following:

Remark A.3. If f has the well-defined Inverse Fourier Transform, *GenusSink* for the generalized Sinkhorn problem with the corresponding function f and leveraging the above method remains asymptotic computational profile (space and time complexity wise) of its regular counterpart (for $f = \exp$), as long as $m = \text{polylog}(|A| + |B|)$. Note that here ***cross_compute*** is approximated rather than computed exactly (as it was the case for all scenarios considered before).

A.2 Separator Subsampling

We quantify the role of separator coverage in the bounded-diameter motivation given in Sec. 3.3 (for a single original separation (A, S, B) and exact parent-graph distances).

Lemma A.4 (Separator coverage radius)**.** *Let $Q \subseteq S$ be a nonempty retained subset of a separator S in a connected weighted graph, and let $\rho = \max_{s\in S} \min_{q\in Q} d_G(s, q)$. For $a \in A$ and $b \in B$, define $\widehat{d}_Q(a, b) = \min_{q\in Q}(d_G(a, q) + d_G(q, b))$. Then*

$$d_G(a, b) \leq \widehat{d}_Q(a, b) \leq d_G(a, b) + 2\rho. \tag{20}$$

Consequently, for the exponential kernel,

$$e^{-2\rho/\epsilon} e^{-d_G(a,b)/\epsilon} \leq e^{-\widehat{d}_Q(a,b)/\epsilon} \leq e^{-d_G(a,b)/\epsilon}. \tag{21}$$

Proof. The lower distance bound follows from the triangle inequality. A shortest path from a to b meets some $s \in S$, so $d_G(a, b) = d_G(a, s) + d_G(s, b)$. Choose $q \in Q$ with $d_G(s, q) \leq \rho$ and apply the triangle inequality to each term in $d_G(a, q) + d_G(q, b)$, which gives the upper bound. The kernel inequalities follow by exponentiation. □

When the graph discretizes a fixed physical object, refinement can increase n while the diameter remains bounded. The bound shows why a retained set that covers the separator well can preserve accurate cross-kernel actions; the quantitative requirement is that ρ be small relative to ϵ. This is a single-split estimate using exact parent-graph distances, rather than a guarantee for the complete recursive heuristic that randomly reassigns discarded separator vertices. We therefore evaluate the practical approximation empirically through the depth and retention experiments in Appendix B.4.

A.3 Exact geodesic Sinkhorn on small-treewidth graphs

We make explicit the numerical-equivalence statement for graphs of treewidth $O(\log \log n)$. The result follows from exact S-GFI integration with full separators and the construction and integration bounds established in Remarks 3.1 and 3.2.

Corollary A.5 (Exact geodesic Sinkhorn on small-treewidth graphs)**.** *Let G be an n-vertex graph with $tw(G) = O(\log \log n)$ and $\epsilon > 0$. With full separators, metric-completed child graphs, and exact cross computations for $f_\epsilon(t) = \exp(-t/\epsilon)$, S-GFI computes $\mathbf{Kx}$ exactly in real arithmetic for every $\mathbf{x}$, where $\mathbf{K}_{ij} = \exp(-d_G(i, j)/\epsilon)$. Construction, storage, and each kernel action require $n^{1+o(1)}$ resources under the bounds established in Remarks 3.1 and 3.2. Consequently, Sinkhorn initialized identically and run for the same number of iterations produces the same iterates and transport plan as the dense geodesic implementation, up to floating-point effects.*

Proof. At a leaf, the kernel action is computed explicitly. At each internal node, metric completion preserves the child distances, the separator minimum gives the exact cross distances, and Eq. 8 combines these actions without double counting. Induction over the S-GFI tree proves exactness of every kernel action and therefore of the resulting Sinkhorn iterates. The resource bounds follow from Remarks 3.1 and 3.2. □

B Numerical Experiments

We first describe the synthetic graphs and real meshes used in Section 4, including additional tests on nonzero-genus surfaces and meshes with larger separators. We then vary recursion depth and separator retention to distinguish the effect of recursive decomposition from the effect of subsampling. All experiments were run on an Apple M3 machine with 8 CPU cores, a 10-core integrated GPU, and 16 GB of unified memory, running macOS 26.3.1.

B.1 Details on the pseudo-genus family

To build some intuition for how GenusSink constructs the separation tree and applies the kernel, we begin by considering a particularly stylized example, that is, a dumbbell-shaped planar graph, shown in Fig. 7. This geometry admits a small separator that splits the graph into two nearly equal subgraphs, making it a natural setting for a depth-1 separation tree.

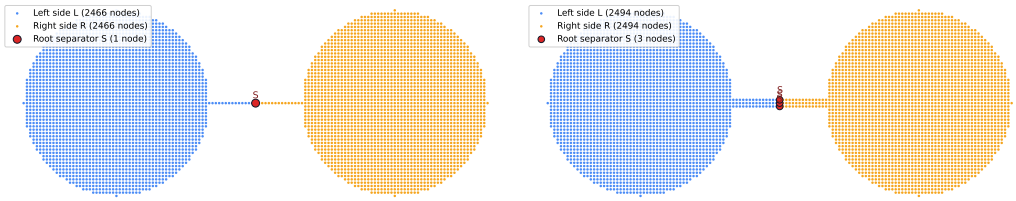

Figure 7: Planar dumbbell graphs after the root separator split. The two lobes form the left and right subgraphs, while the narrow bridge induces a small separator; the examples above correspond to two different handle widths.

We evaluate the time required for Sinkhorn to converge to stopping tolerance $\tau = 10^{-7}$ on dumbbell graphs with radii $r \in \{10, 18, 26, 34, 42, 50, 56\}$, bridge length equal to the radius, handle width $w \in \{1, 2, 3\}$, and initial/target distributions given by two-component Gaussian mixtures localized on the left and right lobes, respectively, and normalized over the graph vertices. The entropic regularization parameter is set to $\varepsilon = 0.3r$. Since the handle width also controls the separator size in this construction, these experiments illustrate how the performance of GenusSink changes as the separator becomes larger. We compare against the dense full-kernel geodesic Sinkhorn baseline, Greenkhorn [2], Screenkhorn [1], Sparse Sinkhorn [37], Geodesic heat kernel [20], and two Linear Sinkhorn baselines [36]: the original ambient-Euclidean feature construction and a geodesic Nyström approximation. With separation height fixed to 1, GenusSink converges substantially faster than the brute-force baseline that materializes the full dense kernel and then performs matrix–vector products; see Fig. 8. As the graph size grows, constructing the dense kernel becomes increasingly expensive due both to memory costs and to the need to compute all pairwise shortest-path distances. In contrast, GenusSink remains exact up to floating-point precision, whereas the approximate baselines incur visibly larger error.

In addition to the dumbbell graph, we evaluate our method on a family of boundaryless sphere-topology "pseudo-genus" surface graphs embedded in $\mathbb{R}^3$ (as seen in Section 4). These graphs remain planar, but become progressively more complicated geometrically. Specifically, increasing the pseudo-genus order introduces additional narrow, highly squashed neck regions that visually resemble higher-genus shapes while preserving planarity, which is essential for our separator-tree construction. The resulting family therefore provides a more challenging synthetic testbed than the dumbbell graph while still lying in the regime targeted by our method (see Fig. 9).

For these timing and accuracy experiments, we use pseudo-genus graphs of orders 1, 2, and 3, with node counts matched to the width-1 dumbbell size sequence, namely $n \in \{644, 2036, 4268, 7283, 11090, 15740, 19745\}$. On each graph, we compute Sinkhorn transport between two smooth nonuniform probability distributions, defined as two-component Gaussian mixtures in the surface parameter coordinates and normalized over the graph vertices. We again compare GenusSink against the dense geodesic full-kernel Sinkhorn baseline, Greenkhorn, Ambient LinearSinkhorn, Geod. Nyström, Sparse Sinkhorn, Screenkhorn, and Geod. heat kernel. The approximate baselines avoid the exact separator-tree computation but incur visible transport-cost error; in particular, Ambient LinearSinkhorn uses Euclidean features rather than the intrinsic graph geometry.

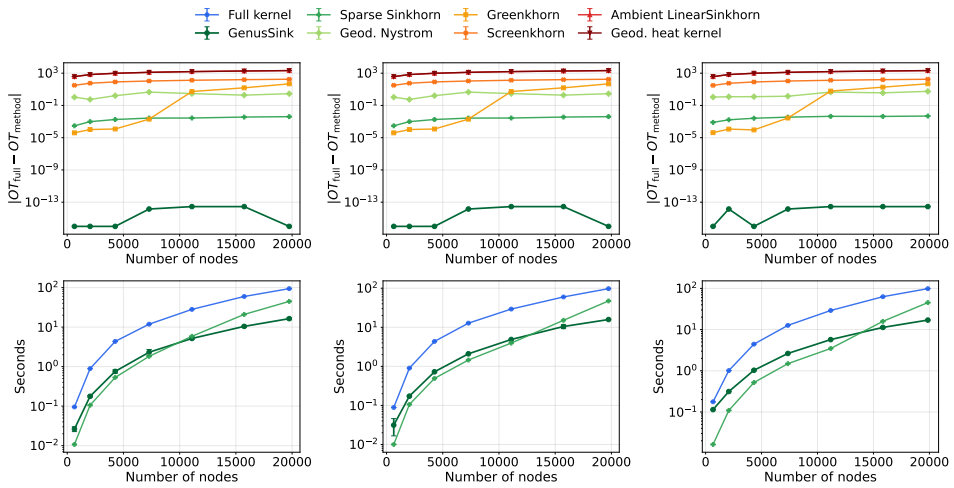

Figure 8: Planar dumbbell experiments for handle widths $w = 1, 2, 3$ (left to right). **Top:** absolute Sinkhorn-cost error relative to the dense geodesic full-kernel baseline. **Bottom:** total runtime for the baseline and two most accurate efficient methods: Sparse Sinkhorn and GenusSink. Across all widths, GenusSink remains numerically exact while providing significant computational gains.

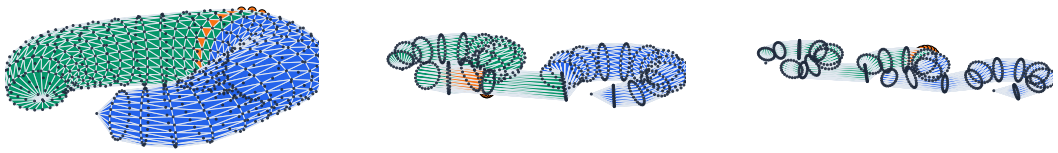

Figure 9: Pseudo-genus surface family used in our synthetic experiments. We show planar sphere-topology surface meshes embedded in $\mathbb{R}^3$ for pseudo-genus orders 1, 2, and 3, colored according to the root separator split used to initialize the separation tree.

B.2 Details using Thingi10K data

As mentioned in Section 4, we evaluate GenusSink on a variety of Thingi10K triangle meshes converted to weighted graph instances. For each mesh, every unique triangle edge is used as an undirected graph edge with weight equal to its Euclidean length. The experiment uses 28 meshes, with mesh IDs

$$636808, 472186, 52562, 636795, 636813, 149255, 636794, 650181, 51809, 642501,$$
$$650180, 39879, 131453, 650183, 636818, 650184, 79741, 780015, 636800, 636796,$$
$$636816, 650185, 636799, 636817, 636797, 636803, 159695, 41272.$$

The corresponding vertex counts range from 2771 to 12472. Source and target measures are deterministic smooth geodesic Gaussian mixtures. PCA anchors define left, right, low, and high mesh locations; the source uses anchors $(\text{left}, \text{low})$ with weights $(0.7, 0.3)$, while the target uses $(\text{right}, \text{high})$ with weights $(0.65, 0.35)$. For graph distance d_G, each mixture component is proportional to

$$\exp\left(-\frac{d_G(x,c)^2}{2\sigma^2}\right), \qquad \sigma = \max\{0.18 \times \text{diam},\ 3\,\overline{w},\ 10^{-8}\},$$

where $\overline{w}$ is the mean mesh-edge length and diam is the Euclidean diagonal length of the mesh's axis-aligned bounding box. The entropic scale is $\varepsilon = 0.05 \times \text{diam}$. We compare against dense geodesic Sinkhorn as the reference and report both dense kernel construction time versus GenusSink total runtime, and absolute OT-cost error relative to the dense reference. Additional accuracy baselines are Greenkhorn with 50,000 updates and tolerance 10^{-4}, Geodesic Nyström with rank 64, Ambient LinearSinkhorn with 64 random features, Screenkhorn with budget fraction 0.5 and `maxiter` $=$ `maxfun` $= 500$, sparse truncated Sinkhorn retaining entries with $K_{ij} \geq 10^{-4}$, and

geodesic heat-kernel Sinkhorn with $\tau = 0.25\varepsilon$, Chebyshev order 20, and inverse-distance graph Laplacian weights. The smaller speedups on Thingi10K, as compared to the pseudo-genus graphs, are primarily due to implementation-level factors and separator size. Larger separators increase GenusSink's preprocessing and kernel-application costs. In addition, the current implementation incurs overhead from recursive preprocessing, tree traversal, and less optimized memory access. Improved vectorized, compiled, and parallel implementations could reduce these constants.

B.2.1 Thingi10K Meshes with Larger Separators

In addition to the original Thingi10K experiments, which contain meshes with relatively small separators, we tested ten additional meshes with larger separators, ranging from 144 to 433 vertices, and retained 64 points from each separator. This helps us examine the practical approximation when retaining the complete separator is more expensive. The reported subset uses a depth-one tree and k-center selection of the retained separator vertices. The experiments show that GenusSink's relative OT-cost error remains low after sampling, ranging from 0.1664% to 2.3851% on these meshes. These sampled-separator results are approximate, in contrast to the floating-point agreement obtained on the original mesh set.

Table 1: Additional Thingi10K meshes with larger separators, using depth $h = 1$ and 64 retained separator vertices selected by k-center sampling. Separator sizes refer to the selected root partition, and relative OT-cost errors are percentages.

Mesh ID	Vertices	Original separator	Relative OT-cost error
1454388	11,863	183	0.2826%
856334	10,607	241	0.2396%
101636	8,972	187	0.1664%
94662	6,465	220	2.3851%
204243	9,211	144	0.6217%
472113	10,489	245	0.2076%
77019	10,304	364	1.1352%
1036466	6,768	433	0.4855%
1454016	6,310	342	0.8770%
472059	7,265	265	0.2207%

B.3 Graphs of Genus 1, 2, and 3

The pseudo-genus examples above remain planar, so we additionally evaluate mesh graphs on closed surfaces with one, two, and three handles. These surfaces are constructed by removing disjoint tunnels from a solid block and triangulating its boundary. Source and target distributions are smooth Gaussian mixtures defined in the construction coordinates and normalized over the graph vertices. Tables 2 and 3 show experiment results on graphs of genus 1, 2, and 3, with root separators of size 40, 44, and 40, respectively, and a depth of 3 across all graphs. All separator vertices are retained in these runs. We see that GenusSink is approximately two times faster while retaining floating-point accuracy in the reported OT-cost comparison.

Table 2: Experiments on graphs of genus 1, 2, and 3, with depth $h = 3$ and full separator retention. Runtimes are mean $\pm$ standard deviation over four repetitions with alternating execution order, including preprocessing and solve time.

Genus	Vertices	Root separator	Full kernel (s)	GenusSink (s)	Speedup
1	9,760	40	11.69 ± 1.94	5.75 ± 0.62	2.03×
2	10,310	44	13.10 ± 1.27	6.91 ± 0.67	1.89×
3	10,860	40	14.63 ± 1.49	7.29 ± 0.30	2.01×

B.4 Ablations over Depth and Separator Retention

We next distinguish the effect of recursion depth from the effect of separator subsampling. We vary the depth from 1 to 4 and retain between 10% and 100% of each separator. With full retention,

Table 3: Absolute regularized OT-cost errors for the genus 1–3 experiments, using the dense geodesic full-kernel baseline as the reference. Ambient denotes Ambient LinearSinkhorn, and heat denotes the geodesic heat-kernel baseline.

Genus	GenusSink	Greenkhorn	Sparse	Nyström	Screenkhorn	Ambient	Heat
1	1.33×10^{-15}	1.17×10^{-6}	1.59×10^{-3}	8.30×10^{-3}	6.44×10^{-2}	2.41×10^{-1}	2.56×10^{1}
2	8.88×10^{-16}	3.23×10^{-6}	1.50×10^{-3}	1.30×10^{-3}	6.52×10^{-2}	2.39×10^{-1}	2.55×10^{1}
3	4.44×10^{-16}	3.23×10^{-6}	1.41×10^{-3}	1.62×10^{-3}	6.56×10^{-2}	2.38×10^{-1}	2.56×10^{1}

the kernel action agrees with the dense reference to floating-point accuracy at every tested depth; reducing retention introduces approximation error, which generally increases with deeper recursion and smaller retained fractions.

Relative kernel-action error measures how well the sampled-separator method approximates the kernel operator, whereas relative regularized OT-cost error measures the final scalar Sinkhorn objective for the chosen source and target distributions. For each run, kernel-action error is the mean of $\|\widehat{K}x - Kx\|_2/\|Kx\|_2$ over five positive random probe vectors normalized to unit sum, and relative OT-cost error is $|\widehat{F} - F|/|F|$ for the regularized objective values. We first take the median over sampling seeds within each mesh and then the median across meshes. The ablation uses the original 28 meshes, five fixed sampling seeds below full retention, and a seed-independent exact control at full retention. The entropic scale is $\varepsilon = 0.05$ times the mesh bounding-box diagonal, and errors are evaluated at the final iterate after at most 100 iterations. Of the 2912 runs, 2832 converged within this iteration budget; the reported errors include iteration-limited runs. With 100% separator retention, GenusSink remains numerically equal to the dense reference at all tested depths. In this ablation, error arises from separator subsampling rather than recursion depth alone.

Table 4: Relative kernel-action error as a function of recursion depth h and retained separator fraction. Values are relative errors, not percentages.

h	100%	90%	70%	50%	30%	10%
1	1.668×10^{-15}	2.786×10^{-3}	8.597×10^{-3}	1.258×10^{-2}	2.153×10^{-2}	6.845×10^{-2}
2	1.160×10^{-15}	1.027×10^{-2}	2.155×10^{-2}	3.785×10^{-2}	7.011×10^{-2}	1.323×10^{-1}
3	9.149×10^{-16}	1.403×10^{-2}	3.389×10^{-2}	5.576×10^{-2}	1.103×10^{-1}	1.906×10^{-1}
4	8.090×10^{-16}	1.798×10^{-2}	4.449×10^{-2}	8.138×10^{-2}	1.555×10^{-1}	2.354×10^{-1}

Table 5: Relative regularized OT-cost error as a function of recursion depth h and retained separator fraction. Values are relative errors, not percentages.

h	100%	90%	70%	50%	30%	10%
1	7.541×10^{-16}	4.356×10^{-4}	2.011×10^{-2}	4.626×10^{-2}	1.328×10^{-1}	4.873×10^{-1}
2	6.881×10^{-16}	3.215×10^{-3}	3.276×10^{-2}	8.038×10^{-2}	1.749×10^{-1}	5.300×10^{-1}
3	6.262×10^{-16}	3.686×10^{-3}	3.436×10^{-2}	8.496×10^{-2}	1.812×10^{-1}	5.536×10^{-1}
4	6.704×10^{-16}	3.893×10^{-3}	3.482×10^{-2}	8.650×10^{-2}	1.858×10^{-1}	5.579×10^{-1}

B.5 Broader Impact & Limitations

Broader Impact. This paper might potentially have a positive impact on the new classes of Transformer architectures that leverage Optimal Transport methods in their attention mechanisms, by making these mechanisms much more computationally efficient.

Societal Impact. We are not aware of any societal impact that this paper might have, as it is mainly of the algorithmic flavor, proposing a new class of efficient Sinkhorn algorithms operating on bounded genus graphs and with cost functions defined via shortest-path-distance metrics.

Limitations. This paper focuses on bounded genus graphs. In future work, we plan to extend to larger classes of graphs, in particular for minor-free graphs (every bounded genus graph is minor-free, and the class of minor-free graphs is a strict super-class of the bounded genus graphs). An analogue of

Theorem 3.3 exists for minor-free graphs [16], but its construction time is not necessarily near-linear and can instead be polynomial. Potential near-linear approximations of that procedure might suffice, and the separator-subsampling procedure may also work in practice for these larger graph families.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We provide detailed theoretical analysis of the algorithm in the paper (both: its computational time and accuracy) and complement it with thorough empirical studies, involving **seven** other methods. That proves claims made in the abstract and introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes, we discuss limitations of the work and its broader impact in Sec. B.5.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Yes, the paper provides the full set of assumptions for the theoretical claims made and correct proofs of the presented statements. For the relevant statements with proofs in the already peer-reviewed papers, we provide exact references.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Yes, the paper fully discloses all the information needed to reproduce the main experimental results of the paper. We provide all the details regarding experimental scenarios, clearly list the methods we compare our algorithm against, as well as include additional details such as the hyperparameter setups (in the Appendix).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We will make the code available upon the acceptance of the paper. As mentioned above, provided details are sufficient to reconstruct all presented empirical results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies all the training and test details necessary to understand as well as accurately reproduce the results in the main body and the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Yes, we report statistical significance metrics via error bars.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes, we provide all those details in Section B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes, the research conducted in the paper conforms, in every respect, with the NeurIPS Code of Ethics. In particular, we preserve full anonymity.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: The paper discusses the above explicitly in Section B.5.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [N/A]

    Justification: The paper does not use assets requiring licenses.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.