SCALABLE OPTIMAL TRANSPORT VIA SPARSE EDGE RECOMBINATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In large-scale optimal transport (OT) between two equally sized, uniformly weighted discrete distributions, there are N^2 possible correspondences between N source and N target points, although an optimal assignment uses only N of them. This motivates constructing a small candidate edge set that still contains a low-cost perfect matching. BSP-OT offers one such approach: it constructs sparse approximate assignments through randomized paired partitions. However, its random directions in high dimensions, refinement to singleton pairs, and sequential merging can miss or discard useful candidate edges. To address these limitations, we propose *SER-OT* (Sparse Edge Recombination for Optimal Transport), retaining BSP-OT's paired-partitioning framework. SER-OT recursively partitions the source and target point sets using shared data-dependent projection directions formed from pairs of target points, and solves an exact linear assignment problem within each leaf. Repeating this procedure over k randomized partition trees produces multiple sparse candidate matchings, whose retained edges are recombined into a single candidate graph with at most kN edges. This reduces the candidate space from N^2 possible correspondences to at most kN edges, while allowing edges discovered by different partition trees to be jointly optimized in the final assignment. Experiments show that SER-OT achieves small relative cost gaps across large-scale, high-dimensional, and geometrically diverse benchmarks while maintaining practical runtime and memory usage.

1 INTRODUCTION

Optimal transport (OT) provides a fundamental framework for finding minimum-cost correspondences between distributions (Peyré and Cuturi, 2019), with applications in machine learning (Courty et al., 2017), computer graphics (Mérigot, 2011), and statistics (Panaretos and Zemel, 2019). We consider OT between two uniformly weighted discrete distributions with the same number N of points. In this setting, OT reduces to the linear assignment problem (LAP): finding a minimum-cost one-to-one matching between the two point sets. Although the final assignment contains only N matched edges, there are N^2 possible source–target pairs. Evaluating and storing all pairwise costs is therefore prohibitive at large scale. A central challenge is to identify a small candidate edge set that still contains a low-cost perfect matching.

To address this challenge, projection-based methods provide a way to efficiently identify plausible correspondences without explicitly examining all N^2 pairs. Among them, BSP-OT (Genest et al., 2025) constructs approximate assignments through randomized paired partitions without evaluating every source–target cost (see Section 3 for details). In this work, we retain this partitioning mechanism while seeking a more effective way to discover and use candidate edges.

From this perspective, BSP-OT has three limitations in how it discovers and uses candidate edges. First, the random projection directions used in high dimensions may fail to capture informative variation in the point clouds, resulting in poor paired partitions. Second, each tree is recursively refined to singleton pairs, which can miss lower-cost correspondences that could instead be found by directly optimizing small local subproblems before further splitting. Third, the matchings obtained from different trees are merged sequentially, which can lose opportunities to jointly optimize useful edges discovered across different trees.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

To address these limitations, we propose *SER-OT* (Sparse Edge Recombination for Optimal Transport). SER-OT retains the paired-partition structure of BSP-OT while changing how candidate edges are discovered and used. First, at each partitioning step, we use the difference between two randomly sampled target points as a data-dependent projection direction. Second, rather than refining every tree to singleton pairs, SER-OT stops when each paired leaf contains at most b points per side and solves the LAP exactly within each leaf, retaining only the resulting matching edges. Third, SER-OT retains the matching edges discovered by all k randomized trees, pools them into a single candidate edge set, and solves one exact global LAP over their union. By postponing final assignment selection until the global optimization, SER-OT preserves cross-tree recombination opportunities that can be lost through sequential merging. Although introducing a global LAP may appear computationally expensive, each tree contributes only N retained edges, so the candidate edge set contains at most kN edges. For fixed k, b, and d, fixed finite-precision costs, and $O(d)$-time pair-cost evaluations, the complete procedure admits an $O(N^{1.5} \log N)$ worst-case implementation with a classical exact sparse-assignment backend.

Beyond this bound, the two exact LAP stages satisfy non-worsening guarantees under fixed paired leaves or candidate edge sets. These guarantees do not establish end-to-end dominance over BSP-OT, since the two methods generally construct different candidate edges. We therefore demonstrate the overall performance of SER-OT through large-scale experiments. Across the evaluated benchmarks, SER-OT achieves low relative cost gaps with practical runtime and compares favorably with BSP-OT and other recent OT solvers.

Our main contributions are:

1. We introduce SER-OT, combining inexpensive data-dependent partitioning with exact recombination of candidate edges across trees.
2. We show a super-linear worst-case time bound of $O(N^{1.5} \log N)$ for an exact SER-OT implementation under the assumptions above, using $O(N)$ memory.
3. We show empirically that cross-tree edge recombination yields accurate assignments with practical runtime and memory usage on large-scale and high-dimensional problems.

2 PRELIMINARIES

2.1 NOTATION

Ordinary lowercase letters, such as a, denote scalars; bold lowercase letters, such as $\mathbf{a}$, denote column vectors; and bold uppercase letters, such as $\mathbf{A}$, denote matrices. Their entries are written in ordinary type, e.g., a_i for the i-th entry of $\mathbf{a}$ and A_{ij} for the entry of $\mathbf{A}$ at position (i, j). Calligraphic letters, such as $\mathcal{A}$, denote sets, and $|\mathcal{A}|$ denotes the cardinality of a finite set $\mathcal{A}$. For disjoint sets $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{A} \sqcup \mathcal{B}$ denotes their union. We write $\| \cdot \|$ for the Euclidean (ℓ_2) norm. For a positive integer N, let $[N] := \{1, \dots, N\}$. We write $\mathbb{R}$ for the set of real numbers, $\mathbb{R}_+ := [0, \infty)$ for the set of nonnegative real numbers, $\mathbb{R}^d$ for the d-dimensional real vector space, and $\mathbb{R}^{m \times n}$ for the set of $m \times n$ real matrices. Finally, $\mathbf{1}_N \in \mathbb{R}^N$ denotes the all-ones vector.

2.2 OPTIMAL TRANSPORT

Let $\mathbf{X} := [\mathbf{x}_1, \dots, \mathbf{x}_N]$ and $\mathbf{Y} := [\mathbf{y}_1, \dots, \mathbf{y}_N]$ in $\mathbb{R}^{d \times N}$ denote the source and target point clouds, respectively, where $\mathbf{x}_i, \mathbf{y}_j \in \mathbb{R}^d$ are the i-th source and j-th target points. We use nonbold X and Y as labels for the source and target sides, respectively. We assign uniform weight $1/N$ to each point and define the corresponding empirical measures as $\mu := \frac{1}{N} \sum_{i=1}^{N} \delta_{\mathbf{x}_i}$ and $\nu := \frac{1}{N} \sum_{j=1}^{N} \delta_{\mathbf{y}_j}$, where $\delta_{\mathbf{x}}$ denotes a unit mass at $\mathbf{x}$. Given a finite-valued cost function $c : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, let $c_{ij} := c(\mathbf{x}_i, \mathbf{y}_j)$.

The balanced OT problem between μ and ν is defined as

$$\mathrm{OT}_c(\mu, \nu) := \min_{\substack{\mathbf{P} \in \mathbb{R}_+^{N \times N} \\ \mathbf{P}\mathbf{1}_N = \mathbf{1}_N / N \\ \mathbf{P}^\top \mathbf{1}_N = \mathbf{1}_N / N}} \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} P_{ij}. \tag{1}$$

where P_{ij} denotes the mass transported from source point $\mathbf{x}_i$ to target point $\mathbf{y}_j$ (Peyré and Cuturi, 2019). Because both measures consist of N uniformly weighted points, there exists an optimal

transport plan that is a scaled permutation matrix. Therefore, the OT problem reduces to a linear assignment problem (LAP).

For any equal-sized source and target index sets $\mathcal{I}, \mathcal{J} \subseteq [N]$, let $\mathcal{P}(\mathcal{I}, \mathcal{J}) := \{\mathcal{M} \subseteq \mathcal{I} \times \mathcal{J} \mid \forall i \in \mathcal{I},\ \exists! j \in \mathcal{J} \text{ such that } (i,j) \in \mathcal{M};\ \forall j \in \mathcal{J},\ \exists! i \in \mathcal{I} \text{ such that } (i,j) \in \mathcal{M}\}$ be the set of perfect matchings between them. For $\mathcal{M} \in \mathcal{P}([N],[N])$, we define its cost as $C(\mathcal{M}) := \frac{1}{N}\sum_{(i,j)\in\mathcal{M}} c_{ij}$. Then

$$C^\star := \min_{\mathcal{M}\in\mathcal{P}([N],[N])} C(\mathcal{M}) = \mathrm{OT}_c(\mu,\nu).$$

Although every $\mathcal{M} \in \mathcal{P}([N],[N])$ contains only N edges, the complete bipartite graph contains N^2 possible source–target pairs.

2.3 Related Work

For a comprehensive review of computational optimal transport, we refer to Peyré and Cuturi (2019). We focus on methods for large-scale discrete OT that avoid explicitly handling all N^2 source–target pairs while still seeking an explicit transport plan or assignment. Two lines of work are particularly relevant to this goal: coarse-to-fine methods that progressively restrict the candidate support, and projection-based methods that use low-dimensional structure to simplify high-dimensional OT computations.

Coarse-to-fine hierarchical OT methods exploit data geometry to progressively refine transport solutions across multiple scales (Mérigot, 2011; Schmitzer, 2016; Gerber and Maggioni, 2017). HiRef (Halmos et al., 2025) uses low-rank OT subproblems to construct dynamic multiscale partitions and progressively refines coarse correspondences into a bijection. HALO (Xia et al., 2026) solves a hierarchy of OT problems, warm-starting each finer level from the coarser solution while updating the active candidate support and solving restricted linear programs. In both cases, optimization across successive levels is used to progressively identify a smaller set of plausible transport edges.

Low-dimensional projections provide another route to reducing the computational burden of high-dimensional OT. A prominent example is the Sliced Wasserstein framework (Rabin et al., 2011; Bonneel et al., 2015), which computes one-dimensional Wasserstein distances along multiple projection directions and aggregates them into an efficiently computable discrepancy between distributions. Because each one-dimensional OT problem can be solved efficiently by sorting, this approach enables inexpensive comparison of high-dimensional distributions. Its primary purpose, however, is not to construct a single explicit transport plan in the original high-dimensional space.

Several works have therefore explored how projections can instead be used to construct explicit transport plans. One approach lifts and combines one-dimensional transport plans back in the original space (Liu et al., 2025), while another derives a transport plan from a single optimized projection direction (Mahey et al., 2023). BSP-OT (Genest et al., 2025) takes a different approach: rather than determining the transport plan directly from projected one-dimensional problems, it uses projection orderings to recursively partition the source and target point sets into paired subsets. This yields an approximate assignment without evaluating all N^2 source–target costs. Unlike random-projection trees, which partition a single point cloud (Dasgupta and Freund, 2008), BSP-OT performs coordinated partitioning of the two point sets. Its lightweight partitioning mechanism therefore provides a natural framework for constructing candidate matchings at scale. We take BSP-OT as the starting point for SER-OT and review its procedure in Section 3.

3 BSP-OT and Motivation

We use BSP-OT (Genest et al., 2025) as the starting point for SER-OT. Accordingly, this section first reviews the BSP-OT procedure, which consists of *tree construction* and *sequential merging* and is summarized in Algorithm 1, and then discusses the limitations that motivate our modifications.

Tree construction. Figure 1(a–d) illustrates the paired partitioning procedure of BSP-OT. Although the source and target point clouds lie in the same space $\mathbb{R}^d$, Figure 1 draws them separately as the X and Y sides to make their corresponding regions explicit. Because each paired region is recursively divided into two child regions, the procedure naturally forms a binary tree; Figure 1 also depicts the corresponding tree structure.

Algorithm 1 BSP-OT

Input: Source $\mathbf{X}$, target $\mathbf{Y}$; cost function c
Number of trees k
Output: A perfect matching $\mathcal{M}_{\mathrm{BSP}}$

for $r = 1, \ldots, k$ **do**
  Start from the full clouds $\mathbf{X}, \mathbf{Y}$
  while a leaf node has size > 1 **do**
    Select such a node v
    Choose a shared direction $\mathbf{w}$
    Project both sides onto $\mathbf{w}$
    Split both rankings at one pivot rank
    Form two paired child nodes of v
  end while
  Form $\mathcal{M}^{(r)}$ from the singleton pairs
end for
$\mathcal{M} \leftarrow \mathcal{M}^{(1)}$
for $r = 2, \ldots, k$ **do**
  $\mathcal{M} \leftarrow \mathrm{BSPMerge}(\mathcal{M}, \mathcal{M}^{(r)})$
end for
$\mathcal{M}_{\mathrm{BSP}} \leftarrow \mathcal{M}$
return $\mathcal{M}_{\mathrm{BSP}}$

Algorithm 2 SER-OT

Input: Source $\mathbf{X}$, target $\mathbf{Y}$; cost function c
Number of trees k
Partition stopping size b
Output: A perfect matching $\mathcal{M}_{\mathrm{SER}}$

for $r = 1, \ldots, k$ **do**
  Start from the full clouds $\mathbf{X}, \mathbf{Y}$
  while a leaf node has size $> b$ **do**
    Select such a node v
    Choose $\mathbf{w}$ from a target pair
    Project both sides onto $\mathbf{w}$
    Split both rankings at one pivot rank
    Form two paired child nodes of v
  end while
  Solve an exact LAP in each paired leaf
  Form $\mathcal{M}^{(r)}$ from the leaf matchings
end for
$\mathcal{E} \leftarrow \bigcup_{r=1}^{k} \mathcal{M}^{(r)}$
Solve the exact LAP on $\mathcal{E}$ for $\mathcal{M}_{\mathrm{SER}}$
return $\mathcal{M}_{\mathrm{SER}}$

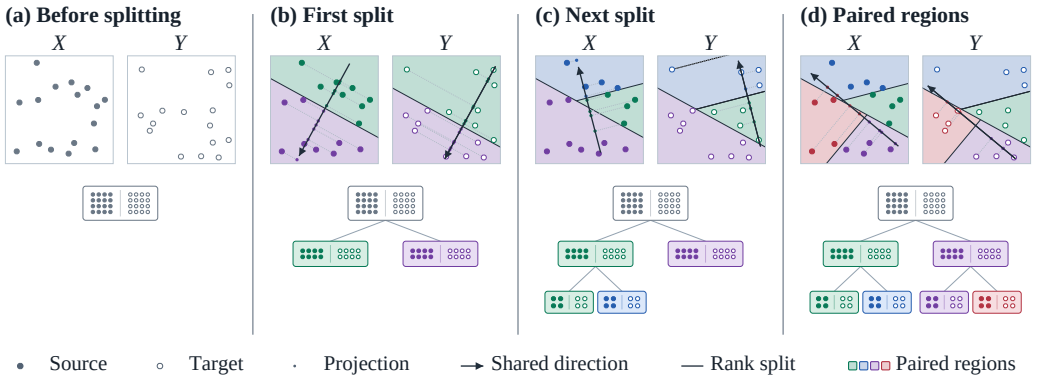


Figure 1: Illustration of the paired rank-partitioning mechanism shared by BSP-OT and SER-OT.

For either method, let $\mathcal{T}^{(r)}$ be the r-th partition tree and $\mathcal{V}^{(r)}$ its node set; nodes in different trees are distinct. Each node $v \in \mathcal{V}^{(r)}$ has a source index set $\mathcal{I}_v^X \subseteq [N]$ and a target index set $\mathcal{I}_v^Y \subseteq [N]$, with $|\mathcal{I}_v^X| = |\mathcal{I}_v^Y|$. The superscripts X and Y likewise identify the source and target sides. For $\mathcal{S} \subseteq [N]$, write $\mathbf{X}_{\mathcal{S}}$ and $\mathbf{Y}_{\mathcal{S}}$ for the submatrices with columns indexed by $\mathcal{S}$ in increasing order. In Figure 1(a), the root $\rho^{(r)}$ contains both full clouds, so $\mathcal{I}_{\rho^{(r)}}^X = \mathcal{I}_{\rho^{(r)}}^Y = [N]$.

To split a node v, BSP-OT projects $\mathbf{X}_{\mathcal{I}_v^X}$ and $\mathbf{Y}_{\mathcal{I}_v^Y}$ onto a common direction $\mathbf{w}$. Under squared Euclidean cost, BSP-OT uses data-dependent Gaussian slicing in low dimensions and random directions in high dimensions. As illustrated in Figure 1(b), the source points indexed by $i \in \mathcal{I}_v^X$ and target points indexed by $j \in \mathcal{I}_v^Y$ are projected onto $\mathbf{w}$ and ranked independently according to their projected values. The two rankings are then split at the same rank. The lower-ranked source and target subsets form one child node v_L, and the upper-ranked subsets form the other child node v_R. Thus the index sets at v are partitioned into those of its children on each side: $\mathcal{I}_v^X = \mathcal{I}_{v_L}^X \sqcup \mathcal{I}_{v_R}^X$ and $\mathcal{I}_v^Y = \mathcal{I}_{v_L}^Y \sqcup \mathcal{I}_{v_R}^Y$. Splitting both rankings at the same rank preserves equal cardinalities in each child. The same operation is recursively applied to the resulting paired regions, as illustrated in Figure 1(c,d), thereby growing the binary tree. BSP-OT continues the recursion until every leaf contains exactly one source point and one target point. The resulting singleton pairs collectively define the r-th tree matching $\mathcal{M}^{(r)} \in \mathcal{P}([N], [N])$. Repeating the randomized construction independently k times yields $\mathcal{M}^{(1)}, \ldots, \mathcal{M}^{(k)}$.

This construction has two limitations that motivate our design. **(L1)** First, the random directions used in high dimensions need not align with informative variation in the point clouds and may therefore produce less useful paired regions. **(L2)** Second, BSP-OT continues partitioning until singleton pairs are obtained. Once a paired region is sufficiently small, however, its assignment can instead be optimized directly by solving a local LAP. Further partitioning incurs additional projections and splits and, by restricting matches to separate child regions, can exclude lower-cost assignments that remain feasible within the parent region.

Sequential merging. After constructing the k tree matchings, BSP-OT combines them sequentially. It initializes the current matching $\mathcal{M} \leftarrow \mathcal{M}^{(1)}$ and, for $r = 2, \ldots, k$, updates it by $\mathcal{M} \leftarrow \mathrm{BSPMerge}(\mathcal{M}, \mathcal{M}^{(r)})$. Here $\mathrm{BSPMerge}$ denotes the two-matching operation defined in Appendix B. For each merge, BSP-OT considers the union of the two matchings, which decomposes into connected components. Within each component, it first keeps the lower-cost of the two induced matchings and then applies cost-decreasing local swaps when available. Only the resulting perfect matching is retained for the next iteration. The final matching is denoted by $\mathcal{M}_{\mathrm{BSP}}$.

This creates a third limitation that motivates our design. **(L3)** Useful edges from earlier trees may be discarded before later matchings are incorporated, so edges discovered by different trees need not be considered jointly.

Motivation for SER-OT. The projection-based partitioning of BSP-OT provides a computationally inexpensive mechanism for discovering candidate correspondences, and related projection-based methods such as sliced optimal transport have shown that this strategy can work well in practice. We therefore retain this basic framework and focus on addressing **(L1)–(L3)**. This motivates SER-OT, which modifies the direction selection, stopping rule, and recombination strategy of BSP-OT.

4 SER-OT: Sparse Edge Recombination for Optimal Transport

SER-OT builds on BSP-OT's paired rank partitioning to address the three limitations **(L1)–(L3)** identified in Section 3. Algorithm 2 summarizes the method.

4.1 Projection Directions

To address **(L1)**, SER-OT replaces the random directions used by BSP-OT in high dimensions with inexpensive data-dependent directions. At each node v to be split, we sample two distinct target indices $a, a' \in \mathcal{I}_v^Y$ uniformly as an ordered pair and use $\mathbf{w} := (\mathbf{y}_a - \mathbf{y}_{a'})/\|\mathbf{y}_a - \mathbf{y}_{a'}\|$ as the shared projection direction. If the sampled difference is zero or numerically negligible, we use an independent random direction instead. The source and target subsets are then partitioned using the same rank-based split described in Section 3. Constructing $\mathbf{w}$ requires only $O(d)$ time. The conditional directional structure of this sampling rule is analyzed in Section 5.1.

4.2 Early Stopping and Exact Leaf Matching

Let $b \geq 1$ be an integer stopping size. To address **(L2)**, SER-OT stops partitioning once a paired region contains at most b points per side, rather than continuing to singleton pairs. For the r-th tree, let $\mathcal{L}^{(r)} \subseteq \mathcal{V}^{(r)}$ be its set of leaf nodes. We use $\ell \in \mathcal{L}^{(r)}$ for a leaf node itself, not its ordinal number. Each leaf ℓ satisfies $|\mathcal{I}_\ell^X| = |\mathcal{I}_\ell^Y| \leq b$. Within each leaf ℓ, SER-OT solves the exact local LAP

$$\mathcal{M}_\ell \in \operatorname*{arg\,min}_{\mathcal{M} \in \mathcal{P}(\mathcal{I}_\ell^X, \mathcal{I}_\ell^Y)} \sum_{(i,j) \in \mathcal{M}} c_{ij}.$$

Only the resulting matching edges are retained. Write $\mathcal{M}^{(r)} := \bigcup_{\ell \in \mathcal{L}^{(r)}} \mathcal{M}_\ell$ for their union, the r-th tree matching. Because the paired leaves partition both the source and target indices, $\mathcal{M}^{(r)} \in \mathcal{P}([N], [N])$. Thus, each tree contributes exactly N retained edges before duplicate edges across trees are removed.

4.3 Exact Recombination over Retained Edges

To address **(L3)**, SER-OT pools matching edges from all k trees instead of merging their matchings sequentially, allowing the final assignment to combine edges from different trees. We form the

candidate edge set $\mathcal{E} := \bigcup_{r=1}^{k} \mathcal{M}^{(r)}$. Since each tree contributes exactly N edges, $|\mathcal{E}| \leq kN$. Moreover, $\mathcal{E}$ contains at least one perfect matching. SER-OT finally solves the exact LAP restricted to this sparse candidate set:

$$\mathcal{M}_{\mathrm{SER}} \in \underset{\substack{\mathcal{M}\in\mathcal{P}([N],[N]) \\ \mathcal{M}\subseteq\mathcal{E}}}{\arg\min} C(\mathcal{M}).$$

5 Properties and Complexity of SER-OT

Section 4 introduced three modifications to BSP-OT. We examine them in the same order, explaining the motivation behind each design choice and the resulting properties of SER-OT. We then analyze the computational cost of the complete procedure. Proofs are provided in Appendix C.

5.1 Angular Structure of Sampled Directions

We describe the geometric properties of the shared projection direction $\mathbf{w}$ used in Section 4.1. Fix a node v, let $m := |\mathcal{I}_v^Y| \geq 2$, and assume that distinct target points in the node have nonzero pairwise differences.

Proposition 1 (Moment of the sampled direction)**.** *Sample distinct indices $a, a' \in \mathcal{I}_v^Y$ uniformly as an ordered pair and set $\mathbf{w} := (\mathbf{y}_a - \mathbf{y}_{a'})/\|\mathbf{y}_a - \mathbf{y}_{a'}\|$. Conditional on the node's target points,*

$$\mathbb{E}_{a,a'}[\mathbf{w}] = \mathbf{0},$$
$$\mathbf{A}_v := \mathbb{E}_{a,a'}[\mathbf{w}\mathbf{w}^\top] = \frac{2}{m(m-1)} \sum_{\substack{a<a' \\ a,a'\in\mathcal{I}_v^Y}} \frac{(\mathbf{y}_a - \mathbf{y}_{a'})(\mathbf{y}_a - \mathbf{y}_{a'})^\top}{\|\mathbf{y}_a - \mathbf{y}_{a'}\|^2}. \tag{2}$$

Moreover, $\mathbf{A}_v \succeq 0$ and $\mathrm{tr}(\mathbf{A}_v) = 1$; for every unit vector $\mathbf{u} \in \mathbb{R}^d$, $\mathbf{u}^\top \mathbf{A}_v \mathbf{u} = \mathbb{E}_{a,a'}[(\mathbf{u}^\top \mathbf{w})^2]$.

The conditional second moment $\mathbf{A}_v$ equals the empirical symmetrized spatial sign covariance matrix of the target subset (Visuri et al., 2000; Taskinen et al., 2012). Since each pairwise difference is normalized, $\mathbf{A}_v$ characterizes the angular structure of the target subset. The sampled direction follows the empirical angular distribution of local target-point pairs, allowing the split direction to reflect their geometry rather than being independent of the data.

5.2 Exact Local Optimization and Global Recombination

Next, we consider exact optimization within each paired leaf. Stopping before singleton pairs may leave several possible assignments within each paired leaf. Solving its LAP exactly selects the least-cost one among them. The following comparison makes this benefit precise while keeping the paired leaves fixed.

Proposition 2 (Leaf optimization under a fixed partition)**.** *Fix a tree r with leaf set $\mathcal{L}^{(r)}$ and tree matching $\mathcal{M}^{(r)} = \bigcup_{\ell\in\mathcal{L}^{(r)}} \mathcal{M}_\ell$. Let $\mathcal{M}_{\mathrm{split}}^{(r)}$ be any perfect matching obtained by further partitioning these paired leaves and matching only within the resulting subleaves. Then $C(\mathcal{M}^{(r)}) \leq C(\mathcal{M}_{\mathrm{split}}^{(r)})$.*

Further splitting removes feasible source–target pairs from each original leaf. Its matching remains feasible for the original leaf LAPs, so it cannot cost less than their joint optimum.

We then consider the exact global recombination of the edges retained across trees. Since each $\mathcal{M}^{(r)}$ is a perfect matching, the candidate graph has maximum degree at most k and contains a feasible perfect matching.

Proposition 3 (Exact recombination)**.** (a) *For any $\mathcal{M} \in \mathcal{P}([N],[N])$ with $\mathcal{M} \subseteq \mathcal{E}$, $C^\star \leq C(\mathcal{M}_{\mathrm{SER}}) \leq C(\mathcal{M})$. In particular, $C(\mathcal{M}_{\mathrm{SER}}) \leq \min_{r\in[k]} C(\mathcal{M}^{(r)})$.*

(b) *For $s \in [k]$, let $\mathcal{E}_s := \bigcup_{r=1}^{s} \mathcal{M}^{(r)}$ and choose $\mathcal{M}_s$ as an exact minimum-cost perfect matching contained in $\mathcal{E}_s$, with $\mathcal{M}_k = \mathcal{M}_{\mathrm{SER}}$. Then $C(\mathcal{M}_{s+1}) \leq C(\mathcal{M}_s)$ for $1 \leq s < k$.*

The first inequality says that the final LAP can combine retained edges from different trees without doing worse than any matching that uses only those edges. This includes an edge-preserving merge,

but not necessarily BSP-OT's full merge: its assignment-swapping stage may introduce an edge outside $\mathcal{E}$ (see Appendix B for details). The second inequality follows because each additional tree only enlarges the feasible edge set.

5.3 Computational Complexity and Memory

For fixed k, b, and d, building the trees and candidate graph takes $O(N \log^2 N)$ time and $O(Nd+kN)$ memory. The remaining cost comes from the final sparse LAP. In theory, recent exact minimum-cost-flow results yield an almost-linear-time implementation with complexity $O(N^{1+\varepsilon})$ for any fixed $\varepsilon > 0$ under fixed-precision costs (van den Brand et al., 2023). However, such algorithms rely on sophisticated data structures and remain challenging to implement in practice. A classical scaling algorithm (Gabow and Tarjan, 1989) provides a simpler worst-case guarantee of $O(N^{1.5} \log N)$.

Remark (Practical sparse-LAP solver). The bound above is a worst-case theoretical guarantee for SER-OT. LAPMOD (Volgenant, 1996) has outperformed a cost-scaling implementation on some assignment benchmarks (Dell'Amico and Toth, 2000), although this does not establish the same ranking on our sparse graphs. We use LAPMOD for its empirical performance on those graphs, despite its less favorable worst-case bound.

6 Experiments

We evaluate SER-OT through four complementary experiments. First, a large-scale synthetic benchmark tests accuracy and runtime as the problem size and dimension grow. Next, Gaussian-to-image-latent matching tests performance on high-dimensional features derived from real images. We then use varied input families to examine geometric sensitivity and projection-direction choice. Finally, we test whether SER-OT's gains can be explained by giving BSP-OT more trees or by globally recombining the candidate matches found by its trees.

Experimental setup. We use squared Euclidean cost and seeds 0, 1, and 2. For an output matching $\mathcal{M}$ and reference $\mathcal{M}_{\mathrm{ref}}$, we report the relative cost gap $\mathrm{Gap} := 100[C(\mathcal{M})/C(\mathcal{M}_{\mathrm{ref}}) - 1]$ (%). When the reference is a certified optimum, Gap measures the relative difference from the optimal cost. Tables report mean Gap and runtime over the three runs; sample standard deviations appear in Appendix F. Each method uses 16 CPU threads on a dual-socket server (two 56-core Intel Xeon Platinum 8480+ processors, 512 GiB RAM, Ubuntu 24.04.4). Runs have a one-hour, 512 GiB limit; OOT and OOM indicate these limits.

6.1 Synthetic assignment benchmark

Design. We first evaluate SER-OT at large scale and examine whether it maintains high accuracy and computational efficiency as both the problem size N and dimension d increase. This requires reference optimal solutions for large, high-dimensional OT problems. However, at the scale of one million points, constructing all N^2 pairwise costs and solving the full problem exactly is impractical. We therefore use synthetic OT instances for which the optimal assignment is known by construction. Specifically, we generate the problems using the convex-gradient construction of Korotin et al. (2021) and evaluate 36 settings with varying N and d. BSP-OT and SER-OT each use $k = 64$, and SER-OT uses leaf size $b = 64$. **Results.** SER-OT completes all 36 settings with a maximum mean gap of 0.005730% and is faster than BSP-OT in 35 of them. Thus, SER-OT maintains low relative cost gaps as both N and d increase, while scaling beyond one million points.

6.2 Gaussian-to-image-latent assignment benchmark

Design. We next test whether SER-OT remains effective on high-dimensional features derived from real images, beyond the synthetic benchmark. We match samples from $\mathcal{N}(0, \mathbf{I}_d/d)$ to latent features extracted from ImageNet images (Deng et al., 2009) using the Stable Diffusion VAE (Rombach et al., 2022) and flattened to $d = 4096$. This benchmark is motivated by the minibatch OT coupling used in OT-CFM (Tong et al., 2024). We evaluate nested prefixes from $N = 2048$ to 131,072. BSP-OT and SER-OT each use $k = 64$, and SER-OT uses leaf size $b = 64$. We obtain a reference assignment from a dense full-graph LAP and evaluate it and the solver outputs using squared Euclidean costs on the stored coordinates. **Results.** SER-OT achieves lower mean gaps and shorter runtimes than

Table 1: Synthetic assignment benchmark. Mean Gap (%) and runtime (s).

N	d	HiRef		BSP-OT		HALO		SER-OT	
		Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
2,048	16	25.966	17.20	0.000	0.20	0.000	53.04	0.000	0.04
	64	21.223	17.64	0.000	0.21	0.000	20.66	0.000	0.05
	128	26.520	17.74	0.000	0.31	0.000	31.88	0.000	0.06
	256	19.142	17.93	0.000	0.53	0.000	52.01	0.000	0.09
	512	30.951	18.21	0.000	0.96	0.000	97.63	0.000	0.15
	1,024	34.949	18.96	0.000	1.86	0.000	181.15	0.000	0.27
8,192	16	91.484	61.56	0.000	0.61	0.000	382.99	0.000	0.19
	64	102.773	64.30	0.000	0.57	0.021	276.23	0.000	0.22
	128	124.483	65.09	0.000	1.04	3.974	491.03	0.000	0.28
	256	139.212	63.41	0.000	1.95	1.056	1204.72	0.000	0.42
	512	122.313	64.50	0.000	3.90	OOT		0.000	0.68
	1,024	125.583	67.47	0.000	8.31	OOM		0.000	1.21
32,768	16	52.202	246.46	4.623	0.80	OOT		0.000	0.82
	64	98.740	269.37	0.000	2.21	OOM		0.000	0.95
	128	123.855	254.54	0.000	4.24	OOM		0.000	1.32
	256	126.287	252.90	0.000	8.79	OOM		0.000	1.89
	512	154.107	268.29	0.000	17.20	OOM		0.000	3.13
	1,024	160.362	275.29	0.000	32.34	OOM		0.000	5.01
131,072	16	91.846	982.97	0.000	8.64	OOM		0.005	3.87
	64	127.363	1025.08	0.060	10.27	OOM		0.000	4.25
	128	148.300	1027.42	0.000	21.07	OOM		0.000	5.54
	256	154.523	1026.46	0.000	36.64	OOM		0.000	8.08
	512	178.855	1041.63	0.000	58.58	OOM		0.000	12.61
	1,024	191.933	1114.56	0.000	113.92	OOM		0.000	21.10
524,288	16	OOT		0.000	33.85	OOM		0.006	16.27
	64	OOT		0.448	40.68	OOM		0.000	19.19
	128	OOT		0.009	74.62	OOM		0.000	25.62
	256	OOT		0.000	131.41	OOM		0.000	36.22
	512	OOT		0.000	237.77	OOM		0.000	51.89
	1,024	OOT		0.000	468.89	OOM		0.000	88.55
1,048,576	16	OOT		0.001	71.87	OOM		0.003	33.51
	64	OOT		0.635	84.53	OOM		0.000	38.93
	128	OOT		0.023	152.57	OOM		0.000	52.72
	256	OOT		0.000	268.51	OOM		0.000	76.51
	512	OOT		0.000	486.88	OOM		0.000	114.77
	1,024	OOT		0.000	971.31	OOM		0.000	174.46

Table 2: Gaussian-to-image-latent assignment benchmark. Mean Gap (%) and runtime (s).

N	HiRef		BSP-OT		HALO		SER-OT	
	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
2,048	0.033181	21.54	0.012101	8.62	0.003134	1295.60	0.000146	1.06
8,192	0.037032	43.49	0.015377	30.63	OOM		0.001013	5.57
32,768	0.040248	173.94	0.018590	128.80	OOM		0.002734	31.12
131,072	0.043256	1421.56	0.021456	488.69	OOM		0.004822	129.97

BSP-OT at every size. HiRef yields substantially larger gaps, while HALO exceeds the memory limit beyond the smallest setting. Thus, SER-OT combines low relative cost gaps with practical runtime on high-dimensional features derived from real images. At $N = 131{,}072$, both its mean gap and runtime are about one quarter of BSP-OT's.

Table 3: Geometrically Diverse Assignment Benchmark. Mean Gap (%) and runtime (s).

Family (d)	N	BSP-OT		SER-OT (U)		SER-OT	
		Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
Nonlinear (16)	2,048	0.871	0.076	0.000	0.050	0.000	0.041
Anisotropic (64)	2,048	0.707	0.152	0.000	0.049	0.000	0.039
Multimodal (16)	2,048	6.764	0.073	0.000	0.105	0.000	0.080
Manifold (16)	2,048	11.524	0.071	0.064	0.410	0.015	0.361
Geometry mismatch (16)	2,048	2.526	0.060	0.101	0.387	0.017	0.425
Torus–trefoil (3)	2,048	7.367	0.037	0.421	0.493	0.208	0.457
Geometry mismatch (16)	8,192	3.891	0.197	0.298	7.090	0.056	7.757
Torus–trefoil (3)	8,192	7.963	0.099	1.039	7.048	0.608	7.118

6.3 Geometrically Diverse Assignment Benchmark

Design. To evaluate SER-OT across different distributions and geometric structures, we consider six transport problems: a three-component Gaussian mixture to a nonlinear distribution (Nonlinear), an anisotropic Gaussian to its affine-transformed distribution (Anisotropic), an eight-component Gaussian mixture to a nonlinearly transformed multimodal distribution (Multimodal), a noisy harmonic manifold to its affine-transformed manifold (Manifold), a standard Gaussian to a Gaussian mixture (Geometry mismatch), and a torus to a trefoil tube (Torus–trefoil). Reference solutions are obtained with an exact LAP solver. We compare BSP-OT, SER-OT, and SER-OT (U). All methods use $k = 64$, and both SER-OT variants use $b = 64$. SER-OT (U) replaces the proposed data-dependent projection direction with a uniformly random direction. **Results.** SER-OT achieves a lower mean Gap than BSP-OT across all input families and also a lower Gap than SER-OT (U), indicating that target-difference-based projection directions are more effective than uniformly random directions on these benchmarks.

6.4 BSP-OT budget and recombination controls

Design. We use the Geometry mismatch and Torus–trefoil benchmarks from Section 6.3. We compare SER-OT with 64 trees against BSP-OT with 64, 1024, 2048, and 4096 trees. We also compare against BSP+LAP, which unions the candidate edges obtained from 64 BSP-OT trees and then solves an exact global LAP over the resulting graph. **Results.** Even when the number of trees is increased to 4096, BSP-OT exhibits larger Gaps than 64-tree SER-OT on both benchmarks. BSP+LAP also yields larger Gaps than SER-OT despite solving an exact global LAP. These results suggest that the improvement of SER-OT cannot be explained solely by using more trees or by global recombination, but instead stems from generating more useful candidate edges.

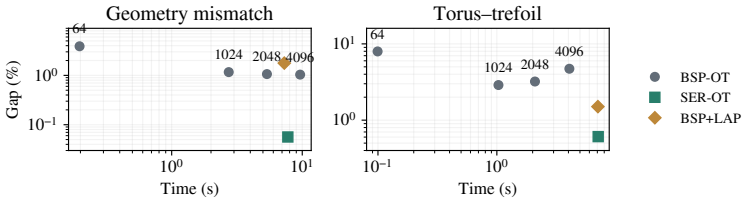


Figure 2: BSP-OT budget and recombination controls on the $N = 8192$ Geometry mismatch and Torus–trefoil benchmarks. Numeric labels indicate the number of BSP-OT trees.

7 Conclusion

We consider large-scale discrete OT problems where the source and target distributions have the same number of points with uniform weights, and propose SER-OT for this setting. Extensive experiments show that SER-OT achieves high solution accuracy with practical runtime. Since this is a standard OT formulation, SER-OT may apply to a broad range of problems. However, the current formulation is limited to this setting, and extending it to unequal cardinalities or non-uniform weights is an important direction for future work. In addition, although exact almost-linear-time algorithms can in principle be applied, efficient practical implementations remain challenging and are currently limited to super-linear worst-case guarantees. Developing practical sparse LAP solvers with strong theoretical guarantees and high empirical efficiency remains an important research direction.

AI USE STATEMENT

In this work, we used generative AI tools to assist with language editing, translation, improving the clarity and organization of the manuscript, and software development. We also used generative AI tools to help identify possible improvements to the presentation of the method and experiments. All AI-assisted text was reviewed and revised by the authors. Mathematical claims, theoretical arguments, citations, and experimental results were independently checked by the authors, and AI-assisted code was inspected and tested before use. The authors take full responsibility for the final content of this work, including all text, claims, results, and artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

Our work is methodological and focuses on improving the scalability of discrete optimal transport. We do not use human subjects, private data, or personally identifiable information, and we are not aware of any immediate ethical concerns arising directly from the proposed method. As with general-purpose optimization methods, potential ethical concerns may depend on the downstream applications in which the method is used.

REPRODUCIBILITY STATEMENT

The SER-OT algorithm and its main components are described in Sections 3–4, and the theoretical results are stated with their assumptions and proved in Appendix C. Experimental settings, evaluation metrics, implementation details, and hardware configurations are reported in Section 6 and the appendix. We use synthetic and publicly available benchmark datasets and report results over multiple runs where applicable. We will release the implementation necessary to reproduce the experiments.

REFERENCES

Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. Sliced and Radon Wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 51(1):22–45, 2015.

Yann Brenier. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on Pure and Applied Mathematics*, 44(4):375–417, 1991.

Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9):1853–1865, 2017.

Sanjoy Dasgupta and Yoav Freund. Random projection trees and low dimensional manifolds. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pp. 537–546, 2008.

Mauro Dell'Amico and Paolo Toth. Algorithms and codes for dense assignment problems: The state of the art. *Discrete Applied Mathematics*, 100(1–2):17–48, 2000.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, 2009.

Harold N. Gabow and Robert E. Tarjan. Faster scaling algorithms for network problems. *SIAM Journal on Computing*, 18(5):1013–1036, 1989.

Baptiste Genest, Nicolas Bonneel, Vincent Nivoliers, and David Coeurjolly. BSP-OT: Sparse transport plans between discrete measures in loglinear time. *ACM Transactions on Graphics*, 44(6):246:1–246:15, 2025.

Samuel Gerber and Mauro Maggioni. Multiscale strategies for computing optimal transport. *Journal of Machine Learning Research*, 18(72):1–32, 2017.

Peter Halmos, Julian Gold, Xinhao Liu, and Benjamin Raphael. Hierarchical refinement: Optimal transport to infinity and beyond. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 21629–21661, 2025.

Alexander Korotin, Lingxiao Li, Aude Genevay, Justin Solomon, Alexander Filippov, and Evgeny Burnaev. Do neural optimal transport solvers work? A continuous Wasserstein-2 benchmark. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 14593–14605, 2021.

Xinran Liu, Rocío Díaz Martín, Yikun Bai, Ashkan Shahbazi, Matthew Thorpe, Akram Aldroubi, and Soheil Kolouri. Expected sliced transport plans. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

Guillaume Mahey, Laetitia Chapel, Gilles Gasso, Clément Bonet, and Nicolas Courty. Fast optimal transport through sliced generalized Wasserstein geodesics. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Quentin Mérigot. A multiscale approach to optimal transport. *Computer Graphics Forum*, 30(5):1583–1592, 2011.

Victor M. Panaretos and Yoav Zemel. Statistical aspects of Wasserstein distances. *Annual Review of Statistics and Its Application*, 6:405–431, 2019.

Gabriel Peyré and Marco Cuturi. Computational optimal transport: With applications to data science. *Foundations and Trends in Machine Learning*, 11(5–6):355–607, 2019.

Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. Wasserstein barycenter and its application to texture mixing. In *Scale Space and Variational Methods in Computer Vision (SSVM)*, pp. 435–446, 2011.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, 2022.

Bernhard Schmitzer. A sparse multiscale algorithm for dense optimal transport. *Journal of Mathematical Imaging and Vision*, 56(2):238–259, 2016.

Sara Taskinen, Inge Koch, and Hannu Oja. Robustifying principal component analysis with spatial sign vectors. *Statistics & Probability Letters*, 82(4):765–774, 2012.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024.

Jan van den Brand, Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, Sushant Sachdeva, and Aaron Sidford. A deterministic almost-linear time algorithm for minimum-cost flow. In *Proceedings of the 64th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 503–514, 2023.

Ton Volgenant. Linear and semi-assignment problems: A core oriented approach. *Computers & Operations Research*, 23(10):917–932, 1996.

Samuli Visuri, Visa Koivunen, and Hannu Oja. Sign and rank covariance matrices. *Journal of Statistical Planning and Inference*, 91(2):557–575, 2000.

Wenzhou Xia, Ya-Nan Zhu, Jingwei Liang, and Xiaoqun Zhang. A memory-efficient hierarchical algorithm for large-scale optimal transport problems. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026.

APPENDIX OVERVIEW

The appendix follows the development in the paper. Appendix A collects the notation; Appendix B details the BSP-OT merge used in Section 3; Appendix C proves the properties in Section 5; Appendix D derives the complexity bounds; Appendix E documents the benchmark constructions and reference assignments; and Appendix F provides the complete experimental results and controls.

A NOTATION

Table A.1 summarizes the notation used in the paper and appendix. Symbols local to a proof or experiment are defined where they first appear.

Table A.1: Notation used in the paper and appendix.

Symbol	Meaning
Point sets and assignments	
N	Number of points in each point cloud.
d	Ambient dimension of the point clouds.
$[N]$	Index set $\{1, \dots, N\}$.
$\mathbf{X}$	Source point cloud in $\mathbb{R}^{d\times N}$.
$\mathbf{Y}$	Target point cloud in $\mathbb{R}^{d\times N}$.
$\mathbf{x}_i$	Source point indexed by $i \in [N]$.
$\mathbf{y}_j$	Target point indexed by $j \in [N]$.
$\|\cdot\|$	Euclidean (ℓ_2) norm of a vector.
μ	Uniform empirical measure on the source points.
ν	Uniform empirical measure on the target points.
$\delta_{\mathbf{x}}$	Unit point mass at $\mathbf{x}$.
c	Finite-valued source–target pair-cost function.
c_{ij}	Cost $c(\mathbf{x}_i, \mathbf{y}_j)$ of matching source i to target j.
$\mathbf{P}$	Coupling matrix in the discrete OT formulation.
$\mathbf{1}_N$	All-ones vector in $\mathbb{R}^N$.
$\mathcal{I}$	Source index subset of $[N]$, paired with an equally sized $\mathcal{J}$.
$\mathcal{J}$	Target index subset of $[N]$, paired with an equally sized $\mathcal{I}$.
$\mathcal{S}$	Generic index subset of $[N]$.
$\mathbf{X}_{\mathcal{S}}$	Source submatrix indexed by $\mathcal{S}$, with columns in increasing index order.
$\mathbf{Y}_{\mathcal{S}}$	Target submatrix indexed by $\mathcal{S}$, with columns in increasing index order.
$\mathcal{P}(\mathcal{I}, \mathcal{J})$	Perfect matchings between $\mathcal{I}$ and $\mathcal{J}$.
$\sqcup$	Union of two disjoint sets.
$\mathcal{M}$	A matching on the index sets specified by its context.
$C(\mathcal{M})$	Mean pair cost of a full matching $\mathcal{M}$.
$C^\star$	Minimum mean pair cost over all full perfect matchings.
Paired partition trees	
k	Number of independently constructed trees.
b	Maximum number of points per side in a paired leaf.
r	Tree index in $[k]$.
$\mathcal{T}^{(r)}$	The r-th paired partition tree.
$\mathcal{V}^{(r)}$	Set of all nodes in $\mathcal{T}^{(r)}$.
$\rho^{(r)}$	Root node of $\mathcal{T}^{(r)}$.
v	An arbitrary node in $\mathcal{V}^{(r)}$.
v_L	Left paired child node produced by splitting v.
v_R	Right paired child node produced by splitting v.
$\mathcal{I}_v^X$	Source indices belonging to node v.
$\mathcal{I}_v^Y$	Target indices belonging to node v.
$\mathcal{L}^{(r)}$	Set of leaf nodes in $\mathcal{T}^{(r)}$.
ℓ	A leaf node in $\mathcal{L}^{(r)}$, not its ordinal number.

Table A.1 (continued)

Symbol	Meaning
$\mathcal{I}_\ell^X$	Source indices belonging to leaf ℓ.
$\mathcal{I}_\ell^Y$	Target indices belonging to leaf ℓ.
a	First target index in a sampled ordered pair at node v.
a'	Second target index in the pair, distinct from a.
$\mathbf{w}$	Normalized target-point difference sampled as a projection direction.
m	In Section 5.1, the node size $\lvert\mathcal{I}_v^Y\rvert$.
$\mathbf{A}_v$	Conditional second moment $\mathbb{E}[\mathbf{w}\mathbf{w}^\top]$ at node v.
$\mathcal{M}_\ell$	Exact local-LAP matching within leaf ℓ.
$\mathcal{M}^{(r)}$	Full tree matching formed from the leaf matchings of tree r.
Merging and recombination	
BSPMerge	BSP-OT's bijective merge of two perfect matchings.
H_h	Component h of the two-matching union graph in Appendix B.
$\mathcal{M}_h$	Edges of the first input matching in component H_h.
$\mathcal{M}'_h$	Edges of the second input matching in component H_h.
$\mathcal{Q}_h$	Lower-cost matching edges selected in component H_h.
$\mathcal{R}_h$	Unselected matching edges in component H_h.
$\mathcal{M}^{\mathrm{mono}}$	Union of the componentwise selected edge sets $\mathcal{Q}_h$.
$\mathcal{M}_{\mathrm{BSP}}$	Final matching produced by BSP-OT.
$\mathcal{M}_{\mathrm{SER}}$	Final matching produced by SER-OT.
$\mathcal{M}_{\mathrm{split}}^{(r)}$	Comparator obtained by further splitting fixed paired leaves.
$\mathcal{E}$	Union $\bigcup_{r=1}^k \mathcal{M}^{(r)}$ of retained candidate edges.
$\mathcal{E}_s$	Candidate edges contributed by the first s trees.
$\mathcal{M}_s$	Exact minimum-cost perfect matching contained in $\mathcal{E}_s$.
Complexity and experiments	
L	Maximum number of split levels, $\max\{0, \lceil \log_2(N/b) \rceil\}$.
e	Number $\lvert\mathcal{E}\rvert$ of distinct candidate edges in Appendix D.
T_c	Time needed for one source–target pair-cost query.
D	Maximum source degree of the candidate graph in Appendix D.
P	Number of concurrent tree builders in Appendix D.
ϕ	Convex potential used to generate synthetic targets.
$\nabla\phi$	Gradient map applied to synthetic source points.
$\mathcal{M}_{\mathrm{ref}}$	Reference matching for an experimental input.
Gap	Relative cost gap with respect to the reference matching, $100[C(\mathcal{M})/C(\mathcal{M}_{\mathrm{ref}}) - 1]$ (%).
$\mathrm{Cov}(\mathcal{E}; \mathcal{M}_{\mathrm{ref}})$	Fraction of reference edges present in the candidate set.
OOT	Time limit exceeded.
OOM	Memory limit exceeded.

B BSP-OT MERGING

BSP-OT combines two perfect matchings at a time using BSPMerge (Genest et al., 2025). Let $\mathcal{M}, \mathcal{M}' \in \mathcal{P}([N], [N])$. Regard the source and target copies of $[N]$ as distinct vertex sets and superimpose the edges of the two matchings. Every vertex has one edge from each matching. A shared edge forms a two-vertex component; all other components are alternating even cycles. Writing $H_1, \ldots, H_q$ for these components, let $\mathcal{M}_h$ and $\mathcal{M}'_h$ be the two matchings restricted to H_h.

Componentwise selection. For each h, BSP-OT compares the total costs of $\mathcal{M}_h$ and $\mathcal{M}'_h$ and selects the cheaper one, retaining $\mathcal{M}_h$ in a tie. Denote the selected edges by $\mathcal{Q}_h$ and the other edges by $\mathcal{R}_h$. Because components have disjoint vertices, $\mathcal{M}^{\mathrm{mono}} := \bigcup_{h=1}^q \mathcal{Q}_h$ is a perfect matching. Its total cost is the sum of the smaller componentwise costs, so $C(\mathcal{M}^{\mathrm{mono}}) \le \min\{C(\mathcal{M}), C(\mathcal{M}')\}$. The choice can differ between components: the merged matching may use edges from both inputs before any swaps are considered.

Cost-decreasing assignment swaps. The second stage considers edges of the unselected $\mathcal{R}_h$ as proposals. For a proposed edge (i, j'), let (i, j) be the current edge at source i and (i', j') the unique current edge entering target j'. Replacing these two current edges by (i, j') and (i', j) preserves a perfect matching. BSP-OT accepts the replacement only when

$$c_{ij'} + c_{i'j} < c_{ij} + c_{i'j'}.$$

The procedure visits the source indices of each component and applies such cost-decreasing swaps when available. The complementary edge (i', j) is evaluated for the swap; it need not have appeared in either input matching. Each accepted swap preserves bijectivity and strictly decreases cost, so the final $\mathrm{BSPMerge}(\mathcal{M}, \mathcal{M}')$ is a perfect matching with cost no greater than either input.

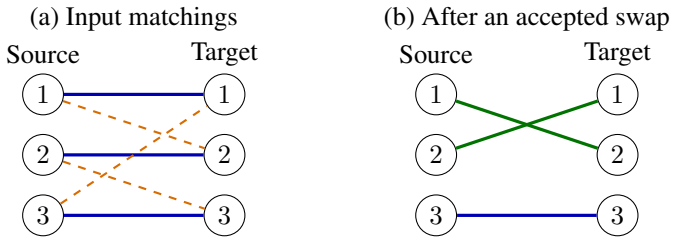


Figure 3: A BSP-OT merge on three points. In (a), solid blue edges form $\mathcal{M}$ and dashed orange edges form $\mathcal{M}'$. Componentwise selection keeps $\mathcal{M}$; the green edges in (b) replace $(1, 1)$ and $(2, 2)$, reducing their cost from 8 to 2.

A three-point example. Take the cost matrix and two matchings

$$(c_{ij}) := \begin{pmatrix} 4 & 1 & 10 \\ 1 & 4 & 9 \\ 9 & 10 & 4 \end{pmatrix}, \qquad \mathcal{M} := \{(1,1),(2,2),(3,3)\}, \qquad \mathcal{M}' := \{(1,2),(2,3),(3,1)\}.$$

Their union is one alternating cycle. The unnormalized cost sums are 12 and 19, respectively, so componentwise selection retains $\mathcal{M}$. The unselected edge $(1, 2)$ can then be proposed. The current edges $(1, 1)$ and $(2, 2)$ cost $4 + 4 = 8$, whereas $(1, 2)$ and its complementary edge $(2, 1)$ cost $1 + 1 = 2$. The accepted swap yields $\{(1, 2), (2, 1), (3, 3)\}$, whose unnormalized cost is 6. Figure 3 shows the input edges and the resulting matching. In particular, $(2, 1)$ is created by the swap rather than taken from either input matching.

BSP-OT applies this two-matching operation sequentially: starting with $\mathcal{M}^{(1)}$, it merges the current result with $\mathcal{M}^{(2)}, \ldots, \mathcal{M}^{(k)}$ in order. At each step, the result is again a perfect matching and is the only matching passed to the next step.

C Proofs of SER-OT properties

We prove Propositions 1–3 in the order used in Section 5. The moment calculation conditions on the target points at one node; the optimization statements use finite costs and exact assignment on their stated feasible sets.

C.1 Proof of Proposition 1

Proof. Condition on the target points at node v and write $m := |\mathcal{I}_v^Y|$. There are $m(m-1)$ admissible ordered pairs, all equally likely. For each unordered pair $a < a'$, put $\mathbf{s}_{aa'} := (\mathbf{y}_a - \mathbf{y}_{a'})/\|\mathbf{y}_a - \mathbf{y}_{a'}\|$. The two orderings generate $\mathbf{s}_{aa'}$ and $-\mathbf{s}_{aa'}$. Consequently, their contributions to the first moment cancel:

$$\mathbb{E}_{a,a'}[\mathbf{w}] = \frac{1}{m(m-1)} \sum_{\substack{a<a' \\ a,a' \in \mathcal{I}_v^Y}} (\mathbf{s}_{aa'} - \mathbf{s}_{aa'}) = \mathbf{0}.$$

Their outer products are identical. Summing the two copies for every unordered pair gives

$$\mathbf{A}_v = \frac{2}{m(m-1)} \sum_{\substack{a<a' \\ a,a' \in \mathcal{I}_v^Y}} \mathbf{s}_{aa'} \mathbf{s}_{aa'}^\top,$$

which is Equation (2) after substituting the definition of $\mathbf{s}_{aa'}$. For any $\mathbf{u} \in \mathbb{R}^d$, $\mathbf{u}^\top \mathbf{A}_v \mathbf{u} = \mathbb{E}_{a,a'}[(\mathbf{u}^\top \mathbf{w})^2] \geq 0$, proving positive semidefiniteness and the asserted identity for unit $\mathbf{u}$. Finally, every sampled direction has unit norm, so $\mathrm{tr}(\mathbf{A}_v) = \mathbb{E}_{a,a'}[\mathrm{tr}(\mathbf{w}\mathbf{w}^\top)] = \mathbb{E}_{a,a'}[\|\mathbf{w}\|^2] = 1$. □

C.2 Proof of Proposition 2

Proof. Fix the original paired leaves of tree r. Further splitting partitions each leaf ℓ into smaller paired regions; it neither moves indices between original leaves nor matches indices across them. Let $\mathcal{S}_\ell := \mathcal{M}^{(r)}_{\mathrm{split}} \cap (\mathcal{I}^X_\ell \times \mathcal{I}^Y_\ell)$. Every source and every target in leaf ℓ has exactly one partner in $\mathcal{S}_\ell$, since the subleaf matching is perfect. Hence $\mathcal{S}_\ell \in \mathcal{P}(\mathcal{I}^X_\ell, \mathcal{I}^Y_\ell)$, making it feasible for the original leaf LAP. Exact local optimization gives

$$\sum_{(i,j)\in\mathcal{M}_\ell} c_{ij} \leq \sum_{(i,j)\in\mathcal{S}_\ell} c_{ij} \qquad (\ell \in \mathcal{L}^{(r)}).$$

The original leaves are disjoint and cover both point sets, so $\mathcal{M}^{(r)} = \bigcup_{\ell\in\mathcal{L}^{(r)}} \mathcal{M}_\ell$ and $\mathcal{M}^{(r)}_{\mathrm{split}} = \bigcup_{\ell\in\mathcal{L}^{(r)}} \mathcal{S}_\ell$. Summing the leaf inequalities and dividing by N yields $C(\mathcal{M}^{(r)}) \leq C(\mathcal{M}^{(r)}_{\mathrm{split}})$. □

C.3 Proof of Proposition 3

Proof. For an edge set $F \subseteq [N]\times[N]$, define its feasible matchings by $\mathfrak{F}(F) := \{\mathcal{M} \in \mathcal{P}([N],[N]) : \mathcal{M} \subseteq F\}$. Each tree matching belongs to $\mathfrak{F}(\mathcal{E})$, so this set is nonempty. By Section 4.3, $\mathcal{M}_{\mathrm{SER}}$ minimizes C over $\mathfrak{F}(\mathcal{E})$. Thus, for every $\mathcal{M} \in \mathfrak{F}(\mathcal{E})$, $C(\mathcal{M}_{\mathrm{SER}}) \leq C(\mathcal{M})$. Substituting each $\mathcal{M}^{(r)}$ proves the upper bounds in part (a). The feasible set $\mathfrak{F}(\mathcal{E})$ is contained in $\mathcal{P}([N],[N])$; minimizing over the latter can only lower the optimum. Therefore $C^\star \leq C(\mathcal{M}_{\mathrm{SER}})$ as well.

For part (b), $\mathcal{E}_s \subseteq \mathcal{E}_{s+1}$ implies $\mathfrak{F}(\mathcal{E}_s) \subseteq \mathfrak{F}(\mathcal{E}_{s+1})$. In particular, $\mathcal{M}_s$ is feasible at stage $s+1$, whereas $\mathcal{M}_{s+1}$ is an optimizer there. Hence $C(\mathcal{M}_{s+1}) \leq C(\mathcal{M}_s)$. At $s = k$, $\mathcal{E}_k = \mathcal{E}$, so the chosen $\mathcal{M}_k = \mathcal{M}_{\mathrm{SER}}$ is consistent with part (a). □

D Computational complexity and memory

We derive the bounds stated in Section 5.3. We count arithmetic operations and pair-cost queries separately, excluding input I/O. Let $N \geq 2$, $b \geq 1$, $L := \max\{0, \lceil \log_2(N/b) \rceil\}$, and $e := |\mathcal{E}|$. The compact main-text bounds use $1 \leq k \leq N$; for unrestricted k, retain the deduplication term below. Bounds use unit-cost arithmetic and counters large enough to avoid overflow.

D.1 Candidate-graph construction

At split depth h, every node contains at most $\lceil N/2^h \rceil$ points per side. By the definition of L, this quantity is at most b when $h = L$, so every branch stops by depth L. For each $\ell \in \mathcal{L}^{(r)}$, write $m_\ell := |\mathcal{I}^X_\ell|$ for its leaf size. The leaves partition the N source indices and satisfy

$$\sum_{\ell\in\mathcal{L}^{(r)}} m_\ell = N, \qquad m_\ell \leq b.$$

The pointwise inequalities $m_\ell^2 \leq b m_\ell$ and $m_\ell^3 \leq b^2 m_\ell$ therefore give

$$\sum_{\ell\in\mathcal{L}^{(r)}} m_\ell^2 \leq Nb, \qquad \sum_{\ell\in\mathcal{L}^{(r)}} m_\ell^3 \leq Nb^2.$$

A leaf of size m_ℓ evaluates m_ℓ^2 source–target costs and solves an LAP in $O(m_\ell^3)$ work using a cubic algorithm. Summing the leaf-size bounds over k trees gives at most kNb pair-cost queries and $O(kNb^2)$ leaf-LAP work. These counts include repeated evaluations across trees. If one pair-cost query takes T_c time, the queries contribute $O(kNbT_c)$ additional work; direct squared Euclidean evaluation has $T_c = O(d)$.

At any fixed depth, the nodes partition each point set. A tree therefore projects at most N source and N target points at that depth. Across at most L split levels and k trees, projection work is $O(kNdL)$.

The $O(d)$ work needed to form one direction at each internal node is absorbed by this bound when $L \geq 1$; when $L = 0$, no split or direction is required.

After projection, linear-time rank selection would contribute $O(kNL)$ work. The implementation instead sorts within each node. A node containing m points requires $O(m \log m) \leq O(m \log N)$ comparisons. Summing node sizes at one depth gives N, and summing over depths and trees gives $O(kNL \log N)$ comparisons.

Every exact tree matching contributes one edge per source, hence exactly N candidate records. Across all trees there are kN raw records. After deduplication,

$$N \leq e = |\mathcal{E}| \leq \min(kN, N^2),$$

and each source and target has degree at most $\min(k, N)$. The selected-edge costs are cached, so assembly does not trigger new pair-cost queries. Sorting and deduplicating at most k records for each of N sources costs $O(kN \log(k+1))$ work, and packing the distinct records is linear.

Excluding the separately counted pair-cost queries, these bounds give

$$T_{\text{build}} = O\big(kNdL + kNL \log N + kNb^2 + kN \log(k+1)\big). \tag{3}$$

Index-array initialization and candidate writes add $O(kN)$ work and are already absorbed. For direct squared Euclidean costs, the pair-cost queries add $O(kNbd)$ work. Finally, for fixed k, b, d, we have $L = O(\log N)$, so the sorting-based construction takes $O(N \log^2 N)$ time.

D.2 Classical exact sparse-assignment bound

Derivation of the bound in Section 5.3. Candidate assembly gives $N \leq e = |\mathcal{E}| \leq kN$, and each $\mathcal{M}^{(r)} \subseteq \mathcal{E}$ is a perfect matching. Thus the final candidate-graph LAP is feasible. A fixed finite-precision binary format has a common power-of-two denominator for all finite representable costs. Multiplying every stored candidate cost by that denominator makes the costs integral. For any perfect matching, the scaled total cost is the same positive multiple of its original total cost, so its minimizers are unchanged. Because the format is fixed, the integer magnitudes are bounded independently of N.

The exact scaling assignment algorithm of Gabow and Tarjan (1989) takes $O(\sqrt{N}\, e \log N)$ time on this graph. Since $e \leq kN$, this becomes $O(N^{1.5} \log N)$ for fixed k. Equation (3), together with $O(kNb)$ cost evaluations taking $O(d)$ time each, gives $O(N \log^2 N)$ construction time for fixed k, b, d. Adding the two stages yields $O(N \log^2 N + N^{1.5} \log N) = O(N^{1.5} \log N)$.

With trees built sequentially, the input clouds, the kN candidate records, the index arrays, and the leaf workspaces require $O(Nd + kN + b^2 + dL)$ words. A sparse scaling solver requires $O(N + e)$ additional workspace, which is $O(kN)$. For fixed b, d, the total is $O(Nd + kN)$ words. □

D.3 Almost-linear theoretical backend

For completeness, the candidate-graph LAP can also be written as an integral minimum-cost-flow problem. Direct each edge from its source to its target, assign unit capacity and cost, and require one unit of flow out of every source and into every target. The integral feasible flows are exactly the perfect matchings on $\mathcal{E}$; their objectives differ from C only by the factor N. This network has $O(N + e) = O(e)$ edges, since $e \geq N$. The same common scaling of stored costs makes its data integral and polynomially bounded. The deterministic algorithm of van den Brand et al. (2023) then gives an $O(e^{1+\varepsilon})$-time exact solution for any fixed $\varepsilon > 0$. For fixed k, b, d and $O(d)$-time cost evaluations, candidate construction is $O(N \log^2 N)$, so the complete procedure admits an $O(N^{1+\varepsilon})$ theoretical time bound. This is an alternative backend, not the LAPMOD solver used for the reported measurements.

D.4 Experimental sparse-LAP backend

For completeness, we account for the LAPMOD backend used for the reported measurements, separating heap-based augmentations from its row-reduction preprocessing.

The sparse graph has $2N$ vertices and e candidate edges. During one search, each matched target is settled at most once and exposes at most one matched source. Consequently, even if the search reaches

the entire graph, it inspects only $O(e)$ row edges and matched edges in total. These inspections create at most $O(e)$ heap insertions and removals, including obsolete entries. Adding $O(N)$ work to initialize distance and settled arrays and to reconstruct the augmenting path gives

$$O(N + e \log(e + 1)) = O(e \log N).$$

Preprocessing may leave $A \leq N$ unmatched targets. Initializing length-N arrays for these searches costs $O(NA)$, which is covered by the bound below. Since at most N augmentations are required, the per-search bound gives

$$T^{\text{worst}}_{\text{augment}} = O(Ne \log N) \leq O(kN^2 \log N).$$

Column reduction takes $O(N + e)$ work. The implementation then performs at most two augmenting row-reduction passes. Let D be the maximum source degree. In one pass, the current-row index can remain unchanged when a displaced row is retried, but it never decreases. The retry rule compares the cumulative iteration count with N times that index. Once the count reaches N^2, the rule permits no further retries, and at most N additional iterations finish the pass. Each pass therefore has $O(N^2)$ iterations. Since one iteration scans at most D candidate edges, both passes inspect $O(N^2 D)$ edges.

Combining preprocessing, dual extraction, and the augmentation bound gives

$$T^{\text{worst}}_{\text{sparse,actual}} = O(N^2 D + Ne \log N) \leq O(kN^2 \log N).$$

The $O(N^2 D)$ term explicitly accounts for repeated row processing instead of identifying preprocessing cost with the heap-search bound.

Combining the sparse-LAP result with Equation (3) gives, for $1 \leq k \leq N$,

$$T^{\text{worst}}_{\text{actual}} = O\big(kNdL + kNL \log N + kNb^2 + N^2 D + Ne \log N\big),$$

plus at most kNb pair-cost queries. For fixed k, b, d, this gives a conservative $O(N^2 \log N)$ upper bound for the experimental LAPMOD backend. The classical scaling and almost-linear backends above establish the theoretical bounds stated in Section 5.3.

D.5 Memory and workspace

We decompose memory into inputs, candidate records, tree-construction workspace, and sparse-solver workspace.

The two point sets require $O(Nd)$ words. There are kN raw candidate records and $e \leq kN$ distinct edges; each record stores an index and its cached cost. Thus input and candidate storage uses $O(Nd + kN + e)$ words.

For P concurrent tree builders, index arrays, temporary projection arrays, one leaf matrix per builder, and direction vectors retained on the recursion stack use

$$O\big(P(N + b^2 + dL)\big)$$

words. A builder releases a node's projection array before descending, so only the direction vectors need remain along a recursion path.

The solver stores $O(N)$ matching, dual, distance, and visit arrays together with an $O(e)$ heap. Its additional workspace is therefore $O(N + e)$. The graph arrays reserved for up to kN entries are already included in the raw-record term above.

Adding these terms yields

$$O\big(Nd + kN + e + P(N + b^2 + dL)\big)$$

words. For fixed k, b, d, P, this is linear in N; in particular, input and candidate storage is $O(Nd + kN)$.

E Benchmark construction and reference assignments

We document the source and target point sets behind Section 6 and the matching used to evaluate each benchmark. The synthetic construction uses the connection between convex gradients and quadratic-cost OT (Brenier, 1991; Peyré and Cuturi, 2019); the finite-assignment proof below applies directly to the sampled point sets.

E.1 SYNTHETIC ASSIGNMENT BENCHMARK: FINITE OPTIMUM

Proposition 4 (Convex-gradient finite optimum)**.** *Let $\phi : \mathbb{R}^d \to \mathbb{R}$ be differentiable and convex, and let $\mathbf{y}_i := \nabla\phi(\mathbf{x}_i)$, followed optionally by a permutation of the targets. For squared Euclidean cost, the pre-shuffle pairing is an optimal finite assignment. If ϕ is λ-strongly convex with $\lambda > 0$ and the $\mathbf{x}_i$ are distinct, that pairing is the unique optimal assignment.*

Proof. Relabel the targets to undo their final shuffle, so that the construction-defined matching is $\mathcal{M}_0 := \{(i, i) : i \in [N]\}$. Relabeling changes neither the set of feasible assignments nor their costs.

Let $\mathcal{M} \in \mathcal{P}([N], [N])$ be arbitrary, and let $\pi : [N] \to [N]$ be the bijection determined by $(i, \pi(i)) \in \mathcal{M}$.

$$C(\mathcal{M}) - C(\mathcal{M}_0) = \frac{1}{N}\sum_{i=1}^{N}\left(\|\mathbf{x}_i - \mathbf{y}_{\pi(i)}\|^2 - \|\mathbf{x}_i - \mathbf{y}_i\|^2\right).$$

For each i, expansion of the squared norms gives

$$\|\mathbf{x}_i - \mathbf{y}_{\pi(i)}\|^2 - \|\mathbf{x}_i - \mathbf{y}_i\|^2 = \|\mathbf{y}_{\pi(i)}\|^2 - \|\mathbf{y}_i\|^2 - 2\mathbf{x}_i^\top(\mathbf{y}_{\pi(i)} - \mathbf{y}_i).$$

The source-norm terms have canceled within each summand. The target-norm terms cancel after summation because π permutes $[N]$. Reindexing the remaining inner products by $j = \pi(i)$ gives

$$C(\mathcal{M}) - C(\mathcal{M}_0) = \frac{2}{N}\sum_i \mathbf{x}_i^\top(\mathbf{y}_i - \mathbf{y}_{\pi(i)}) = \frac{2}{N}\sum_j \mathbf{y}_j^\top(\mathbf{x}_j - \mathbf{x}_{\pi^{-1}(j)}).$$

Since $\mathbf{y}_j = \nabla\phi(\mathbf{x}_j)$, the first-order convexity inequality at $\mathbf{x}_j$, applied to $\mathbf{x}_{\pi^{-1}(j)}$, gives

$$\phi(\mathbf{x}_{\pi^{-1}(j)}) \geq \phi(\mathbf{x}_j) + \mathbf{y}_j^\top(\mathbf{x}_{\pi^{-1}(j)} - \mathbf{x}_j),$$

or equivalently,

$$\mathbf{y}_j^\top(\mathbf{x}_j - \mathbf{x}_{\pi^{-1}(j)}) \geq \phi(\mathbf{x}_j) - \phi(\mathbf{x}_{\pi^{-1}(j)}).$$

The indices $\pi^{-1}(j)$ are a permutation of $[N]$, so

$$\sum_j \phi(\mathbf{x}_j) - \sum_j \phi(\mathbf{x}_{\pi^{-1}(j)}) = 0.$$

Substitution into the cost difference above yields $C(\mathcal{M}) - C(\mathcal{M}_0) \geq 0$. Hence the construction-defined matching is optimal.

If ϕ is λ-strongly convex, the first-order inequality becomes

$$\mathbf{y}_j^\top(\mathbf{x}_j - \mathbf{x}_{\pi^{-1}(j)}) \geq \phi(\mathbf{x}_j) - \phi(\mathbf{x}_{\pi^{-1}(j)}) + \frac{\lambda}{2}\|\mathbf{x}_j - \mathbf{x}_{\pi^{-1}(j)}\|^2.$$

Repeating the telescoping argument gives

$$C(\mathcal{M}) - C(\mathcal{M}_0) \geq \frac{\lambda}{N}\sum_j \|\mathbf{x}_j - \mathbf{x}_{\pi^{-1}(j)}\|^2.$$

If the source points are distinct and $\mathcal{M} \neq \mathcal{M}_0$, then at least one index j satisfies $\pi^{-1}(j) \neq j$, and the corresponding squared distance on the right is strictly positive. Thus every nonidentity permutation has strictly larger cost, proving uniqueness. □

Analytic dual certificate. The same optimum can be certified through the LAP dual. Let

$$u_i := \|\mathbf{x}_i\|^2 - 2\phi(\mathbf{x}_i), \qquad v_j := \|\mathbf{y}_j\|^2 - 2\phi^*(\mathbf{y}_j),$$

whenever these quantities are finite, where ϕ^* is the convex conjugate. Fenchel's inequality, $\phi(\mathbf{x}_i) + \phi^*(\mathbf{y}_j) \geq \mathbf{x}_i^\top\mathbf{y}_j$, implies

$$u_i + v_j \leq \|\mathbf{x}_i\|^2 + \|\mathbf{y}_j\|^2 - 2\mathbf{x}_i^\top\mathbf{y}_j = \|\mathbf{x}_i - \mathbf{y}_j\|^2.$$

Thus $(\mathbf{u}, \mathbf{v})$ is dual feasible. Equality holds for the construction-defined pairs because $\mathbf{y}_i = \nabla\phi(\mathbf{x}_i)$, so their primal and dual objectives coincide.

E.2 Synthetic assignment benchmark: construction and validation

The benchmark samples three centers independently from $\mathcal{N}(0, 1.5^2\mathbf{I}_d)$. Each source point chooses a center uniformly and adds independent $\mathcal{N}(0, 0.7^2\mathbf{I}_d)$ noise. Targets are formed as

$$\mathbf{y}(\mathbf{x}) := 0.65\mathbf{x} + \mathbf{B}\,\mathrm{diag}(\mathbf{s})\mathbf{B}^\top\mathbf{x} + \frac{0.35}{h}\sum_{r=1}^{h}\widetilde{\sigma}(\mathbf{a}_r^\top\mathbf{x} + o_r)\mathbf{a}_r,$$

where $q := \min(8, d)$, $h := \min(16, 2d)$, and $\mathbf{B} \in \mathbb{R}^{d\times q}$ is the orthonormal factor of a QR decomposition of an independent standard Gaussian matrix. The entries of $\mathbf{s} \in [0.15, 0.55]^q$ are evenly spaced between 0.15 and 0.55 (with $s_1 := 0.15$ when $q = 1$). Independently, $\mathbf{a}_r \sim \mathcal{N}(0, \mathbf{I}_d/d)$ and $o_r \sim \mathcal{N}(0, 0.2^2)$. The clipped logistic function is

$$\widetilde{\sigma}(t) := \frac{1}{1 + \exp\{-\,\mathrm{clip}(t, -40, 40)\}}.$$

It is continuous and nondecreasing, so it has a differentiable convex antiderivative F. In particular, the displayed map is the gradient of

$$\phi(\mathbf{x}) := \frac{0.65}{2}\|\mathbf{x}\|^2 + \frac{1}{2}\mathbf{x}^\top\mathbf{B}\,\mathrm{diag}(\mathbf{s})\mathbf{B}^\top\mathbf{x} + \frac{0.35}{h}\sum_{r=1}^{h}F(\mathbf{a}_r^\top\mathbf{x} + o_r).$$

The last two terms are convex because $\mathbf{s} \geq 0$ and F is convex; the first term makes ϕ 0.65-strongly convex. A fixed seed determines all parameters and the final target shuffle. The construction-defined correspondence is used as ground truth, with its optimality justified in exact arithmetic by Proposition 4 for every benchmark size. As an additional implementation check, full LAP audits recover the same permutation for five seeds at each of $N = 512, 1024, 2048$.

E.3 Gaussian-to-image-latent assignment benchmark: reference

The source consists of samples from $\mathcal{N}(0, \mathbf{I}_d/d)$; the target consists of flattened Stable Diffusion VAE posterior-mean features of ImageNet images, with $d = 4096$. Both are stored as point clouds, and the tested sizes use nested prefixes of those clouds. For each size, a dense full-graph LAP maximizes a stored finite-precision affinity matrix computed from $\mathbf{X}^\top\mathbf{Y}$. In exact arithmetic, maximizing this affinity is equivalent to minimizing squared Euclidean assignment cost, because the source and target norm sums are constant across perfect matchings. The dense reference matching and all reported solver matchings are then evaluated using squared Euclidean costs on the stored coordinates. Thus the denominator in Gap is the cost of the same dense reference matching for every method on a given input.

E.4 Geometrically Diverse Assignment Benchmark

For Nonlinear, source points come from a three-component Gaussian mixture and targets are produced by the convex-gradient map above. For Anisotropic, a rotated Gaussian source has unequal coordinate scales; its target is an affine transformation. For Multimodal, the source is an eight-component Gaussian mixture and the target applies an affine transformation with a small nonlinear perturbation. For Manifold, the source is a noisy harmonic embedding of a one-dimensional latent variable and the target is an affine transformation. In these four families, target order is shuffled after generation.

Geometry mismatch transports an isotropic Gaussian source to an independently sampled six-component Gaussian-mixture target. Torus–trefoil transports points sampled from a torus surface to independently sampled points on a tube around a trefoil knot. The full LAP on each tested benchmark instance provides its reference matching. Section 6.4 reuses the $N = 8192$ Geometry mismatch and Torus–trefoil inputs to compare tree budgets and recombination controls.

F Extended experimental results and controls

We first report the complete versions of the three main tables, then examine recombination, parameter sensitivity, and source-degree-matched candidate controls. Unless stated otherwise, entries are means $\pm$ sample standard deviations over three inputs or solver runs. Gap, OOT, and OOM have the same meanings as in Section 6.

F.1 COMPLETE MAIN-TABLE RESULTS

Tables 4–6 correspond to Tables 1–3, respectively, and add sample standard deviations to the reported means.

Table 4: Synthetic assignment benchmark. Mean ± standard deviation of Gap (%) and runtime (s).

N	d	HiRef		BSP-OT		HALO		SER-OT	
		Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
2,048	16	25.966±6.811	17.20±0.48	0.000±0.000	0.20±0.01	0.000±0.000	53.04±3.81	0.000±0.000	0.04±0.00
	64	21.223±18.647	17.64±0.37	0.000±0.000	0.21±0.02	0.000±0.000	20.66±1.53	0.000±0.000	0.05±0.00
	128	26.520±2.850	17.74±0.51	0.000±0.000	0.31±0.01	0.000±0.000	31.88±2.87	0.000±0.000	0.06±0.00
	256	19.142±8.334	17.93±0.35	0.000±0.000	0.53±0.00	0.000±0.000	52.01±3.54	0.000±0.000	0.09±0.01
	512	30.951±1.709	18.21±0.24	0.000±0.000	0.96±0.02	0.000±0.000	97.63±4.64	0.000±0.000	0.15±0.00
	1,024	34.949±7.800	18.96±0.12	0.000±0.000	1.86±0.03	0.000±0.000	181.15±4.55	0.000±0.000	0.27±0.01
8,192	16	91.484±14.628	61.56±0.69	0.000±0.000	0.61±0.06	0.000±0.000	382.99±37.32	0.000±0.000	0.19±0.00
	64	102.773±13.474	64.30±3.69	0.000±0.000	0.57±0.00	0.021±0.036	276.23±32.70	0.000±0.000	0.22±0.00
	128	124.483±29.185	65.09±3.78	0.000±0.000	1.04±0.01	3.974±6.054	491.03±41.82	0.000±0.000	0.28±0.01
	256	139.212±42.358	63.41±1.05	0.000±0.000	1.95±0.02	1.056±1.269	1204.72±655.16	0.000±0.000	0.42±0.01
	512	122.313±7.728	64.50±0.94	0.000±0.000	3.90±0.07	OOT		0.000±0.000	0.68±0.00
	1,024	125.583±11.629	67.47±0.95	0.000±0.000	8.31±0.05	OOM		0.000±0.000	1.21±0.02
32,768	16	52.202±5.576	246.46±6.68	4.623±0.453	0.80±0.01	OOT		0.000±0.000	0.82±0.01
	64	98.740±4.285	269.37±28.64	0.000±0.000	2.21±0.01	OOM		0.000±0.000	0.95±0.05
	128	123.855±14.958	254.54±3.88	0.000±0.000	4.24±0.05	OOM		0.000±0.000	1.32±0.02
	256	126.287±17.944	252.90±7.25	0.000±0.000	8.79±0.04	OOM		0.000±0.000	1.89±0.04
	512	154.107±8.184	268.29±17.06	0.000±0.000	17.20±0.35	OOM		0.000±0.000	3.13±0.04
	1,024	160.362±19.914	275.29±17.07	0.000±0.000	32.34±1.10	OOM		0.000±0.000	5.01±0.01
131,072	16	91.846±12.394	982.97±48.68	0.000±0.000	8.64±0.80	OOM		0.005±0.004	3.87±0.10
	64	127.363±8.358	1025.08±19.39	0.060±0.025	10.27±0.26	OOM		0.000±0.000	4.25±0.09
	128	148.300±10.721	1027.42±39.52	0.000±0.000	21.07±0.53	OOM		0.000±0.000	5.54±0.09
	256	154.523±6.563	1026.46±34.69	0.000±0.000	36.64±1.81	OOM		0.000±0.000	8.08±0.11
	512	178.855±34.483	1041.63±42.13	0.000±0.000	58.58±0.48	OOM		0.000±0.000	12.61±0.62
	1,024	191.933±46.546	1114.56±52.85	0.000±0.000	113.92±0.36	OOM		0.000±0.000	21.10±1.47
524,288	16	OOT		0.000±0.000	33.85±2.48	OOM		0.006±0.002	16.27±0.35
	64	OOT		0.448±0.109	40.68±1.18	OOM		0.000±0.000	19.19±0.27
	128	OOT		0.009±0.001	74.62±2.00	OOM		0.000±0.000	25.62±0.68
	256	OOT		0.000±0.000	131.41±0.87	OOM		0.000±0.000	36.22±1.61
	512	OOT		0.000±0.000	237.77±1.53	OOM		0.000±0.000	51.89±2.69
	1,024	OOT		0.000±0.000	468.89±1.27	OOM		0.000±0.000	88.55±2.49
1,048,576	16	OOT		0.001±0.002	71.87±7.10	OOM		0.003±0.003	33.51±1.27
	64	OOT		0.635±0.163	84.53±1.75	OOM		0.000±0.000	38.93±1.50
	128	OOT		0.023±0.008	152.57±3.28	OOM		0.000±0.000	52.72±0.66
	256	OOT		0.000±0.000	268.51±1.60	OOM		0.000±0.000	76.51±2.85
	512	OOT		0.000±0.000	486.88±1.38	OOM		0.000±0.000	114.77±4.15
	1,024	OOT		0.000±0.000	971.31±7.93	OOM		0.000±0.000	174.46±2.44

Table 5: Gaussian-to-image-latent assignment benchmark. Mean ± standard deviation of Gap (%) and runtime (s).

N	HiRef		BSP-OT		HALO		SER-OT	
	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
2,048	0.033181±0.000024	21.54±19.15	0.012101±0.000060	8.62±0.94	0.003134±0.000000	1295.60±169.31	0.000146±0.000011	1.06±0.01
8,192	0.037032±0.000029	43.49±14.20	0.015377±0.000075	30.63±4.43	OOM		0.001013±0.000006	5.57±0.31
32,768	0.040248±0.000137	173.94±28.87	0.018590±0.000021	128.80±8.61	OOM		0.002734±0.000008	31.12±1.01
131,072	0.043256±0.000025	1421.56±36.67	0.021456±0.000005	488.69±20.66	OOM		0.004822±0.000004	129.97±3.48

Table 6: Geometrically Diverse Assignment Benchmark. Mean ± standard deviation of Gap (%) and runtime (s).

Family (d)	N	BSP-OT		SER-OT (U)		SER-OT	
		Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
Nonlinear (16)	2,048	0.871±1.509	0.076±0.017	0.000±0.000	0.050±0.005	0.000±0.000	0.041±0.003
Anisotropic (64)	2,048	0.707±0.613	0.152±0.018	0.000±0.000	0.049±0.001	0.000±0.000	0.039±0.001
Multimodal (16)	2,048	6.764±1.653	0.073±0.011	0.000±0.000	0.105±0.002	0.000±0.000	0.080±0.006
Manifold (16)	2,048	11.524±2.216	0.071±0.011	0.064±0.021	0.410±0.023	0.015±0.006	0.361±0.018
Geometry mismatch (16)	2,048	2.526±0.038	0.060±0.005	0.101±0.008	0.387±0.039	0.017±0.002	0.425±0.048
Torus–trefoil (3)	2,048	7.367±1.008	0.037±0.003	0.421±0.026	0.493±0.017	0.208±0.033	0.457±0.030
Geometry mismatch (16)	8,192	3.891±0.394	0.197±0.008	0.298±0.044	7.090±2.011	0.056±0.005	7.757±1.500
Torus–trefoil (3)	8,192	7.963±0.580	0.099±0.014	1.039±0.011	7.048±1.920	0.608±0.037	7.118±0.998

F.2 Recombination and parameter sensitivity

These experiments use target-difference vectors, retain only leaf-matching edges, and use the same exact global LAP as the main experiments. We report means $\pm$ sample standard deviations over three fixed inputs on 16 CPU threads, scoring each tree matching before global recombination. Reference-edge coverage is $\mathrm{Cov}(\mathcal{E}; \mathcal{M}_{\mathrm{ref}}) := |\mathcal{E} \cap \mathcal{M}_{\mathrm{ref}}|/N$, reported as a fraction in $[0, 1]$. It measures reference edges present in the candidate graph, not agreement of the final assignment. The reference is the certified optimum for the synthetic assignment benchmark and the Geometrically Diverse Assignment Benchmark and the dense reference from Section 6.2 for the Gaussian-to-image-latent assignment benchmark.

Table 7 varies k and b on the synthetic $N = 8192$, $d = 16$ benchmark. As k increases, complementary tree matchings substantially improve reference-edge coverage and the quality of the globally recombined solution. At $k = b = 64$, their union attains coverage 1.0000 ± 0.0000, and exact recombination recovers the ground-truth cost at displayed precision. This occurs even though no individual tree provides a comparably accurate assignment, demonstrating that the gain arises from cross-tree edge recombination rather than selection of a single favorable tree. On the $N = 2048, d = 4096$ image-latent input, recombination similarly reduces the best tree's excess cost by 98.4%, from a gap of 0.009219% to 0.000146%.

Table 7: Recombination and parameter sensitivity on the synthetic assignment benchmark ($N = 8192, d = 16$). Mean $\pm$ standard deviation of Gap (%), reference-edge coverage (Cov.), and runtime (s). The upper block varies k at $b = 64$; the lower varies b at $k = 16$.

k	b	Best-tree gap	Final gap	$\lvert\mathcal{E}\rvert/N$	Cov.	Time (s)
1	64	248.211±81.327	248.211±81.327	1.000±0.000	0.5116±0.0328	0.039±0.002
4	64	122.152±19.644	31.984±10.554	2.725±0.182	0.9508±0.0204	0.058±0.005
16	64	100.172±17.599	0.220±0.068	7.894±0.121	0.9997±0.0001	0.056±0.005
64	64	94.390±12.264	0.000±0.000	25.988±0.236	1.0000±0.0000	0.182±0.004
16	1	206.597±27.587	3.356±0.746	12.061±0.193	0.9924±0.0012	0.037±0.002
16	4	186.273±16.337	1.640±0.514	11.367±0.079	0.9966±0.0005	0.030±0.003
16	16	140.830±19.036	0.256±0.161	9.508±0.058	0.9994±0.0003	0.029±0.002
16	64	100.172±17.599	0.220±0.068	7.894±0.121	0.9997±0.0001	0.056±0.005
16	256	69.267±10.457	0.178±0.222	6.175±0.240	0.9998±0.0003	0.355±0.077

Because changing b also changes subsequent random-number consumption, these rows provide a marginal sensitivity comparison rather than a pathwise comparison along nested refinements. In these experiments, larger leaves lower the best-tree cost and reduce the number of distinct retained edges, but eventually increase leaf-LAP time.

F.3 Degree-matched candidate controls

These controls compare candidate graphs at the same source degrees. SER-OT and the kNN baselines use the same exact global LAP, and every kNN graph is made feasible by one random-permutation edge per source. These controls are timed separately from the main benchmarks, so their runtimes need not match the corresponding main-table entries.

Table 8 matches every source degree while varying the synthetic dimension. For every input and seed, one random-permutation edge per source ensures feasibility, and exact squared-Euclidean nearest neighbors fill the remaining slots up to that source's SER-OT degree. The same exact global LAP then solves both graphs. Both methods reach the ground-truth cost in all cells except $N = 8192, d = 1024$, where one kNN seed misses an optimal edge. At $N = 32768$, SER-OT is faster throughout; all-pairs matrix products make kNN competitive at the smaller size.

Table 8: Degree-matched exact kNN+LAP controls on synthetic and image-latent inputs. Mean $\pm$ standard deviation of Gap (%), reference-edge coverage (Cov.), and runtime (s). The final row uses image latents.

N	d	$\lvert\mathcal{E}\rvert/N$	SER-OT gap	kNN gap	SER-OT Cov.	kNN Cov.	SER-OT time (s)	kNN time (s)
8,192	16	25.988±0.236	0.000000±0.000000	0.000000±0.000000	1.0000±0.0000	1.0000±0.0000	0.182±0.004	0.233±0.022
	64	25.502±0.837	0.000000±0.000000	0.000000±0.000000	1.0000±0.0000	1.0000±0.0000	0.212±0.014	0.215±0.011
	128	21.832±1.172	0.000000±0.000000	0.000000±0.000000	1.0000±0.0000	1.0000±0.0000	0.262±0.012	0.248±0.102
	256	19.383±1.047	0.000000±0.000000	0.000000±0.000000	1.0000±0.0000	1.0000±0.0000	0.377±0.007	0.199±0.009
	512	16.646±0.211	0.000000±0.000000	0.000000±0.000000	1.0000±0.0000	1.0000±0.0000	0.614±0.003	0.265±0.010
	1,024	12.345±1.518	0.000000±0.000000	0.125718±0.217750	1.0000±0.0000	1.0000±0.0001	1.083±0.011	0.500±0.080
32,768	16	31.845±0.537	0.000000±0.000000	0.000000±0.000000	1.0000±0.0000	1.0000±0.0000	0.805±0.010	2.974±0.433
	64	31.399±0.752	0.000000±0.000000	0.000000±0.000000	1.0000±0.0000	1.0000±0.0000	0.902±0.017	2.223±0.139
	128	29.033±0.859	0.000000±0.000000	0.000000±0.000000	1.0000±0.0000	1.0000±0.0000	1.196±0.021	2.769±0.119
	256	25.419±0.663	0.000000±0.000000	0.000000±0.000000	1.0000±0.0000	1.0000±0.0000	1.765±0.101	3.693±0.381
	512	19.737±1.375	0.000000±0.000000	0.000000±0.000000	1.0000±0.0000	1.0000±0.0000	2.689±0.022	5.421±0.350
	1,024	14.830±0.431	0.000000±0.000000	0.000000±0.000000	1.0000±0.0000	1.0000±0.0000	4.429±0.117	7.567±0.227
2,048	4,096	42.450±0.079	0.000146±0.000011	0.033417±0.000185	0.9259±0.0032	0.0213±0.0010	1.064±0.010	0.391±0.056

On the $N = 2048, d = 4096$ image-latent input, exact kNN covers only 2.13% of dense-reference edges versus SER-OT's 92.59%. The degree-matched random+LAP control has gap $0.013296 \pm 0.000170\%$, coverage 0.0210 ± 0.0017, and time 0.249 ± 0.016 seconds. Exact kNN peaks near 653 MiB versus native SER-OT's 70 MiB. The common exact LAP and matched source degrees isolate the quality of the discovered candidate edges.

In Table 9, Geometry denotes the Geometry mismatch benchmark of Section 6.3.

Table 9: Degree-matched exact kNN+LAP controls on the Geometrically Diverse Assignment Benchmark. Mean $\pm$ standard deviation of Gap (%), reference-edge coverage (Cov.), and runtime (s).

Family	d	N	SER-OT gap	kNN gap	$\lvert\mathcal{E}\rvert/N$	SER-OT Cov.	kNN Cov.	SER-OT time (s)	kNN time (s)
Nonlinear	16	2,048	0.000±0.000	0.000±0.000	19.31±1.29	1.000±0.000	1.000±0.000	0.041±0.003	0.036±0.006
Anisotropic	64	2,048	0.000±0.000	0.000±0.000	16.29±0.74	1.000±0.000	1.000±0.000	0.039±0.001	0.034±0.006
Multimodal	16	2,048	0.000±0.000	742.742±96.240	25.84±1.73	1.000±0.000	0.436±0.012	0.080±0.006	0.229±0.009
Manifold	16	2,048	0.015±0.006	21.333±12.402	39.86±0.75	0.959±0.017	0.904±0.031	0.361±0.018	0.484±0.089
Geometry	16	2,048	0.017±0.002	34.788±3.837	33.78±0.36	0.968±0.006	0.194±0.058	0.425±0.048	0.175±0.030
Torus–trefoil	3	2,048	0.208±0.033	213.066±39.498	38.90±0.96	0.884±0.014	0.799±0.004	0.457±0.030	0.412±0.030
Geometry	16	8,192	0.056±0.005	45.563±2.902	42.98±0.38	0.925±0.007	0.149±0.021	7.757±1.500	1.602±0.209
Torus–trefoil	3	8,192	0.608±0.037	670.923±3.487	54.93±0.36	0.482±0.016	0.567±0.011	7.118±0.998	5.588±0.597

The matched-degree kNN+LAP result is strongly input-dependent: it is exact on the nonlinear and anisotropic inputs, but its mean gap ranges from 21.333% to 742.742% on the other tested families. In contrast, SER-OT remains between 0 and 0.608% on the same cases, showing much more stable accuracy across these inputs. At $N = 8192$ on torus–trefoil, kNN even covers more optimal edges than SER-OT but has a much larger gap; coverage alone does not measure the gain of missing assignment cycles or the quality of substitutes. These results demonstrate SER-OT's empirical stability across the tested input families.