The Modal Voronoi Wasserstein Transport Algorithm: A Fast Near Optimal Transport Algorithm for Point Sets

Zachary Mullaghy

April 30, 2025

Abstract

We introduce a computationally efficient and interpretable algorithm for approximating optimal transport between point sets. The method aligns the first and second moments (centroid and covariance) of the distributions and then reassigns mass locally through a Voronoi diagram defined on the target point set. We examine how many points from the mode-matched set fall within each Voronoi cell of the target point set and then construct a near optimal method to assign further movement. We detail the theoretical structure, optional higher-order extensions, and provide a computational complexity analysis, demonstrating the method's sub-quadratic performance in practice.

1 Introduction

Optimal transport is a fundamental tool for comparing probability distributions and point sets [10, 8]. Traditional methods like the Hungarian algorithm [6] or Sinkhorn iterations [3] may scale poorly in high dimensions or with low regularization and may obscure the physical structure of the transport. We present an alternative framework that exploits modal structure and geometric locality to perform fast, structured transport approximations.

2 Algorithm Overview

Let $X = \{x_i\}_{i=1}^N$ and $Y = \{y_j\}_{j=1}^N$ be point sets in $\mathbb{R}^2$.

2.1 Step 1: Modal Matching of First and Second Moments

We begin by aligning the global structure of X to match that of Y using moment matching[9]:

- **Centroid Matching (Rank-1)**:

$$x_i \leftarrow x_i + (\mu_Y - \mu_X), \quad \text{where } \mu = \frac{1}{N}\sum x_i$$

- **Covariance Matching (Rank-2)**: Let Σ_X and Σ_Y be the empirical covariance matrices. Compute the SVD:

$$\Sigma_X = U_X S_X U_X^T, \quad \Sigma_Y = U_Y S_Y U_Y^T$$

Then transform:

$$T = U_Y \sqrt{S_Y / S_X} U_X^T, \quad x_i \leftarrow T(x_i - \mu_X) + \mu_Y$$

Optional Extensions: Higher-order tensors (rank-3 and above) or rigid body rotation minimizations may be incorporated for users requiring more detailed structure matching.

2.2 Step 2: Voronoi Diagram Construction

Construct the Voronoi diagram of Y using a standard computational geometry package. Each cell is associated with a generator y_j. The domain is partitioned such that every point in $\mathbb{R}^2$ lies in the cell of the nearest y_j.

2.3 Step 3: Generator Identification via Fictitious Reassignment

After modal alignment, assign each transformed point x_i to the Voronoi cell of Y in which it lies. If a cell contains more than one point, we:

- Identify neighboring Voronoi cells with no assigned points.
- Reassign excess points (those beyond the first) to the nearest available empty generator.
- Use Euclidean proximity to minimize fictitious cost.

Fictitious vs. Real Work

The reassignment of points to unoccupied Voronoi cells is treated as *fictitious work*: it facilitates a bijective mapping between point sets, but does not contribute to the physically meaningful transport cost. This is because reassignment serves only to determine the destination for "snapping" into place. No physical movement occurs during this stage.

Only movement that aligns modal structure and snapping to the final generator locations is treated as real.

2.4 Step 4: Snapping to Generators

Once each cell contains exactly one assigned point, we perform a final movement from each point to the generator of its assigned cell:

$$\text{Work}_{\text{snap}} = \sum_{i=1}^{N} \|\tilde{x}_i - y_{\text{assign}(i)}\|^2$$

where $\tilde{x}_i$ is the point after modal alignment and $y_{\text{assign}(i)}$ is its assigned generator.

3 Work Decomposition

We distinguish between real and fictitious work as follows:

$$\text{Total Work}_{\text{modal}} = \underbrace{\sum_{i=1}^{N} \|x_i - \tilde{x}_i\|^2}_{\text{Moment Matching (real)}} + \underbrace{\sum_{i=1}^{N} \|\tilde{x}_i - y_{\text{assign}(i)}\|^2}_{\text{Snap to Generators (real)}}$$

Any intermediate reassignment of points during the Voronoi phase is excluded from the work total.

4 Comparison to Optimal Transport

We compare the modal method against the Hungarian algorithm for optimal transport. While the modal approach is slightly sub-optimal, it achieves within a small fraction of the Wasserstein-2 cost [5] at dramatically reduced runtime.

For a 2D Gaussian-to-Gaussian example with $N = 100$:

$$\begin{aligned} \text{Moment Matching Work} &= 33068.84 \\ \text{Snap to Generators Work} &= 58.27 \\ \text{Total Modal Work} &= 33127.11 \\ \text{Hungarian Work} &= 33025.20 \\ \text{Modal/Hungarian Ratio} &= 1.0031 \end{aligned}$$

The modal method requires only 0.31% more transport cost, while operating in near-linear time.

5 Computational Complexity

The efficiency of this method arises from the following observations:

- **Moment Matching (Centroid + Covariance):** $\mathcal{O}(N)$
- **Voronoi Diagram Construction:** $\mathcal{O}(N \log N)$ in 2D
- **Assignment to Empty Cells:** $\mathcal{O}(N)$
- **Snapping to Generators:** $\mathcal{O}(N)$

Thus, the full method runs in:

$$\boxed{\mathcal{O}(N \log N)} \quad \text{with a small constant factor}$$

This is a major computational improvement over the Hungarian algorithm ($\mathcal{O}(N^3)$) and even entropic regularized methods like Sinkhorn ($\mathcal{O}(N^2)$).

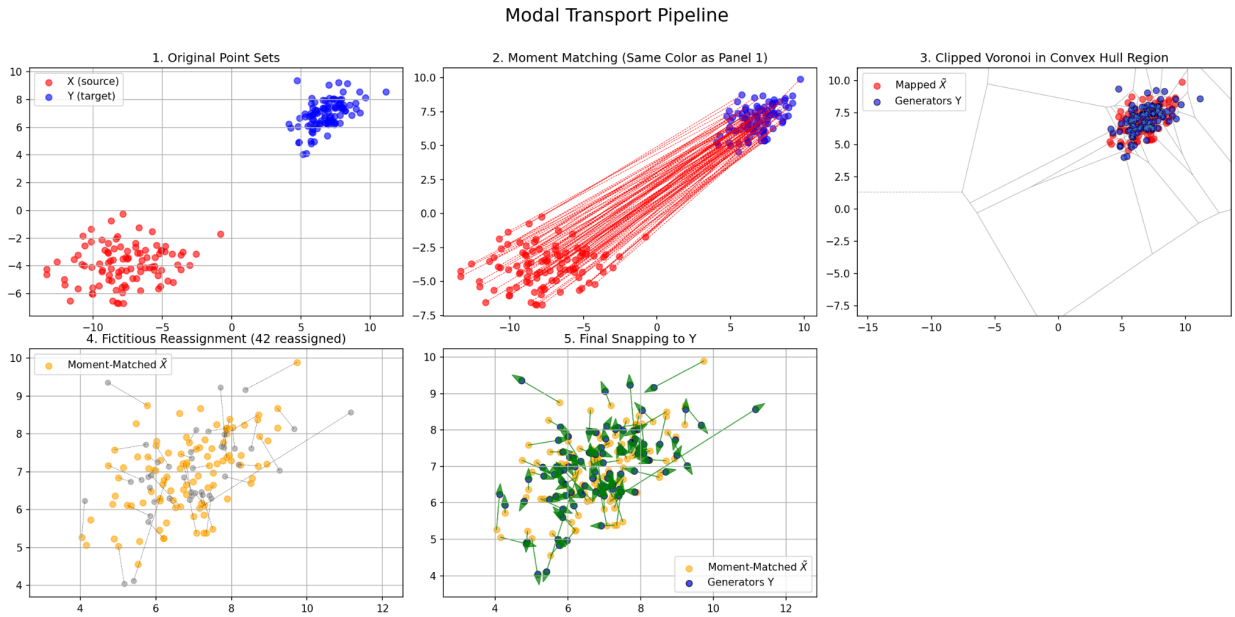

Figure 1: The modal transport algorithm visualized step-by-step. (1) Original point sets X (red) and Y (blue). (2) Moment matching aligns X to match Y's first and second moments. (3) A Voronoi diagram is constructed from Y in the convex hull region. (4) Fictitious reassignment resolves overcrowded cells. (5) Final snapping assigns each $\tilde{X}_i$ to a unique generator y_j.

6 Dimensional Scalability and Computational Considerations

While the proposed modal transport algorithm performs exceptionally well in two dimensions, its scalability to higher dimensions requires careful consideration.

The core components of our algorithm — moment matching, nearest-neighbor queries, and snapping — remain efficient in higher-dimensional spaces. Moment matching involves computing means and covariance matrices, which scales as $\mathcal{O}(Nd^2)$, where d is the dimensionality. Nearest-neighbor queries can be efficiently approximated using KD-trees or Ball trees in moderate dimensions (up to $d \sim 20$) [2, 7].

However, the computational bottleneck arises in the construction of Voronoi diagrams in $\mathbb{R}^d$. While Voronoi diagrams are conceptually well-defined in any dimension, their combinatorial complexity increases exponentially with dimension. In particular, the number of faces of a Voronoi diagram in $\mathbb{R}^d$ is known to be $\mathcal{O}(N^{\lceil d/2 \rceil})$ in the worst case [4, 1]. It is fortunate however that the Voronoi diagram only needs to be computed once.

In future work I intend to consider higher order tensor mode matching and other "fictitious" insightful comptuational point movements or rotations that if framed well could improve the inexactness of the method. If a natural bijection can be shown for a cheap real work cost then this should improve the ratio of Modal to Hungarian work in Wasserstein 2.

References

[1] Franz Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.

[2] Jon L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[3] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in Neural Information Processing Systems*, 26, 2013.

[4] Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry*. EATCS Monographs on Theoretical Computer Science. Springer, 1987.

[5] Jean Feydy, Thibault Séjourné, François-Xavier Vialard, Shun-ichi Amari, Alain Trouvé, and Gabriel Peyré. Interpolating between optimal transport and mmd using sinkhorn divergences. *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 2681–2690, 2019.

[6] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340, 1987.

[7] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP (1)*, volume 2, pages 331–340, 2009.

[8] Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11(5-6):355–607, 2019.

[9] Anuj Srivastava, Ian H Jermyn, Shantanu H Joshi, and Eric P Klassen. Statistical shape analysis: clustering, learning, and testing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):590–602, 2005.

[10] Cédric Villani. *Topics in Optimal Transportation*. Springer, 2003.