An Improved ε-Approximation Algorithm for Geometric Bipartite Matching

Pankaj K. Agarwal
Duke University

Sharath Raghvendra
Virginia Tech

Pouyan Shirzadian
Virginia Tech

Rachita Sowle
Virginia Tech

Abstract

For two point sets $A, B \subset \mathbb{R}^d$, with $|A| = |B| = n$ and $d > 1$ a constant, and for a parameter $\varepsilon > 0$, we present a randomized algorithm that, with probability at least $1/2$, computes in $O(n(\varepsilon^{-O(d^3)} \log\log n + \varepsilon^{-O(d)} \log^4 n \log^5 \log n))$ time, an ε-approximate minimum-cost perfect matching under any L_p-metric. All previous algorithms take $n(\varepsilon^{-1} \log n)^{\Omega(d)}$ time. We use a randomly-shifted tree, with a polynomial branching factor and $O(\log\log n)$ height, to define a tree-based distance function that ε-approximates the L_p metric as well as to compute the matching hierarchically. Then, we apply the primal-dual framework on a compressed representation of the residual graph to obtain an efficient implementation of the Hungarian-search and augment operations.

1 Introduction

Let $A, B \subset \mathbb{R}^d$ be two point sets of size n each, where $d > 1$ is a constant, and $\mathsf{d}(\cdot, \cdot)$ a metric. Let $\mathcal{G} = (A \cup B, A \times B)$ be a weighted complete bipartite graph in which the cost of an edge $(a, b) \in A \times B$ is $\mathsf{d}(a, b)$. A *matching* in $\mathcal{G}$ is a set of vertex-disjoint edges in $\mathcal{G}$. A *perfect matching* in $\mathcal{G}$ is a matching of size n. The cost of a matching M, denoted by $¢(M)$, is $¢(M) = \sum_{(a,b)\in M} \mathsf{d}(a, b)$. The *minimum-cost perfect matching* in $\mathcal{G}$, denoted by M^*, is a perfect matching in $\mathcal{G}$ of the minimum cost. For any $\varepsilon > 0$, a perfect matching M in $\mathcal{G}$ is called an *ε-approximate matching* if $¢(M) \leq (1 + \varepsilon)¢(M^*)$. We consider the case where the cost $\mathsf{d}(a, b)$ is the L_p distance denoted by $\|a - b\|_p$. The optimal transport (OT) distance between two (possibly continuous) distributions can be estimated by taking n samples from both distributions and then computing their minimum-cost perfect matching. The wide applicability of OT in Machine Learning and Computer Vision [5, 16, 19] has motivated the design of fast exact and approximation algorithms that compute a minimum-cost perfect matching. In this paper, for the L_p-norm, we present a new algorithm to computing an ε-approximate matching.

1.1 Related work

For an arbitrary weighted bipartite graph with n vertices and m edges, the Kuhn-Munkres algorithm [12] computes a minimum-weight bipartite matching in a weighted bipartite graph in $O(mn + n^2 \log n)$ time. For bipartite graphs with non-negative integer costs bounded by C, Gabow and Tarjan [9] gave an $O(m\sqrt{n} \log(nC))$-time algorithm. Both the Hungarian and Gabow-Tarjan algorithms are combinatorial algorithms that iteratively find an augmenting path and augment the matching along this path. The augmenting paths are chosen such that the increase in the matching cost after each augmentation is minimized. This cost increase is referred

to as the *net-cost* of the path. Alternate approaches such as the electrical flow [20] method and the matrix multiplication based methods [14] can be used to obtain fast matching algorithms. The current best known execution time is $\tilde{O}(m + n^{1.5})$.

When $A \cup B$ is a 2-dimensional point set in the Euclidean space, a Euclidean minimum-weight matching (EMWM) can be computed in $O(n^2 \operatorname{polylog} n)$ time [11], and in $O(n^{3/2} \operatorname{polylog} n)$ time when the points have integer coordinates [17, 18]. For this case, it is easy to compute a $O(\log n)$-approximate matching in expectation using a randomly shifted quad-tree [6, 7]. Agarwal and Varadarajan [1] used the shifted quad-tree to compute an $O(\log 1/\delta)$-approximate solution in $O(n^{1+\delta})$ time. Following this, there were several results that used such a decomposition; see for instance [4, 8, 10]. The current best-known approximation algorithm for computing EMWM is by Raghvendra and Agarwal [15], which computes an ε-approximate matching with high probability in $n(\varepsilon^{-1} \log n)^{O(d)}$ time. In their algorithm, each cell $\square$ of a randomly shifted quad-tree Q is decomposed by a uniform grid into $(\log n/\varepsilon)^{O(d)}$ subcells. The Euclidean distance between any pair of points u, v with $\square$ as their least common ancestor in Q is ε-approximated by the distance between the subcells of $\square$ that contain u and v respectively. Their algorithm uses Q to compute a minimum net-cost augmenting path P with respect to the new distance and augment the matching along this path, both in time $O(|P| \operatorname{poly} \log n)$. They obtain a near-linear execution time by bounding the total length of all augmenting paths by $O(\varepsilon^{-1} n \log n)$. To compute these paths quickly, they compress the residual graph inside $\square$ into a graph of $(\log n/\varepsilon)^{O(d)}$ size and execute Bellman-Ford algorithm on this graph. Lahn and Raghvendra [13] extended this framework to approximate the 2-Wasserstein distance of planar point sets, i.e., an approximate minimum-cost matching when $\mathsf{d}(u, v)$ is $\|u - v\|_2^2$. Unlike Euclidean distance, approximating the squared-Euclidean distance using Q results in a polynomial sized compressed residual graph at each cell. Since using Bellman-Ford algorithm on such a compressed graph can be prohibitively expensive, they introduce a novel primal-dual framework and define *compressed feasibility* on the compressed residual graph. Using this framework, they are able to find an augmenting path as well as augment it along this path in sub-linear time. Consequently, they achieve an $O(n^{5/4} \operatorname{poly}(\log n, 1/\varepsilon))$ time algorithm for the 2-Wasserstein distance between planar point sets. Recently, Agarwal *et al.* [2] have designed a deterministic algorithm that uses multiple quadtrees to compute a $(1 + \varepsilon)$-approximate Euclidean matching in $n(\varepsilon^{-1} \log n)^{O(d)}$ time.

1.2 Our result

The following theorem states our main result.

Theorem 1.1. *Let A, B be two point sets in $\mathbb{R}^d$ of size n each, for a constant $d > 1$, and let $0 < \varepsilon \leq 1$ be a parameter. With probability at least $1/2$, an ε-approximate matching under any L_p-metric can be computed in $O(n(\varepsilon^{-O(d^3)} \log \log n + \varepsilon^{-O(d)} \log^4 n \log^5 \log n))$ time.*

For the sake of simplicity, we describe the algorithm for the Euclidean metric. It can be extended to other L_p-metrics in a straight forward manner. For any two points a and b, we use $\|a - b\|$ to denote the Euclidean distance between them. Using standard techniques [13, 15], we can preprocess the input points in $O(n \log n)$ time so that the point sets A and B satisfy the following conditions: (P1) All input points have integer coordinates bounded by $n^{O(1)}$. (P2) No integer grid point contains points of both A and B. (P3) $\mathfrak{c}(M^*) \in \left[\frac{3\sqrt{d}n}{\varepsilon}, \frac{9\sqrt{d}n}{\varepsilon}\right]$. Assuming A and B satisfy (P1)–(P3), we present an algorithm that, with probability $1/2$, computes an $(\varepsilon/2)$-approximate matching in $O(n(\varepsilon^{-O(d^3)} + \varepsilon^{-O(d)} \log^4 n \log^4 \log n))$ time. The preprocessing step adds an additional $\log \log n$ factor to the running time of the algorithm, resulting in the running time mentioned in Theorem 1.1. In the following, we provide an overview of our approach and its comparison with existing work.

As in [13, 15], we also define a tree based distance $\mathsf{d_T}(\cdot,\cdot)$ that approximates the Euclidean distance. Unlike [13, 15] that use a quad-tree of height $O(\log n)$, we build a tree T of height $O(\log\log n)$ (see Section 2.1 in [3]). Each cell of T at level i (root is assigned level 0) with a side length of ℓ_i is partitioned using a randomly-shifted grid into children whose side-length $\ell_{i+1} = \ell_i^c$ where $c < 1$ is a constant that depends only on d. Given that the point set have integer coordinates bounded by $n^{O(1)}$ (from (P1)), the height of T is $h = O(\log\log n)$. For any pair of points (u, v) with a cell $\square$ of level i as its least common ancestor, let $\square_u$ and $\square_v$ be the children of $\square$ that contain u and v respectively. As in the case of a randomly shifted-quadtree where we get an $O(h) = O(\log n)$ approximation, one can show that the distance between the centers of $\square_u$ and $\square_v$ is a $O(h) = O(\log\log n)$ approximation of the Euclidean distance (in expectation). We obtain a refined $(1+\varepsilon)$-approximation of the Euclidean distance by partitioning $\square_u$ and $\square_v$ into finer subcells and then using the distance between the centers of those sub-cells that contain u and v. As in [13, 15], one can divide each cell into $O(h^d)$ many subcells and obtain a $(1+\varepsilon)$-approximation of the Euclidean distance. With $h = \log\log n$, this will result in an execution time of $\Omega(n\log^4 n(\varepsilon^{-1}\log\log n)^{d^3})$. Instead, we partition a cell into subcells more carefully (See the definition of subcells in Section 2.2 in [3]). Intuitively, we make the number of subcells a function of the height of the cell, i.e., smaller cells have significantly fewer than $\log^{O(d)}\log n$ subcells. As a result, we are able to improve the dependence of our algorithm from $\log^{O(d^3)}\log n$ to $\log^{O(d)}\log n$. Interestingly, we show that the expected distortion is higher for cells that are closer to the leaves. Nonetheless, we are able to bound the expected error of our distance between any two points u and v by $\varepsilon\|u-v\|$ (See Lemma 3 in [3]).

Similar to [13], our algorithm compactly stores the residual graph (Section 5.2 in [3]) as well as the dual weights (Section 5.3 in [3]) and uses this compact representation to efficiently find augmenting paths. The size of the compressed residual graph inside any cell is bounded by the side-length of its child, i.e., smaller cells have a smaller compressed graph (Lemma 14 in [3]). As a result, finding augmenting paths in smaller cells is significantly faster than that in larger ones. In our analysis, we show that most of the augmenting paths in the algorithm are found in smaller cells which can be computed quickly. In particular, only $O(\frac{n}{\varepsilon\ell_{i+1}})$ augmenting paths are found inside a compressed graph at level i, each of which can be found in $O(\ell_{i+1}\log^2 n)$ time. Combining across all $O(\log\log n)$ levels, we get a near-linear execution time.

Typical matching algorithms that are based on a compressed residual graph modify the dual weights and find an augmenting path with respect to current matching M. The algorithms presented in [13, 15] classify edges into *local* and *non-local* which they use critically in computing a minimum net-cost augmenting path. We remove the need for this classification and make our algorithm and its analysis simpler. Instead of using the classification, our algorithm carefully updates the dual weights, possibly modifies a matching M to another matching M' of the same size and cost, and finds an augmenting path with respect to the new matching M'.

References

[1] Pankaj Agarwal and Kasturi Varadarajan. A near-linear constant-factor approximation for Euclidean bipartite matching? In *Proceedings of the twentieth annual symposium on Computational geometry*, page 247, 2004.

[2] Pankaj K Agarwal, Hsien-Chih Chang, Sharath Raghvendra, and Allen Xiao. Deterministic, near-linear ε-approximation algorithm for geometric bipartite matching. *arXiv preprint arXiv:2204.03875*, 2022.

[3] Pankaj K Agarwal, Sharath Raghvendra, Pouyan Shirzadian, and Rachita Sowle. An improved ε-approximation algorithm for geometric bipartite matching. In *18th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

[4] A. Andoni, K. D. Ba, P. Indyk, and D. P. Woodruff. Efficient sketches for earth-mover distance, with applications. *In Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 324–330, 2009.

[5] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proc. 34th International Conference on Machine Learning*, pages 214–223, 2017.

[6] Moses Charikar. Similarity estimation techniques from rounding algorithms. In *Proc. 34th Annu. ACM Sympos. Theory Comput.*, pages 380–388, 2002.

[7] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *In Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 448–455, 2003.

[8] Kyle Fox and Jiashuai Lu. A near-linear time approximation scheme for geometric transportation with arbitrary supplies and spread. In *Proc. 36th Annual Symposium on Computational Geometry*, pages 45:1–45:18, 2020.

[9] H. N. Gabow and R.E. Tarjan. Faster scaling algorithms for network problems. *SIAM Journal on Computing*, 18:1013–1036, 1989.

[10] Piotr Indyk. A near linear time constant factor approximation for Euclidean bichromatic matching (cost). In *SODA 2007*, page 4, 2007.

[11] Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic planar Voronoi diagrams for general distance functions and their algorithmic applications. *In Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2495–2504, 2017.

[12] Harold Kuhn. Variants of the hungarian method for assignment problems. *Naval Research Logistics*, 3(4):253–258, 1956.

[13] Nathaniel Lahn and Sharath Raghvendra. An $O(n^{5/4})$ time ε-approximation algorithm for RMS matching in a plane. *In Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms*, pages 869–888, 2021.

[14] Marcin Mucha and Piotr Sankowski. Maximum matchings via gaussian elimination. *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 248–255, 2004.

[15] Sharath Raghvendra and Pankaj K. Agarwal. A near-linear time ε-approximation algorithm for geometric bipartite matching. *Journal of the ACM*, 67(3):1–19, June 2020.

[16] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[17] R. Sharathkumar. A sub-quadratic algorithm for bipartite matching of planar points with bounded integer coordinates. In *29th International Symposium on Computational Geometry*, pages 9–16, 2013.

[18] R. Sharathkumar and P. K. Agarwal. Algorithms for transportation problem in geometric settings. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 306–317, 2012.

[19] Justin Solomon, Fernando De Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics*, 34(4):66, 2015.

[20] Jan van den Brand, Danupon Nanongkai Yin-Tat Lee, Richard Peng, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Bipartite matching in nearly-linear time on moderately dense graphs. *IEEE 61st Annual Symposium on Foundations of Computer Science*, pages 919–930, 2020.