A Near-Linear Time ε-Approximation Algorithm for Geometric Bipartite Matching*

R. Sharathkumar
Department of Computer Science
Duke University
sharath@cs.duke.edu

Pankaj K. Agarwal
Department of Computer Science
Duke University
pankaj@cs.duke.edu

ABSTRACT

For point sets $A, B \subset \mathbb{R}^d$, $|A| = |B| = n$, and for a parameter $\varepsilon > 0$, we present an algorithm that computes, in $O(n\text{poly}(\log n, 1/\varepsilon))$ time, an ε-approximate perfect matching of A and B with high probability; the previously best known algorithm takes $\Omega(n^{3/2})$ time. We approximate the L_p norm using a distance function, $\mathtt{d}(\cdot,\cdot)$ based on a randomly shifted quad-tree. The algorithm iteratively generates an approximate minimum-cost augmenting path under $\mathtt{d}(\cdot,\cdot)$ in time proportional to the length of the path. We show that the total length of the augmenting paths generated by the algorithm is $O((n/\varepsilon)\log n)$, implying that the running time of our algorithm is $O(n\text{poly}(\log n, 1/\varepsilon))$.

Categories and Subject Descriptors

G.2.2 [**Discrete Mathematics**]: Graph Theory—*graph algorithms*; F.2.2 [**Theory of Computation**]: Nonnumerical Algorithms and Problems—*geometrical problems and computations*

General Terms

Algorithms, Theory

Keywords

Matching, approximation algorithms

1. INTRODUCTION

Let $\mathcal{G} = (V, E)$, where $V = A \cup B$ and $E \subseteq A \times B$, be a weighted bipartite graph with $|A| = |B| = n$. Let $\mathtt{d}(a, b)$ be the cost of an edge $(a, b) \in E$. A *matching* M in $\mathcal{G}$ is a set of vertex-disjoint edges. The cost of M, $\mathtt{w}(M)$ is $\sum_{(a,b)\in M} \mathtt{d}(a, b)$. M is a *perfect matching* if $|M| = n$. An *optimal* matching is a perfect matching with minimum cost. An *ε-approximate* matching is a perfect matching whose cost is within $(1 + \varepsilon)$ times the cost of an optimal matching. In this paper, we consider geometric settings, namely, A and B are point sets in $\mathbb{R}^d$, $E = A \times B$, and the cost of an edge is the distance between a and b under some L_p-norm. We consider the problem of computing an ε-approximate matching in this setting.

Previous work. Optimal matching on weighted bipartite graphs with n vertices and m edges can be computed using the Hungarian algorithm in $O(mn)$ time [12].[1] If edge costs are positive integers bounded by $n^{O(1)}$, Gabow and Tarjan [5] show that an optimal matching can be computed in $O(m\sqrt{n}\log n)$ time. For unweighted regular bipartite graphs, Goel *et al.* present an $O(n \log n)$ algorithm for computing perfect matchings [7]. There is also extensive work on computing optimal matchings and maximum-cost matchings in non-bipartite graphs [4, 6, 11].

For the case where $A, B \subset \mathbb{R}^2$ and an L_p-norm, Vaidya [15] shows that an optimal matching can be computed in $O(n^{2.5})$ time. Agarwal *et al.* [1] improve the running time of the algorithm for computing optimal matching to $O(n^{2+\delta})$ where $\delta > 0$ is an arbitrarily small constant. For $A, B \subset \mathbb{R}^d$ and the L_1 and L_∞ norms, Vaidya [15] presents $O(n^2 \log^{O(d)} n)$ time algorithm for computing an optimal matching. If $A, B \subseteq [\Delta]^d$, i.e., points in A and B are from a bounded integer grid, Sharathkumar and Agarwal [14] show that an optimal matching can be computed in $O(n^{3/2} \log^{d+O(1)} n \log \Delta)$ time. It is an open question whether a subquadratic algorithm exists for computing an optimal Euclidean bipartite matching in $\mathbb{R}^2$, or even for L_1, L_∞-norms. In contrast, Varadarajan [16] presents an $O(n^{3/2}\,\text{polylog}\, n)$ algorithm for the non-bipartite case — this is surprising because the non-bipartite case seems harder for graphs with arbitrary edge costs. See also [9, 10, 13].

For bipartite graphs on point sets $A, B \subset \mathbb{R}^2$ and for any L_p norm, Agarwal and Varadarajan [17] show that an ε-approximate matching can be computed in $O((n/\varepsilon)^{3/2} \log^5 n)$ time. In [14], Sharathkumar and Agarwal present an algorithm that computes an ε-approximate matching under L_p-norm in $O(n^{3/2}\Phi(n)\log(1/\varepsilon))$ time; here $\Phi(n)$ is the query and update time of a dynamic weighted nearest-neighbor data structure, thereby improving the dependency on ε by an exponential factor. In [2], Agarwal and Varadarajan present a Monte Carlo algorithm for computing an $O(\log(1/\delta))$-approximate matching in time $O(n^{1+\delta})$. Building on the ideas of Agarwal and Varadarajan, Indyk [8] presents an algorithm that estimates in $O(n \log^{O(1)} n)$ time, with probability at least $1/2$, the cost of an optimal matching within a constant factor. It, however, does not return such a matching. It is an open question whether an ε-approximate matching can be computed in near linear-time. Again, the non-bipartite case seems easier and an ε-approximate

*This work is supported by NSF under grants CNS-05-40347, IIS-07-13498, CCF-09-40671, and CCF-1012254, by ARO grants W911NF-07-1-0376 and W911NF-08-1-0452, by an NIH grant 1P50-GM-08183-01, and by a grant from the U.S.–Israel Binational Science Foundation.

STOC'12, May 19–22, 2012, New York, New York, USA.

[1] For arbitrary graphs, a perfect matching may not exist, so the goal is to compute a maximum cardinality minimum-cost matching.

Euclidean matching of a set of points can be computed in near-linear time [3, 17].

Our results. The following theorem states the main result of the paper.

THEOREM 1. *Let $A, B \subset \mathbb{R}^d$ be two point sets with $|A| = |B| = n$ and let $\varepsilon > 0$ be a parameter. For any L_p-norm, an ε-approximate matching of A and B can be computed with high probability in $O(n\text{poly}(\log n, 1/\varepsilon))$ time.*

A simple transformation, as the one discussed in [8], decomposes, with high probability, computing ε-approximate matching of A, B to computing ε-approximate matchings on several subsets

$$\{(A'_1, B'_1), \ldots, (A'_k, B'_k)\}.$$

Here $\bigcup_{i=1}^{k} A'_i = A$, $\bigcup_{i=1}^{k} B'_i = B$ and each pair of subsets $A'_i, B'_i \subseteq [\Delta]^d$, $|A'_i| = |B'_i|$, are point sets from an integer grid with $\Delta \leq n^{O(1)}$. We describe a Monte-Carlo algorithm for the case when $A, B \subseteq [\Delta]^d$, leading to the following result.

THEOREM 2. *Let $A, B \subseteq [\Delta]^d$ be two point sets with $|A| = |B| = n$, and let $\varepsilon > 0$ be a parameter. For any L_p-norm, an ε-approximate matching of A and B can be computed with high probability in $O(n\text{poly}(\log \Delta, 1/\varepsilon))$ time.*

There are three key ingredients of our algorithm.

- We use a randomly-shifted quad-tree Q based distance function $\mathtt{d}(\cdot,\cdot)$ that ε-approximates the L_p-norm. A similar distance function was used in [14]. It therefore suffices to compute an ε-approximate matching under $\mathtt{d}(\cdot,\cdot)$ (Section 2).
- For a matching M, we partition the edges in M into certain classes. The endpoints of all the edges in a class lie sufficiently close to each other and all of them have the same $\mathtt{d}(\cdot,\cdot)$ distance; see Figure 2. Based on these classes, we call certain edges special and set the cost of a special edge (a, b) to $\mathtt{d}(a, b) + \theta$ where $\theta = \varepsilon \mathtt{w}(M^*)/6n$ and $\mathtt{w}(M^*)$ is the cost of an optimal matching under $\mathtt{d}(\cdot,\cdot)$. Doing so allows us to show a bound of $O(n/\varepsilon \log n)$ on the total length of the augmenting paths computed by the algorithm while keeping the cost of the matching within $(1+\varepsilon)\mathtt{w}(M^*)$. Gabow and Tarjan [5] used a similar idea where they add to every non-matching edge a cost of 1. Their scheme introduces an additive error of n while the total length of all the augmenting paths produced by their algorithm is $O(n \log n)$ if $\mathtt{w}(M^*) = O(n)$. (Sections 3, 4).
- Unlike previous algorithms, our algorithm does not explicitly maintain dual variables. Instead we design a data structure that maintains the current matching, produces "shortest" augmenting paths in an output sensitive manner, and updates the current matching quickly. (Section 5).

2. APPROXIMATING L_P-NORM

In this section we describe a distance function $\mathtt{d}(\cdot,\cdot)$ that approximates the L_p-norm. For $A, B \subseteq [\Delta]^d$, we choose integers $i_1, \ldots, i_d \in [\Delta]$ uniformly at random and set $G = [0, 2\Delta]^d - (i_1, \ldots, i_d)$. G is a randomly-shifted hypercube that contains both A and B. We build a quad-tree Q of depth $\delta = \log_2(2\Delta) = 1 + \log_2 \Delta$ on G[2] — the root of Q is associated with G itself and the squares (cells) associated with the children of a node are obtained by splitting the hypercube associated with that node into 2^d equal hypercubes. The nodes at depth i induce a grid G_i in which each cell has a side length $2^{\delta - i}$. We view Q as the sequence of grids $G_0, G_1, \ldots, G_\delta$; G_0 is a single cell which is G itself and G_δ is the finest grid[3] associated with the leaves of Q. For any pair $(a, b) \in A \times B$ we define $\mathtt{d}(a, b)$ as follows: For a parameter $\varepsilon > 0$, we set $\omega = \lceil \log_2(8d^2 \log_2 \Delta/\varepsilon^2) \rceil$ and $\Omega = 2^\omega$ Let $\square$ be the least common ancestor in Q of the leaves containing a and b. Suppose $\square \in G_{\delta - i}$. Let

$$\mathbb{G}[\square] = \square \cap G_{\delta - i + \omega}$$

be a $\Omega \times \Omega$ grid that divides $\square$ into *sub-cells* — sidelength of each sub-cell of $\square$ is $2^{i-\omega} = 2^i/\Omega$. Let $a_\square$ (resp. $b_\square$) be the center of sub-cell of $\square$ that contains a (resp. b). We set

$$\mathtt{d}(a, b) = ||a_\square b_\square||_p + d2^i/\Omega.$$

The following lemma proves that $\mathtt{d}(\cdot,\cdot)$ approximates the L_p-norm in the expected sense.

LEMMA 1. *For any pair $(a, b) \in A \times B$, $\mathtt{d}(a, b) \geq ||ab||_p$. Furthermore, $\mathbb{E}[\mathtt{d}(a, b)] \leq (1 + \varepsilon/2)||ab||_p$.*

PROOF. Let $\square$ be the least common ancestor of a and b. Suppose $\square \in G_{\delta - i}$. Let $\xi_1, \xi_2 \in \mathbb{G}[\square]$ be the sub-cells that contain a and b, respectively. Since the length of both ξ_1 and ξ_2 is $2^i/\Omega$, by triangle inequality, $||ab||_p \leq ||a_\square b_\square||_p + d.2^i/\Omega$. Hence, $\mathtt{d}(a, b) \geq ||ab||_p$. Moreover, $||a_\square b_\square||_p \leq ||ab||_p + d2^i/\Omega$, there-

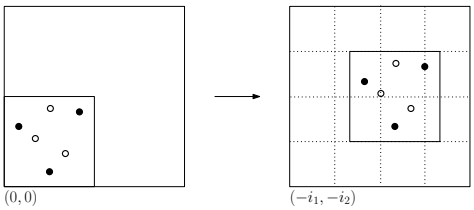

Figure 1. Randomly shifted quad-tree.

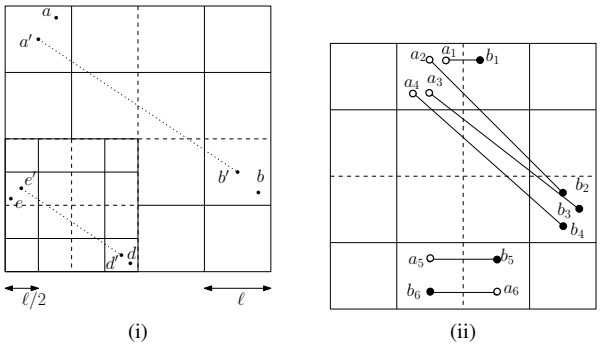

Figure 2. (i) Decomposition of a quad-tree cell and one of its children into sub-cells. a', b', d', e' are center points of sub-cells containing a, b, d, e respectively. $\mathtt{d}(a, b) = ||a'b'|| + 2\ell$, $\mathtt{d}(d, e) = ||d'e'|| + \ell$. (ii) $M = \{(a_1, b_1), (a_2, b_2), (a_3, b_3), (a_4, b_4), (a_5, b_5), (a_6, b_6)\}$, $\mathcal{K}_M = \{(\{a_1\}, \{b_1\}), (\{a_2, a_3, a_4\}, \{b_2, b_3, b_4\}), (\{a_5\}, \{b_5\}), (\{a_6\}, \{b_6\})\}$.

[2]We assume Δ is of the form 2^k for some positive integer k

[3]Although G_δ is the finest grid associated with the lowest level of Q, we can define G_i for $i > \delta$ in a straightforward manner

fore $\mathtt{d}(a,b) \leq ||ab||_p + d2^{i+1}/\Omega$ which implies,

$$\begin{aligned}\mathbb{E}[\mathtt{d}(a,b)] &\leq \sum_{i=1}^{\log\Delta+1} \Pr[\square \in G_{\delta-i}](||ab||_p + d2^{i+1}/\Omega) \\ &\leq ||ab||_p + \frac{d}{\Omega}\sum_{i=1}^{\log\Delta+1} 2^{i+1}\Pr[\square \in G_{\delta-i}].\end{aligned}$$

Since $\square$ is the least common ancestor of a and b, they lie in two different children of $\square$. Recall that $G_{\delta-i+1}$ is shifted uniformly at random, as argued by Arora [3], $\Pr[\square \in G_{\delta-i}] \leq ||ab||_1/2^{i-1} \leq d||ab||_p/2^{i-1}$. Therefore,

$$\begin{aligned}\mathbb{E}[\mathtt{d}(a,b)] &\leq ||ab||_p + \frac{4d^2}{\Omega}\sum_{i=0}^{\delta}||ab||_p \\ &= (1+4d^2\delta/\Omega)||ab||_p \\ &\leq (1+\varepsilon/2)||ab||_p;\end{aligned}$$

the last inequality follows from the fact that $\Omega \geq 8d^2\delta/\varepsilon$. $\square$

COROLLARY 1. *Let $A, B \subseteq [\Delta]^d$ and let $\varepsilon > 0$ be a parameter. Let M^* be an optimal matching of A, B under $\mathtt{d}(\cdot,\cdot)$ and let $\mathtt{w}_{\mathrm{OPT}}$ be the cost of the optimal matching under any L_p-norm. Then $\mathtt{w}(M^*) \leq (1+\varepsilon/2)\mathtt{w}_{\mathrm{OPT}}$.*

3. CLASSIFYING MATCHING EDGES

In this section, we describe how to cluster the edges of M into classes and prove a few useful properties of this classification. For a matching $M = \{(a_1,b_1),\ldots,(a_k,b_k)\}$ of A, B, a vertex is called *free* if it is not incident upon any edge of M, and *matched* otherwise. Let $A_{\mathrm{F}} \subseteq A$ and $B_{\mathrm{F}} \subseteq B$ be the set of free vertices. We classify the edges of M into equivalence classes as follows: Two edges in the matching $(a_l,b_l),(a_m,b_m) \in M$ belong to the same class if and only if: both (a_l,b_l) and (a_m,b_m) have the same least common ancestor $\square$ in Q, a_l, a_m are in the same sub-cell of $\mathbb{G}[\square]$, and b_l, b_m are in the same sub-cell of $\mathbb{G}[\square]$; see Figure 2. Let $\mathcal{K}_M = \{M_1,\ldots,M_r\}$ be the resulting partition of M into equivalence classes. For each M_i, let $A_i = \bigcup_{(a_j,b_j)\in M_i} a_j$ and $B_i = \bigcup_{(a_j,b_j)\in M_i} b_j$. $A \setminus A_F$ is partitioned into sets $A_1,\ldots,A_r$ and the set $B \setminus B_F$ is partitioned into sets $B_1,\ldots,B_r$ and $M_i \subseteq A_i \times B_i$. For an edge $(a,b) \in A \times B$, if there is an $i \leq r$ such that $(a,b) \in A_i \times B_i$, then we call (a,b) a *local* edge. All other edges, including the edges incident on points in A_F and B_F, are *non-local* edges. Note that all edges of M are local edges. All pairs of $A_i \times B_i$ are referred to as edges of *class i*.

We fix a parameter θ. We describe how to choose θ in Section 4. Given a matching M, we define a modified cost function $\Phi_M : A \times B \to \mathbb{R}_{\geq 0}$ called the *adjusted cost* of an edge,

$$\Phi_M(a,b) = \begin{cases} \mathtt{d}(a,b) & \text{if } (a,b) \text{ is a local edge,} \\ \mathtt{d}(a,b)+\theta & \text{otherwise.}\end{cases}$$

For a set of edges $E \subseteq A \times B$, we define

$$\Phi_M(E) = \sum_{(a,b)\in E} \Phi_M(a,b).$$

Therefore, $\Phi_M(M) = \mathtt{w}(M)$.

Given a matching M, an *alternating path* (or cycle) Π is a simple path (resp. cycle) whose edges are alternately in and not in M. We define the *net cost* of Π, $\phi_M(\Pi)$, as

$$\phi_M(\Pi) = \Phi_M(\Pi \setminus M) - \Phi_M(\Pi \cap M).$$

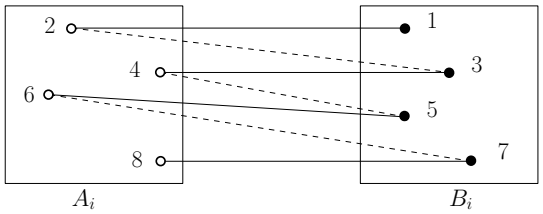

Figure 3. An alternating path consisting of local edges. Let $\pi_1 = \langle 1,2,3,4,5,6,7\rangle, \pi_2 = \langle 1,2,3,4,5\rangle, \pi_3 = \langle 1,2,3,4\rangle, \pi_4 = \langle 2,3,4,5\rangle$, then $\phi_M(\pi_1) = \phi_M(\pi_2) = 0$, $\phi_M(\pi_3) = -\mathtt{d}(1,2)$, $\phi_M(\pi_4) = \mathtt{d}(1,2)$.

An *augmenting path* Π is an alternating path between two free vertices. We *augment* M along Π, by updating M to $M \oplus \Pi$, i.e., remove the edges in $\Pi \cap M$ and add the edges in $\Pi \setminus M$. The following properties of local edges are relatively straightforward(See Figure 3):

(L1) For any $1 \leq i \leq r$, all local edges of class i have the same adjusted cost, say, φ_i.

(L2) Let Π be a path composed only of local edges, then all edges in Π belong to the same class.

(L3) Let $\Pi = \langle e_1, e_2, \ldots, e_k\rangle$ be an alternating path such that all edges in Π are local edges of class i. Then,

$$\phi_M(\Pi) = \begin{cases} \varphi_i & e_1, e_k \notin M, \\ -\varphi_i & e_1, e_k \in M, \\ 0 & \text{otherwise.}\end{cases}$$

(L4) Let $e = (a,b)$ be a local non-matching edge, say, of class i, then the matching edges incident on a and b also belong to the class i.

(L1) and (L2) follow from the definition of local edges, and (L3) follows from (L1) and the definition of net cost.

An alternating path Π is called *compact* if the total number of non-local edges of Π is at least $|\Pi|/4$, where $|\Pi|$ is the length, i.e., the number of edges in the alternating path.

LEMMA 2. *For a matching M and a compact augmenting path Π, let $M' = M \oplus \Pi$. Then*

$$\mathtt{w}(M') - \mathtt{w}(M) + \theta|\Pi| \geq \phi_M(\Pi) \geq \mathtt{w}(M') - \mathtt{w}(M) + \theta|\Pi|/4.$$

PROOF. Suppose Π has t non-local edges. Then,

$$\begin{aligned}\phi_M(\Pi) &= \Phi_M(\Pi \setminus M) - \Phi_M(\Pi \cap M) \\ &= \mathtt{w}(\Pi \setminus M) - \mathtt{w}(\Pi \cap M) + t\theta \\ &= \mathtt{w}(M') - \mathtt{w}(M) + t\theta. \qquad (1)\end{aligned}$$

Since Π is a compact path, $|\Pi| \geq t \geq |\Pi|/4$, the lemma follows from (1). $\square$

The following lemma shows that there is always an augmenting path of minimum net cost that is also a compact path.

LEMMA 3. *Given a matching M and an augmenting path Π, there is a compact augmenting path Π' with $\phi_M(\Pi) = \phi_M(\Pi')$.*

PROOF. Among all augmenting paths of the minimum net cost, let Π be an augmenting path with the fewest number of edges. Let $\pi = \langle e_1, \ldots e_k\rangle$ be the longest sub-path consisting only of local edges. Since π is maximal and all edges of M are local, $e_1, e_k \in$

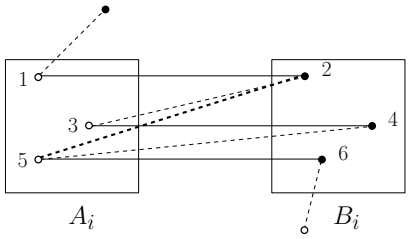

Figure 4. Shortcutting an alternating path — replace the path $\langle(2,3),(3,4),(4,5)\rangle$ with $(2,5)$.

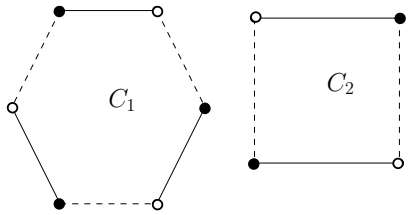

Figure 5. Proof of Lemma 4: Solid edges belong to $\mathbb{M}$ and dashed edges belong to M^*.

M. If $|\pi| \leq 3$, then Π is obviously compact, so assume that $|\pi| > 3$. By (L2), all edges of π belong to the same class, say i. Let $b_1, a_1, b_2, a_2, \ldots, b_k, a_k$ be the sequence of vertices in π, i.e., $e_1 = (b_1, a_1)$ and $e_k = (b_k, a_k)$. Set $\tilde{\pi} = e_1 \circ (a_1, b_k) \circ e_k$. By (L3), $\phi_M(\pi) = \phi_M(\tilde{\pi})$ but $|\pi| > |\tilde{\pi}|$. If we replaces π with $\tilde{\pi}$ in Π (See Figure 4), we obtain another alternating path of the same net cost but with fewer edges, a contradiction. Hence $|\pi| \leq 3$ and Π is compact. $\square$

4. ALGORITHM

Let $A, B \subseteq [\Delta]^d$ be two point sets of n points each, and let $\mathtt{d}(\cdot,\cdot)$ be the quad-tree based distance function defined in Section 2. We present an algorithm for computing an ϵ-approximate matching of A and B under $\mathtt{d}(\cdot,\cdot)$. Let

$$\varepsilon \mathtt{w}(M^*)/6n \leq \theta \leq \varepsilon \mathtt{w}(M^*)/3n.$$

Given a θ that satisfies this inequality, our algorithm produces an ε-approximate matching in $O(n\mathrm{poly}(\log\Delta, 1/\varepsilon))$ time. There are various ways of obtaining θ. For a constant c that depends on the norm and the dimension of the problem, the cost of an optimal matching is $1 \leq \mathtt{w}(M^*) \leq cn\Delta$. For some sufficiently large constant c_2, we execute $c_2 n\mathrm{poly}(\log\Delta, 1/\varepsilon)$ steps of our algorithm by setting $\theta = \theta_i = \varepsilon 2^i/6n$ for $i \in [1, \ldots, \lceil \log_2 cn\Delta \rceil]$. Of all the perfect matchings produced by the $O(\log n\Delta)$ executions of our algorithm, we return the matching with the smallest cost. Since at least one of our guesses of θ is correct, we obtain an ε-approximate matching. In the rest of the paper, we work under the assumption that we have the desired θ.

Our algorithm relies on a data structure, described in Section 5, that stores A,B and a matching M and that supports the following two operations:

- FINDAP(): return a compact augmenting path Π of the smallest net cost.
- AUGMENT(Π): augment M along Π, i.e., update the matching to $M \oplus \Pi$.

Using the data structure we compute an ε-approximate matching $\mathbb{M}$ as follows: The algorithm maintains a matching M and iteratively repeats the following until $M = \mathbb{M}$ is perfect; initially $M = \emptyset$. The algorithm first finds an augmenting path Π using FINDAP(). Next, it augments M along Π using the AUGMENT(Π) procedure. The algorithm maintains the following invariant, which we call the *alternating cycle invariant*.

(ACI) For any alternating cycle C, $\phi_M(C) \geq 0$.

We first bound the cost of $\mathbb{M}$ assuming ACI, then analyze the running time, and finally prove ACI.

LEMMA 4. *Let M^* be an optimal matching of A, B under $\mathtt{d}(\cdot,\cdot)$. Then, $\mathtt{w}(\mathbb{M}) \leq (1+\varepsilon/3)\mathtt{w}(M^*)$.*

PROOF. $\mathbb{M} \cup M^*$ is a set of vertex disjoint alternating cycles $C_1, \ldots, C_k$(See Figure 5). For each i, $C_i \setminus \mathbb{M} = C_i \cap M^*$. Therefore,

$$\mathtt{w}(M^*) - \mathtt{w}(\mathbb{M}) = \sum_{i=1}^{k} (\mathtt{w}(C_i \setminus \mathbb{M}) - \mathtt{w}(C_i \cap \mathbb{M})). \qquad (2)$$

On the other hand, by ACI, $\phi_{\mathbb{M}}(C_i) \geq 0$ for every $0 \leq i \leq k$, implying that

$$\sum_{i=1}^{k} \phi_{\mathbb{M}}(C_i) = \sum_{i=1}^{k} (\Phi_{\mathbb{M}}(C_i \setminus \mathbb{M}) - \Phi_{\mathbb{M}}(C_i \cap \mathbb{M})) \geq 0.$$

For every edge $(a,b) \in A \times B$, $\mathtt{d}(a,b) \leq \Phi_{\mathbb{M}}(a,b) \leq \mathtt{d}(a,b) + \theta$. Therefore,

$$\sum_{i=1}^{k} (\mathtt{w}(C_i \setminus \mathbb{M}) - \mathtt{w}(C_i \cap \mathbb{M})) + n\theta \geq 0. \qquad (3)$$

Combining (2) and (3) and using the fact that $\theta \leq \varepsilon \mathtt{w}(M^*)/3n$, we obtain

$$\mathtt{w}(\mathbb{M}) \leq \mathtt{w}(M^*) + n\theta \leq (1+\varepsilon/3)\mathtt{w}(M^*). \quad \square$$

Next, we bound the running time of the algorithm. Let M_i be the matching after first i steps of the algorithm, and let Π_i be the augmenting path computed in the i^{th} step of the algorithm. $M_0 = \emptyset$, $M_{i+1} = M_i \oplus \Pi_i$, and $M_n = \mathbb{M}$. We show in Section 5 that FINDAP and AUGMENT in step i take $O(|\Pi_i|\mathrm{poly}(\log\Delta, 1/\epsilon))$ time. We prove in the next lemma that $\sum_{i\geq 1} |\Pi_i| = O(n/\varepsilon \log n)$, which implies that the algorithm takes $O(n\mathrm{poly}(\log\Delta, 1/\varepsilon))$ time.

LEMMA 5. $\sum_{i=1}^{n} |\Pi_i| = O((n/\varepsilon)\log n)$.

PROOF. Fix a value of $i \leq n$. $M^* \cup M_i$ consists of a set of alternating cycles and $n-i+1$ augmenting paths $P_1, \ldots, P_{n-i+1}$. By definition of ϕ and Φ and the value of θ

$$\begin{aligned}
\sum_{j=1}^{n-i+1} \phi_{M_{i-1}}(P_j) &\leq \sum_{j=1}^{n-i+1} \Phi_{M_{i-1}}(P_j \setminus M_{i-1}) \\
&\leq \sum_{j=1}^{n-i+1} \mathtt{w}(P_j \setminus M_{i-1}) + n\theta \\
&= \sum_{j=1}^{n-i+1} \mathtt{w}(P_j \cap M^*) + n\theta \\
&\leq (1+\varepsilon/3)\mathtt{w}(M^*).
\end{aligned}$$

Since Π_i is an augmenting path of minimum net cost, $\phi_{M_{i-1}}(\Pi_i) \leq \phi_{M_{i-1}}(P_j)$ for each $j \leq n-i+1$. Therefore,

$$\phi_{M_{i-1}}(\Pi_i) \leq (1+\varepsilon/3)\mathtt{w}(M^*)/(n-i+1),$$

or

$$\sum_{i=1}^{n} \phi_{M_{i-1}}(\Pi_i) \leq (1+\varepsilon/3)\mathtt{w}(M^*)\ln n. \tag{4}$$

Since Π_i is a compact path, by Lemma 2, $\phi_{M_{i-1}}(\Pi_i) \geq \mathtt{w}(M_i) - \mathtt{w}(M_{i-1}) + \theta|\Pi_i|/4$, which implies that

$$\begin{aligned}\sum_{i=1}^{n} \phi_{M_{i-1}}(\Pi_i) &\geq \mathtt{w}(\mathbb{M}) - \mathtt{w}(M_0) + \theta\sum_{i=1}^{n}|\Pi_i|/4 \\ &\geq \mathtt{w}(M^*) + \frac{\varepsilon\mathtt{w}(M^*)}{24n}\cdot\sum_{i=1}^{n}|\Pi_i|. \end{aligned}\tag{5}$$

The lemma follows by combining (4) and (5). □

Proof of ACI. We prove ACI by induction on i. Since $M_0 = \emptyset$, ACI is true for $i = 0$. Suppose ACI holds for all $j < i$. Let us consider the i^{th} step. Let $\Pi_i = \langle p_1, \ldots, p_u\rangle$ where $p_1 \in B$ and $p_u \in A$, be the augmenting path computed by the algorithm in step i. If there is a negative alternating cycle with respect to M_i, then let C be a negative alternating cycle with the *fewest number* of edges.

We call an edge *affected* by Π_i if at least one of its endpoints is a vertex of Π_i. Note that for any *unaffected* edge (p,q), $\Phi_{M_i}(p,q) = \Phi_{M_{i-1}}(p,q)$. Therefore, if C does not contain any affected edge, then C is an alternating cycle after step i and $\phi_{M_{i-1}}(C) = \phi_{M_i}(C)$. By induction hypothesis, C is non-negative. Hence, C must contain an affected edge. Since C is a cycle, i.e., $C \not\subseteq \Pi_i$, it contains an affected edge that does not lie in Π_i. Let (p,q) be such an affected edge, and suppose $p \in \Pi_i$. Since one of the edges of Π_i incident on p. say, (p,q') belongs to M_i, we can conclude $(p,q) \notin M_i$. C being an alternating cycle and (p,q') being the only matching edge incident on p imply that $(p,q') \in C$. Hence, C contains at least one edge of $\Pi_i \cap M_i$.

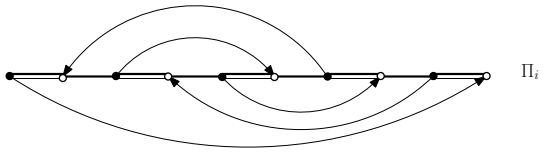

Figure 6. Illustration of forward and backward components in an alternating cycle C; circular arcs are components of $C \setminus \Pi_i$. Black (resp. white) circles are points of B (resp. A). Components oriented toward right (resp. left) are forward (resp. backward) components; double-line segments are components of $\Pi_i \cap C$.

Let $\mathcal{C}^0$ be the set of connected components of $\Pi_i \cap C$, each of which is an alternating path; the above discussion implies that the first edge and the last edge of each connected component are edges of M_i. Each connected component π of $C \setminus \Pi_i$ is an alternating path whose first and last edges do not belong to M_i. Let $p_r \in B$ and $p_s \in A$ be the endpoints of π. We use $\tilde{\pi}$ to denote the sub-path of Π_i between p_r and p_s. We call π a *forward* (resp. *backward*) component if $r < s$ (resp. $r > s$)(See Figure 6). Let $\mathcal{C}^+$ (resp. $\mathcal{C}^-$) denote the set of forward (resp. backward) components of $C \setminus \Pi_i$. We call an affected edge (p,q) of $C \setminus \Pi_i$ *reducing* if $\Phi_{M_i}(p,q) < \Phi_{M_{i-1}}(p,q)$. If (p,q) is a reducing edge, then it is a non-local edge with respect to M_{i-1} and local edge with respect to M_i. Let κ^+ (resp. κ^-) be the total number of reducing edges in forward (resp. backward) components and let κ be the number of edges in Π_i that were non-local edges with respect to M_{i-1}. We claim that

(C1) $\displaystyle\sum_{\pi\in\mathcal{C}^+} \phi_{M_i}(\pi) + \kappa^+\theta \geq \sum_{\pi\in\mathcal{C}^+} \phi_{M_{i-1}}(\tilde{\pi}).$

(C2) $\displaystyle\sum_{\pi\in\mathcal{C}^-} \phi_{M_i}(\pi) + \kappa^-\theta \geq -\sum_{\pi\in\mathcal{C}^-} \phi_{M_{i-1}}(\tilde{\pi}).$

(C3) $\displaystyle\sum_{\pi\in\mathcal{C}^0} \phi_{M_i}(\pi) \geq -\sum_{\pi\in\mathcal{C}^0} \phi_{M_{i-1}}(\pi) + \kappa\theta.$

(C4) $\kappa \geq \kappa^- + \kappa^+.$

Before proving (C1)–(C4), we show that they imply $\phi_{M_i}(C) \geq 0$, which contradicts the assumption that $\phi_{M_i}(C) \leq 0$. Indeed, C being a cycle implies if an edge of $(a,b) \in \Pi_i$ appears in $\tilde{\pi}$'s of j different forward components, then (a,b) has to appear j times in paths $\tilde{\pi}$ of backward components and $\mathcal{C}^0$ put together. Hence, adding equations (C1), (C2) and (C3), we get

$$\sum_{\pi\in\mathcal{C}^+} \phi_{M_i}(\pi) + \sum_{\pi\in\mathcal{C}^-} \phi_{M_i}(\pi) + \sum_{\pi\in\mathcal{C}^0} \phi_{M_i}(\pi) + (\kappa^- + \kappa^+ - \kappa)\theta \geq 0.$$

Since

$$\phi_{M_i}(C) = \sum_{\pi\in\mathcal{C}^+} \phi_{M_i}(\pi) + \sum_{\pi\in\mathcal{C}^-} \phi_{M_i}(\pi) + \sum_{\pi\in\mathcal{C}^0} \phi_{M_i}(\pi)$$

and $\kappa \geq \kappa^- + \kappa^+$, we get $\phi_{M_i}(C) \geq 0$. We now prove (C1)–(C4).

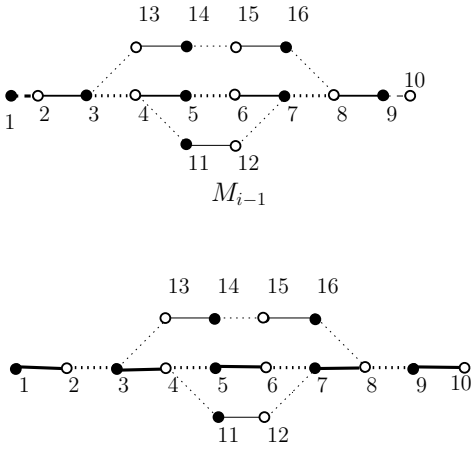

Figure 7. $\Pi_i = \langle 1,2,3,4,5,6,7,8,9,10\rangle$, $C = \langle 3,13,14,15,16,8,7,12,11,4,3\rangle$ is an alternating cycle with respect to M_i; C decomposes into $\pi_1 = \langle 3,13,14,15,16,8\rangle$, $\pi_2 = \langle 4,11,12,7\rangle$ and $\Pi_i \cap C = \langle 3,4\rangle, \langle 7,8\rangle$; the forward component consists of $\mathcal{C}^+ = \{\pi_1\}$ and the backward component consists of $\mathcal{C}^- = \{\pi_2\}$; $\tilde{\pi}_1 = \langle 3,4,5,6,7,8\rangle$, $\tilde{\pi}_2 = \langle 4,5,6,7\rangle$.

Proof of (C1): Let $\pi \in \mathcal{C}^+$ be a forward component. We note that $\overline{\Pi} = (\Pi_i \setminus \tilde{\pi}) \cup \pi$ is an augmenting path with respect to M_{i-1}; see Figure 7. Since Π_i is the smallest augmenting path (with respect to M_{i-1}) $\phi_{M_{i-1}}(\overline{\Pi}) \geq \phi_{M_{i-1}}(\Pi_i)$. This implies $\phi_{M_{i-1}}(\pi) \geq \phi_{M_{i-1}}(\tilde{\pi})$ or

$$\sum_{\pi\in\mathcal{C}^+} \phi_{M_{i-1}}(\pi) \geq \sum_{\pi\in\mathcal{C}^+} \phi_{M_{i-1}}(\tilde{\pi}). \tag{6}$$

Since the total number of reducing edges in forward components is κ^+ and since all other affected edges increase the adjusted cost, we have $\sum_{\pi\in\mathcal{C}^+}\phi_{M_i}(\pi)+\kappa^+\theta \geq \sum_{\pi\in\mathcal{C}^+}\phi_{M_{i-1}}(\pi)$. Combining this with (6), we obtain (C1).

Proof of (C2): For any backward component $\pi\in\mathcal{C}^-$, $\tilde{\pi}\cup\pi$ is an alternating cycle C w.r.t. M_{i-1}; see Figure 7. Since $\phi_{M_{i-1}}(C)\geq 0$, we have $\phi_{M_{i-1}}(\pi)+\phi_{M_{i-1}}(\tilde{\pi})\geq 0$ or

$$\sum_{\pi\in\mathcal{C}^-}\phi_{M_{i-1}}(\pi)+\sum_{\pi\in\mathcal{C}^-}\phi_{M_{i-1}}(\tilde{\pi})\geq 0. \tag{7}$$

There are κ^- reducing edges in $\mathcal{C}^-$, therefore

$$\sum_{\pi\in\mathcal{C}^-}\phi_{M_i}(\pi)+\kappa^-\theta\geq\sum_{\pi\in\mathcal{C}^-}\phi_{M_{i-1}}(\pi).$$

Plugging this into (7), we get (C2).

Proof of (C3): Let κ be the total number of non-local edges in paths of $\mathcal{C}^0$,

$$\begin{aligned}\sum_{\pi\in\mathcal{C}^0}\phi_{M_{i-1}}(\pi) &= \sum_{\pi\in\mathcal{C}^0}\Phi_{M_{i-1}}(\pi\setminus M_{i-1})-\Phi_{M_{i-1}}(\pi\cap M_{i-1})\\ &= \sum_{\pi\in\mathcal{C}^0}\mathtt{w}(\pi\setminus M_{i-1})-\mathtt{w}(\pi\cap M_{i-1})+\kappa\theta.\end{aligned}$$

Therefore

$$\begin{aligned}-\sum_{\pi\in\mathcal{P}}\phi_{M_{i-1}}(\pi)+\kappa\theta &= \sum_{\pi\in\mathcal{C}^0}\mathtt{w}(\pi\cap M_{i-1})-\mathtt{w}(\pi\setminus M_{i-1})\\ &= \sum_{\pi\in\mathcal{C}^0}\mathtt{w}(\pi\setminus M_i)-\mathtt{w}(\pi\cap M_i)\\ &\leq \sum_{\pi\in\mathcal{P}}\phi_{M_i}(\pi).\end{aligned}$$

Proof of (C4): Since C is a negative cycle with smallest length and since there are $\kappa^-+\kappa^+$ reducing edges of $\mathcal{C}^+$ and $\mathcal{C}^-$ that are incident on $C\cap\mathcal{C}^0$, from Lemma 6 below, it follows that there are at least $\kappa^-+\kappa^+$ non-local edges in $C\cap\mathcal{C}^0$. Hence, $\kappa\geq\kappa^++\kappa^-$.

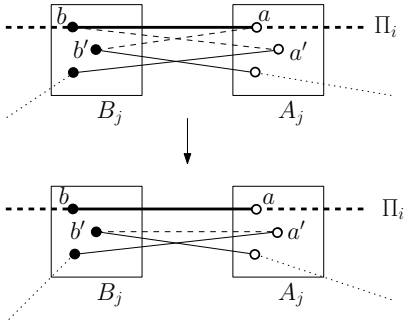

Figure 8. Shortcutting C: replacing $\langle b',a,b,a'\rangle$ with $\langle b',a'\rangle$.

LEMMA 6. *Each edge $e\in M_i\cap\Pi_i\cap C$ is adjacent to at most one reducing edge of C. Furthermore if e is adjacent to a reducing edge, then e was non-local w.r.t. M_{i-1}.*

PROOF. Suppose $e=(a,b)$ is adjacent to two reducing edges (b',a) and (b,a'); see Figure 8. The above discussion implies that both of them are local w.r.t. M_i and neither of them is in M_i or M_{i-1}. By (L2) and (L4), all three of them belong to the same class of $\mathcal{K}_{M_i}$, say, j, and the edge (a',b') also belongs to the class j. If $(a',b')\in M_i$, then $C=\langle a',b',a,b,a'\rangle$ and $\phi_{M_i}=0$, contradicting the assumption that $\phi_{M_i}(C)<0$. Hence $(a',b')\notin M_i$. Let $\pi=(b',a)\circ(a,b)\circ(b,a')$ and $\tilde{\pi}=(b',a')$. By (L3), $\phi_{M_i}(\pi)=\phi_{M_i}(\tilde{\pi})$. If we replace π with $\tilde{\pi}$ in C, we obtain another alternating cycle of the same net cost but with fewer edges, a contradiction; see Figure 8. Hence, at most one of (b',a) and (a',b) is reducing.

Next, suppose (b',a) is reducing. If a was not matched in M_{i-1}, then (a,b) was obviously non-local w.r.t.M_{i-1}, so assume that Π_i has another edge $(a,b'')\in M_{i-1}$; see Figure 9. Let (b',a'') be the other edge of C incident on b', then $(b',a'')\in(M_{i-1}\cap M_i)\setminus\Pi_i$. Since (b',a) is reducing, it is local edge w.r.t.M_i. By (L4), see Figure 9, (b',a'') and (b,a) belong to the same class as (b',a) (w.r.t.M_i). On the other hand, (b',a) being reducing also implies that it is non-local w.r.t. M_{i-1}. By (L4), we can conclude that $(b',a''),(a,b'')\in M_{i-1}$ do not belong to the same class of $\mathcal{K}_{M_{i-1}}$; see Figure 9. Using the condition of two matching edges lying in the same class, we can conclude that (a,b) and (a,b'') also do not belong to the same class w.r.t. M_{i-1}, implying that (a,b) is non-local w.r.t. M_{i-1}. □

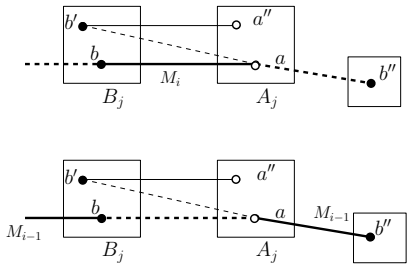

Figure 9. (b',a) is non-local w.r.t. M_{i-1}; b,b' lie in the same sub-cell and b'' lies in a different sub-cell.

5. DATA STRUCTURE

In this section we describe a data structure based on the quad-tree Q constructed in Section 2 that, given point sets A,B and a matching M for which ACI holds, supports FINDAP and AUGMENT operations. We describe a weighted directed graph $\vec{\mathrm{G}}_M(A,B)$ so that the problem of finding a compact augmenting path of minimum net cost (using FINDAP) corresponds to finding an "optimal" path between free vertices in this directed graph. Guided by the partition of the matching edges into classes (Section 3), we describe a hierarchical representation of $\vec{\mathrm{G}}_M$ that allows us to compute an optimal path in $\vec{\mathrm{G}}_M$ efficiently. We describe the information stored at each node of Q and two auxiliary procedures that compute optimal paths. Finally, we describe how FINDAP and AUGMENT are implemented using these auxiliary procedures.

Directed Graph and its properties. For A, B and a matching M, let $\vec{\mathrm{G}}_M(A,B)$ be the weighted graph with the vertex set $A\cup B$ and the edge set $A\times B$. For $(a,b)\in A\times B$, if (a,b) is a non-local edge then (a,b) is directed from b to a with a cost $\mu_M(a,b)=\Phi_M(a,b)$, otherwise, i.e., it is a local edge, there is an edge (a,b) directed from a to b with $\mu_M(a,b)=-\Phi_M(a,b)$. If M is obvious from the context we use $\vec{\mathrm{G}}$ to denote $\vec{\mathrm{G}}_M$ and $\mu(\cdot,\cdot)$ to denote $\mu_M(\cdot,\cdot)$. For any directed path $\vec{\Pi}$ in $\vec{\mathrm{G}}(A,B)$, let the cost of $\vec{\Pi}$, $\mu(\vec{\Pi})$, be the sum of cost of all the edges of $\vec{\Pi}$. Each free vertex of A is a *sink* in $\vec{\mathrm{G}}$, each free vertex of B is a *source* in $\vec{\mathrm{G}}$, and these are the only source and sink vertices in $\vec{\mathrm{G}}$. A path in $\vec{\mathrm{G}}$ alternates

between local and non-local edges, but a local edge may not be a matching edge. The following two lemmas relate alternating paths Π with directed paths $\vec{\Pi}$ in $\vec{\mathrm{G}}$.

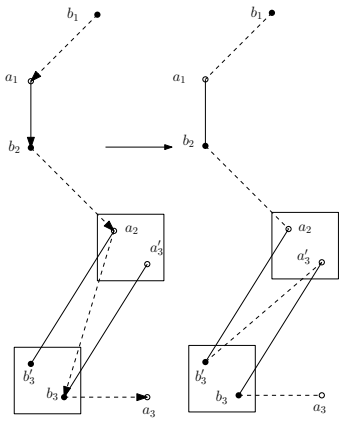

Figure 10. Transforming a directed path in $\vec{\mathrm{G}}$ to an alternating path.

LEMMA 7. *Let M be a matching that satisfies* ACI*, and let $b \in B$ and $a \in A$ be two free vertices w.r.t. M.*

(i) A (directed) path $\vec{\Pi}$ in $\vec{\mathrm{G}}_M(A, B)$ from b to a can be transformed into a compact augmenting path Π between a and b such that $\mu(\vec{\Pi}) = \phi_M(\Pi)$.

(ii) An augmenting path Π can be transformed into a (directed) path $\vec{\Pi}$ in $\vec{\mathrm{G}}(A, B)$ such that $\mu(\vec{\Pi}) \leq \phi_M(\Pi)$.

PROOF. (i) Let $\vec{\Pi} = \langle b = b_1, a_1, \ldots, b_k, a_k = a\rangle$, which alternates between non-local and local edges — each (b_i, a_i) is a non-local edge and each (a_i, b_{i+1}) is a local edge. We transform $\vec{\Pi}$ into Π by processing each local edge (a_i, b_{i+1}) as follows (see Figure 10): if $(a_i, b_{i+1}) \in M$, we keep it in Π. Otherwise, by (L4), there are two matching edges (a_i, b'_i) and (a'_{i+1}, b_{i+1}) that belong to the same class as (a_i, b_{i+1}). Set $\pi_i = (a_i, b'_i) \circ (b'_i, a'_{i+1}) \circ (a'_{i+1}, b_{i+1})$. We replace the edge (a_i, b_{i+1}) with π_i. By construction, the resulting path Π is an augmenting path. By (L3), $\phi_M(\pi_i) = -\Phi_M(a_i, b_{i+1}) = \mu(a_i, b_{i+1})$. Hence, $\phi_M(\Pi) = \mu(\vec{\Pi})$. Finally, since $|\Pi| \leq 4k$ and Π has k non-local edges, it is compact.

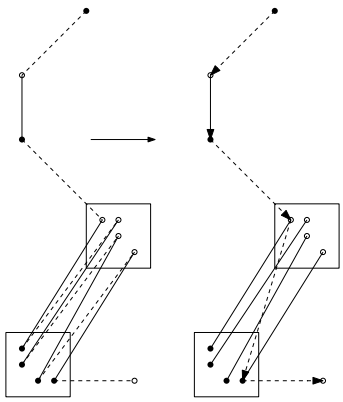

Figure 11. Transforming an alternating path to a directed path in $\vec{\mathrm{G}}$.

(ii) If the augmenting path $\Pi = \langle b = b_1, a_1, \ldots, b_k, a_k = a\rangle$ between b and a does not contain any non-matching local edge, then by the construction of $\vec{\mathrm{G}}$, Π is also a directed path in $\vec{\mathrm{G}}_M(A, B)$ and $\mu(\vec{\Pi}) = \phi_M(\Pi)$. So assume that Π has a non-matching local edge e. Suppose it belongs to class j. Let π be the maximally connected sub-path of Π composed of local edges that contains e. By (L2), all edges of π are in class j. Let $e_r = (a_r, b_r)$ and $e_l = (a_l, b_l), r < l$ be the first and last edges of π. We replace π by a single edge (a_r, b_l) in Π. Since π is maximal and all the edges of π are local, $e_r, e_l \in M$. Therefore (L3) implies that $\phi_M(\pi) = \mu(a_r, b_l)$. Repeating this for all the remaining non-matching local edges of Π, we obtain a path $\vec{\Pi}$ which alternates between non-local and local edges and whose cost $\mu(\vec{\Pi}) = \phi_M(\Pi)$ as desired. □

An immediate corollary of the argument in the proof of Lemma 7 is the following:

COROLLARY 2. *Assuming M satisfies* ACI*, $\vec{\mathrm{G}}_M(A, B)$ does not contain any negative cycle.*

Lemma 7 implies that the problem of computing a compact augmenting path of minimum net cost reduces to finding a minimum-cost directed path from a source to a sink in $\vec{\mathrm{G}}$.

Hierarchical clustering of points. Next, we present a hierarchical clustering of the vertices of $\vec{\mathrm{G}}$ that allows us to compute such a minimum-cost directed path efficiently. For any cell $\square$ of Q, the quad-tree constructed in Section 2, we cluster the vertices of $\vec{\mathrm{G}}$ that lie in $\square$ into $O(\mathrm{poly}(\log \Delta, 1/\varepsilon))$ clusters. Rougly speaking, if two points p, q belong to the same cluster at $\square$, then for any point $r \in (A \cup B) \setminus \square$, $\mu(p, r) = \mu(q, r)$ and (p, r) and (q, r) have identical directions w.r.t.r, i.e., both edges are either directed toward r or away from r. Let $M = \{(a_1, b_1), \ldots, (a_k, b_k)\}$ be the matching, and let

$$\mathcal{K}_M = \{(A_1, B_1), \ldots, (A_u, B_u)\}$$

be the classification of points in A and B induced by M as described in Section 3. For any matched point a_i (resp. b_i), we refer to b_i (resp. a_i) as its *partner* point. Let $\square$ be a non-empty cell of Q, and let $A_\square = A \cap \square$, $B_\square = B \cap \square$. We partition $A_\square$ and $B_\square$ into three types of clusters. For any $\xi \in \mathbb{G}[\square]$, let $A_\xi = A \cap \xi$ and $B_\xi = B \cap \xi$.

- *Free clusters*: All free points of A_ξ and B_ξ belong to a single cluster
$$A^F_\xi = A_F \cap \xi, \quad B^F_\xi = B_F \cap \xi.$$
- *Internal clusters*: All points of A_ξ (B_ξ) whose partner point is also inside $\square$ belong to the same cluster
$$A^I_\xi = \{a_i \in A_\xi \mid (a_i, b_i) \in M, b_i \in B_\square\},$$
$$B^I_\xi = \{b_i \in B_\xi \mid (a_i, b_i) \in M, a_i \in A_\square\}.$$
- *Boundary clusters*: All points of A_ξ whose partner points lie outside $\square$ are partitioned into various clusters. Two such points belong to the same cluster if and only if they lie in the same class $(A_i, B_i) \in \mathcal{K}_M$. More precisely, let $N^*(\square) = \bigcup \mathbb{G}[\square']$ where the union is taken over all siblings $\square'$ of ancestors of $\square$. $|N^*(\square)| = \mathrm{poly}(\log \Delta, 1/\varepsilon)$, as there are $O(\log \Delta)$ ancestors of $\square$, each has three siblings, and each

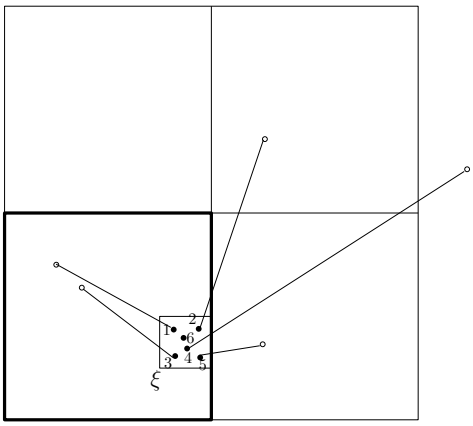

Figure 12. Partitioning of B_ξ into various clusters: $\{6\}$ is a free cluster, $\{1,3\}$ is an internal cluster, and $\{2\},\{4\},\{5\}$ are boundary clusters of the sub-cell ξ of the highlighted cell.

sibling has $\text{poly}(\log\Delta, 1/\varepsilon)$ sub-cells. For every sub-cell $\eta \in N^*(\square)$, we have a cluster

$$A^\eta_\xi = \{a_i \in A_\xi \mid (a_i,b_i) \in M, b_i \in B_\eta\},$$

$$B^\eta_\xi = \{b_i \in B_\xi \mid (a_i,b_i) \in M, a_i \in A_\eta\}.$$

The points in each cluster at $\square$ lie within a single sub-cell of $\mathbb{G}[\square]$. Let $\mathbb{X}_\square$ be the set of clusters at $\square$; $|\mathbb{X}_\square| = \text{poly}(\log\Delta, 1/\varepsilon)$. By definition, each free (resp. internal) cluster at $\square$ is contained in a free (resp. internal) cluster at $\mathtt{p}(\square)$, parent of $\square$, and each boundary cluster at $\square$ is contained in a boundary or an internal cluster at $\mathtt{p}(\square)$.

The relationship between the clusters at $\square$ and those at the children $\square_1, \square_2, \square_3, \square_4$ of $\square$ can be summarized as follows. Each sub-cell $\xi \in \mathbb{G}[\square]$ is contained in one of the four children of $\square$, say $\square_i$. Then ξ has four children $\xi_1, \xi_2, \xi_3, \xi_4$ in $\mathbb{G}[\square_i]$ and we get the following relationship.

$$\begin{aligned}
A^F_\xi &= \bigcup_{i=1}^4 A^F_{\xi_i}, \qquad B^F_\xi = \bigcup_{i=1}^4 B^F_{\xi_i},\\
A^\eta_\xi &= \bigcup_{i=1}^4 A^\eta_{\xi_i}, \qquad B^\eta_\xi = \bigcup_{i=1}^4 B^\eta_{\xi_i}.\\
A^I_\xi &= \bigcup_{i=1}^4 (A^I_{\xi_i} \cup (\bigcup_{\zeta\in N(\xi)} A^\zeta_{\xi_i})),\\
B^I_\xi &= \bigcup_{i=1}^4 (B^I_{\xi_i} \cup (\bigcup_{\zeta\in N(\xi)} B^\zeta_{\xi_i})).
\end{aligned} \tag{8}$$

For $z \in \square_i$, $N(z) = \bigcup_{j\neq i} \mathbb{G}[\square_j]$ is the set of sub-cells that lie inside the siblings of $\square_i$. For a cluster X of $A_\square$ (resp. $B_\square$), we define $D(X)$ as the set of clusters of $A_{\square_1}, A_{\square_2}, A_{\square_3}, A_{\square_4}$ (resp. $B_{\square_1}, B_{\square_2}, B_{\square_3}, B_{\square_4}$) that are contained in X. The following lemma whose proof is straightforward states the property of the above hierarchical clustering scheme

LEMMA 8. *For any cell $\square \in Q$, let $\square_1, \square_2$ be any two of its children. Let $X \in \mathbb{X}_{\square_1}, Y \in \mathbb{X}_{\square_2}$. Then the cost of all edges in $X \times Y$ is the same in $\vec{\mathrm{G}}$ and all edges are oriented in the same direction — either all are oriented from B to A or all of them are oriented from A to B.*

Next, we partition the clusters in $\mathbb{X}_\square$ into two subsets called the *entry* and *exit* clusters respectively,

$$\mathbb{X}^\downarrow_\square = \{B^F_\xi, A^I_\xi, B^\eta_\xi \mid \xi \in \mathbb{G}[\square], \eta \in N^*(\square)\},$$
$$\mathbb{X}^\uparrow_\square = \{A^F_\xi, B^I_\xi, A^\eta_\xi \mid \xi \in \mathbb{G}[\square], \eta \in N^*(\square)\}.$$

Let Π be a path in $\vec{\mathrm{G}}$ from a source to a sink vertex and let $\overline{\pi}$ be a maximal connected sub-path of $\vec{\Pi}$ that lies inside $\square$. Suppose $\overline{\pi}$ contains at least one edge. For the two endpoints p, q of $\overline{\pi}$, we refer to p as entry and q as exit point if $\overline{\pi}$ is directed from p to q. Then we claim that the entry point lies in an entry cluster and exit point lies in an exit cluster. Indeed, if $\overline{\pi}$ is an initial portion of $\vec{\Pi}$, then the entry point is a source vertex, belonging to some B^F_ξ – an entry cluster. Otherwise, if the first edge of $\overline{\pi}$ is a local edge, which lies inside $\square$ by assumption, then, by construction, the entry point belongs to an internal cluster. Since all local edges are oriented from A to B, the entry point must belong to A^I_ξ – an entry cluster. Finally, if the first edge of $\overline{\pi}$ is a non-local edge then the preceding edge in $\vec{\Pi}$ is a local edge. By construction, the entry point belongs to a boundary cluster. Since all non-local edges are directed from B to A, the entry point must belong to B^η_ξ, for some $\xi \in \mathbb{G}[\square], \eta \in N^*(\square)$ – an entry cluster. A similar argument holds for the exit point.

Information stored at each node. For a cell $\square \in Q$, let $\vec{\mathrm{G}}_\square = \vec{\mathrm{G}}_M(A_\square, B_\square)$ be the subgraph of $\vec{\mathrm{G}}(A,B)$ formed by the points $A_\square$, $B_\square$; $\vec{\mathrm{G}}_\square$ contains all directed paths of $\vec{\mathrm{G}}$ that lie completely within $\square$. We define the *mixed cost* of a path Π in $\vec{\mathrm{G}}_\square$, denoted by $\psi(\Pi)$, as $(\mu(\Pi), |\Pi|)$; $\mu(\Pi)$ is the cost of Π and $|\Pi|$ is the number of edges in Π. We order the mixed costs in lexicographic order, i.e., for two paths, Π_1 and Π_2 with $\psi(\Pi_1) = (\mu_1, \ell_1)$ and $\psi(\Pi_2) = (\mu_2, \ell_2)$, Π_1 has smaller mixed cost than Π_2 if $\mu_1 < \mu_2$ or $\mu_1 = \mu_2$ and $\ell_1 < \ell_2$. If $\Pi = \Pi_1 \circ \Pi_2$, then $\psi(\Pi) = (\mu_1 + \mu_2, \ell_1 + \ell_2)$. For a pair of points $a, b \in A_\square \cup B_\square$, let $\psi_\square(a,b)$ be the smallest mixed cost path from a to b in $\vec{\mathrm{G}}_\square$, and we refer to such a path as an *optimal* path from a to b. The definition of mixed cost ensures that an optimal path is always simple.

For any pair of subsets $X \in A_\square, Y \in B_\square$, let

$$\psi_\square(X,Y) = \min_{a\in X, b\in Y} \psi_\square(a,b).$$

At each cell $\square$ in Q, for all $X, Y \in \mathbb{X}^\downarrow_\square \times \mathbb{X}^\uparrow_\square$, we store

$$\psi_\square(X,Y), \tag{9}$$

i.e., the mixed cost of the optimal path between every entry cluster to every exit clusters.

Compressed graph. We show that if we have (9) at the four children of a cell $\square \in Q$, we can compute (9) for $\square$ in $\text{poly}(\log\Delta, 1/\varepsilon)$ time, and that given $(X,Y) \in \mathbb{X}^\downarrow_\square \times \mathbb{X}^\uparrow_\square$, we can compute an optimal path from a vertex $a \in X$ to $b \in Y$ in $\vec{\mathrm{G}}_\square$, in time $O(k\text{poly}(\log\Delta, 1/\varepsilon))$ where k is the length of the optimal path, such that $\psi_\square(a,b) = \psi_\square(X,Y)$. We call these procedures ASCEND($\square$) and EXTRACTPATH($X, Y, \square$).

Let $\square_1, \square_2, \square_3, \square_4$ be the four children of the cell $\square \in Q$. We construct a weighted directed graph $H_\square = (V_\square, E_\square)$ where $V_\square = \bigcup_{i=1}^4 (\mathbb{X}_{\square_i})$. There are two sets of edges in $H_\square$: (i) *interior edges* — between two clusters of the same $\square_i$; (ii) *bridge edges* — between clusters of siblings. For each pair of entry and exit clusters, $(X,Y) \in \mathbb{X}^\downarrow_{\square_i} \times \mathbb{X}^\uparrow_{\square_i}$, we add an interior edge (X,Y) with $\psi_{\square_i}(X,Y)$ as its mixed cost. Next, let $(X,Y) \in (\mathbb{X}_{\square_i} \times \mathbb{X}_{\square_j})$, for $i \neq j$. If there are edges between points of X and Y in $\vec{\mathrm{G}}$, then from Lemma 8, the cost and direction of every edge in $X \times Y$ is

identical. Let this cost be β. We add an edge between X and Y with the direction identical to edges in $X \times Y$ and set its mixed cost to be $(\beta, 1)$. For a pair of vertices $X, Y \in V_\square$, let $\overline{\psi}_\square(X, Y)$ be the smallest mixed cost of a path from X to Y in $H_\square$.

Auxiliary procedures. Before describing the AUGMENT and EXTRACTPATH procedures, we describe a few properties of the compressed graph $H_\square$.

LEMMA 9. *For any pair $X, Y \in \mathbb{X}_\square^\downarrow \times \mathbb{X}_\square^\uparrow$,*

$$\min_{\substack{X' \in D(X) \\ Y' \in D(Y)}} \overline{\psi}_\square(X', Y') \leq \psi_\square(X, Y).$$

PROOF. Let $a \in X$, $b \in Y$ be the points such that $\psi_\square(a, b) = \psi_\square(X, Y)$, and let Π be an optimal path in $\vec{\mathrm{G}}_\square$ from a to b. Suppose $a \in X' \in D(X)$ and $b \in Y' \in D(Y)$. We express Π as a sequence $\pi_1 \circ \pi_2 \circ \cdots \circ \pi_u$, where each π_i is either an edge of Π whose endpoints lie in different children of $\square$ or a maximal sub-path of Π that lies inside a single child of $\square$. If $\pi_i = (p_i, p_{i+1})$ is an edge of Π, then by construction, there is an edge (X_i, X_{i+1}) in $H_\square$ such that $p_i \in X_i$ and $p_{i+1} \in X_{i+1}$ and the mixed cost of (X_i, X_{i+1}) is $(\mu(p_i, p_{i+1}), 1)$. If π_i is a maximal path lying in the child $\square_j$ of $\square$ then its initial and final endpoints lie in the entry and exit clusters, say, X_i and X_{i+1}, of $\mathbb{X}_{\square_j}$. By construction, $H_\square$ has an edge from X_i to X_{i+1} whose mixed cost is at most $\psi(\pi_i)$. In other words, $\Gamma = \langle X' = X_1, X_2, \ldots, X_{r+1} = Y' \rangle$ is a path in $H_\square$ whose mixed cost is at most $\psi(\Pi)$. Hence, $\overline{\psi}_\square(X', Y') \leq \psi_\square(X, Y)$, which implies the lemma. $\square$

Next, let Γ be an optimal path in $H_\square$ from an entry cluster $X' \in \mathbb{X}_\square^\downarrow$ to an exit cluster $Y' \in \mathbb{X}_\square^\uparrow$. We show how to retrieve a path Π in $\vec{\mathrm{G}}_\square$ from a vertex $a \in X'$ to a vertex $b \in Y'$ such that $\psi_\square(\Pi) = \overline{\psi}_\square(X', Y')$.

Suppose $\Gamma = \langle X = X_0, X_1, \ldots, X_{u-1}, X_u = Y \rangle$. By construction, if (X_{i-1}, X_i) is an interior edge, then (X_i, X_{i+1}) has to be a bridge edge. If (X_{i-1}, X_i) is an interior edge, we recursively construct a path from X_{i-1} to X_i in the corresponding child of $\square$. Suppose a path π_i with the endpoints $u_{i-1} \in X_{i-1}$ and $u_i \in X_i$ is returned. If (X_{i+1}, X_{i+2}) is also an interior edge and the path π_{i+2} with the endpoints $u_{i+1} \in X_{i+1}$, $u_{i+2} \in X_{i+2}$ is returned, we connect π_i and π_{i+1} by the edge (u_i, u_{i+1}). If (X_{i+1}, X_{i+2}) is also a bridge edge, we choose any point $u_{i+1} \in X_{i+1}$ and add the edge (u_i, u_{i+1}). Let $\vec{\Pi}$ be the path constructed by this procedure. Clearly, $\vec{\Pi}$ is a directed path in $\vec{\mathrm{G}}(A_\square, B_\square)$. Using an inductive argument on the height of nodes, one can show that $\psi(\Pi) = \overline{\psi}_\square(X', Y')$. Lemma 9 and this procedure imply the following:

LEMMA 10. *For every node $\square \in Q$ and for any $(X, Y) \in \mathbb{X}_\square^\downarrow \times \mathbb{X}_\square^\uparrow$,*

$$\psi_\square(X, Y) = \min_{\substack{X' \in D(X) \\ Y' \in D(Y)}} \overline{\psi}_\square(X', Y').$$

The following is an immediate corollary of the above lemma

COROLLARY 3. *$H_\square$ has no negative cycles.*

By Lemma 10 and Corollary 3, assuming we have computed $\psi_{\square_i}$ at all children $\square_i$ of $\square$, we can compute $\psi_\square(X, Y)$ for all $X, Y \in \mathbb{X}_\square^\downarrow \times \mathbb{X}_\square^\uparrow$ by constructing the graph $H_\square$ and computing optimal paths between every pair of vertices. The total time taken by ASCEND procedure is $\mathrm{poly}(\log \Delta, 1/\varepsilon)$.

EXTRACTPATH$(X, Y, \square)$, where $X, Y \in \mathbb{X}_\square^\downarrow \times \mathbb{X}_\square^\uparrow$, is implemented by first constructing the graph $H_\square$ and computing optimal paths for all pairs of vertices $X', Y' \in D(X) \times D(Y)$. Let Γ be the path with the smallest mixed cost among these paths. We now retrieve a path Π in $\vec{\mathrm{G}}_\square$ from a vertex $a \in X'$ to a vertex $b \in Y'$ such that $\psi(\Pi) = \overline{\psi}(X', Y') = \psi_\square(X, Y)$. The total time spent is $O(k\mathrm{poly}(\log \Delta, 1/\varepsilon))$ where k is the number of edges in Π. Since $H_\square$ has no negative cycles and since minimizing mixed cost ensures that among minimum-cost paths we compute a path with fewest edges, Π is a simple path.

Implementing FINDAP and AUGMENT. Using EXTRACTPATH and ASCEND procedure, FINDAP and AUGMENT can be implemented in a straightforward manner. Let ∇ be the root of Q and let $S_\nabla \subseteq \mathbb{X}_\nabla^\downarrow$ (resp. $T_\nabla \subseteq \mathbb{X}_\nabla^\uparrow$) be the subset of free clusters at ∇. By definition, $B_F = \bigcup S_\nabla$ and $A_F = \bigcup T_\nabla$. Set

$$S, T = \arg\min_{X, Y \in S_\nabla \times T_\nabla} \psi_\nabla(X, Y).$$

Then, $\psi_\nabla(S, T)$ is the smallest mixed cost of a directed path in $\vec{\mathrm{G}}_M(A, B)$ from a source to sink. We compute an optimal path $\vec{\Pi}$ from a vertex $b \in S$ to a vertex $a \in T$ in $\vec{\mathrm{G}}_M(A, B) = \vec{\mathrm{G}}_\nabla$, s.t., $\psi_\nabla(b, a) = \psi_\nabla(S, T)$, using EXTRACTPATH$(S, T, \nabla)$. As mentioned above, $\vec{\Pi}$ is a simple path. $\vec{\Pi}$ can be converted into a compact augmenting path Π, as described in Lemma 7, such that $\phi_M(\Pi) = \mu(\vec{\Pi})$. By Lemma 7, Π is a compact augmenting path of the smallest net cost. Hence, FINDAP takes $O(|\Pi| \mathrm{poly}(\log \Delta, 1/\varepsilon))$ time.

Finally, the AUGMENT(Π) updates M to $M' = M \oplus \Pi$ and the information stored at Q, as follows. For a point $p \in A \cup B$, let $\mathbb{C}(p)$ be the set of ancestors of the leaf of Q that contains p. Set $\mathbb{C}(\Pi) = \bigcup_{p \in \Pi} \mathbb{C}(p)$. We observe that the only change in $\vec{\mathrm{G}}_{M'}$ from $\vec{\mathrm{G}}_M$ is that the direction and cost of the edges in Π change and the cost of edges incident on any vertex of Π may change. Hence, for a node $\square \in Q$, $\vec{\mathrm{G}}_\square$ does not change if $\square$ does not contain any vertex of Π, i.e., $\square \notin \mathbb{C}(\Pi)$. For each point $p \in \Pi$, we update the information at nodes of $\mathbb{C}(p)$ in a bottom-up manner. More precisely, suppose we have already updated $\psi_\square$ values for some cell $\square \in \mathbb{C}(\Pi)$. Next, we call ASCEND$(\mathtt{p}(\square))$ and update the information at $\mathtt{p}(\square)$. Since $|\mathbb{C}(\Pi)| = O(|\Pi| \log \Delta)$, the total time spent by AUGMENT(Π) is $O(|\Pi| \mathrm{poly}(\log \Delta, 1/\varepsilon))$. We thus obtain the following:

LEMMA 11. *A, B and a matching M that satisfies* ACI *can be maintained in a data structure so that* FINDAP *and* AUGMENT *take $O(k\mathrm{poly}(\log \Delta, 1/\varepsilon))$ time where k is the length of the output and input path respectively.*

6. CONCLUSION

In this paper, we presented a near-linear time Monte-Carlo algorithm for computing ε-approximate bipartite matching of point sets in $\mathbb{R}^d$ under any L_p-metric. First, we presented a quad-tree based distance function $\mathtt{d}(\cdot, \cdot)$ that approximates the L_p-norm. Next, we presented an algorithm for computing an ε-approximate matching under $\mathtt{d}(\cdot, \cdot)$. By exploiting the quad-tree structure of the distance function, our algorithm iteratively generates minimum-cost augmenting paths in time proportional to its length. The total length of the augmenting paths generated by the algorithm is $O((n/\varepsilon) \log n)$ implying its near-linear running time. For simplicity of exposition, we did not minimize the exponent in $\mathrm{poly}(\log n, 1/\varepsilon)$ factor. One way of reducing the exponent is by using an exponential grid instead of a uniform grid for $\mathbb{G}[\square]$. This reduces the number of subcells to roughly $\log(n)/\varepsilon$, instead of $(\log(n)/\varepsilon)^2$. We conclude by stating two natural open questions:

- Is there a near-linear time ε-approximation algorithm for a

generalization of bipartite matching called the *transportation problem* in geometric settings?

- Is there a sub-quadratic algorithm for computing an optimal Euclidean bipartite matching in $\mathbb{R}^2$?

7. REFERENCES

[1] P. K. Agarwal, A. Efrat, and M. Sharir, Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications, *SIAM J. Comput.*, 29 (1999), 39–50.

[2] P. K. Agarwal and K. R. Varadarajan, A near-linear constant-factor approximation for euclidean bipartite matching?, *Proc. of 12th Annual Sympos. on Comput. Geom.*, 2004, pp. 247–252.

[3] S. Arora, Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems, *J. ACM*, 45 (1998), 753–782.

[4] R. Duan and S. Pettie, Approximating maximum weight matching in near-linear time, *Proc. 51st Annual IEEE Sympos. on Foundations of Comp. Sc.*, 2010, pp. 673–682.

[5] H. N. Gabow and R. Tarjan, Faster scaling algorithms for network problems, *SIAM J. Comput.*, 18 (1989), 1013–1036.

[6] H. N. Gabow and R. E. Tarjan, Faster scaling algorithms for general graph-matching problems, *J. ACM*, 38 (1991), 815–853.

[7] A. Goel, M. Kapralov, and S. Khanna, Perfect matchings in o(n log n) time in regular bipartite graphs, *Proc. 42nd Annual Sympos. on Theory of Comput.*, 2010, pp. 39–46.

[8] P. Indyk, A near linear time constant factor approximation for euclidean bichromatic matching (cost), *Proc. 18th Annual Sympos. on Discrete Algorithms*, 2007, pp. 39–42.

[9] P. Indyk, R. Motwani, and S. Venkatasubramanian, Geometric matching under noise: Combinatorial bounds and algorithms, *Proc. 10th Annual Sympos. on Discrete Algorithms*, 1999, pp. 457–465.

[10] P. Indyk and S. Venkatasubramanian, Approximate congruence in nearly linear time, *Comput. Geom.*, 24 (2003), 115–128.

[11] S. Micali and V. V. Vazirani, An o(sqrt(v)e) algorithm for finding maximum matching in general graphs, *Proc. Annual 21st IEEE Sympos. on Foundations of Comp. Sc.*, 1980, pp. 17–27.

[12] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*, Prentice-Hall, Inc., 1982.

[13] J. M. Phillips and P. K. Agarwal, On bipartite matching under the rms distance, *Proc. 18th Canadian Conf. on Comput. Geom.*, 2006.

[14] R.Sharathkumar and P. K. Agarwal, Algorithms for transportation problem in geometric settings, *Proc. of 23rd Annual Sympos. on Discrete Algorithms*, 2012, pp. 306–317.

[15] P. M. Vaidya, Geometry helps in matching, *SIAM J. Comput.*, 18 (1989), 1201–1225.

[16] K. R. Varadarajan, A divide-and-conquer algorithm for min-cost perfect matching in the plane, *Proc. 39th Annual IEEE Sympos. on Foundations of Comp. Sc.*, 1998, pp. 320–331.

[17] K. R. Varadarajan and P. K. Agarwal, Approximation algorithms for bipartite and non-bipartite matching in the plane, *Proc. 10th Annual ACM-SIAM Sympos. on Discrete Algorithms*, 1999, pp. 805–814.