

# A Near-Linear Time $\varepsilon$ -Approximation Algorithm for Geometric Bipartite Matching\*

R.Sharathkumar<sup>†</sup>

Pankaj K. Agarwal<sup>‡</sup>

## Abstract

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

## 1 Introduction

Let  $\mathcal{G}(V, E)$ , where  $V = A \cup B$  and  $E \subseteq A \times B$ , be a weighted bipartite graph with  $|A| = |B| = n$  and  $|E| = m$ . Let  $d(a, b)$  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  $M$  is  $w(M) = \sum_{(a,b) \in M} d(a, b)$ .  $M$  is a *perfect matching* if  $|M| = n$ . A perfect matching in  $\mathcal{G}$  is called *optimal* and denoted by  $M^*$ , if its cost is minimum among all perfect matchings of  $\mathcal{G}$ . A perfect matching  $M$  in  $\mathcal{G}$  is called an  $\varepsilon$ -approximate matching if  $w(M) \leq (1 + \varepsilon)w(M^*)$ . 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  $(a, b)$  is the distance between  $a$  and  $b$  under some  $L_p$ -norm. We consider the problem of computing an  $\varepsilon$ -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 [13]. If a perfect matching does not exist, it computes a maximum cardinality minimum-cost matching. Hopcroft and Karp show that if the weight of each edge is 1, then a maximum cardinality matching can be computed in  $O(m\sqrt{n})$  time [8]. If  $\mathcal{G}$  is regular, then Goel *et al.* present a randomized  $O(n \log n)$  algorithm for computing a perfect matching [7]. 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. There is also extensive work on computing optimal matchings and maximum-cost matchings in non-bipartite graphs [4, 6, 12].

---

\*This work is supported by NSF under grants CCF-09-40671, CCF-10-12254, and CCF-11-61359, by ARO grants W911NF-07-1-0376 and W911NF-08-1-0452, and by an ERDC contract W9132V-11-C-0003.

A preliminary version of this paper appeared in Proc. 44th Annual ACM Sympos. on Theory Comput., 2012, pp. 385–394.

<sup>†</sup>Department of Computer Science, Stanford University, Stanford, CA 94305 sharathk@stanford.edu. This work was done while the author was a PhD student at Duke University.

<sup>‡</sup>Department of Computer Science, Box 90129, Duke University, Durham NC 27708-0129 pankaj@cs.duke.edu

For the case where  $A, B \subset \mathbb{R}^2$  and an  $L_p$ -norm, Vaidya [16] shows that an optimal matching can be computed in  $O(n^{2.5})$  time. Agarwal *et al.* [1] improve the running time to  $O(n^{2+\delta})$  where  $\delta > 0$  is an arbitrarily small constant.<sup>1</sup> For  $L_1$ - and  $L_\infty$ -norms, the running time can be improved to  $n^2 \log^{O(d)} n$  [16]. For  $A, B \subseteq [\Delta]^d$ , i.e., points lie on a  $d$ -dimensional integer grid bounded by  $\Delta$  in each dimension, the result of [15] presents an  $O(n^{3/2} \log^{d+O(1)} n \log \Delta)$  time algorithm for computing an optimal matching. It is an open question whether a subquadratic algorithm exists for computing an optimal Euclidean bipartite matching in  $\mathbb{R}^2$ , or even for  $L_1, L_\infty$ -norms. In contrast, Varadarajan [17] presents an  $O(n^{3/2} \text{polylog } n)$  algorithm for the non-bipartite case under any  $L_p$ -norm — this is surprising because the non-bipartite case seems harder for graphs with arbitrary edge costs. See also [10, 11, 14].

For bipartite graphs on point sets  $A, B \subset \mathbb{R}^2$  and for any  $L_p$ -norm, Varadarajan and Agarwal [18] show that an  $\varepsilon$ -approximate matching can be computed in  $O((n/\varepsilon)^{3/2} \log^5 n)$  time. In [15], Sharathkumar and Agarwal present an algorithm that computes an  $\varepsilon$ -approximate matching under an  $L_p$ -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  $\varepsilon$  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 [9] 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. His algorithm, however, does not return such a matching. It is an open question whether an  $\varepsilon$ -approximate matching can be computed in near linear-time. Again, the non-bipartite case seems easier and an  $\varepsilon$ -approximate Euclidean matching of a set of points can be computed in near-linear time [3, 18].

**Our results.** The following theorem states the main result of this 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  $\varepsilon$ -approximate matching of  $A$  and  $B$  can be computed in  $O(n \text{poly}(\log n, 1/\varepsilon))$  time with probability at least  $1 - 1/n^{O(1)}$ .*

We describe a Monte-Carlo algorithm for computing an  $\varepsilon$ -approximate matching of  $A, B \subseteq [\Delta]^d$  leading to Theorem 1.

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

There are three key ingredients of our algorithm.

- We use a quad-tree  $Q$  based distance function  $d(\cdot, \cdot)$  that  $\varepsilon$ -approximates the  $L_p$ -norm. It therefore suffices to compute an  $\varepsilon$ -approximate matching under  $d(\cdot, \cdot)$  (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  $d(\cdot, \cdot)$  distance; see Figure 2. Based on these classes, we call certain edges special and set the cost of a special edge  $(a, b)$  to  $d(a, b) + \theta$  where  $\theta = \varepsilon w(M^*)/6n$  and  $w(M^*)$  is the cost of an optimal matching under  $d(\cdot, \cdot)$ . Doing so allows us to prove 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  $(1 + \varepsilon)w(M^*)$ . Gabow and Tarjan [5] use a similar idea where they add a cost of 1 to every non-matching edge. 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  $w(M^*) = O(n)$ . (Sections 3, 4).

---

<sup>1</sup>The constant of proportionality hidden in the big- $O$  notation depends on  $\delta$  and tends to  $\infty$  as  $\delta$  tends to 0.

- Unlike many previous minimum-weight matching algorithms – both exact and approximate – our algorithm does not 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

Let  $\Delta > 0$  be any integer. For simplicity, we assume  $\Delta$  to be of the form  $2^k$  for some integer  $k > 0$ . In this section we describe a distance function  $d(\cdot, \cdot)$  that approximates the distance between every pair of points in  $[\Delta]^d$  under any  $L_p$ -norm.<sup>2</sup> We choose integers  $i_1, \dots, i_d \in [\Delta]$  uniformly at random and set  $H = [0, 2\Delta]^d - (i_1, \dots, i_d)$ .  $H$  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  $H$  — the root of  $Q$  is associated with  $H$  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. See Figure 1. 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  $G_0, G_1, \dots, G_\delta$ ;  $G_0$  is a single cell which is  $G$  itself and  $G_\delta$  is the finest grid associated with the leaves of  $Q$ . Although  $G_\delta$  is the finest grid associated with the lowest level of  $Q$ , we can define  $G_i$  for  $i > \delta$  in a straightforward manner. For any pair  $(a, b) \in [\Delta]^d$  we define  $d(a, b)$  as follows: For a parameter  $\varepsilon > 0$ , we set  $\omega = \lceil \log_2(8d^2 \log_2(2\Delta)/\varepsilon) \rceil$  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

$$d(a, b) = \|a_\square b_\square\|_p + d2^i/\Omega. \quad (1)$$

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

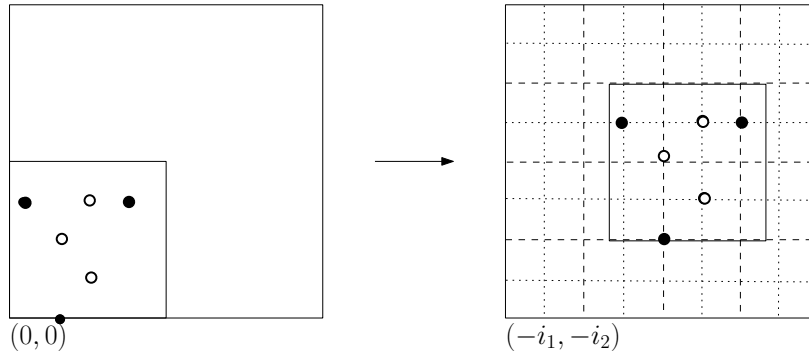


Figure 1. Randomly shifted quad-tree.

**Lemma 1** For any pair  $(a, b) \in [\Delta]^d$ ,  $d(a, b) \geq \|ab\|_p$ . Furthermore,

$$\mathbb{E}[d(a, b)] \leq (1 + \varepsilon/2)\|ab\|_p.$$

<sup>2</sup>Strictly speaking, the function  $d(\cdot, \cdot)$  depends on the value of  $p$  and we should use the notation  $d_p(\cdot, \cdot)$ , but for simplicity we assume that  $p$  is fixed and we drop the subscript  $p$ .

**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 side-length of both  $\xi_1$  and  $\xi_2$  is  $2^i/\Omega$ , by triangle inequality,

$$\|ab\|_p \leq \|a_{\square}b_{\square}\|_p + d \cdot 2^i/\Omega.$$

Hence,  $d(a, b) \geq \|ab\|_p$ . Moreover,

$$\|a_{\square}b_{\square}\|_p \leq \|ab\|_p + d \cdot 2^i/\Omega,$$

therefore  $d(a, b) \leq \|ab\|_p + d \cdot 2^{i+1}/\Omega$  which implies,

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

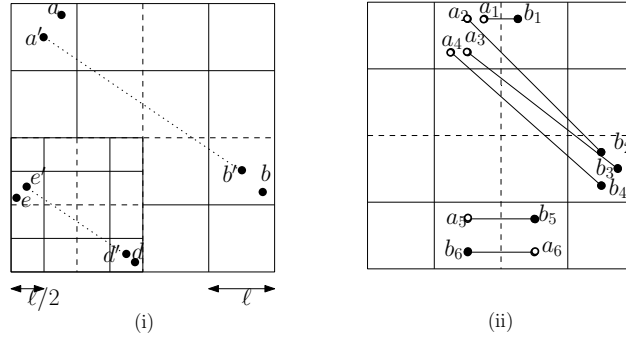
Since  $\square$  is the least common ancestor of  $a$  and  $b$ , they lie in two different children of  $\square$ . Recall that  $G_{\delta-i+1}$  is shifted uniformly at random. As argued by Arora [3],

$$\Pr[\square \in G_{\delta-i}] \leq \frac{\|ab\|_1}{2^{i-1}} \leq \frac{d\|ab\|_p}{2^{i-1}}.$$

Therefore,

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

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



**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.  $d(a, b) = \|a'b'\| + 2\ell$ ,  $d(d, e) = \|d'e'\| + \ell$ . (ii) Classifying Matching edges:  $M = \{(a_1, b_1), (a_2, b_2), (a_3, b_3), (a_4, b_4), (a_5, b_5), (a_6, b_6)\}$ ,  $\mathcal{K}_M = \{\{(a_1, b_1)\}, \{(a_2, b_2), (a_3, b_3), (a_4, b_4)\}, \{(a_5, b_5)\}, \{(a_6, b_6)\}\}$ .

**Corollary 1** Let  $A, B \subseteq [\Delta]^d$ , let  $p > 0$  be an integer, and let  $\varepsilon > 0$  be a parameter. Let  $M^*$  be an optimal matching of  $A, B$  under  $d(\cdot, \cdot)$  and let  $\mu$  be the cost of the optimal matching under the  $L_p$ -norm. Then  $\mu \leq w(M^*) \leq (1 + \varepsilon/2)\mu$  with probability at least  $1/2$ .

### 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  $M = \{(a_1, b_1), \dots, (a_k, b_k)\}$  of  $A, B$ , a vertex is called *free* if it is not incident on any edge of  $M$ , and *matched* otherwise. Let  $A_F \subseteq A$  and  $B_F \subseteq B$  be the set of free vertices. We partition 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(ii).

Let  $\mathcal{K}_M = \{M_1, \dots, M_r\}$  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  $A_1, \dots, A_r$ , the set  $B \setminus B_F$  is partitioned into sets  $B_1, \dots, B_r$ , 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 (local edges) of  $A_i \times B_i$  are referred to as edges of *class*  $i$ .

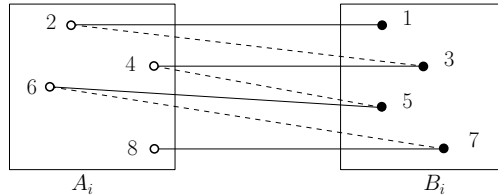
We fix a parameter  $\theta$  whose value will be chosen in Section 4. Given a matching  $M$ , we define a modified cost function  $\Phi_M : A \times B \rightarrow \mathbb{R}_{\geq 0}$ , called the *adjusted cost* of an edge,

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

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,  $\Phi_M(M) = w(M)$ .



**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  $\phi_M(\pi_1) = \phi_M(\pi_2) = 0$ ,  $\phi_M(\pi_3) = -d(1, 2)$ ,  $\phi_M(\pi_4) = d(1, 2)$ .

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

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

An *augmenting path*  $\Pi$  is an alternating path between two free vertices. We *augment*  $M$  along  $\Pi$ , by updating  $M$  to  $M \oplus \Pi$ , where  $\oplus$  is the symmetric difference. That is, the augmentation removes the edges in  $\Pi \cap M$  and adds the edges in  $\Pi \setminus M$  to  $M$ . 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,  $\varphi_i$ .
- (L2) Let  $\Pi$  be a path composed only of local edges, then all edges in  $\Pi$  belong to the same class.
- (L3) Let  $\Pi = \langle e_1, e_2, \dots, e_k \rangle$  be an alternating path such that every edge  $e_j$  of  $\Pi$  is a local edge 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 matching edges are incident on both  $a$  and  $b$ , and they also belong to the class  $i$ .

(L1), (L2) and (L4) follow from the definition of local edges, and (L3) follows from (L1) and the definition of net cost. Note that (L1)–(L4) strongly rely on the fact that the term  $\theta$  is not added to local non-matching edges.

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

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

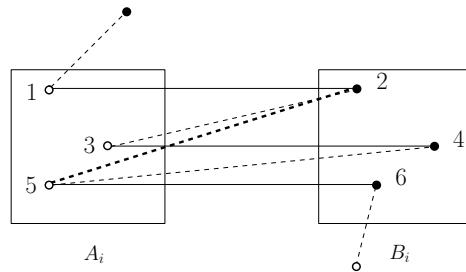
$$\mathfrak{w}(M') - \mathfrak{w}(M) + \theta|\Pi| \geq \phi_M(\Pi) \geq \mathfrak{w}(M') - \mathfrak{w}(M) + \theta|\Pi|/4.$$

**Proof:** Suppose  $\Pi$  has  $t$  non-local edges. Then,

$$\begin{aligned} \phi_M(\Pi) &= \Phi_M(\Pi \setminus M) - \Phi_M(\Pi \cap M) \\ &= \mathfrak{w}(\Pi \setminus M) - \mathfrak{w}(\Pi \cap M) + t\theta \\ &= \mathfrak{w}(M') - \mathfrak{w}(M) + t\theta. \end{aligned} \tag{2}$$

Since  $\Pi$  is a compact path,  $|\Pi| \geq t \geq |\Pi|/4$ , the lemma follows from (2).  $\square$

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



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

**Lemma 3** Given a matching  $M$  and an augmenting path  $\Pi$  with the smallest net-cost, there is a compact augmenting path  $\Pi'$  with  $\phi_M(\Pi) = \phi_M(\Pi')$ .

**Proof:** Among all augmenting paths of the minimum net cost, let  $\Pi$  be an augmenting path with the fewest number of edges. Let  $\pi = \langle e_1, \dots, e_k \rangle$  be the longest sub-path consisting only of local edges. Since  $\pi$  is maximal and all edges of  $M$  are local,  $e_1, e_k \in M$ . If  $|\pi| \leq 3$ , then  $\Pi$  is obviously compact, so assume that  $|\pi| > 3$ . By (L2), all edges of  $\pi$  belong to the same class, say  $i$ . Let  $b_1, a_1, b_2, a_2, \dots, b_k, a_k$  be the sequence of vertices in  $\pi$ , i.e.,  $e_1 = (b_1, a_1)$  and  $e_k = (b_k, a_k)$ . Set  $\tilde{\pi} = e_1 \circ (a_1, b_k) \circ e_k$  (See Figure 4). By (L3),  $\phi_M(\pi) = \phi_M(\tilde{\pi})$  but  $|\pi| > |\tilde{\pi}|$ . If we replace  $\pi$  with  $\tilde{\pi}$  in  $\Pi$ , we obtain another alternating path of the same net cost but with fewer edges, a contradiction. Hence  $|\pi| \leq 3$  and  $\Pi$  is compact.  $\square$

## 4 Algorithm

Let  $A, B \subseteq [\Delta]^d$  be two point sets of  $n$  points each, and let  $d(\cdot, \cdot)$  be the distance function defined in Section 2. We present an algorithm for computing an  $\varepsilon$ -approximate matching of  $A$  and  $B$  under  $d(\cdot, \cdot)$ . Let  $w^*$  be the cost of the optimal matching between  $A$  and  $B$  under  $d(\cdot, \cdot)$ . We choose the parameter  $\theta$ , such that

$$\varepsilon w^*/6n \leq \theta \leq \varepsilon w^*/3n. \quad (3)$$

Given a  $\theta$  that satisfies (3), our algorithm produces an  $\varepsilon$ -approximate matching in  $O(n \text{poly}(\log \Delta, 1/\varepsilon))$  time. There are various ways of obtaining  $\theta$ . Since  $A, B \subseteq [\Delta]^d$ , by (1),  $1 \leq w^* \leq 2dn\Delta$ . For every integer  $i \in [1, \dots, \lceil \log_2 2dn\Delta \rceil]$ , we set  $\theta = \theta_i := \varepsilon 2^i / 6n$  and execute  $c n \text{poly}(\log \Delta, 1/\varepsilon)$  steps of our algorithm, where  $c$  is a sufficiently large constant. Of all the perfect matchings produced by the  $O(\log n\Delta)$  executions of our algorithm, we return the one with the smallest cost. Since at least one of our guesses of  $\theta$  satisfies (3), we obtain an  $\varepsilon$ -approximate matching between  $A$  and  $B$ . In the rest of the paper, we assume that the parameter  $\theta$  we work with satisfies (3).

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  $\Pi$  of the smallest net cost (with respect to current  $M$ ).
- AUGMENT( $\Pi$ ): augment  $M$  along  $\Pi$ , i.e., update the matching to  $M \oplus \Pi$ .

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

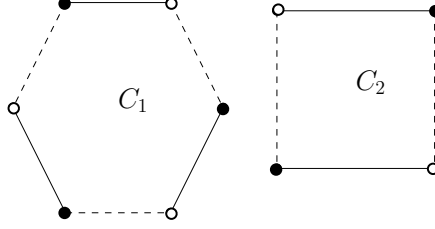
$$\boxed{\text{(ACI) For any alternating cycle } C, \phi_M(C) \geq 0.}$$

Let  $\mathbb{M}$  be the perfect matching computed by the algorithm. 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  $A, B$  of cost  $w^*$  under  $d(\cdot, \cdot)$ . Then,  $w(\mathbb{M}) \leq (1 + \varepsilon/3)w^*$ .*

**Proof:**  $\mathbb{M} \oplus M^*$  is a set of vertex disjoint alternating cycles  $C_1, \dots, C_k$  (see Figure 5). For each  $i$ ,  $C_i \setminus \mathbb{M} = C_i \cap M^*$ . Therefore,

$$w^* - w(\mathbb{M}) = \sum_{i=1}^k (w(C_i \setminus \mathbb{M}) - w(C_i \cap \mathbb{M})). \quad (4)$$



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

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

$$\sum_{i=1}^k (\mathbf{w}(C_i \setminus \mathbb{M}) - \mathbf{w}(C_i \cap \mathbb{M})) + n\theta \geq 0. \quad (5)$$

Combining (4) and (5) and using the fact that  $\theta \leq \varepsilon \mathbf{w}^*/3n$ , we obtain

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

□

Next, we bound the running time of the algorithm. Let  $M_i$  be the matching after  $i$  steps of the algorithm, and let  $\Pi_i$  be the augmenting path computed in the  $i^{\text{th}}$  step of the algorithm.  $M_0 = \emptyset$ ,  $M_i = M_{i-1} \oplus \Pi_i$ , and  $M_n = \mathbb{M}$ . We show in Section 5 that FINDAP and AUGMENT in step  $i$  take  $O(|\Pi_i| \text{poly}(\log \Delta, 1/\varepsilon))$  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  $O(n \text{poly}(\log \Delta, 1/\varepsilon))$  time.

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

**Proof:** Fix a value of  $i \leq n$ .  $M^* \oplus M_{i-1}$  consists of a set of alternating cycles and  $n - i + 1$  augmenting paths  $P_1, \dots, P_{n-i+1}$ . By definition of  $\phi$  and  $\Phi$  and the value of  $\theta$ ,

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

Since  $\Pi_i$  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,

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



or

$$\sum_{i=1}^n \phi_{M_{i-1}}(\Pi_i) \leq (1 + \varepsilon/3)w^* \ln n. \quad (6)$$

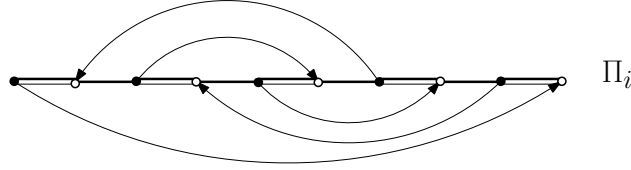
Since  $\Pi_i$  is a compact path, by Lemma 2,  $\phi_{M_{i-1}}(\Pi_i) \geq w(M_i) - w(M_{i-1}) + \theta|\Pi_i|/4$ , which implies that

$$\begin{aligned} \sum_{i=1}^n \phi_{M_{i-1}}(\Pi_i) &\geq w(\mathbb{M}) - w(M_0) + \theta \sum_{i=1}^n |\Pi_i|/4 \\ &\geq w^* + \frac{\varepsilon w^*}{24n} \cdot \sum_{i=1}^n |\Pi_i|. \end{aligned} \quad (7)$$

The lemma follows by combining (6) and (7) and using the fact that  $\theta \geq \frac{\varepsilon w^*}{6n}$ .  $\square$

**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  $i^{\text{th}}$  step. Let  $\Pi_i = \langle p_1, \dots, p_u \rangle$  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  $\Pi_i$  if at least one of its endpoints is a vertex of  $\Pi_i$ . 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,  $\phi_{M_{i-1}}(C) \geq 0$ . 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  $\Pi_i$ . Let  $(p, q)$  be such an affected edge, and suppose  $p \in \Pi_i$ . Since one of the edges of  $\Pi_i$  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 (e.g. double line segments in  $\Pi_i$  of Figure 6); the above discussion implies that the first edge and the last edge of each connected component are edges of  $M_i$ . Each connected component  $\pi$  of  $C \setminus \Pi_i$  (e.g. arcs in Figure 6) 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  $\pi$ . We use  $\tilde{\pi}$  to denote the sub-path of  $\Pi_i$  between  $p_r$  and  $p_s$ . We call  $\pi$  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  $\kappa^+$  (resp.  $\kappa^-$ ) be the total number of reducing edges in forward (resp. backward) components and let  $\kappa$  be the number of edges in  $\Pi_i$  that were non-local edges with respect to  $M_{i-1}$ . We claim that

$$\begin{aligned}
\text{(C1)} \quad & \sum_{\pi \in \mathcal{C}^+} \phi_{M_i}(\pi) + \kappa^+ \theta \geq \sum_{\pi \in \mathcal{C}^+} \phi_{M_{i-1}}(\tilde{\pi}). \\
\text{(C2)} \quad & \sum_{\pi \in \mathcal{C}^-} \phi_{M_i}(\pi) + \kappa^- \theta \geq - \sum_{\pi \in \mathcal{C}^-} \phi_{M_{i-1}}(\tilde{\pi}). \\
\text{(C3)} \quad & \sum_{\pi \in \mathcal{C}^0} \phi_{M_i}(\pi) \geq - \sum_{\pi \in \mathcal{C}^0} \phi_{M_{i-1}}(\pi) + \kappa \theta. \\
\text{(C4)} \quad & \kappa \geq \kappa^- + \kappa^+.
\end{aligned}$$

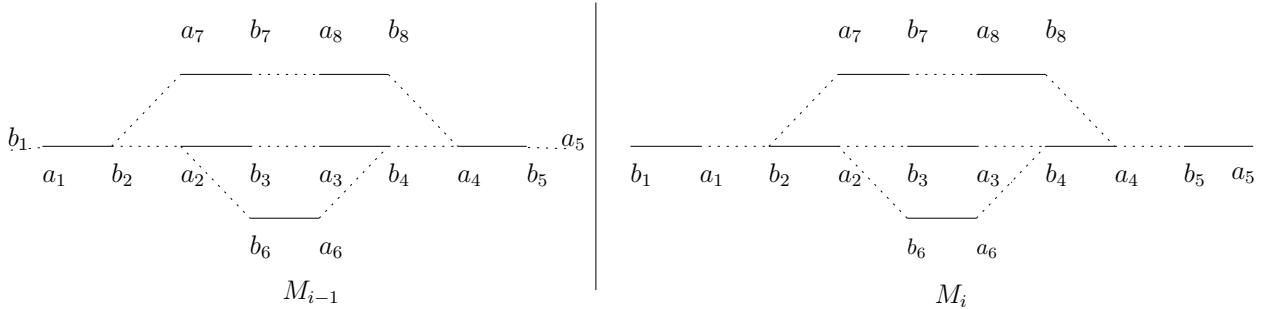
Before proving (C1)–(C4), we show that they imply  $\phi_{M_i}(C) \geq 0$ , which contradicts the assumption that  $\phi_{M_i}(C) < 0$ . Indeed,  $C$  being a cycle implies if an edge of  $(a, b) \in \Pi_i$  appears in  $\tilde{\pi}$ 's of  $j$  different forward components, then  $(a, b)$  has to appear  $j$  times in paths  $\tilde{\pi}$  of backward components and  $\mathcal{C}^0$  put together. Hence, adding equations (C1)–(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  $\bar{\Pi} = (\Pi_i \setminus \tilde{\pi}) \cup \pi$  is an augmenting path with respect to  $M_{i-1}$ ; see Figure 7. Since  $\Pi_i$  is the minimum net-cost augmenting path (with respect to  $M_{i-1}$ )  $\phi_{M_{i-1}}(\bar{\Pi}) \geq \phi_{M_{i-1}}(\Pi_i)$ . This implies  $\phi_{M_{i-1}}(\pi) \geq \phi_{M_{i-1}}(\tilde{\pi})$  or

$$\sum_{\pi \in \mathcal{C}^+} \phi_{M_{i-1}}(\pi) \geq \sum_{\pi \in \mathcal{C}^+} \phi_{M_{i-1}}(\tilde{\pi}). \tag{8}$$

Since the total number of reducing edges in forward components is  $\kappa^+$  and the adjusted cost of other affected edges does not decrease after the augmentation. (8) can be written as

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

which proves (C1).

*Proof of (C2):* For any backward component  $\pi \in \mathcal{C}^-$ ,  $\tilde{\pi} \cup \pi$  is an alternating cycle  $C$  with respect to  $M_{i-1}$ ; 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

$$\sum_{\pi \in \mathcal{C}^-} \phi_{M_{i-1}}(\pi) + \sum_{\pi \in \mathcal{C}^-} \phi_{M_{i-1}}(\tilde{\pi}) \geq 0. \quad (9)$$

There are  $\kappa^-$  reducing edges in  $\mathcal{C}^-$ , therefore (9) becomes

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

which proves (C2).

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

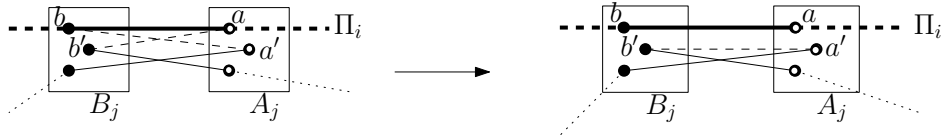
$$\begin{aligned} \sum_{\pi \in \mathcal{C}^0} \phi_{M_{i-1}}(\pi) &= \sum_{\pi \in \mathcal{C}^0} \Phi_{M_{i-1}}(\pi \setminus M_{i-1}) - \Phi_{M_{i-1}}(\pi \cap M_{i-1}) \\ &= \sum_{\pi \in \mathcal{C}^0} \mathbf{w}(\pi \setminus M_{i-1}) - \mathbf{w}(\pi \cap M_{i-1}) + \kappa \theta. \end{aligned}$$

Therefore

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

which proves (C3).

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

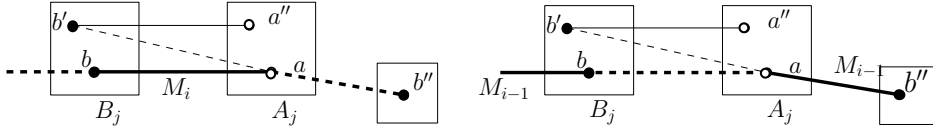


**Figure 8.** Shortcutting  $C$ : replacing  $\langle b', a, b, a' \rangle$  with  $\langle b', a' \rangle$ .

**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 with respect to  $M_{i-1}$ .*

**Proof:** Suppose  $e = (a, b)$  is adjacent to two reducing edges  $(b', a)$  and  $(b, a')$ ; see Figure 8. The above discussion following the discussion of reduced edges implies that both  $(b', a)$  and  $(b, a')$  are local with respect to  $M_i$  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  $\pi$  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 with respect to  $M_{i-1}$ , so assume that  $\Pi_i$  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 with respect to  $M_i$ . By (L4), see Figure 9,  $(b', a'')$  and  $(b, a)$  belong to the same class as  $(b', a)$  (with respect to  $M_i$ ). On the other hand,  $(b', a)$  being reducing also implies that it is non-local with respect to  $M_{i-1}$ . By (L4), we can conclude that  $(b', a''), (a, b'') \in M_{i-1}$  do not belong to the same class of  $\mathcal{K}_{M_{i-1}}$ . 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 with respect to  $M_{i-1}$ , implying that  $(a, b)$  is non-local with respect to  $M_{i-1}$ .  $\square$

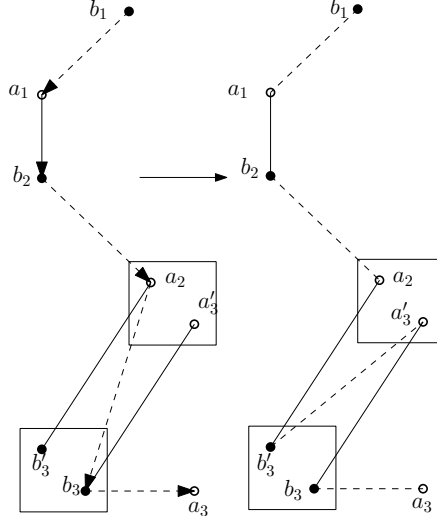


**Figure 9.**  $(b', a)$  is non-local with respect to  $M_{i-1}$ ;  $b, b'$  lie in the same sub-cell and  $b''$  lies in a different sub-cell.

## 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 reduce the problem of finding a compact augmenting path of minimum net cost (using FINDAP) to finding an “optimal” path between free vertices in a weighted directed graph,  $\vec{G}_M(A, B)$ , described below. Guided by the partition of the matching edges into equivalence classes (Section 3), we describe a hierarchical representation of  $\vec{G}_M$  that allows us to compute an optimal path in  $\vec{G}_M$  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  $\vec{G}_M(A, B)$  be the weighted graph with the vertex set  $A \cup B$  and the edge set  $A \times B$ . The orientation and weight of an edge  $(a, b) \in A \times B$  is defined as follows. 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,  $(a, b)$  is directed from  $a$  to  $b$  with  $\mu_M(a, b) = -\Phi_M(a, b)$ . If  $M$  is obvious from the context we use  $\vec{G}$  to denote  $\vec{G}_M$  and  $\mu(\cdot, \cdot)$  to denote  $\mu_M(\cdot, \cdot)$ . For any directed path  $\vec{\Pi}$  in  $\vec{G}(A, B)$ , 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  $\vec{G}$ , each free vertex of  $B$  is a *source* in  $\vec{G}$ , and these are the only source and sink vertices in  $\vec{G}$ . A path in  $\vec{G}$  alternates between local and non-local edges, but a local edge may not be a matching edge. The following lemma relates alternating paths  $\Pi$  with directed paths  $\vec{\Pi}$  in  $\vec{G}$ .



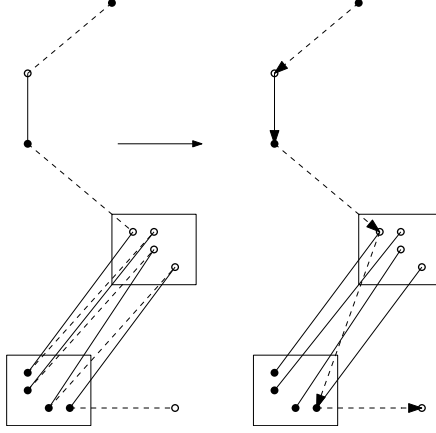
**Figure 10.** Transforming a directed path in  $\vec{G}$  to an alternating path.

**Lemma 7** Let  $M$  be a matching that satisfies ACI, and let  $b \in B$  and  $a \in A$  be two free vertices with respect to  $M$ .

- (i) A (directed) path  $\vec{\Pi}$  in  $\vec{G}_M(A, B)$  from  $b$  to  $a$  can be transformed into a compact augmenting path  $\Pi$  between  $a$  and  $b$  such that  $\mu(\vec{\Pi}) = \phi_M(\Pi)$  and  $|\Pi| \leq 2|\vec{\Pi}| + 2$ .
- (ii) An augmenting path  $\Pi$  can be transformed into a (directed) path  $\vec{\Pi}$  of length at most  $|\Pi|$  in  $\vec{G}(A, B)$  such that  $\mu(\vec{\Pi}) \leq \phi_M(\Pi)$ .

**Proof:** (i) Let  $\vec{\Pi} = \langle b = b_1, a_1, \dots, b_k, a_k = a \rangle$ , 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  $\Pi$  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  $\Pi$  (e.g.  $(a_1, b_2)$  in Figure 10). 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  $(a_i, b_{i+1})$  (e.g.  $(a_2, b_3)$  in Figure 10). 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  $\pi_i$ . By construction, the resulting path  $\Pi$  is an augmenting path and  $|\Pi| \leq 4k = 2|\vec{\Pi}| + 2$ . 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  $\Pi$  has  $k$  non-local edges, it is compact.

(ii) If the augmenting path  $\Pi = \langle b = b_1, a_1, \dots, b_k, a_k = a \rangle$  between  $b$  and  $a$  does not contain any non-matching local edge, then by the construction of  $\vec{G}$ ,  $\Pi$  is also a directed path in  $\vec{G}_M(A, B)$  and  $\mu(\vec{\Pi}) = \phi_M(\Pi)$ . So assume that  $\Pi$  has a non-matching local edge, and let  $e = (b_i, a_i)$  be the first non-matching local edge. Then  $\langle b_1, a_1, \dots, a_{i-1}, b_i \rangle$  is a directed path in  $\vec{G}$ . By definition of local edges,  $b_i$  and  $a_i$  are not free. Suppose it belongs to class  $j$ . Let  $\pi$  be the maximally connected sub-path of  $\Pi$  composed of local edges that contains  $e$ . By (L2), all edges of  $\pi$  are in class  $j$ . Let  $e_r = (a_r, b_{r+1})$  and  $e_l = (a_l, b_{l+1})$ ,  $r < l$  be the first and last edges of  $\pi$ . We replace  $\pi$  by a single edge  $(a_{i-1}, b_{l+1})$  in  $\Pi$ ; see Figure 11. Since  $\pi$  is maximal and all the edges of  $\pi$  are local,  $e_r, e_l \in M$ . (L3) implies that  $\phi_M(\pi) = \mu(a_r, b_l)$ . Since  $(b_i, a_i)$  is the first local non-matching edge,  $r = i - 1$  and  $(b_{i-1}, a_{i-1})$  is a non-local edge. Hence  $\langle b_1, a_1, \dots, b_{i-1}, a_{i-1}, b_{l+1} \rangle$  is a directed path in  $\vec{G}$ . Repeating this step for all the non-matching local edges of  $\Pi$ , we obtain a path  $\vec{\Pi}$  of length at most  $|\Pi|$  that alternates between non-local and local edges and  $\mu(\vec{\Pi}) = \phi_M(\Pi)$  as desired.  $\square$



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

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

**Corollary 2** *Assuming  $M$  satisfies ACI,  $\vec{G}_M(A, B)$  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  $\vec{G}$ .

**Hierarchical clustering of points.** Next, we hierarchically cluster the vertices of  $\vec{G}$  using the partition  $\mathcal{K}_M$  of matching edges, that allows us to compute a minimum-cost directed path in  $\vec{G}$  efficiently. For any cell  $\square$  of  $Q$ , the quad-tree constructed in Section 2, we cluster  $(A \cup B) \cap \square$  into  $O(\text{poly}(\log \Delta, 1/\varepsilon))$  clusters. Roughly 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 with respect to  $r$ , i.e., both edges are either directed toward  $r$  or away from  $r$ . Let  $M = \{(a_1, b_1), \dots, (a_k, b_k)\}$  be the matching, and let  $\{A_1, \dots, A_u\}$  (resp.  $\{B_1, \dots, B_u\}$ ) be the partition of  $A \setminus A_F$  (resp.  $B \setminus B_F$ ) induced by  $\mathcal{K}_M$ . 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; see Figure 12. 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  $A_\xi$  and  $B_\xi$  belong to a single cluster

$$A_\xi^F = A_F \cap \xi, \quad B_\xi^F = B_F \cap \xi.$$

- *Internal clusters:* All points of  $A_\xi$  ( $B_\xi$ ) whose partner point is also inside  $\square$  belong to a single cluster

$$A_\xi^I = \{a_i \in A_\xi \mid (a_i, b_i) \in M, b_i \in B_\square\},$$

$$B_\xi^I = \{b_i \in B_\xi \mid (a_i, b_i) \in M, a_i \in A_\square\}.$$

- *Boundary clusters:* All points of  $A_\xi$  whose partner points lie outside  $\square$  are partitioned into various clusters. Two such points belong to the same cluster if and only if they belong to the same class  $A_i$  for some  $i \leq u$ . More precisely, let  $N^*(\square) = \bigcup \mathbb{G}[\square']$  where the union is taken over all siblings

$\square'$  of ancestors of  $\square$  (including  $\square$ ).  $|N^*(\square)| = \text{poly}(\log \Delta, 1/\varepsilon)$ , as there are  $O(\log \Delta)$  ancestors of  $\square$ , each has three siblings, and each sibling has  $\text{poly}(\log \Delta, 1/\varepsilon)$  sub-cells. For every sub-cell  $\eta \in N^*(\square)$ , we have a cluster

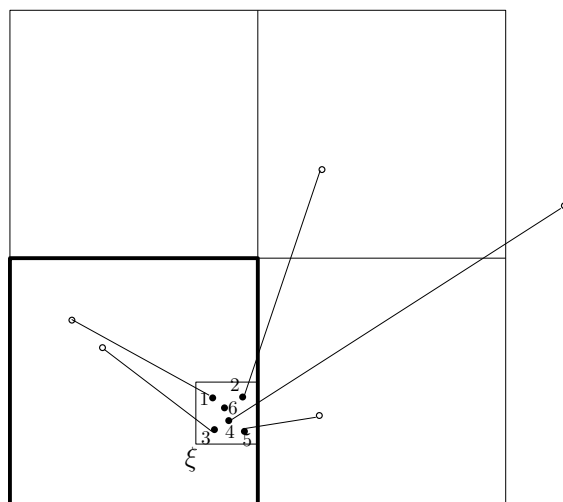
$$A_\xi^\eta = \{a_i \in A_\xi \mid (a_i, b_i) \in M, b_i \in B_\eta\},$$

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

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  $p(\square)$ , parent of  $\square$ , and each boundary cluster at  $\square$  is contained in a boundary or an internal cluster at  $p(\square)$ .

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  $\xi$  has four children  $\xi_1, \xi_2, \xi_3, \xi_4$  in  $\mathbb{G}[\square_i]$  and we get the following relationship.

$$\begin{aligned} A_\xi^F &= \bigcup_{i=1}^4 A_{\xi_i}^F, & B_\xi^F &= \bigcup_{i=1}^4 B_{\xi_i}^F, \\ A_\xi^\eta &= \bigcup_{i=1}^4 A_{\xi_i}^\eta, & B_\xi^\eta &= \bigcup_{i=1}^4 B_{\xi_i}^\eta. \\ A_\xi^I &= \bigcup_{i=1}^4 (A_{\xi_i}^I \cup (\bigcup_{\zeta \in N(\xi)} A_{\xi_i}^\zeta)), \\ B_\xi^I &= \bigcup_{i=1}^4 (B_{\xi_i}^I \cup (\bigcup_{\zeta \in N(\xi)} B_{\xi_i}^\zeta)). \end{aligned} \tag{10}$$



**Figure 12.** Partitioning of  $B_\xi$  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  $\xi$  of the highlighted cell.

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 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  $\vec{G}$  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,

$$\begin{aligned} \mathbb{X}_\square^\downarrow &= \{B_\xi^F, A_\xi^I, B_\xi^\eta \mid \xi \in \mathbb{G}[\square], \eta \in N^*(\square)\}, \\ \mathbb{X}_\square^\uparrow &= \{A_\xi^F, B_\xi^I, A_\xi^\eta \mid \xi \in \mathbb{G}[\square], \eta \in N^*(\square)\}. \end{aligned}$$

Let  $\vec{\Pi}$  be a directed path in  $\vec{G}$  from a source to a sink vertex and let  $\bar{\pi}$  be a maximal connected sub-path of  $\vec{\Pi}$  that lies inside  $\square$ . Suppose  $\bar{\pi}$  contains at least one edge. For the two endpoints  $p, q$  of  $\bar{\pi}$ , we refer to  $p$  as entry and  $q$  as exit point if  $\bar{\pi}$  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  $\bar{\pi}$  is an initial portion of  $\vec{\Pi}$ , then the entry point is a source vertex, belonging to some  $B_\xi^F$ , an entry cluster. Otherwise, if the first edge of  $\bar{\pi}$  is a local edge, which lies inside  $\square$  by assumption that at least one edge of  $\bar{\pi}$  lies inside  $\square$ , 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  $A_\xi^I$ , an entry cluster. Finally, if the first edge of  $\bar{\pi}$  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  $B_\xi^\eta$ , 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  $\vec{G}_\square = \vec{G}_M(A_\square, B_\square)$  be the subgraph of  $\vec{G}(A, B)$  induced by the point sets  $A_\square, B_\square$ ;  $\vec{G}_\square$  contains all directed paths of  $\vec{G}$  that lie completely within  $\square$ . We define the *mixed cost* of a path  $\Pi$  in  $\vec{G}_\square$ , denoted by  $\psi(\Pi)$ , as  $(\mu(\Pi), |\Pi|)$ ;  $\mu(\Pi)$  is the cost of  $\Pi$  and  $|\Pi|$  is the length of  $\Pi$ . We rank the mixed costs in lexicographic order, i.e., for two paths,  $\Pi_1$  and  $\Pi_2$  with  $\psi(\Pi_1) = (\mu_1, \ell_1)$  and  $\psi(\Pi_2) = (\mu_2, \ell_2)$ ,  $\Pi_1$  has smaller mixed cost than  $\Pi_2$ , i.e.,  $\Pi_1 \leq \Pi_2$ , 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  $\vec{G}_\square$ , and we refer to such a path as an *optimal* path from  $a$  to  $b$  in  $\vec{G}_\square$ . The definition of mixed cost and Corollary 2 ensure 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  $Q$ , we store

$$\psi_\square(X, Y) \quad \text{for all } X, Y \in \mathbb{X}_\square^\downarrow \times \mathbb{X}_\square^\uparrow, \quad (11)$$

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



**Compressed graph.** We show that if we have (11) at the four children of a cell  $\square \in Q$ , we can compute (11) for  $\square$  in  $\text{poly}(\log \Delta, 1/\varepsilon)$  time, and that given  $(X, Y) \in \mathbb{X}_{\square}^{\downarrow} \times \mathbb{X}_{\square}^{\uparrow}$ , we can compute an optimal path from a vertex  $a \in X$  to  $b \in Y$  in  $\vec{G}_{\square}$ , 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  $\text{ASCEND}(\square)$  and  $\text{EXTRACTPATH}(X, Y, \square)$ .

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  $H_{\square}$ <sup>3</sup>:

- (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,  $(X, Y) \in \mathbb{X}_{\square_i}^{\downarrow} \times \mathbb{X}_{\square_i}^{\uparrow}$ , 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 both  $X$  and  $Y$  are subsets of  $A$  (or of  $B$ ), there is no edge between  $X$  and  $Y$ . Otherwise, by Lemma 8, the cost and direction of every edge in  $X \times Y$  is identical. Let this cost be  $\beta$ . 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  $\bar{\psi}_{\square}(X, Y)$  be the smallest mixed cost of a path from  $X$  to  $Y$  in  $H_{\square}$ .

**Auxiliary procedures.** Before describing the  $\text{ASCEND}$  and  $\text{EXTRACTPATH}$  procedures, we describe a few properties of the compressed graph  $H_{\square}$ .

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

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

**Proof:** Let  $a \in X, b \in Y$  be the points such that  $\psi_{\square}(a, b) = \psi_{\square}(X, Y)$ , and let  $\Pi$  be an optimal path in  $\vec{G}_{\square}$  from  $a$  to  $b$ . Suppose  $a \in X' \in D(X)$  and  $b \in Y' \in D(Y)$ . We express  $\Pi$  as a sequence  $\pi_1 \circ \pi_2 \circ \dots \circ \pi_u$ , where each  $\pi_i$  is either an edge of  $\Pi$  whose endpoints lie in different children of  $\square$  or a maximal sub-path of  $\Pi$  that lies inside a single child of  $\square$ . If  $\pi_i = (p_i, p_{i+1})$  is an edge of  $\Pi$ , 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  $\pi_i$  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,  $\Gamma = \langle X' = X_1, X_2, \dots, X_{r+1} = Y' \rangle$  is a path in  $H_{\square}$  whose mixed cost is at most  $\psi(\Pi)$ . Hence,  $\bar{\psi}_{\square}(X', Y') \leq \psi_{\square}(X, Y)$ , which implies the lemma.  $\square$

Next, let  $\Gamma$  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  $\Pi$  in  $\vec{G}_{\square}$  from a vertex  $a \in X'$  to a vertex  $b \in Y'$  such that  $\psi_{\square}(\Pi) = \bar{\psi}_{\square}(X', Y')$ . Suppose  $\Gamma = \langle X = X_0, X_1, \dots, X_{u-1}, X_u \rangle$ . We replace each  $X_i$  with a vertex  $u_i \in V_{\square}$  and each edge  $(X_{i-1}, X_i)$  either with a path or a single edge, as follows. If  $(X_{i-1}, X_i)$  is an internal edge, we recursively construct a path from  $X_{i-1}$  to  $X_i$  in the corresponding child of  $\square$ . Suppose a path  $\pi_i$  with end points  $u_{i-1} \in X_{i-1}$  and  $u_i \in X_i$  is returned. We replace  $X_{i-1}$  and  $X_i$  with  $u_{i-1}$  and  $u_i$  respectively and the edge  $(X_{i-1}, X_i)$  in  $\Gamma$  with a path  $\pi_i$ . Since  $\Gamma$  has minimum mixed cost, no two consecutive edges in  $\Gamma$  can be internal edges. Therefore, the above step will replace an  $X_i$  at most once. Each  $X_i$  that is incident on one internal edge has been replaced by a vertex  $u_i$ . If both edges incident on  $X_i$  are bridge edges, we replace  $X_i$

<sup>3</sup>Abusing the notation slightly, we use  $X$  to denote a cluster of  $\mathbb{X}_{\square}$  as well as a node of  $H_{\square}$

with an arbitrary vertex  $u_i$  of the cluster  $X_i$ . Finally, we replace each bridge edge  $(X_{i-1}, X_i)$  with the edge  $(u_{i-1}, u_i)$ .

Let  $\vec{\Pi}$  be the path constructed by this procedure. Clearly,  $\vec{\Pi}$  is a directed path in  $\vec{G}(A_\square, B_\square)$ . Using an inductive argument on the height of nodes, one can show that  $\psi(\Pi) = \bar{\psi}_\square(X', Y')$ . Lemma 9 and this procedure imply the following:

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

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

The following is an immediate corollary of Lemma 10 and Corollary 2.

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

We are now ready to describe the two procedures. As for ASCEND, 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, say using Floyd-Warshall algorithm. The total time taken by ASCEND procedure is  $\text{poly}(\log \Delta, 1/\varepsilon)$ .

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  $\Gamma$  be the path with the smallest mixed cost among these paths. We now retrieve a path  $\Pi$  in  $\vec{G}_\square$  from a vertex  $a \in X'$  to a vertex  $b \in Y'$  such that  $\psi(\Pi) = \bar{\psi}(X', Y') = \psi_\square(X, Y)$ . The total time spent is  $O(k \text{poly}(\log \Delta, 1/\varepsilon))$  where  $k$  is the number of edges in  $\Pi$ . 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,  $\Pi$  is a simple path.

**Implementing FINDAP and AUGMENT.** Using EXTRACTPATH and ASCEND procedures, FINDAP and AUGMENT can be implemented in a straightforward manner:

*FINDAP procedure.* Let  $\nabla$  be the root of  $Q$  and let  $S_\nabla \subseteq \mathbb{X}_\nabla^\downarrow, T_\nabla \subseteq \mathbb{X}_\nabla^\uparrow$  be the subset of free clusters at  $\nabla$ . 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  $\vec{G}_M(A, B)$  from a source to a sink. using EXTRACTPATH( $S, T, \nabla$ ), we compute an optimal path  $\vec{\Pi}$  from a vertex  $b \in S$  to a vertex  $a \in T$  in  $\vec{G}_M(A, B) = \vec{G}_\nabla$ , s.t.,  $\psi_\nabla(b, a) = \psi_\nabla(S, T)$ . As mentioned above,  $\vec{\Pi}$  is a simple directed path.  $\vec{\Pi}$  can be converted into a compact augmenting path  $\Pi$ , as described in Lemma 7, such that  $\phi_M(\Pi) = \mu(\vec{\Pi})$ . By Lemma 7,  $\Pi$  is a compact augmenting path of the smallest net cost. Hence, FINDAP takes  $O(|\Pi| \text{poly}(\log \Delta, 1/\varepsilon))$  time.

The AUGMENT( $\Pi$ ) 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  $\mathbb{C}(\Pi) = \bigcup \mathbb{C}(p)$  where the union is taken over all vertices  $p$  of  $\Pi$ . We observe that the only change in  $\vec{G}_{M'}$  from  $\vec{G}_M$  is that the direction and cost of the edges in  $\Pi$  change and the cost of edges incident on any vertex of  $\Pi$  may change. Hence, for a node  $\square \in Q$ ,  $\vec{G}_\square$  does not change if  $\square$  does not contain any vertex of  $\Pi$ , 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  $\psi_{\square_i}$  for all children  $\square_i \in \mathbb{C}(\Pi)$  of a node  $\square$ , then we call ASCEND( $\square$ ) and update the information at  $\square$ . Since  $|\mathbb{C}(\Pi)| = O(|\Pi| \log \Delta)$ , the total time spent by AUGMENT( $\Pi$ ) is  $O(|\Pi| \text{poly}(\log \Delta, 1/\varepsilon))$ . We thus obtain the following:

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

This completes the proof of Theorem 2. We now describe how Theorem 2 can be used to obtain Theorem 1. Suppose  $T > 0$  is an estimate such that  $T \leq w(M^*) \leq \lambda T$ , for  $\lambda = O(\log \log n)$ ; such a  $T$  can be computed in  $O(n \text{poly} \log n)$  time [2]. We scale the coordinates of points in  $A \cup B$  by multiplying old coordinates with  $(2n/\varepsilon)/T$  and rounding the resulting coordinate to the nearest integer. Let the largest coordinate of any point in  $A \cup B$  after scaling be  $N$ . Let  $G'$  be a grid defined on a hypercube  $[0 : 2N]^d$  with each cell of the grid having a side-length  $4(n/\varepsilon) \log \log n$ . We choose integers  $i_1, \dots, i_d \in [N]$  uniformly at random and set  $G = G' - (i_1, \dots, i_d)$ .  $G$  decomposes  $A$  and  $B$  into several subsets  $\{(A'_1, B'_1), \dots, (A'_k, B'_k)\}$ . Here  $\bigcup_{i=1}^k A'_i = A$ ,  $\bigcup_{i=1}^k B'_i = B$ . As shown in [9], the problem of computing  $\varepsilon$ -approximate matching decomposes, with probability  $1/2$  into the problem of computing  $\varepsilon$ -approximate matchings on the subsets  $\{(A'_1, B'_1), \dots, (A'_k, B'_k)\}$ . Each pair of subsets  $A'_i, B'_i \subseteq [\Delta]^d$ ,  $|A'_i| = |B'_i|$  are point sets from an integer grid where  $\Delta \leq O((n/\varepsilon) \log \log n)$ . Therefore, combining the  $\varepsilon$ -approximate matching returned by the algorithm of Theorem 2, we obtain an  $\varepsilon$ -approximate matching of  $A$  and  $B$  implying Theorem 1.

## 6 Conclusion

In this paper, we presented a near-linear time Monte-Carlo algorithm for computing an  $\varepsilon$ -approximate bipartite matching of point sets in  $\mathbb{R}^d$  under any  $L_p$ -metric. For simplicity of exposition, we did not attempt to minimize the exponent in  $\text{poly}(\log n, 1/\varepsilon)$  factor of the running time. One way of reducing the exponent is by using an exponential grid instead of a uniform grid for  $\mathbb{G}[\square]$ , but new ideas are required to bring the running time to  $O(n \log n \text{poly}(1/\varepsilon))$ .

We conclude by stating two natural open questions:

- Is there a near-linear algorithm for computing an optimal bottleneck matching, i.e., a matching that minimizes the length of its largest edge?
- Is there a sub-quadratic algorithm for computing an optimal Euclidean bipartite matching in  $\mathbb{R}^2$ ?

## 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 Foundat. Comput. 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 ACM Sympos. on Theory Comput.*, 2010, pp. 39–46.
- [8] J. E. Hopcroft and R. M. Karp, An  $n^{5/2}$  algorithm for maximum matchings in bipartite graphs, *SIAM J. Comput.*, 2 (1973), 225–231.
- [9] P. Indyk, A near linear time constant factor approximation for Euclidean bichromatic matching (cost), *Proc. Annual ACM-SIAM Sympos. on Discrete Algorithms*, 2007, pp. 39–42.
- [10] P. Indyk, R. Motwani, and S. Venkatasubramanian, Geometric matching under noise: Combinatorial bounds and algorithms, *Proc. 10th Annual ACM/SIAM Sympos. on Discrete Algorithms*, 1999, pp. 457–465.
- [11] P. Indyk and S. Venkatasubramanian, Approximate congruence in nearly linear time, *Comput. Geom.*, 24 (2003), 115–128.
- [12] S. Micali and V. V. Vazirani, An  $O(\sqrt{v}e)$  algorithm for finding maximum matching in general graphs, *Proc. 21st Annual IEEE Sympos. Foundat. Comput. Sc.*, 1980, pp. 17–27.
- [13] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1982.
- [14] J. M. Phillips and P. K. Agarwal, On bipartite matching under the rms distance, *Proc. 18th Canadian Conf. on Comput. Geom.*, 2006.
- [15] R. Sharathkumar and P. K. Agarwal, Algorithms for transportation problem in geometric settings, *Proc. 23rd Annual ACM-SIAM Sympos. on Discrete Algorithms*, 2012, pp. 306–317.
- [16] P. M. Vaidya, Geometry helps in matching, *SIAM J. Comput.*, 18 (1989), 1201–1225.
- [17] K. R. Varadarajan, A divide-and-conquer algorithm for min-cost perfect matching in the plane, *Proc. 39th Annual IEEE Sympos. on Foundat. Comput. Sc.*, 1998, pp. 320–331.
- [18] 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.