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

R.Sharathkumar Pankaj K. Agarwal

November 2, 2011

Abstract

For point sets $A, B \in \mathbb{R}^d$, $|A| = |B| = n$, and for a parameter $\varepsilon > 0$, we present an algorithm that computes, in $O(n\text{poly}(\log n, 1/\varepsilon))$ time, a matching whose cost is within $(1+\varepsilon)$ of optimal perfect matching 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. Our 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 $O(n\text{poly}(\log n, 1/\varepsilon))$.

1 Introduction

For a weighted bipartite graph $\mathcal{G}(A, B) = (A \cup B, E \subseteq A \times B)$, $|A| = |B| = n$, let $\mathtt{d}(a, b)$ be the cost of any edge $(a, b) \in E$. A *perfect matching* $M \subseteq E$ is a set of edges such that every vertex $a \in A \cup B$ is incident on exactly one edge of M. An *optimal matching* M_{OPT} is a matching that minimizes the cost $\mathtt{w}(M_{\mathrm{OPT}}) = \sum_{(a,b)\in M_{\mathrm{OPT}}} \mathtt{d}(a, b)$. For $\varepsilon > 0$, an *ε-approximate matching* is a matching M such that $\mathtt{w}(M) \leq (1 + \varepsilon)\mathtt{w}(M_{\mathrm{OPT}})$. In this paper, we consider the problem of computing ε-approximate matching for the case in which $A, B \subset \mathbb{R}^d$, $\mathcal{G}(A, B) = (A \cup B, A \times B)$ is the complete bipartite graph and $\mathtt{d}(a, b)$ is the distance between a and b under the L_p-metric.

Previous work. Optimal matching on weighted bipartite graphs with n vertices and m edges can be computed using the Hungarian algorithm in $O(n^3)$ time [12]. For unweighted bipartite graphs, Hopcraft-Karp show that a maximum-cardinality matching can be computed in time $O(m\sqrt{n})$ [7]. For the case where edge costs are positive integers bounded by $n^{O(1)}$, Gabow and Tarjan show that the optimal matching can be computed in $O(m\sqrt{n}\log n)$ time. The non-bipartite version of the matching problem, where given an arbitrary weighted graph, we want to compute a matching of vertices such that the total cost of the corresponding matching is minimized is also well-studied. Micali and Vazirani [11] give an $O(m\sqrt{n})$ time algorithm for computing maximum cardinality matchings on unweighted graphs. For the case where edge weights are bounded integers, Gabow and Tarjan [5] give a scaling algorithm for computing optimal non-bipartite matching in time $O(m\sqrt{n}\log^{3/2} n)$. For unweighted regular bipartite graphs, Goel *et al.* [6] present an $O(n \log n)$ algorithm for computing perfect matchings.

In the case where $A, B \subset \mathbb{R}^2$ and $\mathtt{d}(\cdot, \cdot)$ is the L_2 norm, Vaidya [15] shows that the optimal matching on $\mathcal{G}(A, B)$ can be computed in time $O(n^{2.5})$. 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 the optimal matching. For the case where $A, B \subseteq [\Delta]^d$ are points 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] presented an $O(n^{3/2}\,\mathrm{polylog}\, n)$ algorithm for the non-bipartite case — this is surprising because the non-bipartite case is harder for general graphs. See also [13, 9, 10].

For bipartite graphs on point sets $A, B \subset \mathbb{R}^2$ and for the L_2 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 time $O(n^{3/2}\Phi(n)\log(1/\varepsilon))$; here $\Phi(n)$ is the query and update time of a dynamic weighted nearest-neighbor data structure. In [2], Agarwal and Varadarajan present a Monte Carlo algorithm for computing an $O(\log(1/\varepsilon))$-approximate matching in time $O(n^{1+\varepsilon})$. 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 $1/2$, a cost that is at most $O(1)$ times the cost of the optimal matching. It, however, does not return such a matching. All these algorithms work for L_p-norms as well. It is an open question whether an ε-approximate matching can be computed in near linear-time. Again, the non-bipartite case seems to be easier and an ε-approximate euclidean matching of a set of points can be computed in near-linear time [3, 17].

Our Results. For $A, B \subset \mathbb{R}^d$, a parameter $\varepsilon > 0$, we present a Monte-Carlo algorithm that computes, with high probability, an ε-approximate matching of A, B in $O((n/\varepsilon^{O(d)})\,\mathrm{polylog}\, n)$ time under any L_p-norm.

Unlike previous algorithms, our algorithm does not explicitly conduct a primal-dual based search for the shortest augmenting path. Instead, we present a data structure to directly computes shortest augmenting path in an output sensitive manner. There are three key ingredients of our algorithm.

- We use a randomly-shifted quad-tree Q based distance function $\mathtt{d}_Q(\cdot,\cdot)$ that ε-approximates the L_p-norm. A similar distance function was introduced in [14]. The problem of computing ε-approximate matching reduces to the problem of computing ε-approximate matching on $\mathtt{d}_Q(\cdot,\cdot)$ (Section 2).

- For a partial matching M, we provide a decomposition of the edges in M into certain clusters. The two end points of all the edges in the cluster lie sufficiently close to each other and all of them have the same $\mathtt{d}_Q(\cdot,\cdot)$ distance. On this clustering of edges, we introduce the notion of adjusted costs of edges which is a charging scheme wherein we charge edges that are between clusters by a factor $\varepsilon\mathtt{w}(M^*)/6n$, where $\mathtt{w}(M^*)$ is the cost of the optimal matching under $\mathtt{d}_Q(\cdot,\cdot)$. Doing so allows us to bound the total length of the augmenting paths by $O((n/\varepsilon)\log n)$. Our charging scheme introduces an error of $\varepsilon.\mathtt{w}(M^*)$. Gabow and Tarjan [4] used a similar charging scheme where they charge every non-matching edge a cost of 1. Consequently, their charging scheme introduces an additive error of n while the total length of all the augmenting paths produced by their algorithm is $O(n\log n)$ (Sections 3, 4).

- We describe a data structure on Q, that maintains the current partial matching, produces augmenting paths in an output sensisitve manner and updates the current matching quickly. (Section 5).

2 Approximating L_p-norm

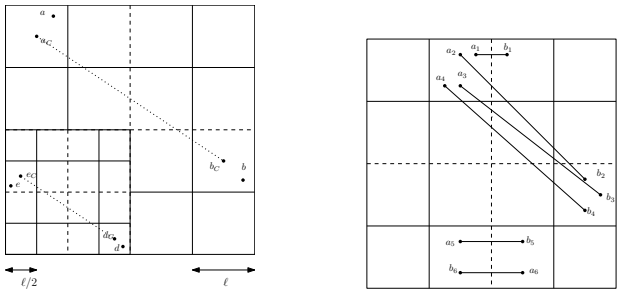
A simple transformation, as the one discussed in [8], decomposes computing approximate matching of A, B to computing approximate matchings on several subsets $\{(A'_1, B'_1)\ldots,(A'_k, B'_k)\}$ with high probability. Here $\bigcup_{i=1}^{k} A'_i = A$, $\bigcup_{i=1}^{k} B'_i = B$ and each pair of subsets $A'_i, B'_i \subseteq [\Delta]^2$, $|A'_i| = |B'_i| = n_i$, are point sets from an integer grid with $\Delta \leq n^{O(1)}$. In order to compute an approximate bipartite matching of A, B in time $O(n\mathrm{poly}(\log n, 1/\varepsilon))$, it suffices that we present an $O(n_i\mathrm{poly}(\log\Delta, 1/\varepsilon))$ time algorithm for computing approximate matching of point sets A'_i, B'_i. In the rest of the paper, we show that for $A, B \subseteq [\Delta]^2$, $|A| = |B| = n$, there is an algorithm that computes an ε-approximate bipartite matching of A, B in $O(n\mathrm{poly}(\log\Delta, 1/\varepsilon))$ time. Next, we present our distance function $\mathtt{d}_Q(\cdot,\cdot)$ that approximates the L_p-norm. We show that the ε-approximate matching under $\mathtt{d}_Q(\cdot,\cdot)$ is, in expectation, a good approximation of optimal matching under the L_p norm.

For $A, B \subseteq [\Delta]^2$, we choose integers $i, j \in [0, \Delta]$ chosen uniformly at random and set $G = [0, 2\Delta] \times [0, 2\Delta] - (i, j)$. G is a randomly-shifted square that contains both A and B. We build a quad-tree Q of height $\log_2(2\Delta) = 1 + \log_2\Delta$ on G — 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 square associated with that node into four equal squares. The nodes at height i induce a grid G_i in which each cell has a side length 2^i. We view Q as the sequence of grids $G_0, G_1, \ldots$; the final grid $G_{\log_2\Delta+1}$ is G itself with a single cell. For two points $a \in A, b \in B$ we define $\mathtt{d}_Q(a, b)$ as follows: For a constant $c_1 > 0$, a parameter $\varepsilon > 0$, we set $j = \lceil\log_2(c_1\log_2\Delta/\varepsilon^2)\rceil$ and $\mu = 2^j$ Let $\mathcal{C}$ be the cell of Q that is the least common ancestor of the leaves containing a and b. Suppose $\mathcal{C} \in G_i$. Let $G_{\mathcal{C}}$ be a $\mu \times \mu$ grid that divides $\mathcal{C}$ into *subcells* — each subcell of $\mathcal{C}$ has a length 2^{i-j}. Let $a_{\mathcal{C}}$ (resp. $b_{\mathcal{C}}$) be the center point of subcells of $\mathcal{C}$ that contain a (resp. b). We define the *representatives* of (a, b), $\mathrm{Rep}(a, b)$, as the ordered pair $(a_{\mathcal{C}}, b_{\mathcal{C}})$. We set

$$\mathtt{d}_Q(a, b) = ||a_{\mathcal{C}}b_{\mathcal{C}}||_p + 2^{i+1}/\mu,$$

here $||\cdot||_p$ is the distance in the L_p norm. In the following lemma, we show that $\mathtt{d}_Q$ approximates the L_p norm in the expected sense.

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



(i) (ii)

Figure 1. (i) For points a, b, d, e, the dotted line depicts the distance between a_C, b_C and (d_C, e_C). $\mathtt{d}_Q(a,b) = ||a_C b_C|| + 2\ell$, $\mathtt{d}_Q(d,e) = ||d_C e_C|| + \ell$. (ii) $M = \{(a_1,b_1),(a_2,b_2),(a_3,b_3),(a_4,b_4),\ (a_5,b_5),(a_6,b_6)\}$, $\mathcal{K}_M = \{(\{a_1\},\{b_1\}),(\{a_2,a_3,a_4\},\{b_2,b_3,b_4\}),(\{a_5\},\{b_5\}),(\{a_6\},\{b_6\})\}$.

Proof: We prove the lemma for L_1 norm; the proof easily extends to any L_p-norm. Let $\mathcal{C} \in G_i$ be the least common ancestor of a and b. Let $\xi_1, \xi_2 \in G_{\mathcal{C}}$ be the subcells that contain a and b, respectively. Since the length of ξ_1, ξ_2 is $2^i/\mu$, the distance between any pair $a' \in \xi_1, b' \in \xi_2$ is at most $||a_{\mathcal{C}} b_{\mathcal{C}}||_1 + 2.2^i/\mu$. Hence, $\mathtt{d}_Q(a,b) \geq ||ab||_1$. Since $||a_{\mathcal{C}} b_{\mathcal{C}}||_1 \leq ||ab||_1 + 2^{i+1}/\mu$, we have $\mathtt{d}_Q(a,b) \leq ||ab||_1 + 2^{i+2}/\mu$. Therefore,

$$\mathbb{E}[\mathtt{d}_Q(a,b)] \ \leq \ \sum_{i=1}^{\log\Delta+1} \Pr[\mathcal{C} \in G_i](||ab||_1 + 2^{i+2}/\mu) \leq ||ab||_1 + \sum_{i=1}^{\log\Delta+1} \Pr[\mathcal{C} \in G_i](2^{i+2}/\mu).$$

$\mathcal{C}$ is the least common ancestor of a and b. Therefore, two different cells $\mathcal{C}_1, \mathcal{C}_2 \in G_{i-1}$ contain a and b respectively, i.e., $a \in \mathcal{C}_1$, $b \in \mathcal{C}_2$. We have, $\Pr[\mathcal{C} \in G_i] \leq \Pr[a \in \mathcal{C}_1, b \in \mathcal{C}_2]$. Since G_{i-1} is shifted uniformly at random, the probability that a and b lie in different cells of G_{i-1} is at most $||ab||/2^{i-1}$. Therefore we have,

$$\mathbb{E}[\mathtt{d}_Q(a,b)] \leq ||ab||_1 + \sum_{i=1}^{\log\Delta+1} (||ab||_1/2^{i-1})(2^{i+2}/\mu) = ||ab||_1 + 8/\mu \sum_{i=1}^{\log\Delta+1} ||ab||_1 \leq ||ab|| + 8(\log\Delta+1)||ab||/\mu$$

Putting $c_1 = 17$ and $\mu = 17\log\Delta/\varepsilon$, we have $\mathbb{E}[\mathtt{d}_Q(a,b)] \leq (1+\varepsilon/2)||ab||_1$. □

Let M^* and M_{OPT} be the optimal bipartite matchings of A, B under $\mathtt{d}_Q(\cdot,\cdot)$ and L_p-norm. From Lemma 1 and linearity of expectation, we obtain $\mathtt{w}(M^*) \geq \mathtt{w}(M_{\mathrm{OPT}})$ and $\mathbb{E}[\mathtt{w}(M^*)] \leq (1+\varepsilon/2)\mathtt{w}(M_{\mathrm{OPT}})$. It thus suffices to compute an ε-approximate matching under $\mathtt{d}_Q(\cdot,\cdot)$.

3 Clique Partition

For a partial matching $M = \{(a_1,b_1),(a_2,b_2)\ldots,(a_k,b_k)\}$ of A, B, let a *free vertex* be a vertex that has not been matched. Let A_{F} and B_{F} be the vertices of A and B that are free. We partition the graph $\mathcal{G}(A \setminus A_F, B \setminus B_F)$ into bipartite cliques $\mathcal{K}_M = \{(A_1,B_1),(A_2,B_2),\ldots(A_r,B_r)\}$ where

- For $(a,b) \in M$, $\exists j$, $a \in A_j, b \in B_j$.
- $a_l, a_m \in A_j$ and $b_l, b_m \in B_j$ for some j if and only if $\mathrm{Rep}(a_l,b_l) = \mathrm{Rep}(a_m,b_m)$.

For any $a \in A, b \in B$, suppose $a \in A_i$ and $b \in B_i$. We refer to the edge (a,b) as a *clique edge*. All other edges, including the edges incident on points in A_F and B_F are *non-clique* edges. Note that all edges in the matching are clique edges.

Let $\varepsilon \mathtt{w}(M^*)/6n \leq \theta \leq \varepsilon \mathtt{w}(M^*)/3n$. Given such a θ our algorithm produces an ε-approximate matching in $O(n\text{poly}(\log \Delta, 1/\varepsilon))$ 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 the optimal matching is $1 \leq \mathtt{w}(M^*) \leq cn\Delta$. For some sufficiently large constant c_2, we execute $c_2 n\text{poly}(\log \Delta, 1/\varepsilon)$ 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.

For a partial bipartite matching M and a corresponding clique partition $\mathcal{K}_M$, we define a modified cost function $\Phi_M(\cdot, \cdot)$. Suppose M^* is the optimal matching on A, B. For any $a \in A$, $b \in B$, the *adjusted cost* of (a, b),

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

By definition, every edge in the matching M is a clique-edge. 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, $\Phi_M(M) = \mathtt{w}(M)$. Also, all edges in the same clique have the same adjusted cost.

Given a matching M on $\mathcal{G}(A, B)$, an *alternating path* (or cycle) P is a simple path (resp. cycle) whose edges are alternately in and not in M. We define the *net cost* of P, $\phi_M(P)$ as

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

An *augmenting path* P is an alternating path between two free vertices. We *augment* along P if we modify M to M' as follows. From the matching M, we remove $P \cap M$ and add $P \setminus M$ to produce M', i.e., $M' = (M \setminus P) \cup (P \setminus M)$. An alternating path P is *good* if the total number of non-clique edges of P is at least $|P|/4$, where $|P|$ is the length, i.e., the number of edges in the alternating path. For a good augmenting path P, the following lemma, whose proof is included in the appendix, establishes relationship between the cost of matchings M, M' and the length of P with respect to $\phi_M(P)$.

Lemma 2 *For a matching M and a good augmenting path P, let M' be the matching after augmenting M along P. Then*

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

Our algorithm chooses smallest net-cost augmenting path P in each step. From the above lemma, if P is good, then by minimizing the net-cost, we are implictly minimizing a function that also depends on the length of P. A similar idea was used by Gabow and Tarjan [4] in their scaling algorithm where they defined the weight (adjusted-cost) of every non-matching edge to be 1 plus the weight of the edge. By doing so, they obtain a matching whose weight is within an additive error n from the optimal. In contrast, our choice of θ ensures that we compute an ε-approximate matching. We will show that the total length of the augmenting paths generated by our algorithm is $O(n \log n)$. The following lemma, whose proof is included in the appendix, shows that there is always a minimum adjusted cost augmenting path that is also a good path.

Lemma 3 *For a matching M, $\mathcal{K}_M$, given any augmenting path P, there is a good augmenting path P' with $\phi_M(P) = \phi_M(P')$.*

4 Algorithm

In this section, we present our algorithm. Our algorithm, at any stage, maintains a partial matching M that satisfies the following invariant.

NON-NEGATIVE CYCLE INVARIANT For any alternating cycle C, $\phi_M(C) \geq 0$.

For a point set A, B and a partial matching M that satisfies the NON-NEGATIVE CYCLE INVARIANT we present a data structure in Section 5 that supports the following two operations.

- FINDGAP(A, B, M) returns a good augmenting path P with the smallest net cost.
- AUGMENT(A, B, P, M) augments M along P.

Each of these operations take $O(|P|\text{poly}(\log \Delta, 1/\varepsilon))$ time. Using this data structure we compute a 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 using FINDGAP(A, B, M). Next, it augments M along P using the AUGMENT(A, B, P, M) procedure.

We first bound the cost of the matching assuming the NON-NEGATIVE CYCLE INVARIANT, then analyze the running time, and finally prove the invariant.

Cost of the matching. In each iteration of our algorithm, augmentation increase the cardinality of the matching by 1. Hence, our algorithm terminates in n iterations with a perfect matching. The next lemma bounds the cost of the matching produced by our algorithm.

Lemma 4 *Let M^* be the optimal matching and let M be the perfect matching produced by our algorithm. Then, $\mathtt{w}(M) \leq (1 + \varepsilon/3)\mathtt{w}(M^*)$.*

Proof: Let $M' = M \cap M^*$. The union of $M \setminus M'$ and $M^* \setminus M'$ is a set of vertex disjoint cycles $C = \{C_1, C_2, \ldots, C_k\}$. Each of these cycles is an alternating cycle with respect to M. From NON-NEGATIVE CYCLE INVARIANT, we have $\sum_{i=1}^{k} \phi_M(C_i) \geq 0$ or

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

For any edge $(a, b) \in A \times B$, $\Phi(a, b) \leq \mathtt{d}_Q(a, b) + \theta$. Since $C_i \setminus M \subseteq M^*$, $|C_i \setminus M| \leq n$ and we get

$$\sum_{i=1}^{k} \left(\sum_{(a,b) \in C_i \setminus M} \mathtt{d}_Q(a, b) - \sum_{(a,b) \in C_i \cap M} \mathtt{d}_Q(a, b) \right) + n\theta \geq 0.$$

Adding and subtracting the cost of the edges in M', we have

$$\sum_{C_i \in C} \left(\sum_{(a,b) \in C_i \setminus M} \mathtt{d}_Q(a, b) \right) + \sum_{(a,b) \in M'} \mathtt{d}_Q(a, b) - \left(\sum_{C_i \in C} \left(\sum_{(a,b) \in C_i \cap M} \mathtt{d}_Q(a, b) \right) + \sum_{(a,b) \in M'} \mathtt{d}_Q(a, b) \right) + n\theta \geq 0.$$

By construction, M^* is $\bigcup_{C_i \in C}(C_i \setminus M) \cup M'$ and M is $\bigcup_{C_i \in C}(C_i \cap M) \cup M'$. Hence we have, $\mathtt{w}(M^*) - \mathtt{w}(M) + n\theta \geq 0$. Since θ is at most $\varepsilon \mathtt{w}(M^*)/3n$, we get $(1 + \varepsilon/3)\mathtt{w}(M^*) - \mathtt{w}(M) \geq 0$ or $\mathtt{w}(M) \leq (1 + \varepsilon/3)\mathtt{w}(M^*)$. □

Running time. Each iteration of our algorithm requires one execution of FINDGAP and AUGMENT procedures both of which run in $O(|P_i|\text{poly}(\log\Delta, 1/\varepsilon))$ time. Therefore the total running time of the algorithm over all iterations is $O(\sum_{i=1}^{n}|P_i|\text{poly}(\log\Delta, 1/\varepsilon))$. Next, we show that since the augmenting path in each iteration is a good path, we have $\sum_{i=1}^{n}|P_i| = O(n\ln n/\varepsilon)$ implying that the running time of our algorithm is $O(n\text{poly}(\log\Delta, 1/\varepsilon))$.

Lemma 5 $\sum_{i=1}^{n}|P_i| \leq (6n/\varepsilon)\ln n$.

Proof: Let M_i be the matching at the beginning of iteration i of our algorithm and let P_i be the good augmenting path with minimum net cost returned in iteration i by the FINDGAP procedure. The union of M_i with the optimal matching M^* is a set of cycles $\{C_1, C_2, \ldots C_k\}$, a set of edges $M' = M_i \cap M^*$ and a set of $n-i+1$ augmenting paths $P' = \{P'_1, \ldots P'_{n-i+1}\}$. Since P_i is an augmenting path with minimum net cost, it follows that

$$\sum_{j=1}^{n-i+1}\phi_{M_i}(P'_j) \geq (n-i+1)\phi_{M_i}(P_i). \tag{1}$$

But,

$$\begin{aligned}\sum_{j=1}^{n-i+1}\phi_{M_i}(P'_j) + \sum_{j=1}^{k}\phi_{M_i}(C_j) + \Phi_{M_i}(M') &\leq \sum_{j=1}^{n-i+1}(\Phi_{M_i}(P'_j \setminus M_i)) + \sum_{j=1}^{k}\Phi_{M_i}(C_j \setminus M_i) + \Phi_{M_i}(M') \\ &\leq \sum_{(a,b)\in M^*}\mathtt{d}_Q(a,b) + n\theta \leq (1+\varepsilon)\mathtt{w}(M^*)\end{aligned}$$

Since $\phi_{M_i}(C_j) \geq 0$ and $\Phi_{M_i}(M') \geq 0$, we get $\sum_{j=1}^{n-i+1}\phi_{M_i}(P'_j) \leq (1+\varepsilon)\mathtt{w}(M^*)$. From (1) it follows that $(n-i+1)\phi_{M_i}(P_i) \leq (1+\varepsilon)\mathtt{w}(M^*)$ or $\phi_{M_i}(P_i) \leq (1+\varepsilon)\mathtt{w}(M^*)/(n-i+1)$. Summing over all iterations,

$$\sum_{i=1}^{n}\phi_{M_i}(P_i) \leq (1+\varepsilon)\mathtt{w}(M^*)\sum_{i=1}^{n}1/(n-i+1) \leq (1+\varepsilon)\mathtt{w}(M^*)\ln n. \tag{2}$$

Since every augmenting path in the algorithm is a good path, from Lemma 2, we obtain $\phi_{M_i}(P_i) \geq \mathtt{w}(M_{i+1}) - \mathtt{w}(M_i) + \theta|P_i|/4$. Hence, $\sum_{i=1}^{n}\phi_{M_i}(P_i) \geq \mathtt{w}(M_{n+1}) - \mathtt{w}(M_1) + \theta\sum_{i=1}^{n}|P_i|/4$. Since $\mathtt{w}(M_1) = 0$ and $\mathtt{w}(M_{n+1}) \leq (1+\varepsilon)\mathtt{w}(M^*)$, we have

$$\sum_{i=1}^{n}\phi_{M_i}(P_i) \geq (1+\varepsilon)\mathtt{w}(M^*) + (\varepsilon\mathtt{w}(M^*)/cn)\sum_{i=1}^{n}|P_i|/4.$$

From the above equation and (2), we get $(\varepsilon\mathtt{w}(M^*)/6n)\sum_{i=1}^{n}|P_i|/4 \leq (1+\varepsilon)\mathtt{w}(M^*)\ln n$ implying that $\sum_{i=1}^{n}|P_i| \leq (6n/\varepsilon)\ln n$. □

Proof of invariant. We now show that the NON-NEGATIVE CYCLE INVARIANT of our algorithm holds after each iteration. At the beginning of the algorithm, there are no matching edges and there are no alternating cycles. Hence the invariant holds. Let us assume that the invariant holds after the first $i-1$ iteration. We show that the invariant holds at the end of iteration i.

An augmentation of M_i along P_i to produce M_{i+1} can change the adjusted costs of all the edges on P_i and the non-matching edges that are incident on some vertex of P_i. The adjusted cost of a non-matching edge (a,b) changes only if one of the two matching edges incident on a and b change implying that a or b

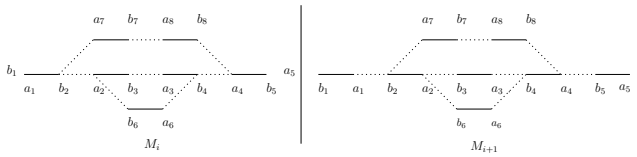


Figure 2. $P_i = \{b_1, a_1, b_2, a_2, b_3, a_3, b_4, a_4, b_5, a_5\}$, $C = \{b_2, a_7, b_7, a_8, b_8, a_4, b_4, a_6, b_6, a_2, b_2\}$ is a cycle with respect to M_{i+1};C decomposes into $F = \{b_2, a_7, b_7, a_8, b_8, a_4\}$, $B = \{a_2, b_6, a_8, b_4\}$ and $P_i \cap C = \{a_2, b_2\}, \{a_4, b_4\}$; $N_F = \{b_2, a_2, b_3, a_3, b_4, a_4\}$, $N_B = \{a_2, b_3, a_3, b_4\}$.

appears on P_i. We call all the edges that are not in P_i and whose adjusted cost change as *affected* edges. Of particular interest are affected edges whose adjusted cost reduce after an augmentation. We call such edges *reducing* edges. We refer to any cycle C with $\phi(C) < 0$ as a negative cycle. The proof of the algorithm relies on the following lemma.

Lemma 6 *Assuming that the invariant holds after $i-1$ iterations, after the ith iteration, Any negative alternating cycle must involve at least one edge of $P_i \cap M_{i+1}$.*

Lemma 7 *For any alternating cycle C, $\phi_{M_{i+1}}(C) \geq 0$.*

Proof: Let C be a negative alternating cycle with the smallest length formed after augmenting M_i along P_i. Let $P_i = \langle p_1, p_2, \ldots, p_k \rangle$ in that order, where $p_1 \in B$ and $p_k \in A$.

By Lemma 6, C contains at least one edge of P_i. A path is called *forward* path F if it is an alternating path between some $p_m \in B$ and $p_j \in A$, $m < j$ and $F \cap P_i = \{p_m, p_j\}$. Let $P_F = \langle p_m \ldots p_j \rangle$. A *backward* path B is an alternating path between $p_m \in B$ and $p_j \in A$, $m > j$ and $B \cap P_i = \{p_m, p_j\}$. Let $P_B = \langle p_j \ldots p_m \rangle$. C can be decomposed into a set of forward paths $\mathcal{F}$, a set of backward paths $\mathcal{B}$, and a set of subpaths in $P_i \cap C$.

Suppose the total number of reducing edges in $\mathcal{F}$ is k_1, the number of reducing edges in $\mathcal{B}$ is k_2 and the number of edges of $P_i \cap C$ that were non-clique before the augmentation is k. Then the following inequalities, whose proof have been included in the appendix (Lemma 14) hold.

(i) $\sum_{F \in \mathcal{F}} \phi_{M_{i+1}}(F) + k_1\theta \geq \sum_{F \in \mathcal{F}} \phi_{M_i}(P_F)$.

(ii) $\sum_{B \in \mathcal{B}} \phi_{M_{i+1}}(B) + k_2\theta \geq -\sum_{B \in \mathcal{B}} \phi_{M_i}(P_B)$.

(iii) $\sum_{P \in P_i \cap C} \phi_{M_{i+1}}(P) \geq -\sum_{P \in P_i \cap C} \phi_{M_i}(P \cap C) + k\theta$.

(iv) For a negative cycle of smallest length, $k \geq k_1 + k_2$.

Since C is a cycle, if an edge of $(a, b) \in P_i$ appears in j different P_F's, then (a, b) has to appear j times in paths P_B and $P_i \cap C$ put together. Hence, adding equations (i), (ii) and (iii), we get $\sum_{F \in \mathcal{F}} \phi_{M_{i+1}}(F) + \sum_{B \in \mathcal{B}} \phi_{M_{i+1}}(B) + \sum_{P \in P_i \cap C} \phi_{M_{i+1}}(P) + (k_1 + k_2 - k)\theta \geq 0$. Since $\phi(C) = \sum_{F \in \mathcal{F}} \phi_{M_{i+1}}(F) + \sum_{B \in \mathcal{B}} \phi_{M_{i+1}}(B) + \sum_{P \in P_i \cap C} \phi_{M_{i+1}}(P)$ and $k \geq k_1 + k_2$, we get $\phi_{M_{i+1}}(C) \geq 0$. □

5 Data Structure

In this section we describe a data structure that, given point sets A, B and a matching M for which NON-NEGATIVE CYCLE INVARIANT hold, supports FINDGAP and AUGMENT operations. We describe a weighted directed graph $\overrightarrow{G}$ on $A \times B$ such that an alternating path with respect to M corresponds to a

directed path in $\overrightarrow{\mathrm{G}}$. We formulate FINDGAP operation as finding a shortest path in $\overrightarrow{\mathrm{G}}$ between free vertices. In order to find such a path efficiently we heirarchically cluster the points of $A \cup B$ using the quad-tree Q and store a compressed graph of size $O(\mathrm{poly}(\log \Delta, 1/\varepsilon))$ at each node of Q, and use them recursively to compute a shortest path in $\overrightarrow{\mathrm{G}}$. The AUGMENT operation along a path P can change the compressed graphs in $O(|P| \log \Delta)$ nodes, each of which can be efficiently recomputed from the children in $O(\mathrm{poly}(\log \Delta, 1/\varepsilon))$ time. We now describe the details.

Directed Graph and its properties. For A, B, a partial matching M and a corresponding clique decomposition $\mathcal{K}_M$, let $\overrightarrow{\mathrm{G}}_M(A, B)$ be a graph where for each $a \in A$ $b \in B$, if (a, b) is a non-clique edge then there is a directed edge directed from b to a with a cost $\Phi(a, b)$. If M is obvious from the context we use $\overrightarrow{\mathrm{G}}$ to denote $\overrightarrow{\mathrm{G}}_M$. Otherwise, for a clique edge, there is an edge (a, b) directed from a to b with a cost $-\Phi(a, b)$. For any directed path P in $\overrightarrow{\mathrm{G}}(A, B)$, let the cost of P, $w(P)$, be the sum of cost of all the edges of P. Each free vertex of A is a sink in $\overrightarrow{\mathrm{G}}$, each free vertex of B is a source in $\overrightarrow{\mathrm{G}}$, and these are the only source and sink vertices in $\overrightarrow{\mathrm{G}}$. A path in $\overrightarrow{\mathrm{G}}$ alternates between clique and non-clique edges, but a clique edge may not be a matching edge. The following two lemmas prove useful properties of paths in $\overrightarrow{\mathrm{G}}$.

Lemma 8 *Given a directed path (cycle) P in $\overrightarrow{\mathrm{G}}(A, B)$, there is an alternating path (cycle) P' in $\mathcal{G}(A, B)$ where the cost of P, $w(P) = \Phi(P')$ and $|P'| \leq 3|P|$.*

Lemma 9 *Given A, B, a matching M and a clique decomposition $\mathcal{K}_M$. Suppose for every alternating cycle C, $\phi_M(C) \geq 0$. Then, for any alternating path P, there is a path P' in $\overrightarrow{\mathrm{G}}(A, B)$ that for every clique in $\mathcal{K}_M$, there is at most one edge of the clique in P'. Furthermore, $w(P') \leq \phi_M(P)$.*

Proof: Let P be a path that passes through the vertices $\{a_1, b_1, a_2, b_2, \ldots a_t, b_t\}$. Let us assume that every (a_j, b_j) is a clique edge. Let $(A_i, B_i) \in \mathcal{K}_M$, $a_j, a_k \in A_i$, $b_j, b_k \in B_i$ be the clique such that (a_j, b_j) and (a_k, b_k) appear in P. Without loss of generality, let us assume that (a_j, b_j) appears before (a_k, b_k) in P. Let $w(a_k, b_k) = w(a_j, b_j) = \eta$. Observe that there is a directed edge (a_j, b_k) and (a_k, b_j) whose cost is also η. Replacing (a_j, b_j) and (a_k, b_k) in P with (a_j, b_k) and (a_k, b_j) results in a path P' from $a_1, \ldots, a_j, b_k, \ldots b_t$ and a cycle C formed by the portion of P between b_j and a_k and the edge a_k, b_j. But $w(P) = w(P') + w(C)$. Since $w(C) \geq 0$, it follows that $w(P') \leq w(P)$. □

These lemmas imply that the minimum-weight path between a source and sink can be transformed into a good augmenting path of the same net cost. and that each augmenting path is a path in $\overrightarrow{\mathrm{G}}$. Hence FINDGAP reduces to finding a minimum-weight path between a source and a sink of $\overrightarrow{\mathrm{G}}$ that visits each clique at most once.

Clustering nodes. For a cell $\mathcal{C}$ of the quad tree Q, let $A_{\mathcal{C}} = A \cap \mathcal{C}$ and $B_{\mathcal{C}} = B \cap \mathcal{C}$. Similarly for a subcell $\xi \in G_{\mathcal{C}}$, let $A_\xi = A \cap \xi$ and $B_\xi = B \cap \xi$. Given $M = \{(a_1, b_1), \ldots, (a_k, b_k)\}$ and a clique partition $\mathcal{K}_M = \{(A_1, B_1), \ldots, (A_u, B_u)\}$, we cluster A_ξ, B_ξ as follows:

$$A_\xi^F = A_F \cap \xi, \qquad B_\xi^F = B_F \cap \xi, \text{ (free clusters).}$$
$$A_\xi^I = \{a_i \in A_\xi \setminus A_F \mid b_i \in B_{\mathcal{C}}\}, \qquad B_\xi^I = \{b_i \in B_\xi \setminus B_F \mid a_i \in A_{\mathcal{C}}\} \text{(internal clusters).}$$

For each subcell $\eta \not\subseteq \mathcal{C}$ of a proper ancestor of $\mathcal{C}$ in Q, we define

$$A_\xi^\eta = \{a_i \in A_\xi \setminus A_F \mid b_i \in B_\eta\}, \qquad B_\xi^\eta = \{b_i \in B_\xi \setminus B_F \mid a_i \in A_\eta\} \text{(boundary clusters.)}$$

A_ξ^I, B_ξ^I correspond to the matching edges that lie inside $\mathcal{C}$, and A_ξ^η, B_ξ^η correspond to the matching edges that cross the boundary of $\mathcal{C}$ and thus lie inside an ancestor of $\mathcal{C}$. Note that the boundary clusters is empty

for the root of Q.

$$\mathbb{X}_{\mathcal{C}} = \{B^F_\xi, A^I_\xi, B^\eta_\xi \mid \xi \in G_{\mathcal{C}}\}, \qquad \mathbb{Y}_{\mathcal{C}} = \{A^F_\xi, B^I_\xi, A^\eta_\xi \mid \xi \in G_{\mathcal{C}}\}.$$

Observing that for a fixed ξ there are $\mathrm{poly}(\log \Delta, 1/\varepsilon)$ boundary clusters — one for each subcell of an ancestor of $\mathcal{C}$ — and there are $\mathrm{poly}(\log \Delta, 1/\varepsilon)$ subcells in $G_{\mathcal{C}}$, we obtain the following

Lemma 10 *For any cell $\mathcal{C}$ of the quad-tree, $|\mathbb{X}_{\mathcal{C}}|, |\mathbb{Y}_{\mathcal{C}}| = O(\mathrm{poly}(\log \Delta, 1/\varepsilon))$.*

For a cell $\mathcal{C}$ of Q, let $\mathcal{C}_1, \ldots, \mathcal{C}_4$, be its 4 children. Each subcell $\xi \in G_{\mathcal{C}}$ has four children $\xi_1, \xi_2, \xi_3, \xi_4$, in the grid $G_{\mathcal{C}_i}$ if ξ lies inside $\mathcal{C}_i$. The following inequalities are straightforward:

$$A^F_\xi = \bigcup_{i=1}^{4} A^F_{\xi_i}, B^F_\xi = \bigcup_{i=1}^{4} B^F_{\xi_i}, \qquad A^\eta_\xi = \bigcup_{i=1}^{4} A^\eta_{\xi_i}, B^\eta_\xi = \bigcup_{i=1}^{4} B^\eta_{\xi_i}.$$
$$A^I_\xi = \bigcup_{i=1}^{4} (A^I_{\xi_i} \cup (\bigcup_{\eta \subset \mathcal{C}} A^\eta_{\xi_i})), \qquad B^I_\xi = \bigcup_{i=1}^{4} (B^I_{\xi_i} \cup (\bigcup_{\eta \subset \mathcal{C}} B^\eta_{\xi_i})). \tag{3}$$

In other words, each cluster of $\mathbb{X}_{\mathcal{C}}$ and $\mathbb{Y}_{\mathcal{C}}$ can be represented as the union of clusters of the children of $\mathcal{C}$. For each such cluster X of $\mathcal{C}$, let $D(X)$ be the set of these clusters of its children.

Information at each node. For a cell $\mathcal{C} \in Q$, let $\overrightarrow{\mathrm{G}}_{\mathcal{C}} = \overrightarrow{\mathrm{G}}_M(A_{\mathcal{C}}, B_{\mathcal{C}})$ be the subgraph of $\overrightarrow{\mathrm{G}}(A, B)$ formed by the points $A_{\mathcal{C}}, B_{\mathcal{C}}$. For a pair $(a, b) \in A_{\mathcal{C}} \times B_{\mathcal{C}}$, let $\Psi_{\mathcal{C}}(a, b)$ be the cost of the shortest weighted path from a to b in $\overrightarrow{\mathrm{G}}_{\mathcal{C}}$. For any pair of subsets $X \subseteq A_{\mathcal{C}}$, $Y \subseteq B_{\mathcal{C}}$, let $\Psi_{\mathcal{C}}(X, Y) = \min_{a \in X, b \in Y} \Psi_{\mathcal{C}}(a, b)$.

At each cell $\mathcal{C} \in Q$, we store

$$\Psi_{\mathcal{C}}(X, Y) \qquad \text{for all } X, Y \in \mathbb{X} \times \mathbb{Y}. \tag{4}$$

Next, we show that if we have (4) at all children of a cell $\mathcal{C} \in Q$, we can compute it for $\mathcal{C}$ in $\mathrm{poly}(\log \Delta, 1/\varepsilon)$, and that given X, Y, we can compute the path in $\overrightarrow{\mathrm{G}}_{\mathcal{C}}$ that visits each clique of $\mathcal{K}_M$ at most once in time $O(k\mathrm{poly}(\log \Delta, 1/\varepsilon))$, where k is the length of the path. We call these procedure ASCEND($\mathcal{C}$) and EXTRACTPATH($X, Y, \mathcal{C}$).

ASCEND procedure. Let $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4$ be the four children of the cell $\mathcal{C} \in Q$. We construct a weighted directed graph $H_{\mathcal{C}} = (V_{\mathcal{C}}, E_{\mathcal{C}})$ where $V_{\mathcal{C}} = \bigcup_{i=1}^{4} (\mathbb{X}_{\mathcal{C}_i} \cup \mathbb{Y}_{\mathcal{C}_i})$. There are two sets of edges in $H_{\mathcal{C}}$: (i) *internal edges* — between two clusters of the same $\mathcal{C}_i$; (ii) *boundary edges* — between clusters of different children. For a pair $(X, Y) \in \mathbb{X}_{\mathcal{C}_i} \times \mathbb{Y}_{\mathcal{C}_i}$, we add an internal edge (X, Y) with weight $w(X, Y) = \Psi_{\mathcal{C}_i}(X, Y)$. Next, let $(X, Y) \in \mathbb{X}_{\mathcal{C}_i} \times \mathbb{Y}_{\mathcal{C}_j}$, for $i \neq j$, suppose $X \subseteq A_\xi$ for some subcell ξ of $\mathcal{C}_i$ and $Y \subseteq B_{\xi'}$ for some subcell ξ' of $\mathcal{C}_j$; the case when $X \subseteq B_\xi$ and $Y \subseteq A_{\xi'}$ is symmetric. Note that $\Phi_M(a, b)$ is the same for all $(a, b) \in X \times Y$, say σ_{ab}. If X is a free or internal cluster, we add (Y, X) with a weight $w(Y, X) = \sigma_{ab}$. Suppose X and Y are boundary clusters A^η_ξ and $B^{\eta'}_{\xi'}$, respectively. If $\eta = \xi'$ and $\eta' = \xi$, i.e., A^η_ξ and $B^{\eta'}_{\xi'}$ lie in the same clique of $\mathcal{K}_M$, we add the edge (X, Y) with $w(X, Y) = -\sigma_{ab}$.

All the internal edges of $H_{\mathcal{C}}$ incident on some cluster $Y \in \mathbb{Y}_{\mathcal{C}}$ are directed inwards and all internal edges incident on $X \in \mathbb{X}_{\mathcal{C}_i}$ are directed outwards. Hence, any path can reach X only through a boundary edge and any path that reaches Y can exit through a boundary edge. Hence every internal edge is succeeded and preceeded by a boundary edge in any path P in $H_{\mathcal{C}}$.

For a cell $\mathcal{C}$, let $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4$ be its four children. Consider a shortest path π between any two clusters of $\mathcal{C}$ in $\overrightarrow{\mathrm{G}}(A_{\mathcal{C}}, B_{\mathcal{C}})$. Suppose π enters and exits $\mathcal{C}_1$ mutliple times. Note that when π enters (resp. exits) $\mathcal{C}_1$ through a non-clique edge, the edge is incident on (resp. origniating from) a point in A^I_ξ (resp. B^I_ξ(, for

some $\xi \in G_{\mathcal{C}_1}$. Similarly, when π enters (resp. exits) through a clique edge, the edge has to be incident on (resp. originating from) A^{η}_{ξ} (B^{η}_{ξ}) where $\eta \in G_{\mathcal{C}_i}, i \neq 1$.

For any pair of clusters of two siblings, say $\mathcal{C}_1$ and $\mathcal{C}_2$, i.e., $X \in (\mathbb{X}_{\mathcal{C}_1} \cup \mathbb{Y}_{\mathcal{C}_1})$ and $Y_1, Y_2 \in (\mathbb{X}_{\mathcal{C}_2} \cup \mathbb{Y}_{\mathcal{C}_2})$, every pair of edges $(x_1, y_1), (x_2, y_2) \in \pi$, $x_1, x_2 \in X$ and $y_1 \in Y_1, y_2 \in Y_2$ are such that $w(x_1, y_1) = w(x_2, y_1)$ and $w(x_1, y_2) = w(x_2, y_2)$. Using arguments similar to Lemma 9, we can modify π to π' such that π' enters or exits X at most once. By repeating this argument, we obtain a path π' that enters or exits every cluster of a child at most once. Such a path π' maps to a simple path in $H_{\mathcal{C}}$ — replace every maximally connected subpath in $\mathcal{C}_1$ between any two entry and exit points by the corresponding internal edge between the two clusters in $H_{\mathcal{C}}$.

For a pair of vertices $z, z' \in V_{\mathcal{C}}$, let $\overline{\Psi}_{\mathcal{C}}(z, z')$ be the length of the shortest path in $H_{\mathcal{C}}$ from z to z'. We obtain the following Lemma 11, the proof of which is included in the appendix.

Lemma 11 *For any $(X, Y) \in \mathbb{X}_{\mathcal{C}} \times \mathbb{Y}_{\mathcal{C}}$, $\Psi_{\mathcal{C}}(X, Y) = \min_{X' \in D(X), Y' \in D(Y)} \overline{\Psi}_{\mathcal{C}}(X', Y')$.*

Since $H_{\mathcal{C}}$ has no negative cycles, we can use Floyd-Warshall algorithm to compute all the $\Psi_{\mathcal{C}}$-values from $H_{\mathcal{C}}$ in $O(\text{poly}(\log \Delta, 1/\varepsilon))$ time.

EXTRACTPATH procedure. Given $z, z' \in \mathbb{X}_{\mathcal{C}} \times \mathbb{Y}_{\mathcal{C}}$, we construct a simple path from a vertex $u \in z$ and $v \in z'$ whose cost is $\Psi_{\mathcal{C}}(z, z')$ and that visits each clique at most once as follows. We compute a shortest path from z to z' in the graph $H_{\mathcal{C}}$. Among all the shortest paths between z and z', we choose the path with the minimum number of edges. Let $z = z_0, z_1, \ldots, z_{u-1}, z_u = z'$ be the computed path. By construction, if (z_{i-1}, z_i) is an internal edge, then z_i, z_{i+1} has to be a boundary edge. If (z_{i-1}, z_i) is an internal edge, we recursively call the procedure in the corresponding child of $\mathcal{C}$. Suppose a path π_i with the endpoints $u_{i-1} \in z_{i-1}$ and $u_i \in z_i$ are returned. If (z_{i+1}, z_{i+2}) is also internal edge and the path π_{i+2} with the endpoints $u_{i+1} \in z_{i+1}$, $u_{i+2} \in z_{i+2}$ is returned, we connect π_i and π_{i+1} by the boundary edge (u_i, u_{i+1}). If (z_{i+1}, z_{i+2}) is also a boundary edge, we choose any point $u_{i+1} \in z_{i+1}$ and add the edge (u_i, u_{i+1}). Let π be the path construced by this procedure. Then

Lemma 12 *The weight of π is $\Psi_{\mathcal{C}}(z, z')$ and it visits each clique at most once.*

Since π visits every clique atmost once, π is a simple and good augmenting path and the time taken by the procedure is $O(|\pi| \text{poly}(\log \Delta, 1/\varepsilon))$.

Implementing FINDGAP. We find $(X, Y) = \arg\min_{\xi_1, \xi_2 \in G_R} \Psi_R(B^F_{\xi_1}, A^F_{\xi_I})$, where R is the root of Q using EXTRACTPATH(X, Y, R), we compute in time $O(|P| \text{poly}(\log \Delta, 1/\varepsilon))$ an augmenting path with the smallest net-cost.

Implementing AUGMENT. For A, B and a partial matching M, Given a path P, the AUGMENT procedure augments M along P. Let P visit points $\{p_1, p_2, \ldots, p_k\}$ in that order. For each point visited by $p_i \in P$, let $\mathbb{C}(p_i)$ be the set of $\log \Delta + 1$ ancestors of p_i in Q. Let $\mathbb{C} = \bigcup_{i=1}^{k} \mathbb{C}(p_i)$. We recursively update $H_{\mathcal{C}}$ values of all cells $\mathcal{C} \in \mathbb{C}$ in a bottom-up fashion. The proof of correctness follows from Lemma 16 in the appendix. Updating $H_{\mathcal{C}}$ for each cell $\mathcal{C} \in \mathbb{C}$ takes $O(\text{poly}(\log \Delta, 1/\varepsilon)$ time. Hence, the total time taken by the procedure is $O(|P| \text{poly}(\log \Delta, 1/\varepsilon))$. We thus conclude the following

Theorem 1 *For point sets $A, B \subset [\Delta]^d$ and a parameter $\varepsilon > 0$, an ε-approximate matching of A, B can be computed in time $O(n \text{poly}(\log \Delta, 1/\varepsilon))$ with high probability.*

From the discussion in the beginning of Section 2, we obtain:

Theorem 2 *For point sets $A, B \subset \mathbb{R}^d$ and a parameter $\varepsilon > 0$, an ε-approximate matching of A, B can be computed in time $O(n \text{poly}(\log n, 1/\varepsilon))$ with high probability.*

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.*, 29:39–50, 1999.

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

[3] S. Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, 1998.

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

[5] H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for general graph-matching problems. *J. ACM*, 38(4):815–853, 1991.

[6] A. Goel, M. Kapralov, and S. Khanna. Perfect matchings in o($n \log n$) time in regular bipartite graphs. In *STOC*, pages 39–46, 2010.

[7] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973.

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

[9] P. Indyk, R. Motwani, and S. Venkatasubramanian. Geometric matching under noise: Combinatorial bounds and algorithms. In *SODA*, pages 457–465, 1999.

[10] P. Indyk and S. Venkatasubramanian. Approximate congruence in nearly linear time. *Comput. Geom.*, 24(2):115–128, 2003.

[11] S. Micali and V. V. Vazirani. An o(sqrt(v)e) algorithm for finding maximum matching in general graphs. In *FOCS*, pages 17–27, 1980.

[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. In *CCCG*, 2006.

[14] R.Sharathkumar and P. K. Agarwal. Algorithms for transportation problem in geometric settings. In *Proc. Annual Sympos. on Discrete Algorithms*, page To Appear, 2012.

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

[16] K. R. Varadarajan. A divide-and-conquer algorithm for min-cost perfect matching in the plane. In *FOCS*, pages 320–331, 1998.

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

A Appendix

Proof of Lemma 2. Suppose P has t non-clique edges. Then, $\phi_M(P) = \Phi_M(P \setminus M) - \Phi_M(P \cap M) = \sum_{(a,b)\in P\setminus M} \mathtt{d}_Q(a,b) + t\theta - \sum_{(a,b)\in P\cap M} \mathtt{d}_Q(a,b)$. Adding and subtracting $\sum_{(a,b)\in M\setminus P} \mathtt{d}_Q(a,b)$ from above we get,

$$\phi_M(P) \;=\; \sum_{(a,b)\in M'} \mathtt{d}_Q(a,b) - \sum_{(a,b)\in M} \mathtt{d}_Q(a,b) + t\theta = \mathtt{w}(M') - \mathtt{w}(M) + t\theta.$$

Since P is a good path, $|P| \geq t \geq |P|/4$. Hence, we get

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

Proof of Lemma 3. We construct P' from P as follows. Let $\{b_0, a_1, \ldots, b_{m-1}, a_m\}$ be the set of vertices visited by P in that order; here $b_0 \in B$ and $a_k \in A$ are free vertices. We decompose the edges in P into a non-clique edges and a set of maximally connected sub-paths $\mathcal{P}$ such that for any subpath $P_i \in \mathcal{P}$, every edge of P_i lies in the same clique. Since, P_i is maximally connected, every $P_i \in \mathcal{P}$ is preceeded by a non-clique edge. Also, each $P_i \in \mathcal{P}$ starts from some $a_j \in A$, with (a_j, b_j) being a matching edge and ends at some $(a_k, b_k) \in M$. Suppose every $|P_i| \leq 3$, then there is a non-clique edge in P for at most three clique edges and hence P is good. Otherwise, suppose $P_i \in \mathcal{P}$ has a length greater than 3. Let P_i pass through the vertices $\{a_j, b_j, a_{j+1}, b_{j+1} \ldots a_k, b_k\}$. Since P_i starts and ends with a matching edge, and since all edges in the same clique have identical adjusted costs, we have $\phi_M(P_i) = -\Phi(a_j, b_j)$. For every $P_i \in \mathcal{P}$, we can replace P_i with $P_i' = \{(a_j, b_j), (b_j, a_k), (a_k, b_k)\}$ to obtain the path P'. Since $\phi_M(P_i) = \phi_M(P_i')$, the net cost of P and P' are identical. Also, since $|P_i'| \leq 3$ path P' is good.

Lemma 13 *For any reducing edge (a, b), in the clique-partition corresponding to M_{i+1}, a and b are in the same clique.*

Proof: For any (a, b), there are only two possible adjusted costs, i.e., $\mathtt{d}_Q(a, b)$ or $\mathtt{d}_Q(a, b) + \theta$. Since, the adjusted cost after augmentation has reduced, $\Phi_{M_{i+1}}(a, b) = \mathtt{d}_Q(a, b)$, implying a and b are in the same clique in the clique partition corresponding to M_{i+1}. □

Proof of Lemma 6. Any negative cycle should involve at least one edge (a, b) such that $\Phi_{M_i}(a, b) \neq \Phi_{M_{i+1}}(a, b)$. Suppose $(a, b) \in P_i$, then if (a, b) is a non-matching edge then the alternating cycle should contain both the matching edges incident on (a, b). Clearly both these edges are also in P_i. Hence there is at least one edge of $P_i \cap M_{i+1}$ in C. On the other hand, suppose (a, b) is an affected edge that is not in P_i. Clearly, both the matching edges incident on a and b are in C. Since every affected edge is incident on at least one edge in $P_i \cap M_{i+1}$, Hence any such C would be incident on at least one edge in $P_i \cap M_{i+1}$.

Lemma 14 *Let M_{i+1} be the matching formed after augmenting M_i along P_i. Let C be a any alternating cycle formed after augmenting M_i. Suppose the total number of reducing edges in $\mathcal{F}$ is k_1, the number of reducing edges in $\mathcal{B}$ is k_2 and the number of edges of $P_i \cap C$ that were non-clique before the augmentation is k.Then,*

(i) $\sum_{F \in \mathcal{F}} \phi_{M_{i+1}}(F) + k_1\theta \geq \sum_{F \in \mathcal{F}} \phi_{M_i}(P_F)$.

(ii) $\sum_{B \in \mathcal{B}} \phi_{M_{i+1}}(B) + k_2\theta \geq -\sum_{B \in \mathcal{B}} \phi_{M_i}(P_B)$.

(iii) $\sum_{P \in P_i \cap C} \phi_{M_{i+1}}(P) \geq -\sum_{P \in P_i \cap C} \phi_{M_i}(P \cap C) + k\theta$.

(iv) For a negative cycle of smallest length, $k \geq k_1 + k_2$.

Proof:

(i) Before augmentating along P_i, suppose for some $F \in \mathcal{F}$, consider the two portion P' and P'' of paths formed by removing P_F from P_i. Concatenating P', F and P'', we obtain another augmenting path $\overline{P}$. Since P_i is the smallest augmenting path $\phi_{M_i}(\overline{P}) \geq \phi_{M_i}(P_i)$. This implies $\phi_{M_i}(F) \geq \Phi_{M_i}(P_F)$ or

$$\sum_{F \in \mathcal{F}} \phi_{M_i}(F) \geq \sum_{F \in \mathcal{F}} \phi_{M_i}(P_F). \tag{5}$$

Since total number of reducing edges in F is k_1 and since all other affected edges increase the adjusted cost, we have $\sum_{F \in \mathcal{F}} \phi_{M_{i+1}}(F) + k_1\theta \geq \sum_{F \in \mathcal{F}} \phi_{M_i}(F)$. Combining this with (5), we have the desired inequality of (i).

(ii) Before augmentation, for $B \in \mathcal{B}$, the union of B and P_B is a cycle C. Since, $\phi_{M_i}(C) \geq 0$, we have $\phi_{M_i}(B) + \phi_{M_i}(P_B) \geq 0$ or

$$\sum_{B \in \mathcal{B}} \phi_{M_i}(B) + \sum_{B \in \mathcal{B}} \phi_{M_i}(P_B) \geq 0. \tag{6}$$

Since there are k_2 reducing edges in $\mathcal{B}$, we get $\sum_{B \in \mathcal{B}} \phi_{M_{i+1}}(B) + k_2\theta \geq \sum_{B \in \mathcal{B}} \phi_{M_i}(B)$. From (6) and the above equations, the desired inquality in (ii) follows.

(iii) For any set of alternating paths $\mathcal{P}$ in $P_i \cap C$ with a total of k non-clique edges, $\sum_{P \in \mathcal{P}} \phi_{M_i}(P) = \sum_{P \in \mathcal{P}} \sum_{(a,b) \in P \setminus M_i} \mathtt{d}_Q(a,b) + k\theta - \sum_{P \in \mathcal{P}} \sum_{(a,b) \in M_i \cap P} \mathtt{d}_Q(a,b)$

$$\begin{aligned} -\sum_{P \in \mathcal{P}} \phi_{M_i}(P) + k\theta &= \sum_{P \in \mathcal{P}} \Big(\sum_{(a,b) \in M_i \cap P} \mathtt{d}_Q(a,b) - \sum_{(a,b) \in P \setminus M_i} \mathtt{d}_Q(a,b) \Big) \\ &= \sum_{P \in \mathcal{P}} \Big(\sum_{(a,b) \in M_{i+1} \setminus P} \mathtt{d}_Q(a,b) - \sum_{(a,b) \in M_{i+1} \cap P} \mathtt{d}_Q(a,b) \Big) \\ &\leq \sum_{P \in \mathcal{P}} \phi_{M_{i+1}}(P). \end{aligned}$$

(iv) Since C is a negative cycle with smallest length and since there are $k_1 + k_2$ reducing edges of C incident on $C \cap \{P_i \cap M_{i+1}\}$, from Lemma 15, it follows that there are at least $k_1 + k_2$ non-clique edges in $C \cap \{P_i \cap M_{i+1}\}$. Hence, $k \geq k_1 + k_2$.

□

Lemma 15 *Given a negative alternating cycle C with the smallest length, let E be the set of reducing edges in C.*

(i) Every edge of $C \cap \{M_{i+1} \cap P_i\}$ has at most one edge in E incident on it.

(ii) If $(a,b) \in C \cap \{M_{i+1} \cap P_i\}$ has a reducing edge incident a or b, then

$$\Phi_{M_i}(a,b) = \mathtt{d}_Q(a,b) + \theta.$$

Proof: We prove (i) by contradiction. Suppose there are two reducing edges (b',a) and (b,a') incident on the same edge $(a,b) \in C \cap \{M_{i+1} \cap P_i\}$. By Lemma 13 and the fact that a and b are a matching edge, it follows that a', a, b and b' are in the same clique. In particular, a', b' are in the same clique. We can reduce C to another cycle C' with the $\{(a',b)(b,a)(a,b')\}$ replaced with a single edge (a',b'). Since $\phi(C') = \phi(C)$ and $|C'| < |C|$, we reach a contradiction of our assumption that C is a negative cycle with the smallest number of edges.

Suppose there is a reducing edge (b'',a) incident on a. By construction $(a,b) \notin M$. Let a be matched to b', b be matched to a' and b'' be matched to a'' in M. Let us assume that $\Phi_M(a,b) = \mathtt{d}_Q(a,b)$. This implies that (a,b) is a clique edge and hence a, b, a', b' belong to the same clique in M. In particular, $\mathrm{Rep}(a,b) = \mathrm{Rep}(a',b')$. After augmentation, from Lemma 13 the edges $(a,b), (a'',b'')$ are in the same clique. Hence $\mathrm{Rep}(a,b) = \mathrm{Rep}(a'',b'')$. This implies that $\mathrm{Rep}(a,b') = \mathrm{Rep}(a'',b'')$ implying that (a,b'') was a clique edge with respect to M. This contradicts our assumption that (a,b) is a reducing edge. □

Proof of Lemma 8. Let the path P visit vertices $\{a_1, b_1, a_2, b_2 \dots a_k, b_k\}$ in that order. The path alternates between clique and non-clique edges. Without loss of generality let us assume that the first edge (a_1, b_1) of P is a non-clique edge. We construct P' as follows. Let P'_i be the alternating path that ends with a matching edge and generated for the portion of the path P, P_i between a_1 and a_i. P'_{i+1} can be constructed from P'_i and P_i. Suppose (b_i, a_{i+1}), which is a clique edge is also a matching edge, then we add (a_i, b_i) and (b_i, a_{i+1}) to P'_i to generate P'_{i+1}. On the other hand, if (b_i, a_{i+1}) which is a clique edge is not a matching edge. Let b_i be matched to a' and a_{i+1} be matched to b'. a', b', a_{i+1} and b_i are in the same clique. We add $N = \{(a_i, b_i), (b_i, a'), (a', b')(b', a_{i+1})\}$ to path P'_i to generate P'_{i+1}. $\phi(N) = w(a_i, b_i) + w(a_{i+1}, b_i)$. Hence, $\phi(P'_{i+1}) = w(P_{i+1})$.

We replace every clique edge in P with at most 3 edges in P. Hence $|P'| \leq 3|P|$.

Proof of Lemma 11. We prove by induction on height of $\mathcal{C}$. Suppose for a cell $\mathcal{C} \in Q$, let $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4$ be its four children.

For any path P of $\overrightarrow{\mathrm{G}}(A_{\mathcal{C}}, B_{\mathcal{C}})$, a *connected sub-path* $\overline{P}$ of P in any child, say $\mathcal{C}_1$, is a connected region of the path P in $\mathcal{C}_1$ such that every vertex of $\overline{P}$ is in $\mathcal{C}_1$. A *maximally connected sub-path* of P' in a child, say $\mathcal{C}_1$, is a connected sub-path $\overline{P}$ in $\mathcal{C}_1$ that is not succeeded or preceeded in P' by any vertices that are in $\mathcal{C}_1$. P can be represented compactly by replacing every maximally connected sub-path $\hat{P}$ of P by a single internal edge between the end points of $\overline{P}$ directed along $\overline{P}$ and with a cost $w(\overline{P})$. Note that, for any child $\mathcal{C}_1$, a maximal connected subpath can exist only between pairs of nodes from any clusters $X \in \mathbb{X}_{\mathcal{C}_1}$ and $Y \in \mathbb{Y}_{\mathcal{C}_1}$ directed from X to Y. By definition, a maximally connected subpath any edge that succeeds or preceeds it is a boundary edge. We refer to such a compact representation P' of P as a compact path.

Given $H_{\mathcal{C}}$, we show the following:

(i) For every directed path (resp. cycle) P in $H_{\mathcal{C}}$, we first show there is a path (resp. cycle) P' in $\overrightarrow{\mathrm{G}}(A_{\mathcal{C}}, B_{\mathcal{C}})$ such that $w(P) = w(P')$. We construct P' from P as follows. For every internal edge $(X, Y) \in \mathbb{X}_{\mathcal{C}_i} \times \mathbb{Y}_{\mathcal{C}_i}$ in P, let $x \in X$ and $y \in Y$ be such that $\Psi_{\mathcal{C}_i}(x, y) = \Psi_{\mathcal{C}_i}(X, Y) = w(X, Y)$. Let $P(X, Y)$ be this shortest directed path between x and y. We replace every internal edge with $P(X, Y)$. Suppose a boundary edge connects two internal edges (X_1, Y_1) and (X_2, Y_2), we simply replace the boundary edge with the corresponding end points of the path $P(X_1, Y_1)$ and $P(X_2, Y_2)$ to obtain the directed path P'. If two consecutive boundary edges are incident on the same cluster, we choose an arbitrary point from the cluster to obtain the corresponding edges in P'. Since every edge in $\overrightarrow{\mathrm{G}}(A_{\mathcal{C}}, B_{\mathcal{C}})$ that is between two clusters connected by a boundary edge have the same weight and direction as the boundary edge, we have $w(P') = w(P)$. Since there are no negative cycles in $\overrightarrow{\mathrm{G}}(A_{\mathcal{C}}, B_{\mathcal{C}})$, it follows that there are no negative cycles in $H_{\mathcal{C}}$.

(ii) $\Psi_{\mathcal{C}}(X, Y) \leq \min_{X' \in D(X), Y' \in D(Y)} \overline{\Psi}_{\mathcal{C}}(X', Y')$. We first show that for any $X \in \mathbb{X}_{\mathcal{C}_i}, Y \in \mathbb{Y}_{\mathcal{C}_j}$, $i \neq j$, $\Psi_{\mathcal{C}}(X, Y) = \overline{\Psi}_{\mathcal{C}}(X, Y)$. For $X \in \mathbb{X}_{\mathcal{C}_i}, Y \in \mathbb{Y}_{\mathcal{C}_j}$, $i \neq j$, let P be a path from some $x \in X$ and $y \in Y$ such that $\Psi_{\mathcal{C}}(x, y) = \Psi_{\mathcal{C}}(X, Y)$. From Lemma 9, we can assume that P is a directed path that visits every clique at most once. Let P' be the compact path corresponding to P. We modify P' to another compact path P'' such that $w(P'') \leq w(P')$ and P'' visits every cluster in $V_{\mathcal{C}}$ at most once. To construct such a P'', we show that if P' visits the same cluster $X \in V_{\mathcal{C}}$ twice, i.e., two non-adjacent edges of P' are incident on X, then we can obtain another path whose weight is at most the weight of P' and X is visited at most once. Clearly, All edges incident on points in B^F_ξ are outgoing. Hence no path can visit B^F_ξ twice. Similarly, all edges incident on A^F_ξ are incoming; hence X is not a free cluster. By construction, P' visits every clique at most once implying that no boundary cluster is visited twice by P' cannot visit any boundary cluster twice. Therefore, X is not a boundary cluster. Suppose X is an internal cluster, say $X = A^I_\xi$ for some $\xi \in G_{\mathcal{C}_1}$ (The argument for B^I_ξ is symmetric.) Since X is visited twice, there are at least two non-clique boundary edges incident on X. Let the edges be (b, a) and (b', a'). Without loss of generality, let us assume that (b, a)

appears before (b', a') in P'. P' visits vertices $\{x, \ldots, b, a, \ldots, b', a', \ldots, y\}$ in that order. Since a, a' are in the same cluster, $w(b, a') = w(b, a)$ and $w(b', a') = w(b', a)$. We remove (b, a), (b', a') and add (b', a) and (b, a'). This creates a cycle C, consisting of the portion of P between b, a' and the edge $(a'b)$ and a path $\tilde{P}$ consisting of the portion of P' between x and a followed by edge (a, b') and the portion of P' between b' and y. Since $w(C) \geq 0$, we obtain $\tilde{P}$ whose cost is at most the cost of P and $\tilde{P}$ visits B^I_ξ atmost once. Repeating the argument for every cluster that is visited twice, we obtain P''.

Given P'', we can construct a simple path in $H_{\mathcal{C}}$ by replacing every internal edge of P'' with the corresponding internal edge of $H_{\mathcal{C}}$. Hence (ii) holds.

From (ii) and the subset relationship of (3), the desired inequality follows.

Proof of Lemma 12. We prove by induction on the height of the cell $\mathcal{C}$. Assume, that for the internal edge (z_{i-1}, z_i), a directed path with cost $w(z_{i-1}, z_i)$ is returned. We replace the internal edge (z_{i-1}, z_i) with the path without any increase in cost. Furthermore, since every internal edge is preceeded and succeeded by a boundary edge. For any two clusters $X, Y \in \mathbb{X}_{\mathcal{C}_i} \times \mathbb{Y}_{\mathcal{C}_j}$, $i \neq j$, every edge between any two points $x \in X$ and $y \in Y$ have identical in weight and direction. Hence, the choice of u_{i-1} and u_i does not alter the weights of the boundary edges. From Lemma 11, we know that the cost of the shortest path in $H_{\mathcal{C}}$ is equal to $\Psi_{\mathcal{C}}(z, z')$. Therefore, the constructed path is a directed path in $\overrightarrow{\mathrm{G}}(A_{\mathcal{C}}, B_{\mathcal{C}})$ with a weight $\Psi_{\mathcal{C}}(z, z')$.

Next, we argue that this path visits every clique at most once. By induction hypotheses, this is true for all the paths recursively generated for the internal edges. π does not visit any boundary clique twice, since otherwise, using arguments similar to Lemma 9, we can eliminate a cycle thereby reducing the length of the shortest path. This results in a contradiction since our algorithm chooses the shortest path with the smallest length. Next, we show that no internal clique is visited twice. Suppose there is a clique $(A_i, B_i) \in \mathcal{K}_M$ that is visited twice by the path. Since the paths corresponding to the internal edges visit every clique at most once, (A_k, B_k) must be visited by the paths corresponding to two different internal edges. Let $a_l, a_m \in A_k$ and $b_l, b_m \in B_k$, be the corresponding points visited by these paths. Without loss of generality, let us assume that (a_l, b_l) appears before (a_m, b_m) in the path of z, z'. Suppose (z_{i-1}, z_i) and (z_{j-1}, z_j) are the two internal edges and let the paths corresponding to them be $\{u_{i-1} \ldots a_l, b_l, \ldots u_i\}$ and $\{u_{j-1} \ldots, a_m, b_m, \ldots, u_j\}$. Since (a_l, b_l) and (a_m, b_m) belong to the same clique, we have $w(a_l, b_l) = w(a_m, b_m) = w(a_l, b_m) = w(a_m, b_l)$. Hence, we can remove (a_m, b_m) and (a_l, b_l) and add (a_l, b_m) and (a_m, b_l). This creates a vertex disjoint cycle $C = \{b_l \ldots, u_i \ldots, u_j, \ldots a_m, b_l$ and a path $P = \{\ldots, u_{i-1} \ldots a_l, b_m, \ldots, u_j, \ldots\}$. Since $w(C) \geq 0$, we have $w(P)$ is at most the cost of the path between z and z'. Furthermore, the portion between u_{i-1}, u_j is a directed path between clusters z_{i-1} and z_j. This implies that the internal edge (z_{i-1}, z_i) and (z_{j-1}, z_j) can be replaced by z_{i-1}, z_j. Thus there is another path with the same cost and a smaller length resulting in a contradiction.

Lemma 16 *For an augmenting path P and a matching M, let $\mathcal{C}$ be a non-empty quad tree cell such that $\bigcup_{i=1}^{k} p_i \cap \mathcal{C} = \emptyset$. Then, augmenting along P will not change $H_{\mathcal{C}}$.*

Proof: $H_{\mathcal{C}}$ changes only if any point in $A_{\mathcal{C}}$ or $B_{\mathcal{C}}$ changes its cluster or any edge in $\overrightarrow{\mathrm{G}}(A_{\mathcal{C}}, B_{\mathcal{C}})$ changes its weight. Clusters are defined on points being matched inside, outside $\mathcal{C}$ or being unmatched. Therefore, a point changes its cluster only if the edge it is matched to changes. Since $\mathcal{C}$ does not contain any point of P, no point in $A_{\mathcal{C}} \cup B_{\mathcal{C}}$ change their cluster. The cost of any edge in $\overrightarrow{\mathrm{G}}(A_{\mathcal{C}}, B_{\mathcal{C}})$ changes after an augmentation only if the edge is incident on some vertex of P. Since no vertex of P is in $\mathcal{C}$, no edge of $\overrightarrow{\mathrm{G}}(A_{\mathcal{C}}, B_{\mathcal{C}})$ changes its weight. □