Minimum Weight Euclidean Matching and Weighted Relative Neighborhood Graphs

Andranik Mirzaian
Department of Computer Science
York University
Toronto, Ontario, Canada M3J 1P3
andy@cs.yorku.ca

Abstract

The *Minimum Weight Euclidean Matching* (MWEM) problem is: given $2n$ point sites in the plane with Euclidean metric for interpoint distances, match the sites into n pairs so that the sum of the n distances between matched pairs is minimized. The graph theoretic version of this problem has been extensively studied since the pioneering work of Edmonds. The best time bound known for MEWM is $O(n^{2.5}(\log n)^4)$ due to Vaidya. His algorithm requires $O(n \log n)$ space.

We investigate new geometric properties of the problem and propose an $O(n)$ space, $O((n^2 + \mathcal{F}) \log n)$ time algorithm based on the Weighted Voronoi Diagram (WVD) of the sites, where $\mathcal{F}$ is the number of *edge-flips* in the diagram as the weights change during the matching algorithm. We conjecture that $\mathcal{F}$ is close to $O(n^2)$.

The new geometric results established in this paper include the following: We introduce Weighted Relative Neighborhood Graphs (WRNG) and Weighted Gabriel Graphs (WGG). These are generalizations of their unweighted versions studied in the literature. We show WRNG and WGG are straight-line planar graphs; WRNG is a subgraph of WGG; and WGG is a subgraph of the dual of WVD. Furthermore, we show that the admissible edges (and hence, the matching edges) in Edmonds' primal-dual algorithm form a subgraph of WRNG.

Key Words: *Matching, planarity, Weighted Relative Neighborhood Graphs, Weighted Gabriel Graphs, Weighted Delaunay Diagrams.*

1 Introduction

Graph Matching is a classical and well studied combinatorial optimization problem. A matching in a graph is a subset of the edges, no two of which are incident to the same vertex. In a weighted graph (with real valued weights on edges) the weight of a matching is the sum of its edge weights. The *maximum cardinality matching* problem is to find a matching of maximum cardinality in the given (unweighted) graph. The *minimum weight matching* problem is to find a maximum cardinality matching with the least possible weight in a given (weighted) graph. The first polynomial time algorithm for these problems was proposed by the pioneering work of Edmonds [17,18]. Lovász and Plummer [38] provide a comprehensive study of graph matching. Also, [37,45,52] are excellent general sources for the subject. Galil [27] provides a lucid survey of the area up to 1986. Let n and m, respectively, denote the number of vertices and edges of the given graph. The best known time bound for the maximum cardinality matching problem is $O(m\sqrt{n})$ (see [33,41]). The best known time bound for the minimum weight matching problem is $O(n^3)$ for dense graphs [23,37], and $O(n(m + n\log n))$ for sparse graphs [21,24]. (See also [28] for an $O(mn\log n)$ time algorithm.) For the case of integer edge weights a scaling method can be applied (see, for example, [19,26]). Some other related problem areas are: the bottleneck matching problem [10,11,25], sensitivity analysis in matching [55], heuristic matching algorithms [9], and on-line matching [36].

This paper is concerned with the *Euclidean Minimum Weight Matching* (EMWM) problem: given $2n$ point sites in the plane that form the vertices of an underlying complete graph with Euclidean interpoint distances as edge weights, find a minimum weight perfect matching (that is, with cardinality n) of the point sites. The best time complexity we would get by applying any general graph based minimum weight matching algorithm to solve EMWM would be $O(n^3)$ [23,37]. (For some variations of the matching problem more efficient algorithms exist. The bottleneck matching problem mentioned above is one example. Also, the minimum weight matching of points on a simple polygon can be found in $O(n(\log n)^2)$ time [39].) The $O(n^3)$ time bound for MWEM stood for many years until Vaidya [54] gave an $O(n^{2.5}(\log n)^4)$ time, $O(n\log n)$ space, algorithm for it. His algorithm is also based on Edmonds' primal-dual

method, but he uses an efficient geometric query processing technique to further speed up the algorithm.

To break the time barrier any further, there seems to be a need for a better understanding of the interplay between the graph theoretic methods on the one hand, and the underlying (somewhat less understood) geometric properties on the other hand. Perhaps a more challenging proposition would be to ask whether we can solve the problem mainly by computational geometric methods. Two such methods are the *sparsification* and the *lifting* techniques. The sparsification technique is to first compute a sparse subgraph of the underlying complete graph which is guaranteed to contain the solution (for instance, the optimum perfect matching). This technique has been successfully applied to a number of problems such as computing the closest pair, the nearest neighbors, and the Euclidean Minimum Spanning Tree via the Delaunay Triangulation of the sites (see [16,46,56]), and recently for computing the Minimum Weight Euclidean Bottleneck Matching via k-Relative Neighborhood Graphs [10,11]. The lifting technique is to construct a structure in a higher dimension so that its projection down to the lower dimensions gives the desired result. Examples of this technique are computing variations of the Voronoi diagram and the Delaunay Triangulation (see, for example, [5,6,8,16,20,31,32]). Also, recent developments in computational geometry show promise for improved algorithms for MWEM. Some of these techniques are: the *geometric partitioning* techniques, such as in [1,2], and semidynamic and dynamic algorithms [12,47]. (For results on dynamic Voronoi diagrams of *moving points* see for example [22,30].)

In this paper, the *Weighted Voronoi Diagram* and its dual (the *Weighted Delaunay Diagram*) play a prominent role. In fact, subgraphs of the Weighted Delaunay Diagram, namely, Weighted Gabriel Graphs (WGG) and Weighted Relative Neighborhood Graphs (WRNG) also come into play. Although the weighted versions of these graphs do not seem to appear in the literature previously, their unweighted versions have been studied extensively [3,34,35,40,44,49,50,51,53].

We propose an $O((n^2 + \mathcal{F}) \log n)$ time, $O(n)$ space, algorithm for MWEM based on the Weighted Voronoi Diagram of the $2n$ sites, where the weights are related to the linear programming dual variables and dynamically change. The $\mathcal{F}$ term is the number of *edge-flips* in

the Weighted Voronoi Diagram during the matching algorithm. Furthermore, we conjecture that $\mathcal{F}$ is close to $O(n^2)$. Our starting point is Edmonds' linear programming formulation and his primal-dual method. The main contribution of this paper includes further exploration of the geometric properties of the Euclidean version. More specifically:

- We show that if the edge weights form a distance metric (in an underlying complete graph) then the dual variables corresponding to the vertices remain nonnegative. This brings more symmetry between trivial and nontrivial blossoms, hence the primal constraints corresponding to vertices (that is, trivial blossoms) and nontrivial blossoms become virtually alike. For the MWEM problem, this allows us to associate circular disks centered at the point sites whose (nonnegative) radii are related to vertex and blossom dual variables. These radii are considered as the site weights.

- We generalize the Relative Neighborhood Graphs and Gabriel Graphs to their weighted versions. We show that when the weights are associated with the dual variables in Edmonds' primal-dual weighted matching algorithm as applied to EMWM, the Weighted Relative Neighborhood Graph is a subgraph of the Weighted Gabriel Graph. We also show that in that case the Weighted Gabriel Graph is a straight-line connected planar subgraph of the Weighted Delaunay Diagram and spans all the sites.

- We show the *admissible edges* form a subgraph of the Weighted Relative Neighborhood Graph of the sites. Therefore, the admissible edges, considered as straight line segments, are noncrossing and there are only $O(n)$ of them at any given time, whereas the underlying graph contains $\Theta(n^2)$ edges. (The matching edges are a subset of the admissible edges.) This enables us to search the sparse weighted Delaunay (or WGG or WRNG) edges, rather than the underlying complete graph, in order to maintain the admissible edges. However, we now have to pay the overhead for maintaining the Delaunay (or WGG or WRNG) edges, since they change as the weights change. These edge changes are called edge-flips. The latter phenomenon is known for the unweighted case (see, for example, [6,32]).

- An immediate corollary of the above results is that the optimum perfect matching is a

subgraph of the Weighted Voronoi Diagram (and the Weighted Relative Neighborhood Graph), where the weights are associated with the optimum dual variables. A tantalizing question related to the sparsification technique is: can we directly and efficiently compute this *optimum* Weighted Voronoi Diagram (or Weighted Relative Neighborhood Graph)? Once we have the optimum weights, we can construct the WDD in $O(n \log n)$ time using Fortune's sweep algorithm [20], and then compute its optimum perfect matching in $O(n^2 \log n)$ time using any of the algorithms in [24,28].

The rest of the paper is organized as follows. Section 2 gives the preliminaries, including the linear programming formulation of the minimum weight matching problem, an introduction to Edmonds' primal-dual algorithm, some features of Vaidya's algorithm, and some data structuring rudimentaries. Section 3 contains the development of the new geometric results listed above. Section 4 presents the proposed new algorithm. Section 5 contains further discussion and concluding remarks.

2 Preliminaries

In this section we introduce the reader to some of the necessary background, before we get into the details of our new results in the subsequent section.

2.1 The Linear Program

Here we will consider Edmonds' linear programming formulation as adapted by Lovász and Plummer [38]. Let $G = (V, E)$ be a given weighted graph with edge weights $d_e = d_{vu} = d(v, u)$ for $e = (v, u) \in E$. We assume $|V| = 2n$ and G contains a perfect matching. A *blossom* is an odd cardinality subset of V. A blossom is called *trivial* if it is a singleton (a vertex); otherwise it is called *nontrivial*. Let $\mathcal{B}$ denote the set of all blossoms of G. (Note that here any complement of a blossom is also a blossom.) Let x be a real vector with an entry for each edge of G (with the interpretation that $x_e = 1$ if e is a matched edge and $x_e = 0$ otherwise). Similarly, let d denote the vector of edge weights. We say an edge e is incident to a blossom B if exactly one endpoint of e is in B. We let ∇B denote the set of all edges incident to

blossom B. (Lovász and Plummer call ∇B a *cut.*) Let $x(B) = \Sigma\{x_e | e \in \nabla B\}$. The linear programming formulation of the problem is

$$\begin{array}{rll} \mathit{minimize} & d^T \cdot x & \\ \mathit{subject\ to}: & x_e \geq 0 & (\mathit{for\ each}\ e \in E) \\ & x(B) = 1 & (\mathit{for\ each\ trivial\ blossom}\ B) \\ & x(B) \geq 1 & (\mathit{for\ each\ nontrivial\ blossom}\ B)\ . \end{array}$$

Now consider the dual program. We use a variable α_B for each blossom $B \in \mathcal{B}$. Let $\alpha(e) = \alpha(u,v) = \Sigma\{\alpha_B | e \in \nabla B\}$ for any edge $e = (u,v) \in E$. The dual program consist of the following objective and constraints:

$$\begin{array}{rll} \mathit{maximize} & \Sigma_{B \in \mathcal{B}}\ \alpha_B & \\ \mathit{subject\ to}: & \alpha_B \geq 0 & (\mathit{for\ each\ nontrivial\ blossom}\ B) \\ & \alpha(e) \leq d_e & (\mathit{for\ every\ edge}\ e \in E)\ . \end{array}$$

In Section 3 we will show that these linear programming formulations can be further simplified by removing the distinction between trivial and nontrivial blossoms if the underlying graph is complete and the edge weights form a distance metric (such as the Euclidean case).

2.2 Edmonds' Algorithm

Edmonds' algorithm is a primal-dual algorithm and can be described based on the linear programming formulations in the preceding subsection. The algorithm maintains dual feasibility (starting with $\alpha(B) = 0$ for every blossom $B \in \mathcal{B}$). It also maintains an integral primal solution (starting with $x_e = 0$ for each edge $e \in E$) that satisfies all the primal constraints except that it may violate some of the second and third set of primal constraints. That is, for some blossoms B, we may have $x(B) = 0$. Any such solution is a matching of G though not necessarily a perfect matching. The algorithm proceeds through n phases. In each phase, it increases the number of matched edges by one (and thus resolves at least one of the violated primal constraints). We let M denote the set of matched edges during the execution of the algorithm. To ensure optimality, the algorithm maintains the complementary slackness conditions:

$$\begin{array}{l} \mathit{if}\ x(B) = 0\ \mathit{then}\ \alpha(B) = 0,\ \mathit{for\ each\ blossom}\ B \in \mathcal{B} \\ \mathit{if}\ \alpha(e) < d_e\ \mathit{then}\ x_e = 0,\ \mathit{for\ each\ edge}\ e \in E. \end{array}$$

The *slack* for an edge $e = (u, v) \in E$ is the quantity $slack(e) = slack(u, v) = d_e - \alpha(e) \geq 0$. An edge $e \in E$ is called *admissible* if $\alpha(e) = d_e$, that is, with a zero slack. Dual feasibility and the complementary slackness conditions imply that the edges in the (optimum) matching are admissible. We call a blossom B *active* if it is either trivial or $\alpha(B) > 0$. A biproduct of Edmonds' algorithm is that the set of active blossoms are nested. That is any two active blossoms are either disjoint or one includes the other. This allows us to represent the blossoms by the *blossom structure forest*. The trivial blossoms form the leaves of the structure. A blossom B_1 is a child of blossom B_2 in the structure, if B_2 is the smallest active blossom that properly contains B_1. The roots of the blossom structure forest will be called *maximal blossoms*.

A blossom B is called matched if $x(B) = 1$, otherwise it is called *exposed*. By implication, a vertex $v \in V$ is matched if $x_e = 1$ for some edge $e \in E$ incident to v, otherwise it is exposed. A pair of vertices (or blossoms) incident to a matched edge are called mates. The *base* of an active blossom is its unique vertex that is not matched with any other vertex of that blossom (it may be matched with a vertex in another blossom). The blossom is exposed if and only if its base is exposed.

An *alternating path* in G with respect to a matching M is a simple path in G whose edges alternate between M and not in M. An *augmenting path* in G with respect to a matching M is an alternating path between two exposed vertices of G. If there is an augmenting path P, then we can augment the matching M by changing it to $M' = M \oplus P$ (the symmetric difference of M and P). That is, we switch the status of edges of P from matched to unmatched and vice versa. It is easy to check that M' is indeed a matching and $|M'| = |M| + 1$. The converse also holds:

Fact 2.1 *A matching M in G has maximum cardinality if and only if there is no augmenting path in G with respect to M.*

Let $\hat{\mathcal{B}}$ denote the current set of maximal blossoms in the algorithm. The *shrunken graph* $G/\hat{\mathcal{B}}$ is obtained from G by contracting its maximal blossoms to single super vertices. The shrunken matching $M/\hat{\mathcal{B}}$ is defined similarly.

Fact 2.2 *There is an augmenting path in G with respect to a matching M, if and only if there is an augmenting path in $G/\hat{\mathcal{B}}$ with respect to $M/\hat{\mathcal{B}}$.*

At any point in time, $\alpha(B)$ can change only if B is a maximal blossom. The algorithm gives each maximal blossom one of the labels F, S, T. The meaning of these labels are: $label(B) = F$ means $\alpha(B)$ is fixed, $label(B) = S$ means $\alpha(B)$ is increasing, and $label(B) = T$ means $\alpha(B)$ is decreasing. A vertex inherits the label of the maximal blossom that contains it. We shall also refer to a maximal blossom as an S-blossom, a T-blossom, or an F-blossom, according to its label. At the beginning of each phase, the exposed maximal blossoms are S-blossoms, the rest are F-blossoms. To help construct an augmenting path during a phase, the algorithm maintains an *alternating forest* in the (conceptually) shrunken graph $G/\hat{\mathcal{B}}$ using only the admissible edges. The nodes of the forest are the maximal blossoms labeled S or T. The roots of the forest are exactly the exposed maximal blossoms. The path from a root to any leaf in this forest is an alternating path and the nodes alternate between S-blossoms and T-blossoms. A T-blossom has exactly one child — its S-blossom mate — (their bases are matched). Figure 1 shows an example of an alternating tree and blossoms (some of the admissible edges are shown, with the matched edges shown thicker). Now imagine for each maximal S-blossom B we increase $\alpha(B)$, and for each maximal T-blossom we decrease $\alpha(B)$, all by the same amount θ, until either a new edge becomes admissible, or $\alpha(B)$ becomes 0 for some maximal T-blossom B. The quantity θ is obtained from the following equations:

$$\begin{aligned}
\theta_{SS} &= min\ \{\ slack(u,v)/2\ \mid (u,v) \in E, u\ and\ v\ are\ in\ distinct\ maximal\ S-blossoms\} \\
\theta_{FS} &= min\ \{\ slack(u,v)\ \mid (u,v) \in E, u\ is\ an\ F-vertex,\ v\ is\ an\ S-vertex\} \\
\theta_{T} &= min\ \{\ \alpha(B)\ \mid B\ is\ a\ maximal\ T-blossom\ \} \\
\theta &= min\ \{\ \theta_{SS},\ \theta_{FS},\ \theta_{T}\ \}\ .
\end{aligned}$$

Recall that the algorithm consists of n augmenting phases. Each phase consists of a number of stages of the following types:

1. a dual variable change stage,

2. a tree growing stage,

3. a blossom deactivation (or expansion) stage,

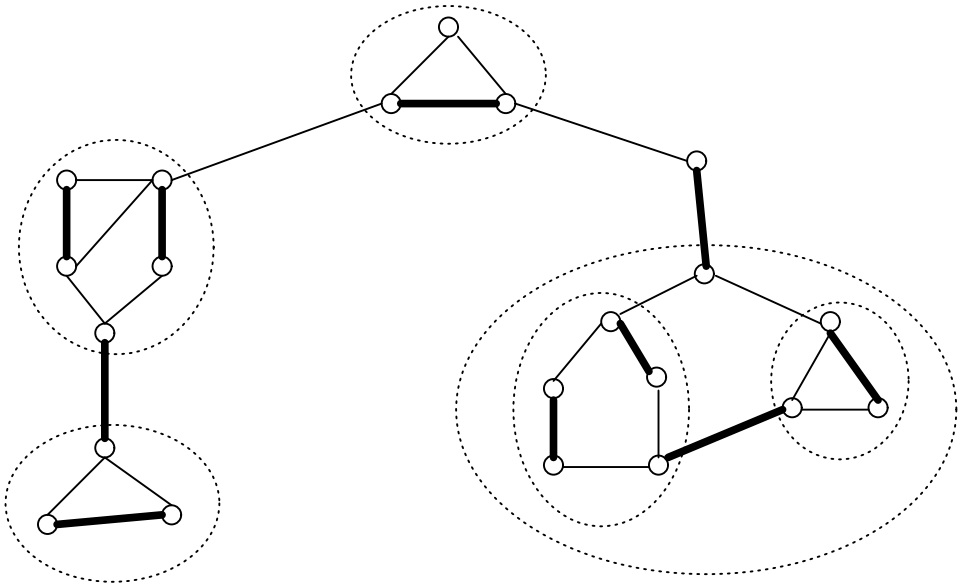

Figure 1: An alternating tree and its blossom clusters.

4. a blossom activation (or shrinking) stage, and

5. an augmentation stage.

Each phase ends with an augmentation stage. The details are as follows.

If $\theta > 0$, then we need to readjust the dual variables corresponding to maximal blossoms as follows:

$$\alpha(B) \;:=\; \begin{cases} \alpha(B) + \theta & \text{if } B \text{ is an } S\text{-blossom} \\ \alpha(B) - \theta & \text{if } B \text{ is a } T\text{-blossom} \end{cases}$$

This increases the dual objective value by θ times the number of exposed maximal blossoms. (The exposed maximal blossoms are S-blossoms and the others are either F type or matched in pairs of S type and T type). It would be too costly to make these repeated dual variable changes. To avoid this, we will use an offset for each S or T blossom similar to [28] as follows. Let Δ be the accumulated θ values for the current phase (Δ is set to 0 at the beginning of the phase). For each maximal blossom B, let $\Delta(B)$ denote the value of Δ at the time B became active most recently. Then, we let $\alpha(B)$ remain fixed, but we use the quantity $\alpha(B)+\Delta-\Delta(B)$ (respectively, $\alpha(B)-\Delta+\Delta(B)$) which is the true up to date value of $\alpha(B)$, for an S-blossom (respectively, T-blossom) B. At the end of each phase, and when B stops

being a maximal blossom (for instance, due to being shrunk into a larger blossom), we update $\alpha(B)$ by adding (subtracting) $\Delta - \Delta(B)$ to it (from it) if B is an S-blossom (T-blossom). We also set $\Delta(B) := 0$ at this point. Fact 2.3 below shows that we will need to perform such an operation for a total of only $O(n)$ times in each phase.

Otherwise, $\theta = 0$. There are three possible subcases:

The case $\theta_{FS} = 0$ corresponds to the tree growing stage. This means that an edge between an S-blossom B_S and an F-blossom B_F becomes admissible. We add B_F as a child of B_S in its alternating tree; change the label of B_F to T and change the label of its mate from F to S. We also add to the alternating tree the mate of B_F as the child of B_F and the matched edge between them.

The case $\theta_{SS} = 0$: This means that an edge between two S-blossoms B and C has become admissible. If B and C are in the same alternating tree, then a new blossom must be activated (the blossom shrinking stage). The new blossom D consists of all the ancestor blossoms of B and C in the alternating tree starting from B and C up to and including their lowest common ancestor B_o. Suppose these ancestors of B along the alternating path are $B_1 = B, B_2, \cdots, B_k, B_o$, and similarly, the ancestors of C are $C_1 = C, C_2, \cdots, C_p, B_o$. Then the alternating cycle of the new blossom D is $B_k, B_{k-1}, \cdots, B_2, B_1, C_1, C_2, \cdots, C_p, B_o$. We contract (or shrink) these blossoms into the newly activated maximal blossom D in the alternating tree, and label it S. Subblossom B_o becomes the base of D. Accordingly, we introduce a new root in the blossom structure forest corresponding to the newly activated blossom D. We make its children those old roots that are no longer maximal and correspond to its subblossoms. The subblossoms loose their labels.

On the other hand, if the two S-blossoms B and C are in two different alternating trees, then we have found an augmenting path, namely the path that consists of the admissible edge between B and C, and the two alternating paths from, respectively, B and C up to their roots in the alternating forest. In the latter case, we augment the matching and end the current phase.

The case $\theta_T = 0$ corresponds to the blossom deactivation (or blossom expansion) stage. This occurs when $\alpha(B)$ has become 0 for some T-blossom B. We need to expand B. This

```
M := ∅; set dual variables and vertex weights to zero;
initialize the blossom structure forest by setting each
singleton vertex as a maximal blossom;
for phase := 1, ···, n do
    initialize the alternating forest for this phase;
    augmented := false;
    while not augmented do
        calculate θ_SS, θ_FS, θ_T, θ;
        case:
            θ > 0 : UPDATE DUAL VARIABLES;
            θ_T = 0: DEACTIVATE BLOSSOM;
            θ_FS = 0: GROW ALTERNATING TREE;
            θ_SS = 0: if the two S-blossoms incident to the new
                      admissible edge are in the same alternating tree
                then ACTIVATE BLOSSOM
                else AUGMENT MATCHING ; augmented := true
    end{while}
end{for}
```

Figure 2: A summary of Edmonds' Minimum Weight Matching Algorithm.

needs restructuring of both the blossom structure forest and the alternating forest. The root B in the blossom structure forest is removed. This causes its previous children to become new roots in the forest. These new roots are now maximal blossoms. In the alternating tree containing B, we similarly expand B. This expansion can be described as follows. First replace B by its alternating cycle of subblossoms. This alternating cycle is partitioned into two chains delimited by the connection to the old parent of B and child of B. One of the chains has odd length, the other one even. Remove the chain with odd length from the alternating tree and relabel the (now) maximal blossoms on it as F-blossoms. The (now) maximal blossoms on the remaining chain of the alternating cycle are relabeled as S-blossoms and T-blossoms so as to maintain the alternating property of the tree.

Figure 2 shows a summary of Edmonds' algorithm.

Fact 2.3 ([28,54]) *Within each of the n phases of the algorithm, the following quantities are $O(n)$: the number of alternating tree growings, blossom deactivations, blossom activations, dual variable changes, the total number of different maximal blossoms, the number of times $\theta_{SS}, \theta_{FS}, \theta_T$,* and *$\theta$ are computed.*

Proof sketch: These mainly follow from the nested structure of the active blossoms and the fact that within a phase each S-blossom corresponds to a unique node in the blossom structure forest at the end of the phase, and each T-blossom corresponds to a unique node in the blossom structure forest at the beginning of the phase. □

Let W be any subset of V. We define $\rho(W) = \Sigma\{\ \alpha(B) \mid W \subseteq B \in \mathcal{B}\}$. Let $lca(u,v)$ denote the blossom that is the lowest common ancestor of vertices u and v in the blossom structure forest. Assume the lowest common ancestor is V if u and v are not in the same tree of the forest. (Note that $\rho(V) = 0$.)

Lemma 2.4 *Consider an arbitrary edge $e = (u,v) \in E$. Then, $\alpha(e) = \rho(u) + \rho(v) - 2\rho(lca(u,v))$, where $\rho(x) = \Sigma\{\ \alpha(B) \mid x \in B\}$ for a vertex x. Thus, if u and v are not in the same active blossom, then $\alpha(e) = \rho(u) + \rho(v)$.*

Proof: This easily follows from the nested structure of active blossoms. □

We represent blossoms, as ordered sets of vertices, by an *offsetted concatenable queue* explained below. (This is the same as priority queue p.q.$_1$ in [28].) The linear ordering is according to the order of subblossoms on the alternating cycle when the blossom was activated. The base of the blossom is considered the last subblossom in this ordering. The same ordering is inductively applied to subblossoms, in the blossom structure forest. We need to perform the following operations on the offsetted concatenable queues:

- $find(v)$: find the maximal blossom containing vertex v,
- $eval(v)$: return up-to-date value of $\rho(v)$ (including the offset $\pm(\Delta - \Delta(B))$ if B, the maximal blossom containing v, is an S-blossom or a T-blossom).

- *concatenate*(B_1, B_2) : concatenate the two maximal blossoms B_1 and B_2,
- *split*(B, v) : split maximal blossom B and its linear ordering at vertex v into two disjoint offsetted concatenable queues.

We need the operation $find$ for deciding whether two ends of an edge belong to the same maximal blossom. The second operation is also needed for edge-slack computations when the two vertices are not in the same blossom. Operations *concatenate* and *split* are required for blossom expansion and shrinking and for augmentation. We can implement an offsetted concatenable queue by 2-3 trees whose leaves correspond to the vertices of the blossom in its linear order [4]. This will allow us to perform $find$, *concatenate* and *split* in $O(\log n)$ time. We also maintain an additional real number in each node of the tree so that the sum of the numbers on the path from the root to any leaf v is $\rho(v)$. We can accomplish this by using the idea of [28]. In this way, the operation *eval* can also be performed in $O(\log n)$ time. A blossom expansion or shrinking involving r subblossoms can be performed in $O(r \log n)$ time (for a total of $O(n \log n)$ time per phase by Fact 2.3). Also, an augmentation stage may cause the base of the blossoms on the augmenting path to be shifted around its alternating cycle. This can be implemented by a split and concatenate on the affected blossoms to maintain their appropriate linear ordering. There is only one augmentation per phase and it can be performed in $O(n \log n)$ time. We also maintain T-blossoms in a priority queue for logarithmic computation of θ_T. We conclude:

Theorem 2.5 *Excluding the maintenance of θ_{FS} and θ_{SS}, Edmonds' algorithm can be implemented to run in $O(n \log n)$ time per phase, $O(n^2 \log n)$ time total, and $O(n)$ space.*

2.3 The Euclidean case

The remaining issue is how to maintain θ_{SS} and θ_{FS} efficiently. The rest of the paper concentrates on this issue for the Euclidean case. So, now G is a complete graph on $2n$ planar point sites as its vertices, and $d(u, v)$ is the Euclidean distance between sites u and v. In the next section we will show that in the Euclidean case $\rho(v)$ is nonnegative for every vertex v. Think of $\rho(v)$ as the weight, or radius of a circular disk centered at

vertex v. Let the *weighted distance* between a pair of sites u and v be the slack value $slack(u,v) = d(u,v) - \alpha(u,v)$. When u and v are not in the same active blossom, then by Lemma 2.4 we have $slack(u,v) = d(u,v) - \rho(u) - \rho(v) \geq 0$. The efficient maintenance of θ_{FS} and θ_{SS} suggests the following two query processing problems respectively:

Problem 1. (*Bichromatic weighted closest-pair maintenance*):
We have two disjoint subsets of the $2n$ point sites: S (red) and F (blue). we want to maintain the minimum weighted red-blue distance dynamically, when weighted red points can be added to S and weighted blue points can be added to or deleted from F.

Problem 2. (*Unichromatic weighted closest-pair maintenance*):
We have a set of at most $2n$ planar point sites S with the nonnegative weight $\rho(v)$ for site v. Think of these as circular disks with radius $\rho(v)$ centered at v. These disks form a number of connected components in the plane. Two sites belong to the same active blossom if and only if their disks belong to the same connected component (see the next section). The problem is to maintain the minimum (nonnegative) weighted distance between connected components dynamically subject only to site insertions.

These are modified versions of the two query processing problems posed by Vaidya [54]. Vaidya was able to show that these two problems can be solved by maintaining data structures so that each of the individual operations can be performed in $O(\sqrt{n}\ polylog(n))$ amortized time, and $O(n \log n)$ space. This resulted in his $O(n^2\sqrt{n}\ polylog(n))$ time algorithms for both the bipartite and the nonbipartite Euclidean Minimum Weight Matching problems. Recently, as noted in [7], new developments in dynamic bichromatic closest-pair algorithms further reduces the running time of Vaidya's bipartite algorithm to $O(n^{2+\epsilon})$, for any $\epsilon > 0$, with the constant of proportionality depending on ϵ [1,2]. Can Problem 2 also be solved in $o(\sqrt{n})$ per operation? Can Problems 1 or 2 be solved in $O(polylog(n))$ time per operation? The main strategy of this paper is developed in the next section. Here let us briefly offer a possible alternative about tackling Problem 2. Maintain the Additively Weighted Voronoi Diagram of the sites in S. Since the weights of all vertices in S grow at the same rate, the Weighted Voronoi diagram remains fixed during a phase, except the only changes occur when new sites

are inserted into S with given weights. How fast can we insert new weighted sites in the diagram? We in fact need the dual of the diagram — the Weighted Delaunay Diagram. To maintain the closest pair, we label the $O(n)$ edges of the dual depending on whether the two end points are in the same S-blossom or not. Maintain the edges whose two ends are in different S-blossoms in a priority queue, with their weighted distances as their priorities. In this way, we can maintain the closest-pair in logarithmic time. The only open question is how fast can we insert new sites in the diagram. For the related randomized problem in the unweighted case see [15,29]. Eventhough in the worst case insertion of $O(n)$ sites may cause $\Omega(n^2)$ combinatorial changes in the diagram even in the unweighted case, such worst cases may not occur in the matching algorithm, due to a fare amount of locality and clustering. We leave this as an open problem. In the rest of the paper we develop an alternative strategy, as well as establish the necessary geometric ideas.

3 New Geometric Results

3.1 The Euclidean Linear Program

The main result of this subsection is the following theorem which removes the distinction between the trivial and nontrivial blossoms in the linear program for the minimum weight matching in the Euclidean case.

Theorem 3.1 *In the minimum weight matching problem on a complete graph if the edge weights form a distance metric, such as the EMWM problem, then the dual variables corresponding to the vertices (that is, trivial blossoms) remain nonnegative in Edmonds' algorithm.*

Proof: A proof based on the primal linear program is possible. Here we give a proof based on the dual program and Edmonds' algorithm. Let us first consider EMWM; the generalization to other cases would be straightforward. Suppose to the contrary that for some vertex v, $\alpha(v)$ becomes negative in Edmonds' algorithm. Consider the first time that this occurs. Just prior to that time, v must have been a maximal T-blossom. At this point $\rho(v) = \alpha(v) > 0$, and v is incident to two maximal active S-blossoms B_1 and B_2 via two admissible edges. These are the parent and the unique child of v in its alternating tree. Suppose these two admissible

edges are (v,u) and (v,w), with $u \in B_1$ and $w \in B_2$. At this point the three disks centered at v,u,w have positive radii $\rho(v),\rho(u),\rho(w)$. Furthermore,

$$\begin{aligned} d(v,u) &= \rho(v)+\rho(u)\ , \\ d(v,w) &= \rho(v)+\rho(w)\ , \\ d(u,w) &\geq \rho(u)+\rho(w)\ . \end{aligned}$$

These follow from Lemma 2.4, the dual feasibility, and the fact that edges (v,u) and (v,w) are admissible. Also, v is a T-vertex and its radius decreases, while u and w are S-vertices and their radii are increasing. This progress will stop when the third inequality in equation 1 becomes equality, before $\rho(v)$ becomes negative. That is, at this point the circles centered at the triple (v,u,w) will form Apollonius circles (that is, three pairwise externally tangent circles [13,14]) with radii

$$\begin{aligned} \rho(v) &= (d(v,u)+d(v,w)-d(u,w))/2\ , \\ \rho(u) &= (d(u,v)+d(u,w)-d(v,w))/2\ , \\ \rho(w) &= (d(w,v)+d(w,u)-d(v,u))/2\ . \end{aligned}$$

These are nonnegative radii since the edge distances satisfy the triangle inequality (and the other metric axioms). Note that at this point $d(u,w)=\rho(u)+\rho(w)$. Thus, the edge (u,w) between two distinct S-blossoms B_1 and B_2 has become admissible. So, v, B_1 and B_2 will be shrunk into a new S-blossom, and $\alpha(v)=\rho(v)\geq 0$ will be fixed. A contradiction.

The generalization to other metric distances is now obvious; use the same argument emphasizing the above equations without mentioning circles. □

Thus, without loss of generality, we can add the constraints

$$\alpha(B) \geq 0 \quad (for\ each\ trivial\ blossom\ B\ )$$

to Edmonds' dual linear program. We thus obtain the following simplified primal-dual linear programs for EMWM:

$$\begin{array}{rll} minimize & d^T \cdot x & \\ subject\ to: & x_{uv} \geq 0 & (for\ each\ pair\ of\ sites\ u,v) \\ & x(B) \geq 1 & (for\ each\ blossom\ B)\ . \end{array}$$

$$
\begin{array}{rll}
\textit{maximize} & \Sigma_{B \in \mathcal{B}}\, \alpha_B & \\
\textit{subject to}: & \alpha_B \geq 0 & (\textit{for each blossom } B) \\
& \alpha(u,v) \leq d(u,v) & (\textit{for each pair of sites } u,v)\ .
\end{array}
$$

Now, dual feasibility implies $\rho(v) \geq \alpha(v) \geq 0$ for each vertex v.

3.2 Disks and blossoms in EMWM

Let $disk(v)$ denote the circular disk centered at v with the nonnegative radius $\rho(v)$. For each active blossom B associate the planar region $Region(B) = \bigcup_{v \in B}\ disk(v)$.

Lemma 3.2 *Suppose Edmonds' algorithm is applied to EMWM. Then, for each active blossom B (ie, $\alpha(B) > 0$), the interior of $Region(B)$ is connected.*

Proof: The proof follows by an easy induction on the cardinality of the blossoms. If B is a trivial active blossom, then $Region(B)$ is a disk and its interior is obviously connected. Otherwise, consider when B is formed from subblossoms when it is activated. By induction the subblossoms satisfy the assertion of the lemma. When B is activated, $Region(B)$ becomes path connected since the admissible edges along its alternating cycle connect its subblossom regions. At this point the only possible disconnection points for interior of $Region(B)$ are where two regions of subblossoms become tangent to each other. However, when $\alpha(B) > 0$, the abutting disks from its different subblossoms overlap and make the interior of $Region(B)$ also connected. □

Lemma 3.3 *Consider any two distinct maximal blossoms B_1 and B_2. Then, the interiors of $Region(B_1)$ and $Region(B_2)$ are disjoint.*

Proof: Otherwise, there must be vertices $v \in B_1$ and $u \in B_2$ such that $disk(v)$ and $disk(u)$ overlap. But then, $d(u,v) < \rho(u) + \rho(v) = \alpha(u,v)$, since u and v are not in the same active blossom. This violates dual feasibility. □

Corollary 3.4 *Two vertices are in the same maximal active blossom if and only if they are in the same connected component of the interior of $\bigcup_{v \in V}\ disk(v)$.*

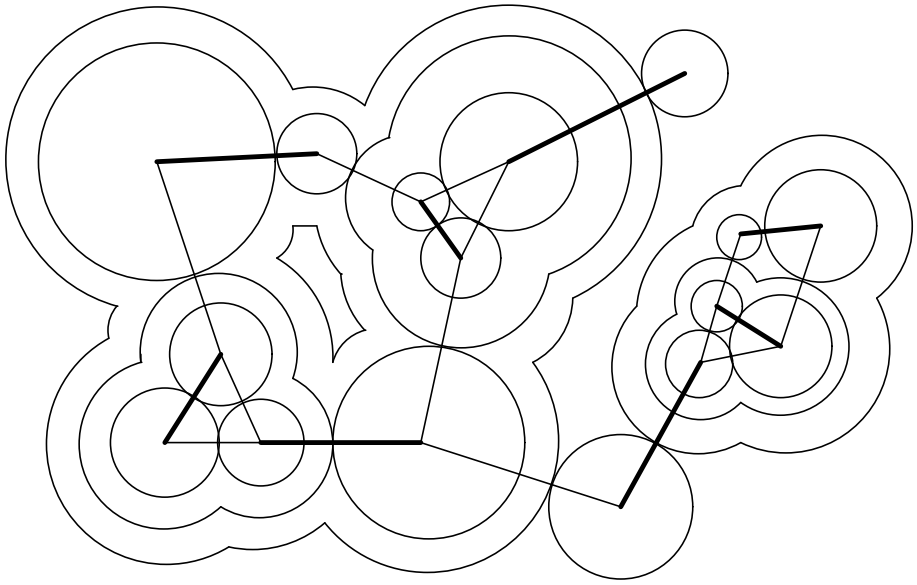

Figure 3: Blossom clusters in EMWM.

Theorem 3.5 *During Edmonds' algorithm applied to EMWM, there is no pair of distinct vertices u and v such that disk(u) is included in the interior of disk(v). That is, $|\rho(u)-\rho(v)| \leq d(u,v)$, for each pair of sites u and v.*

Proof: By dual feasibility and Lemma 2.4, at all times we have $d(u,v) \geq \rho(u) + \rho(v) - 2\rho(lca(u,v))$. Let us define $r_u = \rho(u) - \rho(lca(u,v))$ and $r_v = \rho(v) - \rho(lca(u,v))$. Thus, $d(u,v) \geq r_u + r_v$. We note that $r_u = \Sigma\{\ \alpha(B) \mid u \in B \ \subset lca(u,v)\ \} \geq 0$. Similarly, $r_v \geq 0$. Therefore, the two disks centered at u and v with radii, respectively, r_u and r_v have disjoint interiors. Hence, $|r_u - r_v| \leq d(u,v)$. By adding $\rho(lca(u,v))$ to these two radii, we obtain $|\rho(u) - \rho(v)| \leq d(u,v)$. □

Figure 3 shows the clustering structure of blossoms and subblossoms in an example. The admissible edges are shown in straight line-segments, the bold ones are matched edges.

3.3 The Weighted Voronoi Diagram, Gabriel and Relative Neighborhood Graphs

Weighted Voronoi Diagram: Let $\rho(s)$ be the weight of each site s. For any point x in the plane, the *weighted distance* of x from site s is defined as $\delta_s(x) = d(x,s) - \rho(s)$. Consider

the location of the points that have smaller or equal weighted distance from site u than v. This region is empty if $\rho(u) < \rho(v) - d(u,v)$. Otherwise, it includes u and is bounded by the bisector of u and v, denoted by $H(u,v)$, which is one branch of a hyperbola with foci u and v. This bisector is bending towards the site with smaller weight (it is a line if both sites have equal weight and it is a half line if $|\rho(u) - \rho(v)| = d(u,v)$). This region (if nonempty) is star shaped and u is one of its kernel points. The *Weighted Voronoi cell* (or Voronoi region) of site s, $Vor(s)$, is the set of points in the plane that are at least as close (in the sense of weighted distance) to s than to any other site. A Weighted Voronoi region $Vor(s)$ is star shaped with s as one of its kernel points, and its boundary consists of a chain of hyperbolas. The subdivision of the plane by Weighted Voronoi regions of the sites is their *Weighted Voronoi Diagram* (WVD). The vertices and edges of the subdivision are called Voronoi vertices and Voronoi edges, respectively. In the absence of degeneracy, Voronoi vertices have degree three. For more details on the general properties of WVD see for example [5,6,20,31,48].

Remark: *In this paper the weights are not entirely general due to Theorem 3.5. Some of the properties we prove here do not hold for general weights.*

Weighted Delaunay Diagram: The WDD is the topological dual of the Weighted Voronoi diagram. The sites are the vertices of WDD, and there is an edge between a pair of sites u and v, if $Vor(u)$ and $Vor(v)$ share a common boundary edge. In the unweighted case (excluding degenerate cases) WDD is a straight-line triangulation of the sites and is called Delaunay Triangulation. In the weighted case, WDD is possibly a multi-graph and can be drawn quasi-straight-line as follows. In the weighted case, $Vor(u)$ (if non-empty) is not necessarily convex but it is star-shaped with site u one of its kernel points (that is, the line segment between u and any point $x \in Vor(u)$ does not intersect the exterior of $Vor(u)$). First suppose $Vor(u)$ has a non-empty interior. Draw a line-segment between u and each Voronoi vertex on the boundary of $Vor(u)$. These segments partition $Vor(u)$ into sectors. We can associate each sector with the boundary edge of $Vor(u)$ it contains. This boundary edge is called the *base* of the sector. Now, if $Vor(u)$ and $Vor(v)$ share a common boundary edge, then consider the

sectors of $Vor(u)$ and $Vor(v)$ with the common base. If the common base does not intersect the line segment (u, v), then draw the edge between u and v as two connected straight line segments within the two sectors with the connection point on the common base. Draw the edge as straight-line, if the segment (u, v) intersects the common base. The common boundary between a pair of Voronoi regions may consist of several bases. In that case we will have the corresponding multiple edges in the WDD. In the absence of degeneracy, and if none of the Voronoi regions are empty, this drawing gives a quasi-straight-line topological triangulation (that is, each bounded face is incident to three sites). A site u is on the boundary of the unbounded (exterior) face, if and only if $disk(u)$ touches the boundary of the convex-hull of $\bigcup_{v \in V} disk(v)$. In the presense of degeneracy, a Voronoi region might be a half-line or a line segment, and some internal faces may not be triangles. These cases can be resolved by slight perturbation: in the first case thicken the half-line or the line segment slightly to endow it an interior; in the second case triangulate these non-triangular faces by adding to them a maximal number of non-crossing chords. For additional properties of Voronoi Diagrams and Weighted Delaunay Diagrams and how to compute them in $O(n \log n)$ time see [20,48].

Weighted Gabriel Graph: The WGG is the straight-line graph whose vertices are the given sites, and the line segment between sites u and v is an edge of the graph if and only if the (closed) line segment (u, v) intersects the bisector $H(u, v)$ and this intersection point c_{uv} is on the common boundary of $Vor(u)$ and $Vor(v)$. In other words, let $c_{uv} = (u, v) \cap H(u, v)$ (if it exists). Then (u, v) is an edge of the WGG, if and only if c_{uv} is closest (in terms of weighted distance) to u and v than to any other site (except the degenerate case mentioned below). Point c_{uv} is called the *Weighted Gabriel Center* of sites u and v. The following is now obvious.

Theorem 3.6 *The Weighted Gabriel Graph is a subgraph of the Weighted Delaunay Diagram.*

A degenerate case: Consider a quadruple of sites (a, b, c, d) such that (a, b) and (c, d) cross each other at a point x and $c_{ab} = c_{cd} = x$. In this case x has the same weighted distance from

all four sites a, b, c, d. This is considered a degenerate case. In WGG if the ends of a pair of edges form the degenerate case above, we remove one of these two edges. If, in addition, sites a, b, c, d are collinear, then we remove edge (a, b) if (as a line segment) it is not included in (c, d); similarly, we remove edge (c, d) if it is not included in (a, b). □

Below, we will prove some additional properties of the Gabriel Graphs (and later Weighted Relative Neighborhood Graphs) when the weights satisfy Theorem 3.5.

Lemma 3.7 *The Weighted Gabriel Center c_{uv} of each pair of sites u and v exists and is on the line segment (u, v).*

Proof: Select the point p on the line segment (u, v) such that $d(u, p) = (d(u, v) + \rho(u) - \rho(v))/2$. By Theorem 3.5 $|\rho(u) - \rho(v)| \leq d(u, v)$. Thus, p is indeed on the line segment (u, v). Also, $\delta_u(p) = \delta_v(p) = (d(u, v) - \rho(u) - \rho(v))/2$. Therefore, $p = c_{uv}$ is the desired point. □

Lemma 3.8 *For each site u, $Vor(u)$ is a nonempty star shaped region and u is one of its kernel points.*

Proof: From Lemma 3.7, the bisector $H(u, v)$ of any pair of sites u and v intersects the line-segment (u, v). Hence, u is at least as close (in weighted distance) to itself as to v. This implies $u \in Vor(u)$, and hence, $Vor(u)$ is nonempty. Furthermore, The region of the plane bounded by $H(u, v)$ that includes u is star shaped and u is its kernel point. Since $Vor(u)$ is the intersection of such regions, itself is star shaped and u is its kernel point. □

In fact, we prove a stronger result below.

Theorem 3.9 *The Weighted Gabriel Graph is a connected straight-line planar graph and spans all the sites.*

Proof: The fact that WGG is straight-line planar follows from the quasi-linear drawing and planarity of WDD and Theorem 3.6. Now suppose to the contrary that WGG is not connected. Consider a pair of sites u and v that are disconnected in WGG. Select u and v such that $\delta_u(c_{uv}) = (d(u, v) - \rho(u) - \rho(v))/2$ is minimum possible. If the choice is not unique, select

a pair (u, v) among them such that $d(u, v)$ is minimum. If the choice is still not unique, select a pair (u, v) among them such that the minimum x-coordinate of u or v is as small as possible (break the tie arbitrarily). If there is no other site w, such that $\delta_w(c_{uv}) \leq \delta_u(c_{uv})$, then by definition, the line segment (u, v) is an edge in WGG, a contradiction. Now, assume there is a site w such that $\delta_w(c_{uv}) \leq \delta_u(c_{uv})$. Without loss of generality assume w is not collinear with u and v, or the above inequality is strict (otherwise, by the degeneracy convention, (u, v) would still be an edge in the graph, a contradiction). Then $d(u, w) < d(u, c_{uv}) + d(w, c_{uv})$. Thus, $2\delta_u(c_{uw}) = d(u, w) - \rho(u) - \rho(w) < \delta_u(c_{uv}) + \delta_w(c_{uv}) \leq 2\delta_u(c_{uv})$. Thus, $\delta_u(c_{uw}) < \delta_u(c_{uv})$. Similarly, $\delta_v(c_{vw}) < \delta_v(c_{uv})$. Then, by the selection criterion of (u, v), we conclude that WGG contains a path between u and w and a path between v and w. Thus, u and v are connected, a contradiction. □

Weighted Relative Neighborhood Graph: The WRNG, which is also a straight-line graph with the given sites as vertices, is defined as follows. Let $\delta(a, b) = d(a, b) - \rho(a) - \rho(b)$ be the (symmetric) relative distance between sites a and b. The line segment (a, b) is an edge of WRNG if and only if $\delta(a, b) \leq max\{\delta(a, c), \delta(b, c)\}$, for any other site c.

Theorem 3.10 *The Weighted Relative Neighborhood Graph is a subgraph of the Weighted Gabriel Graph.*

Proof: Suppose (a, b) is an edge of WRNG. Then, for any other site c we have $\delta(a, b) \leq max\{\delta(a, c), \delta(b, c)\}$. Thus,

$$\begin{aligned} \delta_c(c_{ab}) &= d(c, c_{ab}) - \rho(c) \\ &\geq d(a, c) - d(a, c_{ab}) - \rho(c) \\ &= \delta(a, c) - \delta(a, b)/2 \ . \end{aligned}$$

Similarly, $\delta_c(c_{ab}) \geq \delta(b, c) - \delta(a, b)/2$. Thus,

$$\begin{aligned} \delta_c(c_{ab}) &\geq max\{\delta(a, c), \delta(b, c)\} - \delta(a, b)/2 \\ &\geq \delta(a, b)/2 \\ &= \delta_a(c_{ab}) \ . \end{aligned}$$

This implies (a, b) is an edge in WGG. □.

Corollary 3.11 *The Weighted Relative Neighborhood Graph is a straight-line planar subgraph of the Weighted Delaunay Diagram.*

Proof: Follows from Theorems 3.6, 3.9 and 3.10. □

3.4 The Admissible Edges

In this subsection we establish some connection between the admissible edges and the structures discussed in the previous subsection, such as Weighted Relative Neighborhood Graphs.

Lemma 3.12 *Consider any site v during the algorithm. Initially v is an exposed S-vertex. After a while v acquires an incident admissible edge for the first time. From that point on v maintains at least one incident admissible edge.*

Proof: Initially every site v is an exposed S-vertex and hence $\rho(v)$ keeps increasing. Therefore, eventually some edge (v, u) incident to v must become admissible. The only way this edge can become inadmissible later is when both u and v are T-vertices in different blossoms. However, any T-vertex is already matched, and matched vertices do not become exposed. The matched edge incident to v is an admissible edge. □

Recall that $\rho(a, b) = \rho(lca(a, b)) = \Sigma\{\rho(B) \mid a \in B, b \in B\}$. Furthermore, by Lemma 2.4 and dual feasibility, we have $d(a, b) \geq \alpha(a, b) = \rho(a) + \rho(b) - 2\rho(a, b)$. So, for an admissible edge (a, b) we have $\delta_a(c_{ab}) = -\rho(a, b) = (d(a, b) - \rho(a) - \rho(b))/2$. Consider applying Edmonds' algorithm to EMWM. We have the following:

Lemma 3.13 *For any three sites a, b, c, we have $\rho(a, b) \geq min\{\rho(a, c), \rho(b, c)\}$*

Proof: Here we use the nested structure of the blossoms, that is, the blossom structure forest. To simplify the discussion, we convert the blossom structure forest to a tree by adding the superficial "blossom" V as the root and making all maximal blossoms its children. Now every blossom is a descendent of the new root in the tree. Consider the subtree rooted at

$lca(a,b)$. If c is outside this subtree, then $lca(a,b)$ is a descendent of $lca(a,c) = lca(b,c)$. Hence, $\rho(a,b) \geq \rho(a,c) = \rho(b,c)$. If c is in the subtree, there are two case: (i) $lca(a,c)$ is a descendent of $lca(a,b) = lca(c,b)$. In that case $\rho(a,c) \geq \rho(b,c) = \rho(a,b)$. (ii) $lca(b,c)$ is a descendent of $lca(a,c) = lca(a,b)$. In that case $\rho(b,c) \geq \rho(a,c) = \rho(a,b)$. In all cases $\rho(a,b) \geq min\{\rho(a,c), \rho(b,c)\}$. □

Theorem 3.14 *Admissible edges form a subgraph of the Weighted Relative Neighborhood Graph.*

Proof: Let (a,b) be an admissible edge. Let c be any other site. By Lemma 3.13 we have

$$\begin{aligned}
\delta(a,b) &= d(a,b) - \rho(a) - \rho(b) \\
&= -2\rho(a,b) \\
&\leq max\{-2\rho(a,c), -2\rho(b,c)\} \\
&\leq max\{d(a,c) - \rho(a) - \rho(c), d(b,c) - \rho(b) - \rho(c)\} \\
&= max\{\delta(a,c), \delta(b,c)\} \ .
\end{aligned}$$

This implies (a,b) is an edge of WRNG. □

Corollary 3.15 *At any time during the algorithm there are only $O(n)$ admissible edges and they are non-crossing.*

Proof: By Theorems 3.9, 3.10, 3.14, and the convention on resolving the degenerate cases. □

Remark: A necessary condition for optimality is that the Weighted Relative Neighborhood Graph must contain a perfect matching (the alleged optimum matching). The WRNG, for arbitrary weights, may not necessarily contain a perfect matching. So, an open question is to find out under what condition on the weights does the corresponding WRNG contain a perfect matching. Also, how does WRNG change as the weights change? □

3.5 The Edge-Flip Criterion

In this subsection we show how to compute the θ change needed before an edge-flip in WDD occurs. There are two kinds of flips possible and are explained below. Let the triple (x_i, y_i, r_i) denote a circle C_i with radius r_i whose center is at Cartesian coordinates (x_i, y_i). We say three circles C_i, $i = 1, 2, 3$, are *collinear*, if there is a line tangent to the three given circles, and all three circles are on the same side of the line. Also, we say four circles C_i, $i = 1, 2, 3, 4$, are *cocircular*, if there is a circle C tangent to the four given circles, and all four circles are on the same side of C, that is, all external tangents, or all internal tangents to C. (Cocircularity condition corresponds to the degenerate case mentioned earlier with respect to Weighted Delaunay and Voronoi Diagrams.) We need the following two lemmas.

Lemma 3.16 *Three circles $C_i = (x_i, y_i, r_i)$, $i = 1, 2, 3$, are collinear, only if $A^2 - B^2 - C^2 = 0$, where*

$$A = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}, \quad B = \begin{vmatrix} r_1 & y_1 & 1 \\ r_2 & y_2 & 1 \\ r_3 & y_3 & 1 \end{vmatrix}, \quad C = \begin{vmatrix} r_1 & x_1 & 1 \\ r_2 & x_2 & 1 \\ r_3 & x_3 & 1 \end{vmatrix} .$$

Proof: Let the equation of the line tangent to the three circles be $ax + by + c = 0$. Suppose this equation is normalized so that $a^2 + b^2 = 1$. Now consider the signed distance of the centers of the three circles from this line. We get the three equations $ax_i + by_i + c = r_i$, for $i = 1, 2, 3$. Using Cramer's rule, we obtain $a = B/A$, $b = -C/A$. Substitute these in the normalization condition $a^2 + b^2 = 1$ to obtain the lemma. □

Lemma 3.17 *Four circles $C_i = (x_i, y_i, r_i)$, $i = 1, 2, 3, 4$, are cocircular, only if*

$$4AB + C^2 + D^2 - E^2 = 0 ,$$

where

$$A = \begin{vmatrix} 1 & x_1 & y_1 & r_1 \\ 1 & x_2 & y_2 & r_2 \\ 1 & x_3 & y_3 & r_3 \\ 1 & x_4 & y_4 & r_4 \end{vmatrix}, \quad B = \begin{vmatrix} z_1 & x_1 & y_1 & r_1 \\ z_2 & x_2 & y_2 & r_2 \\ z_3 & x_3 & y_3 & r_3 \\ z_4 & x_4 & y_4 & r_4 \end{vmatrix}, \quad C = \begin{vmatrix} 1 & z_1 & y_1 & r_1 \\ 1 & z_2 & y_2 & r_2 \\ 1 & z_3 & y_3 & r_3 \\ 1 & z_4 & y_4 & r_4 \end{vmatrix},$$

$$D = \begin{vmatrix} 1 & z_1 & x_1 & r_1 \\ 1 & z_2 & x_2 & r_2 \\ 1 & z_3 & x_3 & r_3 \\ 1 & z_4 & x_4 & r_4 \end{vmatrix}, \quad E = \begin{vmatrix} 1 & z_1 & x_1 & y_1 \\ 1 & z_2 & x_2 & y_2 \\ 1 & z_3 & x_3 & y_3 \\ 1 & z_4 & x_4 & y_4 \end{vmatrix}, \ and \ \ z_i = x_i^2 + y_i^2 - r_i^2 \ .$$

Proof: Suppose the (unknown) circle tangent to the given four circles C_i is (x, y, r), where r is positive for external tangency and negative for internal tangency. Then, the cocircularity condition is

$$(x - x_i)^2 + (y - y_i)^2 = (r + r_i)^2 \quad , \quad for \ i = 1, 2, 3, 4 \ .$$

To linearize these equations, define the new variable z as

$$z = x^2 + y^2 - r^2 \ .$$

Then, we have the four linear equations

$$-z/2 + x_i x + y_i y + r_i r = z_i/2 \quad , \quad for \ i = 1, 2, 3, 4 \ .$$

Using Cramer's rule, we obtain the solutions $z = -B/A, x = C/2A, y = -D/2A, r = E/2A$. Substitute these in the condition $z = x^2 + y^2 - r^2$ to obtain the lemma. □

As the site weights change, the Weighted Delaunay Diagram changes by edge-flips. There are two kinds of edge-flips (corresponding to Lemmas 3.16 and 3.17).

One kind is when an unbounded Voronoi edge appears or disappears. This happens when the Voronoi vertex incident to the unbounded Voronoi edge moves to (or from) infinity. This occurs when the disks of the three sites that share that Voronoi vertex on their common boundary become collinear. The corresponding condition is given in Lemma 3.16. In Lemma 3.16 the three circles $C_i = (x_i, y_i, r_i)$ are the associated three sites, where r_i is $\rho(i)$ if site i is an F-vertex, $\rho(i) + \theta$ if i is an S-vertex, and it is $\rho(i) - \theta$ if i is a T-vertex. We substitute these values in the stated condition of the lemma and compute the smallest positive value of θ. It should not be hard to see that the condition is a second degree polynomial in θ. The corresponding event in WDD is called an edge-flip of the first kind: the edge of WDD corresponding to the infinite edge in WVD is either deleted or added.

The second kind is when a bounded Voronoi edge shrinks to zero length, that is, its two incident Voronoi vertices coincide. The corresponding condition is given in Lemma 3.17. Similar to the above, this involves solving the stated condition of Lemma 3.17 for the unknown θ. The corresponding event in WDD is called an edge-flip of the second kind. In general, the condition becomes a 6-th degree polynomial in θ. However, if the four sites involved are of at most 2 types (from among the 3 possible types: T, S, F), then the polynomial is of degree 4, since it involves computing the intersection of two known hyperbolas. It is only when all 3 types are present (that is, an S-vertex, a T-vertex, an F-vertex, and one other vertex of any type) that the degree of the polynomial is 6. We will assume that this equation can be solved for the smallest positive root θ in $O(1)$ time. (Implicitly we are also making an assumption that extracting square roots is also done in $O(1)$ time in order to compute inter-point distances.)

4 The New Algorithm

The proposed new algorithm for EMWM is the following modification of Edmonds' algorithm. During the initialization we also construct, in $O(n \log n)$ time, the (unweighted) Weighted Delaunay Diagram (since site weights are zero at this point). As the weights change, the WDD changes when an edge-flip occurs as discussed in the previous subsection. Consider an edge (a, b) of WDD that is incident to two triangles (a, b, c) and (a, b, d). Flipping edge (a, b) means replacing it with edge (c, d). This occurs when the circles corresponding to the quadruple (a, b, c, d) become cocircular. Suppose $\theta(a, b) \geq 0$ is the minimal change needed after which edge (a, b) of WDD should be flipped (an edge-flip of the second kind). Similarly, let $\theta(a, b) \geq 0$ be the minimal change needed after which edge (a, b) must go through an edge-flip of the first kind. Let us modify the definition of θ as follows:

$$\begin{aligned} \theta_V &= min \{ \ \theta(a,b) \ | \ (a,b) \ is \ an \ edge \ of \ WDD \ \} \\ \theta &= min \{ \ \theta_V, \theta_{SS}, \theta_{FS}, \theta_T \ \} \ . \end{aligned}$$

We have already discussed how to maintain θ_T. By Fact 2.3, we perform only $O(n)$

operation on θ_T per phase, for a total of $O(n \log n)$ time per phase. Here θ_V is the minimal amount of weight change needed before an edge-flip occurs. (The subscript V stands for Voronoi.) We maintain each edge (a, b) of WDD, in a priority queue, called PQ_V, with $\theta(a, b)$ as its priority. Thus, we can compute θ_V and execute an edge-flip event in $O(\log n)$ time. When an edge-flip occurs, the neighboring four edges in WDD must also be checked and their priorities in PQ_V must be updated. When a site incident to a WDD edge changes label (S, F, or T), we need to directly access the edge in PQ_V and change its priority.

To maintain θ_{SS}, we store each SS-edge (u, v) of WDD (that is both ends are S-vertices) in another priority queue called PQ_{SS}, with priority $slack(u, v)/2$. We are interested in those edges in PQ_{SS} whose two ends are in distinct S-blossoms. Therefore, when we extract the SS-edge with minimum priority, we check to see if both ends are in the same S-blossom or not. This can be done in $O(\log n)$ time using the offsetted priority queues of the blossoms. If the two ends of the edge are in the same S-blossom, then we simply ignore that edge. This happens only once per edge in a phase. When some sites are relabeled and become S-vertices, we may need to add edges to PQ_{SS} using $O(\log n)$ time per edge. An SS-edge will neither be flipped nor will change label for the duration of the phase. Thus, such an edge remains in WDD for the duration of the phase. Hence, there are only $O(n)$ (insert or delete) operations performed on PQ_{SS} per phase, for a total of $O(n \log n)$ time per phase.

It remains to show how we maintain θ_{FS}. We essentially use the two level priority queue structure of [28] that they call p.q.$_2$. To be self contained, we give the necessary details here. For each F-blossom or T-blossom B, we maintain the set of edges that are incident both to B and an S-blossom into an ordered concatenable priority queue $OCPQ(B)$, where the priority of an edge is its slack value. The linear ordering is an extension of the linear ordering used to implement the offsetted priority queue of that blossom, namely, for each vertex $j \in B$, all edges incident to j appear consecutive (but in arbitrary order) in the linear ordering. This is needed for the split operation to work properly. (We need to maintain $OCPQ(B)$ even if B is T-blossom, since it may subsequently expand and some of its subblossoms may become F-blossoms. Also, some F-blossoms during the tree growing stage may become T-blossoms.) These form the lower level of the two level data structure. At the top level, we

maintain a priority queue PQ_{FS} which stores the minimum priority edge of each F-blossom B from $OCPQ(B)$. Now θ_{FS} corresponds to the minimum priority of PQ_{FS}. When a blossom changes label between T or F, we need to perform a corresponding insert or delete on PQ_{FS}. According to Fact 2.3 this occurs only $O(n)$ times per phase. What happens if a T-blossom or an F-blossom B becomes an S-blossom? This happens during the tree growing stage and the blossom expansion or shrinking stage. We first remove the record corresponding to B from PQ_{FS} if B was an F-blossom. Then, move all edges of $OCPQ(B)$ to PQ_{SS} and at the same time multiply their priorities by 1/2. From the analysis on PQ_{SS} above, we conclude that only a total of $O(n)$ edges per phase will be moved from any $OCPQ(B)$ to PQ_{SS}.

What happens when edges are flipped? Suppose $\mathcal{F}_i$ edge flips occur in phase i. (Recall $\mathcal{F} = \Sigma_{i=1}^{n} \mathcal{F}_i$.) So, the total number of edges involved in phase i is $E_i = O(n + \mathcal{F}_i)$. Since an edge-flip can never remove an SS-edge, no delete operations are done on PQ_{SS}. However, if necessary the corresponding edges must be deleted or added to $OCPQ(B)$ for some F-blossom or T-blossom B. A similar add or delete may also have to be done on PQ_{FS}. The total number of such operations is $O(E_i)$, for a total of $O(E_i \log n)$ time per phase. This concludes the description of the new algorithm and we have:

Theorem 4.1 *The proposed new algorithm solves the MWEM problem in $O((n^2 + \mathcal{F}) \log n)$ time and $O(n)$ space, where $\mathcal{F}$ is the total number of edge-flips during the algorithm.*

5 Discussion

The aim of this paper has been to devise a more geometric solution of the weighted Euclidean matching problem. A number of open questions regarding weighted relative neighborhood graphs and their relation with the EMWM problem remain. On a related note, let us also mention that the fractional version of EMWM gives rise to a circle packing problem which admits a more efficient solution [43].

Another remaining problem is to find a tight upper bound on $\mathcal{F}$. A very crude estimate of $\mathcal{F}$ can be obtained as follows. From [20] we know that the WVD is the projection of the lower envelope of vertical circular cones. These are identical circular cones in one-to-one

correspondence with the sites, such that each of their axes is perpendicular to the $xy-$plane and intersects it at the corresponding site, and the radius of the circular intersection of the cone with the $xy-$plane is the weight of the site. Consider the lower envelope of only the S-cones, corresponding to S-vertices. Call this lower envelope, the S-surface. Define T-surface and F-surface similarly. During a stage, the vertex labels do not change, the S-surface is moving down at a constant rate, the T-surface is moving up at the same constant rate, and the F-surface is stationary. As these three surfaces interact with each other, their lower envelope goes through at most $O(n^2)$ combinatorial changes. Since there are a total of $O(n^2)$ stages, we conclude the total number of combinatorial changes, that is $\mathcal{F}$, is $O(n^4)$. This analysis is of course very crude, since it considers the worst for each stage and does not take into account the correlations inherent in the algorithm. We conjecture that $\mathcal{F}$ is close to $O(n^2)$.

6 Acknowledgement

This research was partly supported by Natural Sciences and Engineering Research Council of Canada Grant OGP00D5516. Preliminary versions of this paper appeared as [42].

References

[1] P.J. Agarwal, D. Epstein, and J. Matoušek. Dynamic half-space reporting and minimum spanning trees. unpublished maniscript, 1992.

[2] P.J. Agarwal and J. Matoušek. Dynamic half-space range reporting and its applications. Technical Report CS-1991-43, Dept. of CS, Duke Univ., Dec. 1991.

[3] P.J. Agarwal and J. Matoušek. Relative neighborhood graphs in three dimensions. In *Proc. 3rd Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 58–65, 1992.

[4] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The design and analysis of computer algorithms*. addison-Wesley Publishing Company, Reading, Mass, 1974.

[5] F. Aurenhammer. Power diagrams : properties, algorithms and applications. *SIAM J. Computing*, 16:78–96, 1987.

[6] F. Aurenhammer. Voronoi diagrams – a survey of a fundamental geometric data structure. Technical Report B 90-09, Freie Universität, 1990.

[7] F. Aurenhammer, F. Hoffmann, and B. Aronov. Minkowski-type theorems and least-squares partitioning. In *Proc. 8th Annual ACM Symp. on Computational Geometry*, pages 350–357, 1992.

[8] F. Aurenhammer and O Schwarzkopf. A simpe on-line randomized incremental algorithm for computing higher order Voronoi diagrams. In *Proc. 7th Annual ACM Symp. on Computational Geometry*, pages 142–151, 1991.

[9] D. Avis. A survey of heuristics for the weighted matching problem. *Networks*, 13:475–493, 1983.

[10] M.S. Chang. *K-relative neighborhood graphs and their applications to some Euclidean bottleneck optimization problems.* PhD thesis, Institute of Comp. Sci., Natl. Tsing Hua Univ., Taiwan, 1991.

[11] M.S. Chang, C. Tang, and R.C.T. Lee. Solving the Euclidean bottleneck matching problem by k-relative neighborhood graph. *Algorithmica*, to apear.

[12] Y.J. Chiang and R. Tamassia. Dynamic algorithms in computational geometry. unpublished manuscript, Dept. of CS, Brown Univ., 1992.

[13] H.M.S. Coxeter. The problem of Apollonius. *American Math. Monthly*, 75:5–15, 1968.

[14] H.M.S. Coxeter. *Introduction to Geometry.* John Wiley and Sons, Inc., New York, second edition, 1980.

[15] O. Devillers, S. Meiser, and M. Teillaud. Fully dynamic Delaunay triangulation in logarithmic expected time per operation. In *Proc. WADS'91, LNCS 519*, pages 42–53. Springer-Verlag, 1991.

[16] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer-Verlag, New York, 1987.

[17] J. Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *J. of Research of National Bureau of Standards*, 69B:125–130, 1965.

[18] J. Edmonds. Paths, trees, and flowers. *Canadian J. Math.*, 17:449–467, 1965.

[19] J. Edmonds and R.M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264, 1972.

[20] S. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153–174, 1987.

[21] M.L. Fredman and R.E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34:596–615, 1987.

[22] J.J. Fu and R.C.T. Lee. Voronoi diagrams of moving points in the plane. *Intl. J. Comp. Geometry and Appl*, 1:23–32, 1991.

[23] H.N. Gabow. *Implementations of algorithms for maximum matching on nonbipartite graphs.* PhD thesis, Comp. Sci. Dept., Stanford Univ., 1973.

[24] H.N. Gabow. Data structures for weighted matching and nearest common ancestors with linking. In *Proc. 1st Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 434–443, 1990.

[25] H.N. Gabow and R.E. Tarjan. Algorithms for two bottleneck optimization problems. *J. Algorithms*, 9:411–417, 1988.

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

[27] Z. Galil. Efficient algorithms for finding maximum matching in graphs. *ACM Comp. Surveys*, 18:23–38, 1986.

[28] Z. Galil, S. Micali, and H.N. Gabow. An $O(EVlogV)$ algorithm for finding a maximal weighted matching in general graphs. *SIAM J. Computing*, 15:120–130, 1986.

[29] L.J. Guibas, D.E. Knuth, and M. Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. In *Proc. 17th Annual Intl. Coll. Automata, Lang., Progr.*, pages 414–431. LNCS 443, Springer-Verlag, 1990.

[30] L.J. Guibas, J.S.B. Mitchell, and T. Roos. Voronoi diagrams of moving points in the plane. unpublished manuscript, 1991.

[31] L.J. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. In *Proc. 15th Annual ACM Symp. on Theory of Computing*, pages 221–234, Boston, 1983.

[32] L.J. Guibas and J. Stolfi. Ruler, compass and computer : The design and analysis of geometric algorithms. In R.A. Earnshaw, editor, *NATO ASI Series, Theoretical Foundations of Computer Graphics and CAD*, volume F40, pages 112–159. Springer-Verlag, 1987.

[33] J. Hopcroft and R. Karp. An $n^{5/2}$ algorithm for maximum matching in bipartite graphs. *SIAM J. Computing*, 2:225–231, 1973.

[34] M. Ichino and J. Sklansky. The relative neighborhood graph for mixed feature variables. *Pattern Recognition*, 18:161–167, 1985.

[35] J. Jarmoczyk, M. Kowaluk, and F.F. Yao. An optimal algorithm for constructing β-skeletons in L_p metric. *SIAM J. Computing*, to appear.

[36] B. Kalyanasundaram and K. Pruhs. On-line weighted matching. In *Proc. 2nd Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 234–240, 1991.

[37] E.L. Lawler. *Combinatrotial Optimization: Networks and Matroids.* Holt, Reinhart and Winston, New York, 1976.

[38] L. Lovász and M.D. Plummer. *Matching Theory.* Annals of Discrete Math 29, North-Holland, 1986.

[39] O. Marcotte and S. Suri. Fast matching for points on a polygon. *SIAM J. Computing*, pages 405–422, 1991.

[40] D.W. Matula and R.R. Sokal. Properties of Gabriel graphs relevant to geographic variation search and the clustering of points in the plane. *Geogr. Analysis*, pages 205–222, 1980.

[41] S. Micali and V. Vazirani. An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs. In *Proc. 21st Annual IEEE Symp. on the Found. of Comp. Sci.*, pages 17–27, 1980.

[42] A. Mirzaian. Minimum weight Euclidean matching and weighted relative neighborhood graphs. In *Proc. of the Workshop on Algorithms and Data Structures, WADS'93*, Montreal, Canada, August 1993, and also as Tech. Rep. CS-92-08, Dept. of Computer Science, York University, Canada, Sept. 1992.

[43] A. Mirzaian. Optimum circle packing with specified centers. in preparation.

[44] J. O'Rourke. Computing the relative neighborhood graph in the L_1 and L_∞ metrics. *Pattern Recognition*, 8:45–55, 1979.

[45] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization : Algorithms and Complexity.* Prentice-Hall, Englewood Cliffs, NJ, 1982.

[46] F.P. Preparata and M.I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, 1985.

[47] C. Schwartz, M. Smid, and J. Snoyink. An optimal algorithm for the on-line closest-pair problem. In *Proc. 8th Annual Symp. on Computational Geometry*, pages 330–336, 1992.

[48] M. Sharir. Intersection and closest-pair problems for a set of planar discs. *SIAM J. Computing*, 14:448–468, 1985.

[49] T.H. Su and R.C. Chang. Computing the constrained relative neighborhood graphs and constrained Gabriel graphs in Euclidean plane. *Pattern Recognition*, 24:221–230, 1991.

[50] T.H. Su and R.C. Chang. Computing the k-relative neighborhood graphs in Euclidean plane. *Pattern Recognition*, 24:231–239, 1991.

[51] K.J. Supowit. The relative neighborhood graph, with an application to minimum spanning trees. *J. ACM*, 30:428–448, 1983.

[52] R.E. Tarjan. *Data Structures and Network Algorithms*. SIAM monograph, Philadelphia, PA, 1983.

[53] G.T. Toussaint. The relative neighborhood graph of a finite planar set. *Pattern Recognition*, 12:261–268, 1980.

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

[55] G.M. Weber. Sensitivity analysis of optimal matchings. *Networks*, 11:41–56, 1981.

[56] F.F. Yao. Computational geometry. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science: Volume A, Algorithms and Complexity*, pages 343–389. The MIT Press/Elsevier, 1990.