A Fast Algorithm for Data Collection along a Fixed Track

Otfried Cheong[1], Radwa El Shawi[2,3], and Joachim Gudmundsson[2,3]

[1] Korea Advanced Institute of Science and Technology, Republic of Korea*
otfried@kaist.edu
[2] University of Sydney, Australia**
joachim.gudmundsson@sydney.edu.au
[3] NICTA***, Sydney, Australia
radwa.elshawi@nicta.com.au

Abstract. Recent research shows that significant energy saving can be achieved in wireless sensor networks (WSNs) with a mobile base station that collects data from sensor nodes via short-range communications. However, a major performance bottleneck of such WSNs is the significantly increased latency in data collection due to the low movement speed of mobile base stations. In this paper we study the problem of finding a data collection path for a mobile base station moving along a fixed track in a wireless sensor network to minimize the latency of data collection. The main contribution is an $O(mn \log n)$ expected time algorithm, where n is the number of sensors in the networks and m is the complexity of the fixed track.

1 Introduction

Wireless sensor networks (WSNs) are a well established technology for many application areas. Their main aims are to monitor physical or environmental conditions and to cooperatively pass their data through the network to a main location. Realizing the full potential of wireless sensor networks poses research challenges ranging from hardware and architectural issues, to programming languages and operating systems for sensor networks, to security concerns, to algorithms for sensor network deployment [14].

WSNs usually consist of a large number of sensor nodes, which are battery-powered tiny devices. These devices perform three basic tasks: (i) sample a physical quantity from the surrounding environment, (ii) process the acquired data, and (iii) transfer them through wireless communications to a data collection point called sink node or base station [1,7]. The traditional WSN architectures

* O.C. was supported in part by NRF grant 2011-0016434 and in part by NRF grant 2011-0030044 (SRC-GAIA), both funded by the government of Korea.
** J.G. was funded by the Australian Research Council FT100100755.
*** NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

D.-Z. Du and G. Zhang (Eds.): COCOON 2013, LNCS 7936, pp. 77–88, 2013.

are based on the assumption that the network is dense, so that any two nodes can communicate with each other through multihop paths. As a consequence, in most cases the sensors are assumed to be static. However, recently mobility has been introduced to WSNs and it has been shown to have several advantages, such as, increased connectivity, lower cost, higher reliability and higher energy efficiency [2,10]. An overview of Wireless Sensor Networks with Mobile Elements (WSN-MEs) can be found in the comprehensive survey by Di Francesco et al. [7].

WSN-MEs have in general three main components [14]:

Regular sensor nodes are the sources of information. They perform sensing and may also forward or relay messages in the network.

Sinks (base stations) are the destinations of information. A network usually has very few sinks.

Special support nodes perform a specific task, such as acting as intermediate data collectors or mobile gateways.

In the setting considered in this paper we have one mobile sink that moves on a fixed track. The model was introduced by Xing et al. [16]. Although this is a very restricted model it simplifies the motion control of the mobile sink and it improves the system reliability and has therefore been adopted by several existing mobile sensor systems [3]. An example is when the sink only can move along fixed cables between trees [12]. The objective is to find a continuous path of length at most L along the track and a set of trees rooted on the path that connect all the sensor nodes, such that the total Euclidean length of the trees is minimized [16]. An example is shown in Fig. 1.

More formally, as input we are given a set $\mathcal{S} = \{s_1, \ldots, s_n\}$ of points (sensor nodes) together with a polygonal path $\mathcal{P} = \langle p_1, \ldots, p_m \rangle$ in $\mathbb{R}^2$. Given two points α and β on $\mathcal{P}$ (not necessarily vertices of $\mathcal{P}$) let $\mathcal{P}_{\alpha\beta}$ be the connected subpath of $\mathcal{P}$ with α and β as endpoints. Let $MST(\mathcal{S}, \mathcal{P}_{\alpha\beta})$ be a minimum spanning tree of $\mathcal{S}$ that contains $\mathcal{P}_{\alpha\beta}$, where an endpoint of an edge in $MST(\mathcal{S}, \mathcal{P}_{\alpha\beta})$ can either be a point of $\mathcal{S}$ or an arbitrary point on $\mathcal{P}_{\alpha\beta}$.

Problem 1. Data Collection on a Fixed Track (DCFT) problem
Given a real value L, a set $\mathcal{S} = \{s_1, \ldots, s_n\}$ of points and a polygonal path $\mathcal{P} = \langle p_1, \ldots, p_m \rangle$ in $\mathbb{R}^2$, find a path $\mathcal{P}_{\alpha\beta} \subseteq \mathcal{P}$ of length L such that $wt(MST(\mathcal{S}, \mathcal{P}_{\alpha\beta}))$ is minimized, where $wt(\cdot)$ denotes the total length of all the edges in the tree.

The path $\mathcal{P}_{\alpha\beta}$ is called the *active* path. Since the weight of the active path is fixed the aim is to find a placement of the active path on $\mathcal{P}$ that minimizes the total weight of the trees connecting $\mathcal{S}$ to it. Note that a point in $\mathcal{S}$ can be connected to any point along $\mathcal{P}_{\alpha\beta}$, not only to the vertices and endpoints of $\mathcal{P}_{\alpha\beta}$. We will sometimes write $\mathcal{P}_{\alpha *}$ or $\mathcal{P}_{*\beta}$ to denote the subpath of $\mathcal{P}$ of length L starting at α and ending at β, respectively.

To the best of the authors' knowledge very little work has been done on this problem from an algorithmic perspective and the only result we are aware of is the paper by Xing et al. [16]. They state without proof that the DCFT-problem is NP-hard, and concentrate on approximation algorithms. They showed how to

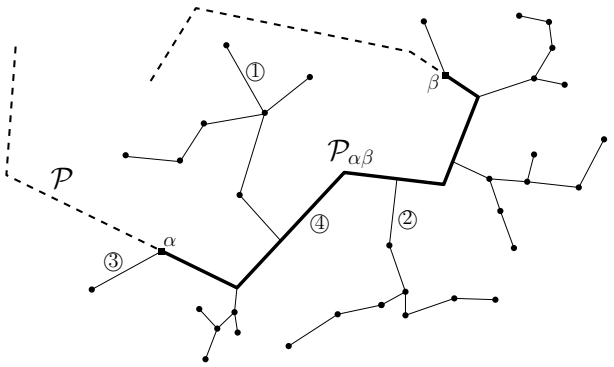

Fig. 1. Illustrating an instance of the DCFT problem with an active path $\mathcal{P}_{\alpha\beta}$ and a minimum spanning tree connecting the sensor nodes to $\mathcal{P}_{\alpha\beta}$. The encircled numbers illustrate the edge types defined in Observation 1.

compute a $3\varepsilon L$-approximation for the DCFT-problem in $O(\frac{wt(\mathcal{P})}{\varepsilon L} \cdot n \log n)$ time, where $\varepsilon > 0$ is a given constant. In this paper we will show that the DCFT-problem can in fact be solved *exactly* in $O(mn \log n)$ expected time (Theorem 5).

Omitted proofs can be found in the full version of the paper.

2 A Polynomial-Time Algorithm for the DCFT Problem

Since the length of $\mathcal{P}_{\alpha\beta}$ is fixed, a natural approach to solve the problem is to sweep an active path of length L along $\mathcal{P}$ while maintaining a minimum spanning tree. We will identify a set of $O(mn)$ event points along $\mathcal{P}$. It will be shown that all topological changes to the minimum spanning tree that we maintain during the sweep will occur when the start or end point of the active path coincides with one of the event points.

Two problems need to be handled: (1) find all event points along $\mathcal{P}$ of the sweep-line algorithm and (2) maintain a $MST(\mathcal{S}, P_{\alpha\beta})$ during the sweep.

2.1 Basic Properties and Notations

Given a point s and a segment ℓ in the plane let $op(s, \ell)$ denote the closest point on ℓ to s, see Fig. 2(a). Similarly, let $op(s, \mathcal{P}) = \cup_{\ell \in \mathcal{P}}\ op(s, \ell)$ and let $op(\mathcal{S}, \mathcal{P}) = \cup_{s \in \mathcal{S}}\ op(s, \mathcal{P})$.

Next we study the edges in an optimal solution in more detail (see Fig. 1).

Observation 1. *An edge $(u, v) \in MST(\mathcal{S}, \mathcal{P}_{\alpha\beta})$ can be one of four types:*
type-1: $u, v \in \mathcal{S}$,
type-2: $u \in \mathcal{S}$ and $v \in op(u, \mathcal{P}_{\alpha\beta})$ and v lies on $\mathcal{P}_{\alpha\beta}$,
type-3: $u \in \mathcal{S}$ and v is either α or β, and
type-4: u and v are consecutive vertices of $\mathcal{P}$ or points in $op(\mathcal{S}, \mathcal{P})$ along $\mathcal{P}$ (these are the edges forming $\mathcal{P}_{\alpha\beta}$).

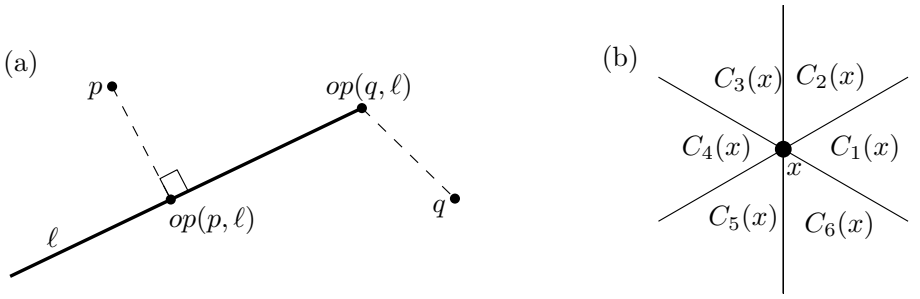

Fig. 2. (a) Illustrating $op(p, \ell)$ and $op(q, \ell)$. (b) Illustrating the definitions of $C_i(x)$, $1 \leq i \leq 6$.

We will frequently refer to these edge types throughout this paper, not only referring to the edges of the spanning tree but to edges of any network.

Now we are ready to start constructing the set of event points, denoted Γ, along $\mathcal{P}$. The set will be the union of three subsets Γ_1 (below), Γ_2 (Theorem 2) and Γ_3 (Theorem 4). Lets start with Γ_1 which will contain the vertices of $\mathcal{P}$ and the set of points $op(\mathcal{S}, \mathcal{P})$.

Observation 2. *The number of points in Γ_1 is $O(nm)$ and can be constructed in $O(nm)$ time.*

We say that an active path $\mathcal{P}'$ is sweeped between two *consecutive* event points x and y in Γ_1 if $\mathcal{P}'$ starts with one endpoint coinciding with x and is then moved along $\mathcal{P}$ until one of the end points of $\mathcal{P}'$ coincides with y. During the sweep no other event point in Γ_1 is encountered by any of the endpoints.

Observation 3. *Consider the inter-point Euclidean distance between two points u and v while sweeping the active path in-between two consecutive event points in Γ_1. If (u, v) is of type-1, type-2 or type-4 then the Euclidean distance $|uv|$ is fixed during the sweep. If (u, v) is of type-3 then the Euclidean distance will monotonically increase or decrease.*

As a direct consequence we have that all events that may induce a topological change to a minimum spanning tree during a sweep between two consecutive event points in Γ_1 will involve a type-3 edge. There are two cases:

Case 1: a type-3 edge will be replaced by another type-3 edge (Section 2.2), or
Case 2: a type-3 edge will replace, or be replaced by, a type-1 or type-2 edge (Section 2.3).

2.2 Generating Event Points for Case 1

In the rest of this section we will only consider sweeping the active path in-between two event points in Γ_1. Consider an arbitrary point x. Let $C_i(x)$,

$1 \leq i \leq 6$ be the six cones, ordered counter clockwise, that partition the plane into six cones with apex at x and interior angle $\pi/3$, see Fig. 2(b).

In the proof of Lemma 1 in [16], Xing et al. show the following observation (adapted to our notations).

Observation 4. *Let $\mathcal{P}_{\alpha\beta}$ be an optimal solution. Assume α lies on a segment (p_j, p_{j+1}) of $\mathcal{P}$ and assume (p_j, p_{j+1}) is horizontal with p_j to the left of p_{j+1}. There exists a $MST(\mathcal{S}, \mathcal{P}_{\alpha\beta})$ such that:*

- *α is connected to at most one point in each $C_i(\alpha)$, $1 \leq i \leq 6$, and β is connected to at most one point in each $C_i(\beta)$, $1 \leq i \leq 6$, and*
- *no point of $\mathcal{S}$ to the right of α, is connected by an edge to α. The symmetric result holds for β.*

From the above observation it immediately follows that for each α and for each β there are at most four *potential* type-3 edges in the minimum spanning tree. Thus, for a type-3 edge to be connected to an endpoint of the active path, say α, in the $MST(\mathcal{S}, \mathcal{P}_{\alpha\beta})$ it has to be a nearest neighbor to α in one of the six cones $C_i(\alpha)$, $1 \leq i \leq 6$.

Consider a case 1 event, and assume that a type-3 edge (s_1, α) is replaced by another type-3 edge, say the type-3 edge (s_2, γ) where $\gamma \in \{\alpha, \beta\}$. Before the event point we have $|s_1\alpha| < |s_2\gamma|$ and after the event point the order has changed to $|s_1\alpha| > |s_2\gamma|$.

From the above discussion it follows that all case 1 events can be found if we, during the entire sweep, can keep track of the twelve (only eight of which are of interest) potential type-3 edges, i.e., the nearest neighbor in each $C_i(\alpha)$ and $C_i(\beta)$, $1 \leq i \leq 6$. We will use the following theorem that will be shown in Section 3. See Fig. 3(a) for an illustration.

Theorem 1. *Given a set $\mathcal{S}$ of n points in the plane, a direction d and an angle $\theta < \pi$ the plane can be partitioned into $O(n)$ regions of total complexity $O(n)$ in $O(n \log n)$ expected time such that every point p in a region has the same nearest neighbor, say s, in $C(p, d, \theta)$, where $C(p, d, \theta)$ is the cone with apex at p, interior angle $\theta \leq \pi$ and with d as its bisector. The resulting partition is called an angle-restricted Voronoi diagram (ar-VD) of $\mathcal{S}$.*

Let $\theta = \pi/3$ and consider the following six directions (counterclockwise angle formed with the positive x-axis) $d_i = (i-1) \cdot \frac{\pi}{3}$, $1 \leq i \leq 6$, see Fig. 2(b). For each i, $1 \leq i \leq 6$, construct an angle-restricted VD, denoted V_i, of $\mathcal{S}$ with parameters θ and d_i using Theorem 1.

Using the ar-VDs one can now construct the set Γ_2. Every intersection point between $\mathcal{P}$ and an edge in any of the six ar-VDs $V_1, \ldots, V_6$, is added to Γ_2. For each segment there are $O(n)$ intersections, thus $O(mn)$ in total. Each intersection point can be found in $O(\log n)$ time using a standard point location data structure [5]. Constructing the six ar-VDs requires $O(mn \log n)$ expected time according to Theorem 1, thus $O(mn \log n)$ expected time in total.

Consider the sweep of the active path in-between two consecutive event points in $\Gamma_1 \cup \Gamma_2$. Note that during this sweep the potential type-3 edges will not

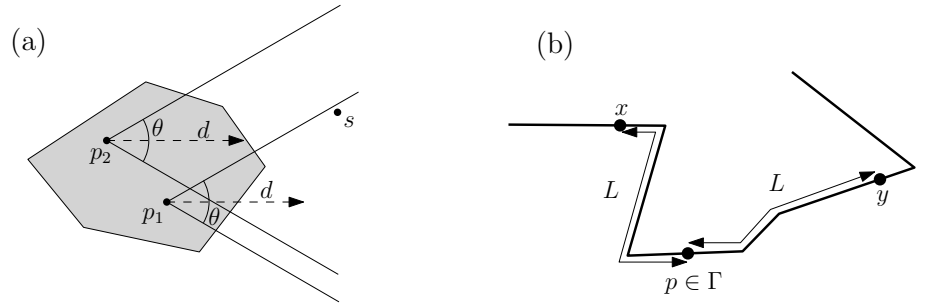

Fig. 3. (a) Every point p in a region of the angle-restricted Voronoi diagram have the same nearest neighbor in the cone $C(p, d, \theta)$. (b) Adding three vertices x, p and y to $\mathcal{E}(\Lambda)$ along $\mathcal{P}$ for each point in Λ.

change. Thus, to complete the set of all case 1 event points compute all events where the weight of two potential type-3 edges swap order. Since the weight of a potential type-3 edge is monotonically increasing or decreasing, according to Observation 3, there are at most 16 such events in-between two consecutive event points in $\Gamma_1 \cup \Gamma_2$. Given all the potential type-3 edges (at most eight) the events can be computed in constant time. Every point along $\mathcal{P}$ where such an event takes place is added to Γ_2.

We summarize this section with the following theorem, which follows immediately from the above discussion.

Theorem 2. *All event points Γ_2 where a type-3 edge is potentially replaced by another type-3 edge can be found in $O(mn \log n)$ expected time using $O(nm)$ space.*

2.3 Generating Event Points for Case 2

The final set of events is when a type-3 edge is replacing, or is replaced by, a type-1 or type-2 edge. We will show how the dynamic offline graph minimum spanning tree (DMST) algorithm by Eppstein [6] can be used to find Γ_3.

Theorem 3. *[Adapted from Theorem 1 in [6]] Given a sequence of k edge weight modifications in a graph of size N, starting from a state in which all weights are equal, we can compute the corresponding sequence of minimum spanning trees in time*[1] *$O(k \log N)$ and space $O(N)$.*

2.2.1 Build the Graph. To use the above result we will construct a graph $G = (V, E)$ and a sequence of edge weight modifications.

[1] In [6] the linear-time MST algorithm by Fredman and Willard [8] is used, resulting in $O(k \log N)$ running time. If we are constrained to the real RAM model of computation we have to pay an additional factor of $O(\log N)$ to build the MST, and thus an extra $O(\log N)$ factor to the total running time.

Before we define the graph we need the following definition. Let Λ be a set of points along $\mathcal{P}$. From Λ one can generate the set $\mathcal{E}(\Lambda)$ by adding up to three points on $\mathcal{P}$ for each point p in Λ; the point p, the two points at distance L along $\mathcal{P}$ from p, if they exist. See Fig. 3(b) for an illustration.

The node set V corresponds to the points in $\mathcal{S}$ and the points in $\mathcal{E}(\Gamma_1 \cup \Gamma_2)$, and its total size is $O(mn)$. The edge set E is constructed as follows:

1. Consider a minimum spanning tree of $\mathcal{S}$. Add the corresponding edges to E.
2. For every vertex $v \in V$ corresponding to a point $s \in \mathcal{S}$ add the $O(m)$ edges connecting v to the set of vertices in V corresponding to the points $op(s, \mathcal{P})$.
3. For each segment $e_i = (p_i, p_{i+1})$ in $\mathcal{P}$, $1 \leq i \leq m-1$, add an edge between the vertex in V corresponding to p_i and the vertex in V corresponding to the nearest neighbor in $C_j(p_i)$, $1 \leq j \leq 6$.
4. For each point $p \in op(\mathcal{S}, \mathcal{P})$ add an edge between the vertex in V corresponding to p and the vertex in V corresponding to the nearest neighbor in $C_j(p)$, $1 \leq j \leq j$.
5. For every two consecutive event points e_1 and e_2 in $\mathcal{E}(\Gamma_1 \cup \Gamma_2)$ add an edge between the vertices in V corresponding to e_1 and e_2 to E.

Note that the number of edges added to E is $O(mn)$. Consider the time required for the five steps. Step 1 uses $O(n \log n)$ time, steps 2 and 5 requires $O(mn)$ time while steps 3 and 4 can be computed in $O(mn \log n)$ time, using a standard point location query data structure on the ar-VDs.

Lemma 1. *Let $\mathcal{P}_{\alpha\beta}$ be an active path with α and/or β in $\Gamma_1 \cup \Gamma_2$. There exists a $MST(\mathcal{S}, \mathcal{P}_{\alpha\beta})$ whose edge set is a subset of E.*

2.2.2 Sweep the Active Path. To be able to use Theorem 3 we initially set all edges in E to have unit weight. Next we build the sequence of edge weight modifications in two steps; one to initialize the sweep and one to simulate the sweep of the active path along $\mathcal{P}$.

Initialization. The active path starts at p_1, that is, the initial configuration is $\mathcal{P}_{p_1*}$. Next we modify the edge weights to simulate the original DCFT-problem. The weight of every type-1 edge (u, v) is set to $|uv|$ and the weight of every type-4 edge is set to 0. The weight of a type-2 edge (u, v) is set to $|uv|$ if v lies on the active path $\mathcal{P}_{p_1*}$, otherwise it is set to ∞. Similarly, the weight of a type-3 edge $(u \in \mathcal{S}, v)$ is set to $|uv|$ if v is one of the endpoints of $\mathcal{P}_{p_1*}$, otherwise it is set to ∞. Note that $p_1 \in \Gamma_1$ and the opposite endpoint of $\mathcal{P}_{p_1*}$ is in $\mathcal{E}(\Gamma_1)$ thus, both are vertices in V. The total number of edge weight modifications needed for the initialization is $O(mn)$.

This correctly models the DCFT-problem for a fixed active path $\mathcal{P}_{p_1*}$ and will include the edges in a $MST(\mathcal{S}, \mathcal{P}_{p_1*})$.

Model the Sweep. Next we compute the set of edge weight modifications to simulate sweeping the active path along $\mathcal{P}$. Assume that we have the active

path $\mathcal{P}_{\alpha_1\beta_1}$ and that the algorithm correctly computed $MST(\mathcal{S}, \mathcal{P}_{\alpha_1\beta_1})$. Next we move the active path along $\mathcal{P}$ until it encounters the next event point in $\Gamma_1 \cup \Gamma_2$, the new active path is denoted $\mathcal{P}_{\alpha_2\beta_2}$. The following edge weight modifications are performed:

- the weight of any type-2 edge connecting α_1 with a point in $\mathcal{S}$ is set to ∞,
- the weight of any type-2 edge connecting β_2 with a point v in $\mathcal{S}$ is set to $|\beta_2 v|$,
- the weight of the potential type-3 edges connecting α_1 or β_1 with points in $\mathcal{S}$ is set to ∞, and
- the weight of the potential type-3 edges connecting α_2 or β_2 with points in $\mathcal{S}$ is set to be equal to the Euclidean distance between their endpoints,

This correctly models the DCFT-problem for the fixed active path $\mathcal{P}_{\alpha_2\beta_2}$ and will include the edges in a $MST(\mathcal{S}, \mathcal{P}_{\alpha_2\beta_2})$. The total number of edge weight modifications is $O(nm)$, and each one can be computed in constant time provided that the six ar-VDs have been computed in a preprocessing step.

Given the initial graph $G = (V, E)$ and the sequence of $O(nm)$ edge weight modifications we can run Eppstein's algorithm (Theorem 3) in $O(mn \log n)$ time. As output we obtain the corresponding sequence of minimum spanning trees (one for each event point), denoted $T_1, \ldots, T_k$, where $k = O(mn)$.

2.2.3 Compute the Event Points. Recall that the aim of this section is to construct the set of points, Γ_3, along $\mathcal{P}$ where a type-3 edge replaces, or is replaced by, a type-1 or type-2 edge. Above we only computed the minimum spanning trees for all the existing event points in $\Gamma_1 \cup \Gamma_2$, that is, for all cases when one of the endpoints of an active path coincides with a point in $\Gamma_1 \cup \Gamma_2$.

Lemma 2. *While sweeping the active path between two consecutive event points in $\Gamma_1 \cup \Gamma_2$ the minimum spanning tree can change topology at most eight times.*

Lemma 3. *While sweeping the active path between two consecutive event points in $\Gamma_1 \cup \Gamma_2$, every point along the sweep where a topological change is made to the minimum spanning tree can be computed in $O(\log n)$ time.*

Between every two consecutive event points in $\Gamma_1 \cup \Gamma_2$ compute all the event points where a topological change is made to the minimum spanning tree, according to Lemma 3. These points form the set Γ_3.

The following theorem concludes this section:

Theorem 4. *All event points, Γ_3, when a type-3 edge is potentially replacing, or is potentially replaced by, a type-1 or type-2 edge can be found in $O(mn \log n)$ expected time using $O(nm)$ space.*

2.4 Maintaining the Spanning Tree

We can now merge the results. Let $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3$. Consider the sweep-line algorithm introduced in the first paragraph of Section 2. That is, sweep an active

path $\mathcal{P}_{\alpha\beta}$ of length L along $\mathcal{P}$ starting with $\alpha = p_1$ and ending with $\beta = p_m$. During the sweep maintain a minimum spanning tree of $\mathcal{S}$ and $\mathcal{P}_{\alpha\beta}$.

From Observation 3, Lemma 2 and Theorem 4 it follows that Γ contains all the event points where a minimum spanning tree might change its topology. Next simulate a sweep while maintaining the tree. We can use the same approach as we used in Section 2.2.2, but using $\mathcal{S}$ and $\mathcal{E}(\Gamma)$ as the set of vertices instead of $\mathcal{S}$ and $\mathcal{E}(\Gamma_1 \cup \Gamma_2)$. Again, it is easy to see that this correctly models the DCFT-problem for the fixed active path $\mathcal{P}_{\alpha\beta}$ and will include the edges in a $MST(\mathcal{S}, \mathcal{P}_{\alpha\beta})$. The total number of edge weight modifications is $O(nm)$, and each one can be computed in constant time provided that the six ar-VDs are computed in a preprocessing step.

Eppstein's algorithm generates the initial minimum spanning tree and the sequence of $O(nm)$ changes made during the sweep. Consequently the weight of these trees can also be output without increasing the running time. Consider the sweep in-between two consecutive events in Γ. No topological changes are made to the minimum spanning tree. The only change is the weight of the type-3 edge. So to find a minimum weight spanning tree in-between two event points it is sufficient to compute the minimum of a function described by at most six distance functions, each distance function describing the minimum distance between a point and a straight-line segment. The minimum of this function can be computed in constant time, thus for each pair of consecutive event points, e_i and e_{i+1}, in Γ we can compute the minimum solution in-between e_i and e_{i+1} in constant time, given T_i and T_{i+1}. We can now summarize Section 2.

Theorem 5. *Given a real value L, a set $\mathcal{S} = \{s_1, \ldots, s_n\}$ of points and a polygonal path $\mathcal{P} = \langle p_1, \ldots, p_m \rangle$ in $\mathbb{R}^2$, an optimal solution for the DCFT-problem can be computed in $O(mn \log n)$ expected time.*

3 Angle-Restricted Voronoi Diagrams

The abstract Voronoi diagram (AVD) [11] is used to construct the set of event points involving type-3 edges. In the following we show how the AVD is constructed and investigate some of its topological properties.

Chew and Drysdale's [4] divide-and-conquer algorithm for computing the Voronoi diagram under convex distance functions can be extended to the construction of angle-restricted Voronoi diagrams (ar-VD) with some modifications [15]. They proved that the ar-VD can be computed in $O(n \log n)$ time but did not bound the complexity of the diagram (to the best of our knowledge), which we need to bound the size of Γ_3.

3.1 Definition and Properties

A unifying approach to Voronoi diagrams was proposed by Klein [11], which is based on the concept of bisecting curves instead of distance functions. For any two sites, p and q, in S, let $J(p, q)$ denote the curve that is homeomorphic

to a line and divides the plane into two open (unbounded) regions $D(p,q)$ and $D(q,p)$, where $D(p,q)$ contains p and $D(q,p)$ contains q. The Voronoi region of p with respect to $\mathcal{S}$, denoted $VR(p,\mathcal{S})$, is the intersection of all $D(p,q)$ regions as q varies in $S \setminus \{p\}$. The abstract Voronoi diagram is defined as:

$$AVD(p,\mathcal{S}) = \bigcup_{p \in \mathcal{S}} \partial(VR(p,\mathcal{S})).$$

Klein [11] showed that if the AVD is a "dominance system" (Definition 1) and the dominance system is "admissible" (Definition 2) then the AVD has many of the same properties as the concrete Voronoi diagrams. In particular, the complexity of the AVD is then $O(n)$, where n is the number of sites.

Our aim is to define a set of bisecting curves that fits the framework in [11]. First it has to be a dominance system.

Definition 1. *The family $\mathcal{D} = \{D(p,q), p \neq q\}$ is called a dominance system over $\mathcal{S}$ if the following holds:*

- *$D(p,q)$ is a non-empty open subset of the plane,*
- *$J(p,q) = \cap(D(p,q))$ is homeomorphic to the open interval (0,1), and*
- *$\partial(D(p,q)) = \partial(D(q,p))$, where ∂ denotes the perimeter of a region.*

We need to define the set $\mathcal{J}$ of bisecting curves that divides the plane into two open (unbounded) regions $D(p,q)$ and $D(q,p)$ for each pair $p,q \in \mathcal{S}$ of points, such that the resulting family $\mathcal{D} = \{D(p,q), p \neq q\}$ is a *dominance* system. In Lemma 4 it will be shown that the abstract Voronoi diagram defined by these bisectors is a dominance system.

3.2 Defining Bisecting Curves

Given two points p and q in the plane let $\mathcal{H}_p(p,q)$ be the set of points (the halfplane) whose Euclidean distance is smaller to p than to q. Furthermore, given a direction d and a point s in the plane let $\ell_d(s)$ denote the infinite ray originating at s with direction d. Let $C(s,d,\theta)$ be the cone with apex at s, angle $\theta \leq \pi$ and with $\ell_d(s)$ as its bisector.

For any two points $p,q \in \mathcal{S}$ a bisecting curve $J(p,q)$ is defined as follows. Assume w.l.o.g. that p lies to the left of q along the direction d. The bisecting curve $J(p,q)$ is defined as the boundary of $\{\mathcal{H}_q(p,q) \cap C(q,d,\theta)\}$.

Given a direction d, let o_d denote the point at $-\infty$ along the direction d and $-\infty$ in the direction orthogonal to d.

Lemma 4. *Given a set $\mathcal{S}$ of n points in the plane, a direction d and an angle $\theta \leq \pi$ the family $\mathcal{D} = \{D(p,q), p \neq q\}$ is a dominance system over $\mathcal{S} \cup \{o_d\}$.*

Proof. Consider the three properties in Definition 1. Adding o_d to our point set guarantees that the first property of the dominance system holds. The second and third properties follows immediately from the definition of bisectors. □

3.3 Computing the Diagram

The AVD defined by $\mathcal{D}$ is denoted an *angle-restricted Voronoi diagram*. We now address the construction of the ar-VD($\mathcal{S}$). Klein [11] showed that if a dominance system is *admissible* then the abstract Voronoi diagram has total size $O(n)$.

Definition 2. *A dominance system $\mathcal{J} = \{J(p,q) : p,q \in \mathcal{S}, p \neq q\}$ is called admissible if and only if for each subset $\mathcal{S}'$ of $\mathcal{S}$ of size at least three the following conditions are fulfilled:*

(a) *The intersection of two bisecting curves consists of finitely many components.*
(b) *The Voronoi regions are path-connected.*
(c) *Each point of the plane lies in a Voronoi region or on the Voronoi diagram, i.e. $\mathbb{R} = \bigcup_{p \in S} VR(p)$.*

Lemma 5. *The dominance system $\mathcal{J} = \{J(p,q) : p,q \in S \cup o_d, p \neq q\}$ defined in Section 3.2 is admissible.*

Mehlhorn et al. [9] and Klein et al. [13] have shown that one can, without further assumptions, apply the randomized incremental construction technique to abstract Voronoi diagrams.

Theorem 6. *The abstract Voronoi diagram of an admissible system $\mathcal{J} = \{J(p,q) : p,q \in \mathcal{S}, p \neq q\}$, where $|\mathcal{S}| = n$, can be constructed within expected $O(n \log n)$ many steps and expected space $O(n)$, by randomized incremental construction.*

Theorem 7. *Given a $\mathcal{S}$ of points and a point q in the plane. The Voronoi region of the directed $AVD(S,d,\theta)$ containing q corresponds to the point in $\mathcal{S}$ closest to q in $C(q,-d,\theta)$.*

Proof. The theorem follows immediately from the construction. □

Theorem 1 summarizes the results of this section and for convenience we restate it here.

Theorem 1. *Given a set $\mathcal{S}$ of n points in the plane, a direction d and an angle $\theta < \pi$ the plane can be partitioned into $O(n)$ regions of total complexity $O(n)$ in $O(n \log n)$ expected time such that every point p in a region has the same nearest neighbor, say s, in $C(p,d,\theta)$, where $C(p,d,\theta)$ is the cone with apex at p, interior angle $\theta \leq \pi$ and with d as its bisector. The resulting partition is called an angle-restricted Voronoi diagram (ar-VD) of $\mathcal{S}$.*

References

1. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Computer Networks 38(4), 393–422 (2002)

2. Anastasi, G., Conti, M., Di Francesco, M., Passarella, A.: Energy conservation in wireless sensor networks: a survey. Ad Hoc Networks 7(3), 537–568 (2009)
3. Batalin, M., Rahimi, M., Yu, Y., Liu, D., Kansal, A., Sukhatme, G., Kaiser, W.J., Hansen, M., Pottie, G.J., Srivastava, M.B., Estrin, D.: Call and response: experiments in sampling the environment. In: Proceedings of the 2nd International ACM Conference on Embedded Networked Sensor Systems (Sensys), pp. 25–38 (2004)
4. Chew, L.P., Drysdale, R.L.: Voronoi diagrams based on convex distance functions. In: Proceedings of the 1st Annual Symposium on Computational Geometry (SoCG), pp. 235–244 (1985)
5. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Computational geometry – algorithms and applications, 3rd edn. Springer, Heidelberg (2008)
6. Eppstein, D.: Offline algorithms for dynamic minimum spanning tree problems. Journal of Algorithms 17(2), 237–250 (1994)
7. Di Francesco, M., Das, S.K., Anastasi, G.: Data collection in wireless sensor networks with mobile elements: a survey. ACM Transactions on Sensor Networks 8(1), 1–7 (2011)
8. Fredman, M.L., Willard, D.E.: Trans-dichotomous algorithms for minimum spanning trees and shortest paths. Journal of Computer and System Sciences 48(3), 533–551 (1994)
9. Meiser, S., Mehlhorn, K., O'Dúnlaing, C.: On the construction of abstract Voronoi diagrams. Discrete and Computational Geometry 6, 211–224 (1991)
10. Kansal, A., Somasundara, A., Jea, D., Srivastava, M., Estrin, D.: Intelligent fluid infrastructure for embedded networks. In: Proceedings of the 2nd ACM International Conference on Mobile Systems, Applications, and Services (MobiSys), pp. 111–124 (2004)
11. Klein, R.: Concrete and Abstract Voronoi Diagrams. LNCS, vol. 400. Springer, Heidelberg (1989)
12. Pon, R., Batalin, M.A., Gordon, J., Kansal, A., Liu, D., Rahimi, M., Shirachi, L., Yu, Y., Hansen, M., Kaiser, W.J., Srivastava, M., Sukhatme, G., Estrin, D.: Networked infomechanical systems: a mobile embedded networked sensor platform. In: Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN), pp. 376–381 (2005)
13. Mehlhorn, K., Klein, R., Meiser, S.: Randomized incremental construction of abstract Voronoi diagrams. Computational Geometry: Theory and Applications 3, 157–184 (1993)
14. Sahni, S., Xu, X.: Algorithms for wireless sensor networks. International Journal of Distributed Sensor Networks 1(1), 35–56 (2005)
15. Wee, Y.C., Chaiken, S., Willard, D.E.: General metrics and angle restricted Voronoi diagrams. In: Proceedings of the 1st Canadian Conference on Computational Geometry (1989)
16. Xing, G., Wang, T., Jia, W., Li, M.: Rendezvous design algorithms for wireless sensor networks with a mobile base station. In: Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing, pp. 231–240 (2008)