Energy Optimal Routing in Radio Networks Using Geometric Data Structures [⋆]

René Beier, Peter Sanders, and Naveen Sivadasan

Max Planck Insitut für Informatik
Saarbrücken, Germany
[rbeier,sanders,ns]@mpi-sb.mpg.de

Abstract. Given the current position of n sites in a radio network, we discuss the problem of finding routes between pairs of sites such that the energy consumption for this communication is minimized. Though this can be done using Dijkstra's algorithm on the complete graph in quadratic tim, it is less clear how to do it in near linear time. We present such algorithms for the important case where the transmission cost between two sites is the square of their Euclidean distance plus a constant offset. We give an $\mathcal{O}(kn \log n)$ time algorithm that finds an optimal path with at most k hops, and an $\mathcal{O}(n^{1+\epsilon})$ time algorithm for the case of an unrestricted number of hops. The algorithms are based on geometric data structures ranging from simple 2-dimensional Delaunay triangulations to more sophisticated proximity data structures that exploit the special structure of the problem.

1 Introduction

Networks between a number of sites can be set up without additional infrastructure using radio communication between the sites. Such an approach has recently gained considerable interest. We discuss the fundamental problem of finding good routes between pairs of sites.

Since the sites have often only limited energy reserves from batteries or power from solar panels, a prime optimization criterion is the energy consumption. We model this as follows. When two sites, say u and v, communicate directly, the sender node u needs a transmission energy $C_u + |uv|^2$. The *cost offset* C_u accounts for distance independent energy consumption like the energy consumption of the signal processing during sending and receiving. $|uv|^2$ denotes the square of the Euclidean distance of u and v. Although many cost functions for energy consumptions have been proposed this quadratic behavior is the most fundamental one since it accurately describes the behavior of electromagnetic waves in free space.

[⋆] This work was partially supported by DFG grant SA 933/1-1 and the Future and Emerging Technologies programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

In addition to energy consumption, another expensive resource is bandwidth[1] — transmitting a data packet using range r inhibits[2] communication in an area of size $\Theta(r^2)$. Hence, our quadratic cost measure can also be viewed as an estimate for the degree to which a route hinders other connections from using the same frequencies. A third cost measure is the reliability of a connection. The more nodes are involved, the higher is the probability of a failure. This cost measure is indirectly captured by the offset C_u.

We model the network as a complete geometric graph where the nodes of the graphs corresponds to the sites of the network and an edge from node u to v in the graph has weight $|uv|^2 + C_u$. In principle, an optimal path can be easily found in time $\mathcal{O}(n^2)$ using Dijkstra's shortest path algorithm in this graph.

The subject of this paper is to exploit the geometric structure of the problem to find energy optimal routes in time closer to $\mathcal{O}(n)$ than $\mathcal{O}(n^2)$. After developing a number of algorithms for this problem, we were made aware of previous work for geometric shortest path problems by Chan, Efrat, and Har-Peled [6,4] motivated by routing airplanes while taking into account that fuel consumption grows superlinearly with distance. It turned out that we rediscovered a very simple algorithm for cost functions without offset and at least quadratic growth. Here it suffices to search for a path in the Delaunay triangulation and hence there is a simple $\mathcal{O}(n \log n)$ algorithm. This is rather easy to see since a non-Delaunay edge e on a shortest path contains points inside the smallest circle enclosing the edge. Replacing e by two edges going via such a point leads to a smaller sum of squared lengths. The problem of this cost function for radio networks is that in large networks it leads to paths with an unrealistically large number of hops (edges).

For general cost functions, Chan, Efrat, and Har-Peled obtain an algorithm using time and space $\mathcal{O}(n^{4/3+\epsilon})$ for any positive constant ϵ.

Although this is a remarkable theoretical achievement, we believe that faster algorithms are so important for many applications that they are worth investigating even for more specialized classes of cost functions. Quadratic cost functions with offset seem a good compromise for radio networks. The quadratic component implies nice geometric properties and models energy consumption in free space and bandwidth consumption. The offsets model energy not invested in radio waves and allow us to penalize paths with too many hops.

1.1 New results

In Section 2 we start with a very simple algorithm based on Voronoi diagrams that finds optimal 2-hop paths for uniform offset costs in time $\mathcal{O}(\log n)$ after a preprocessing time of $\mathcal{O}(n \log n)$ and space $\mathcal{O}(n)$. Using the dynamic programming principle this can easily be extended to an $\mathcal{O}(n \log n)$ algorithm for optimal

[1] 120 MHz of UMTS frequency range in Germany cost about $50 \cdot 10^9$ Euro

[2] In channel based systems like GSM, inhibition basically means blocking the channel for other concurrent communication. In CDMA systems like UMTS the inhibition is more subtle but nevertheless present.

paths with at most four hops. We then give a more general algorithm for finding optimal routes with up to k hops and non-uniform offset costs with running time $\mathcal{O}(kn \log n)$ and $\mathcal{O}(n)$ space by reducing it to a special case of 3D nearest neighbor problem and solving it efficiently. All the above algorithms are quite practical. Finally Section 3 gives a more theoretical solution for the general case of non-uniform costs and arbitrary number of hops with running time and space $\mathcal{O}(n^{1+\epsilon})$ for any constant $\epsilon > 0$. The solution is based on 3D closest bichromatic pair queries [7] similar to the general algorithm in [4]. We mainly improve the nearest neighbor queries involved to take advantage of the special structure of the problem.

1.2 Further Related Work

Energy efficient communication in radio network is a widely studied problem. Besides simple constant cost models [9] the most common simple model is our model with $f(|uv|) = |uv|^\rho$ for some constant $\rho \geq 2$ [12, 10].[3] Rabaey et al. [8] mention sensor and monitoring networks as a natural application for radio networks where positions of stations are known because they are also needed on the application level. Energy-efficient communication by solving network optimization problems in a graph with transmission power as edge weights is an intensively studied approach. However, so far more complex problems like broadcasting and multi-cast routing have been studied (e.g. [15, 14]) whereas the quadratic complexity of solving the shortest path problem in a complete graph has not been challenged.

An intensive area of research is distributed routing in dynamic situations without complete information.[4] Unfortunately, the worst case performance of such systems is not encouraging. Therefore we believe that research should be driven from both sides, namely : making pessimistic distributed models more powerful and making static models with complete information more dynamic. We view our work as a contribution in the latter direction because faster routing means that a centralized scheduler can handle more routing requests.

Although we are mainly applying computational geometry to radio communication, one could also take the more abstract point of view that we are interested in solving fundamental network problems in implicitly defined (geometric) graphs. There are results in this direction on spanning trees [11] and matchings [13].

2 Bounded Number of Hops

There are several reasons why one might prefer routes that traverse only a small number of edges (hops) in the communication graph. If there are more hops,

[3] Powers $\rho < 2$ are only relevant for nowadays rather special cases like short wave transmission with reflections on the ionosphere.

[4] A list of papers can be found at `http://www1.ics.uci.edu/~atm/adhoc/paper-collection/papers.html`.

latency might get too large, reliability becomes an issue and distance independent energy consumption in the intermediate nodes may become too large. Therefore, we might be interested in the energy minimal path with the additional constraint that at most k hops should be used. Section 2.1 presents a simple algorithm for the case of two-hop paths with uniform offsets based on 2-dimensional closest point queries and extends this approach to paths with three and four-hops. Then Section 2.2 gives a generalization for k-hop paths with non-uniform offsets that needs more sophisticated and less standard data structures.

2.1 Two, Three, or Four Hops

Theorem 1. *In a complete geometric graph defined by n points in the plane with edge weights $C + |uv|^2$, queries for optimal 2-hop paths can be answered in time $\mathcal{O}(\log n)$ after preprocessing time $\mathcal{O}(n \log n)$ using $\mathcal{O}(n)$ space.*

Proof. Given source s and target t we can express the cost μ_w of the 2-hop path (s, w, t) as a function of $d = |st|$ and the distance r between w and the mid-point M of segment st. Fig. 1 illustrates this situation. We have

$$\begin{aligned} h^2 &= r^2 - (d/2 - c)^2 = r^2 - d^2/4 - c^2 + dc, \\ \mu_w &= a^2 + b^2 = c^2 + h^2 + (d - c)^2 + h^2 = d^2 + 2c^2 - 2dc + 2h^2 \\ &= d^2/2 + 2r^2. \end{aligned}$$

Hence, we can minimize the path length by picking the intermediate node with the smallest Euclidean distance from M. Finding the nearest neighbor is a standard problem from computational geometry and can be solved in time $\mathcal{O}(\log n)$ after $\mathcal{O}(n \log n)$ preprocessing. For example we can build a Voronoi diagram of the nodes and a point location data structure for locating w in the Voronoi diagram [5, Section 7]. ∎

The 2-hop algorithm from Theorem 1 can be generalized for a dynamically changing set of transmitters. Using the dynamic closest point data structure [1] we can answer two-hop route request in time $\mathcal{O}(\text{Polylog } n)$. For any given $\epsilon > 0$, the data structure can be maintained dynamically in amortized time $\mathcal{O}\big(n^{1+\epsilon}\big)$

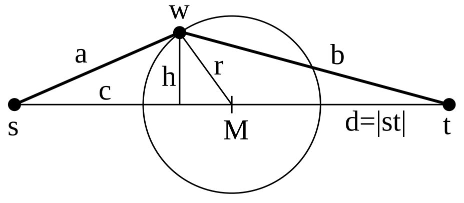

Fig. 1. Points with equal cost for two-hop routing lie on a circle around the mid point M between source s and target t.

per insert/delete operation. If we are content with approximate results within a factor $(1+\delta)$, an answer can be obtained in time $\mathcal{O}(\log(n)/\delta)$ [3]. Although these worst case bounds are only obtained for rather complicated theoretical algorithms we expect that simple data structures will work in practical situations. For example, if transmitter positions are uniformly distributed, all operations run in constant expected time using a simple grid based data structure [2].

We now explain how the 2-hop algorithm leads to an efficient algorithm for three and four hops.

Theorem 2. *In a complete geometric graph defined by n points in the plane with edge weights $C + |uv|^2$, an optimal path with three or four hops between nodes s and t can be found in time $\mathcal{O}(n \log n)$ using space $\mathcal{O}(n)$.*

Proof. We first explain the method for a three-hop path (s, u, v, t). We build a data structure for answering nearest neighbor queries as in the case of two-hop in time $\mathcal{O}(n \log n)$. Then we consider all $n-2$ possibilities for choosing v. We can find the shortest three-hop path (s, u, v, t) by adding $3C + |vt|^2$ to the shortest two-hop path from s to v. This is possible in time $\mathcal{O}(\log n)$ by Theorem 1. We get an overall execution time of $\mathcal{O}(n \log n)$.

To find a four-hop path (s, u, v, w, t) we proceed in a similar way only that we now have to find two such 2-hop paths ($s \to v$ and $t \to v$) for each possible choice of v. ∎

For a practical implementation of the above four-hop algorithm, several optimizations suggest themselves that might improve the constant factors involved. One could exploit that all the $2(n-2)$ nearest neighbor queries are independent of each other. Hence, it suffices to implement a *batched* nearest neighbor algorithm that runs in time $\mathcal{O}(n \log n)$. Another more heuristic approach abandons batched access and considers candidate points close to the mid point M first. Let l be the length of the shortest path found so far. It can be shown that points outside the disc centered at M with radius $\sqrt{l - |st|^2/4}$ cannot possibly yield improved routes and the search can be stopped.

The basic approach from Section 2.1 for 2–4 hop routing can also be generalized for non-quadratic cost functions. We first locate "ideal" relays position in the Voronoi diagram or a grid file and then search in the neighborhood for good points using a more accurate cost measure. We have not analyzed the complexity of this approach but possibly it could lead to efficient approximate algorithms or exact algorithms that are efficient in some average case model.

2.2 k Hops

For any $p \in P$, let $\mu_k(p)$ denote the cost of the shortest path from the source node s to p using at most k hops. In this section we describe a dynamic programming algorithm that determines $\mu_k(t)$ in time $\mathcal{O}(kn \log n)$ for any $k \leq n-1$. Clearly, $\mu_1(p) = C_s + |sp|^2$. For $i > 1$ the function μ_i can be computed recursively as follows:

$$\mu_{i+1}(p) = \min\{\mu_i(p), \min_{q \in P}\{\mu_i(q) + C_q + |qp|^2\}\}. \tag{1}$$

A trivial implementation of this formula yields an algorithm with running time $\mathcal{O}(kn^2)$: We start with μ_1 and iteratively determine μ_i for i increasing from 2 to k. In each of the $k-1$ phases, equation 1 is applied for all $p \in P$. We can exploit the geometric structure of the problem to speed up the computation.

Theorem 3. *In a complete geometric graph defined by n points in the plane with edge weights $C_u + |uv|^2$, the single source shortest path problem restricted to paths with at most k-hops can be solved in time $\mathcal{O}(kn\log n)$ using space $\mathcal{O}(n)$.*

Proof. Applying Lemmas 1 and 3 below, $\mu_{i+1}(p)$ can be computed from μ_i for all $p \in P$ in time $\mathcal{O}(n\log n)$. This corresponds to one phase of the algorithm. For all $k-1$ phases, a total of $\mathcal{O}(kn\log n)$ time is sufficient. ■

The problem of finding the point $q \in P$ that minimizes the expression $\mu_i(q) + C_q + |pq|^2$ on the right hand side of equation 1 can be reduced to a 3D nearest neighbor query. Consider the points P embedded into the plane $z = 0$ in 3-dimensional space. Thus a point $p \in P$ has the form $p = (x_p, y_p, 0) \in \mathbb{R}^3$. We code the cost of a i-hop path from s to p as well as the offset cost C_p using the third dimension: Define $f_i : P \to \mathbb{R}^3; f_i(x_p, y_p, 0) := (x_p, y_p, \sqrt{\mu_i(p) + C_p})$. Let $f_i(P) := \{f_i(p) : p \in P\}$. Since the vector from p to $f_i(p)$ is orthogonal to the plane $z = 0$ we have for each pair $p, q \in P$:

$$|f_i(q)p|^2 = |qp|^2 + \left(\sqrt{\mu_i(q) + C_q}\right)^2 = \mu_i(q) + C_q + |qp|^2 .$$

Let $p \in P$ be fixed and choose $q \in P$ such that $f_i(q)$ is closest to p among the points in $f_i(P)$. Then q minimizes the expression $\mu_i(q) + C_q + |pq|^2$. Using equation 1 we can compute $\mu_{i+1}(p)$ by answering one nearest neighbor query for p. This justifies the following lemma:

Lemma 1. *The problem of computing $\mu_{i+1}(p)$ from $\mu_i(p)$ reduces to an instance of the nearest neighbor problem with sites $f_i(P)$ and query point $p = (p_x, p_y, 0)$.*

A standard approach for solving the nearest neighbor problem uses Voronoi diagrams together with a point location algorithm. The worst case combinatorial complexity of the Voronoi diagram $\mathcal{V}(S)$ for a 3D point set with n elements is $\Theta(n^2)$. Therefore, a straightforward application does not improve the asymptotic running time of our algorithm. Note however that for our purposes all query points lie in the plane $e := (z = 0)$. Hence it suffices to compute the subdivision of plane e that is induced by the Voronoi diagram $\mathcal{V}(f_i(P))$. Let $\mathcal{V}_e$ denote this subdivision of e, which can be regarded as the intersection of $\mathcal{V}(f_i(P))$ and plane e. It is very similar to a 2D Voronoi diagram except that there might be less than n cells (faces). The worst case combinatorial complexity of $\mathcal{V}_e$ is only linear in the number of points in P.This can easily be verified using Euler's formula for planar graphs.

In the following we give an algorithm for computing $\mathcal{V}_e(S)$ in time $\mathcal{O}(n \log n)$. We use the well known relationship between Voronoi diagrams and the intersection of upper half spaces [5]. Given a set S of n points in $\mathbb{R}^3$ we map each point $p = (p_x, p_y, p_z) \in S$ to the hyperplane h_p in $\mathbb{R}^4$ which is the graph of the 3-variate linear function

$$h_p(x, y, z) = 2p_x x + 2p_y y + 2p_z z - (p_x^2 + p_y^2 + p_z^2).$$

A given query point $q = (q_x, q_y, q_z)$ is closest to $p \in S$ (among all points in S) if and only if

$$p = \arg\max_{s \in S} h_s(q_x, q_y, q_z).$$

This can be interpreted geometrically in 4-dimensional space with dimensions x, y, z and v as the height[5]: Consider the half spaces bounded from below (with respect to v) by the hyperplanes $H(S) := \{h_s : s \in S)\}$. The upper envelope $\mathcal{UE}(H(S))$ of $H(S)$ is defined as the boundary of the intersection of these half spaces which is the graph of the 3-variate function $f(x, y, z) = \max_{s \in S} h_s(x, y, z)$. Hence $p \in S$ is a nearest neighbor to query point q if and only if $f(q) = h_p(q)$. In this case h_p is a supporting hyperplane that contributes a facet to $\mathcal{UE}(H(S))$. Consequently, the orthogonal projection of $\mathcal{UE}(H(S))$ to the hyperplane $v = 0$ is the Voronoi diagram $\mathcal{V}(S)$. For our purposes we need to intersect the Voronoi diagram of the 3D point set $f_i(P)$ with the hyperplane $(z = 0)$. The worst case complexity of the resulting subdivision $\mathcal{V}_e$ and $\mathcal{V}(f_i(P))$ differ by a factor of n. To avoid computing unnecessary information, we schedule the transition to the subspace $z = 0$ before computing the upper envelope and obtain the following algorithm:

Algorithm 1

```
1. Compute the set of hyperplanes $H = H(f_i(P))$.
2. Intersect each hyperplane $h \in H$ with the hyperplane $z = 0$ to get a set of
   2D hyperplanes $H_z$.
3. Compute the upper envelope $\mathcal{UE}(H_z)$.
4. Project $\mathcal{UE}(H_z)$ onto the $xy$-plane $(v = z = 0)$.
```

Lemma 2. *Algorithm 1 computes $\mathcal{V}_e(f_i(P))$ correctly and has running time $\mathcal{O}(n \log n)$.*

Proof. Let $q = (q_x, q_y, 0)$ be any query point on the xy-plane lying in the Voronoi cell of some site $p \in f_i(P) = S$ (i.e, p is a nearest neighbor for q). So we have $h_p(q) = \max_{s \in S} h_s(q)$. Note that for any $s \in S$ the 2D hyperplane $h_s^z \in H_z$ is the graph of a 2-variate $h_s^z(x, y) = h_s(x, y, 0)$. Hence, for q, $h_p(q) = h_p^z(q_x, q_y) = \max_{s \in S}\{h_s^z(q_x, q_y)\}$. This means that h_p^z contributes a facet

[5] So we can use the notion of above and below.

to $\mathcal{UE}(H_z)$. Moreover, in the projection of the $\mathcal{UE}(H_z)$ to the xy-plane, q is part of the Voronoi region that corresponds to site p.

Since the worst case combinatorial complexity of $\mathcal{UE}(H_z)$ is $\mathcal{O}(n)$, it can be computed in time $\mathcal{O}(n \log n)$ [5]. It is easy to verify that all other steps of algorithm 1 can be done in linear time. ■

Having computed $\mathcal{V}_e(f_i(P))$, we can make use of a point location algorithm to answer 2D nearest neighbor queries efficiently.

Lemma 3. *Let P be a set of n points in $\mathbb{R}^3$. Allowing $\mathcal{O}(n \log n)$ preprocessing time, the nearest neighbor problem for sites P and query points lying in the xy-plane can be answered in time $\mathcal{O}(\log n)$.*

3 The General Case

In this section we consider the case of computing the shortest path without any restrictions in the number of hops used. Since all edge weights are non-negative, this corresponds to computing $\mu_{n-1}(t)$ for target node t. Using the results of the last section one could solve the problem in time $\mathcal{O}\big(n^2 \log n\big)$ which is even worse than applying Dijkstra's algorithm to the complete graph. Therefore, we apply a different shortest path algorithm which was already used in [4]. It is similar to Dijkstra's algorithm in that it settles nodes in order of the distance from the source. But edges are never explicitly relaxed. Instead a powerful geometric data structure replaces both edge relaxation and the priority queue data structure of Dijkstra's algorithm.

Theorem 4. *In a complete geometric graph defined by n points in the plane with edge weights $C_u + |uv|^2$, the single source shortest path problem can be solved in $\mathcal{O}\big(n^{1+\epsilon}\big)$ time and space for any constant $\epsilon > 0$.*

Let $\mu(v) = \mu_{n-1}(v)$ denote the length of a shortest path from source s to $v \in V$. During the execution of the algorithm each node is either reached (red) or unreached (blue). Initially all nodes except the source node are blue. Nodes change their color from blue to red in order of increasing $\mu()$ values. At the time a node becomes red, its $\mu()$ value is known. Let R and B denote the set of red and blue nodes respectively. In each iteration, we determine a blue node with smallest distance to the source:

$$v = \arg\min_{b \in B} \min_{r \in R} \{\mu(r) + |rb|^2 + C_r\} \tag{2}$$

This can be modeled as a closest bichromatic point problem with Euclidean distance function using one additional dimension. We define

$$f : R \longrightarrow \mathbb{R}^3; \qquad f(r_x, r_y, 0) = (r_x, r_y, \sqrt{\mu(r) + C_r}).$$

For any red-blue pair $(r, b) \in R \times B$, we have $|f(r)b|^2 = |rb|^2 + \sqrt{\mu(r) + C_r}^2$. Let $f(R) = \{f(r) : r \in R\}$. Then the closest bichromatic pair between the sets $f(R)$ and B gives the solution to Equation 2. Algorithm 2 states the pseudo code for the general case.

Algorithm 2

```
$B \leftarrow V \backslash \{s\}$, $R \leftarrow \{s\}$, $\mu[s] \leftarrow 0$;
while ($B \neq \emptyset$)
  Find closest bichromatic pair $(r, b)$ between sets $f(R)$ and $B$.
  $R \leftarrow R \cup \{b\}$, $B \leftarrow B \backslash \{b\}$, $\mu[b] \leftarrow |f(r)b|^2$;
```

Correctness can be shown by induction with the following invariants: For any red-blue pair (r, b) we have $\mu(r) \leq \mu(b)$ and the $\mu()$-values of the red nodes are determined correctly. When a point p is discovered then Equation 2 ensures that all the remaining blue points have a $\mu()$-value at least as large as $\mu(p)$. Since edge weights are strictly positive, there cannot be a blue point on a shortest $s \to p$ path and hence $\mu(p)$ is computed correctly. The dynamic bichromatic closest points problems we have to solve are of special type: all blue points are lying in the xy-plane and the distance function is simply the Euclidean distance. This enables us to speed up the computation compared to the general 3D dynamic closest pair problem.

Corollary 1. *In a complete geometric graph defined by n points in the plane with edge weights $C_u + |uv|^2$, the single source shortest path problem can be reduced to answer a sequence of $n - 1$ closest bichromatic neighbor queries for dynamically changing point sets $R, B \in \mathbb{R}^3$ of cumulative size n where all points in B are lying in the xy-plane.*

A general technique for solving the dynamic bichromatic closest pair problem is given by Eppstein [7], who reduces the problem to the *dynamic nearest neighbors problem* for dynamically changing sets of sites R and B. For blue sites B the query points are red ($q \in R$) and vice versa. Since in our application the blue set is basically a 2D point set, we distinguish two types of nearest neighbor queries:

1. Sites S are restricted to the xy-plane ($S \subset \mathbb{R}^2$, query $q \in \mathbb{R}^3$).
2. Queries q are restricted to the xy-plane ($S \subset \mathbb{R}^3$, $q \in \mathbb{R}^2$).

We show that these special types of the 3D problem can be regarded as 2D nearest neighbor problems. Consider the first type. Given a query point $q = (q_x, q_y, q_z)$ we simply find the nearest neighbor for $q' = (q_x, q_y, 0)$. It is easy to verify that this is also a solution for q.

The second type of nearest neighbor queries we considered already in the last subsection for a static set of sites. Here we again exploit the relation between Voronoi diagrams and the upper envelope of hyperplanes. Instead of precomputing the upper envelope, we determine for each individual query point $q = (q_x, q_y, 0, 0)$ the first hyperplane that is intersected by the ray emanating from the point $(q_x, q_y, 0, +\infty)$ in $-v$ direction (see [1]). Since all query points lie in xy-plane, the corresponding query rays are part of the subspace $z = 0$. Hence it suffices to restrict the calculations to 3D space (with dimensions x, y, v). We maintain a set of 2D planes in $\mathbb{R}^3$ which are obtained by intersecting the hyperplanes $H(f(R)) = \{h_p : p \in f(R)\}$ (using the notation from section 2.2) with hyperplane $z = 0$. Agarwal and Matoušek [1] describe algorithms for dynamic ray shooting problems and state the following lemma.

Lemma 4. *Given a convex polytope in $\mathbb{R}^3$ described as the intersection of n half-spaces, one can process it in time $\mathcal{O}(n^{1+\epsilon})$ into a data structure of size $\mathcal{O}(n^{1+\epsilon})$, so that the first point of the polytope boundary hit by a query ray can be determined in $\mathcal{O}(\log^5 n)$ time. The data structure can be maintained dynamically in amortized time $\mathcal{O}(n^\epsilon)$ per insert/delete operation.*

Combined with the result of Eppstein [7] for maintaining the bichromatic closest pair, this proves Theorem 4. In contrast to the work of [4], we considered a more restrictive cost function which leads to a better running time. They used the lower envelope of a set of bivariate functions to solve the nearest neighbor problem whereas we maintain a set of linear object, namely 2D planes.

We showed how to solve the problem by answering a sequence of closest bichromatic point queries for dynamic sets R and B. These queries have more special properties that we have not exploited:

1. We never insert points into set B.
2. We never delete points from set R.

Our application is even more specific when we view a deletion followed by an insertion as one operation, which is then the only operation we need: A node turns its color from blue to red and is lifted up in the third dimension keeping its x- and y-coordinates. This might be a starting points for finding more practical algorithms.

4 Discussion

We have proposed efficient algorithms for energy and bandwidth efficient communication in a model designed for simplicity and resemblance to the actual physical conditions. The problems of communication in real radio networks are vastly more complicated and probably do not allow similarly clean solutions. However, we believe that the algorithmic principles developed here may nevertheless remain useful. In particular, the 2–4 hop routing algorithms from Section 2.1 are so simple and flexible that they might be useful in practice.

Acknowledgments We would like to thank Edgar Ramos for contributing valuable geometric know-how.

References

1. P. D. Agarwal and J. Matousek. Ray shooting and parametric search. In N. Alon, editor, *Proceedings of the 24th Annual ACM Symposium on the Theory of Computing*, pages 517–526, Victoria, B.C., Canada, May 1992. ACM Press.
2. J. Bentley, B. W. Weide, and A. C. Yao. Optimal expected-time algorithms for closest point problems. *ACM Transactions on Mathematical Software*, 6(4):563–580, December 1980.

3. T. Chan. Approximate nearest neighbor queries revisited. In *Proceedings of the 13th International Annual Symposium on Computational Geometry (SCG-97)*, pages 352–358, New York, June 4–6 1997. ACM Press.
4. T. Chan and A. Efrat. Fly cheaply: On the minimum fuel consumption problem. *Journal of Algorithms*, 41(2):330–337, November 2001.
5. M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry Algorithms and Applications.* Springer-Verlag, Berlin Heidelberg, 2., rev. ed. edition, 2000.
6. A. Efrat and S. Har-Peled. Fly cheaply: On the minimum fuel-consumption problem. In *Proceedings of the Fourteenth Annual Symposium on Computational Geometry (SCG'98)*, pages 143–145, New York, June 1998. Association for Computing Machinery.
7. D. Eppstein. Dynamic euclidean minimum spanning trees and extrema of binary functions. *Disc. Comp. Geom.*, 13:111–122, 1995.
8. J. M. Rabaey et al. Picoradio supports ad hoc ultra-low power wireless networking. *IEEE Computer Magazine*, pages 42–48, July 2000.
9. K. Nakano, S. Olariu, and A. Y. Zomaya. Energy-efficient permutation routing in radio networks. *IEEE Transactions on Parallel and Distributed Systems*, 12(6):544–557, 2001.
10. D. Patel. Energy in ad-hoc networking for the picoradio. Master's thesis, UC Berkeley, 2000.
11. F. P. Preparata and M. I. Shamos. *Computational Geometry.* Springer, 1985.
12. T. S. Rappaport. *Wireless Communication.* Prentice Hall, 1996.
13. K. R. Varadarajan and P. K. Agarwal. Approximation algorithms for bipartite and non-bipartite matching in the plane. In *SODA*, pages 805–814, 1999.
14. Peng-Jun Wan, G. Calinescu, Xiang-Yang Li, and Ophir Frieder. Minimum-energy broadcast routing in static ad hoc wireless networks. In *IEEE Infocom*, 2001.
15. J. E. Wieselthier, G. D. Nguyen, and A. Ephremides. On the construction of energy-efficient broadcast and multicast trees in wireless networks. In *IEEE Infocom*, volume 2, pages 585–594. IEEE, 2000.