Alcom-FT Technical Report Series
ALCOMFT-TR-03-28

Approximating Energy Efficient Paths in Wireless Multi-Hop Networks

Stefan Funke, Domagoj Matijevic, Peter Sanders

Max-Planck-Institut f. Informatik, 66123 Saarbrücken, Germany
{funke,dmatijev,sanders}@mpi-sb.mpg.de *

Abstract. Given the positions of n sites in a radio network we consider the problem of finding routes between any pair of sites that minimize energy consumption and do not use more than some constant number k of hops. Known exact algorithms for this problem required $\Omega(n \log n)$ per query pair (p, q). In this paper we relax the exactness requirement and only compute approximate $(1+\epsilon)$ solutions which allows us to guarantee constant query time using linear space and $O(n \log n)$ preprocessing time. The dependence on ϵ is polynomial in $1/\epsilon$.
One tool we employ might be of independent interest: For any pair of points $(p, q) \in P \subseteq \mathbb{Z}^2$ we can report in constant time the cluster pair (A, B) representing (p, q) in a well-separated pair decomposition of P.

1 Introduction

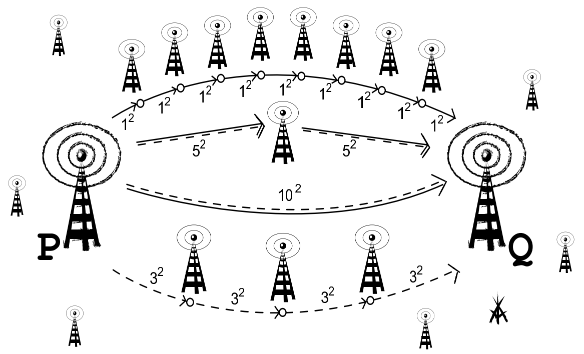

Fig. 1. A Radio Network and 9, 4, 2, 1-hop paths from P to Q with costs 9, 36, 50, 100

Radio networks connecting a number of stations without additional infrastructure have recently gained considerable interest. Since the sites often have limited power supply, the energy consumption of communication is an important optimization criterion. We study this problem using the following simple geometric graph model: Given a set P of n points in $\mathbb{Z}^2$, we consider the complete graph $(P, P \times P)$ with edge weight $\omega(p, q) = |pq|^\delta$ for some constant $\delta > 1$ where $|pq|$ denotes the Euclidean distance between p and q. The objective is to find an approximate shortest path between two query points subject to the

* This work was partially supported by the IST Programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

constraint that at most k edges of the graph are used in the path. For $\delta = 2$ the edge weights reflect the exact energy requirement for free space communication. For larger values of δ (typically between 2 and 4), we get a popular heuristic model for absorption effects [Rap96,Pat00]. Limiting the number of 'hops' to k can account for the distance independent overhead for using intermediate nodes. For a model with node dependent overheads refer to Section 4.

Our main result is a data structure that uses linear space and can be built in time $O(n \log n)$ for any constants k, $\delta > 1$, and $\epsilon > 0$. In constant time it allows to compute k-hop paths between arbitrary query points that are within a factor $(1 + \epsilon)$ from optimal. When k, δ, and ϵ are considered variables, the query time remains constant and the preprocessing time is bounded by a polynomial in k, δ, and $1/\epsilon$.

The algorithm has two main ingredients that are of independent interest. The first part, discussed in Section 2, is based on the observation that for approximately optimal paths it suffices to compute a shortest path for a constant size subset of the points — one point for each square cell in some grid that depends on the query points. This subset can be computed in time $O(\log n)$ using well known data structures supporting (approximate) quadratic range queries [BKOS,AM00]. These data structures and in particular their space requirement are independent of k, δ, and ϵ. Some variants even allow insertion and deletion of points in $O(\log n)$ time. Section 3 discusses the second ingredient. Well separated pair decompositions [CK92] allow us to answer arbitrary approximate path queries by precomputing a linear number of queries. We develop a way to access these precomputed paths in constant time using hashing. This technique is independent of path queries and can be used for retrieving any kind of information stored in well separated pair decompositions. Section 4 discusses further generalizations and open problems.

This extended abstract omits most proofs which can be found in the long version of the paper.

Related Work Chan, Efrat, and Har-Peled [EH98,CE01] observe that for $\omega(p, q) = f(|pq|^\delta)$ and $\delta \geq 2$, exact geometric shortest paths are equivalent to shortest paths in the Delaunay triangulation of P, i.e., optimal paths can be computed in time $O(n \log n)$. Note that this approach completely collapses for k hop paths because most Delaunay edges are very short. The main contribution of Chan et al. is a sophisticated $O(n^{4/3+\gamma})$ time algorithm for computing exact geometric shortest paths for monotone cost functions $\omega(p, q) = f(|pq|)$ where γ is any positive constant. For quadratic cost functions with offsets $\omega(p, q) = |pq|^2 + C$, Beier, Sanders, and Sivadasan reduce that to $O(n^{1+\gamma})$, to $O(kn \log n)$ for k-hop paths, and to $O(\log n)$ time queries for *two hop* paths using linear space and $O(n \log n)$ time preprocessing. The latter result is very simple, it uses Voronoi diagrams and an associated point location data structure.

Thorup and Zwick [ThoZwi01] show that for general graphs and unrestricted k, it is impossible to construct a distance oracle which answers queries $2a - 1$ approximatively using space $o(nn^{1/a})$.

2 Fast Approximate k-hop Path Queries

We consider the following problem: Given a set P of n points in $\mathbb{Z}^2$ and some constant k, report for a given query pair of points $p, q \in P$, a polygonal path $\pi = \pi(p,q) = v_0 v_1 v_2 \dots v_l$, with vertices $v_i \in P$ and $v_0 = p, v_l = q$ which consists of at most k segments, i.e. $l \leq k$, such that its weight $\omega(\pi) = \sum_{0 \leq i < l} \omega(v_i, v_i + 1)$ is minimized. By $\pi_{opt} = \pi_{opt}(p,q)$ we denote an optimal path from p to q under this criterion.

In the following we assume that the weight function ω is of the form $\omega(a,b) = |ab|^\delta$ with $\delta > 1$ (the case $\delta \leq 1$ is trivial as we just need to connect p and q directly by one hop).

2.1 Preliminaries

Before we introduce our procedure for reporting approximate k-hop paths, we need to refer to some standard data structures from Computational Geometry which will be used in our algorithm.

Theorem 1 (Exact Range Query). *Given a set P of n points in $\mathbb{Z}^2$ one can build a data structure of size $O(n \log n)$ in time $O(n \log n)$ which for a given axis aligned query rectangle $R = [x_l, x_u] \times [y_l, y_u]$ reports in $O(\log n)$ time either that R contains no point or outputs a point $p \in P \cap R$.*

The data structure can be maintained dynamically such that points can be inserted and deleted in $O(\log n \log \log n)$ amortized time. The preprocessing time then increases to $O(n \log n \log \log n)$ and the query time to $O(\log n \log \log n)$. All the $\log \log n$ factors can be removed if only either *insertions* or *deletions are allowed.*

Proof. We use the standard 2-level range-tree construction. From the resulting $O(\log^2 n)$ query time we can get rid of one $\log n$ by fractional cascading, see [BKOS]. The whole construction can be maintained dynamically allowing insertions and deletions using the results in [MN90]. □

In fact, the algorithm we will present will also work with an approximate range reporting data structure such as the one presented in [AM00,AM98]. The part of their result relevant for us can be stated in the following theorem:

Theorem 2 (Approximate Range Query). *Given a set P of n points in $\mathbb{Z}^2$ one can build a data structure of size $O(n)$ in time $O(n \log n)$ which for a given axis aligned query rectangle $R = [x_l, x_u] \times [y_l, y_u]$ with diameter ω reports in $O(\log n + \frac{1}{\alpha})$ time either that the rectangle $R' = [x_l + \alpha\omega, x_u + \alpha\omega] \times [y_l + \alpha\omega, y_u + \alpha\omega]$ contains no point or outputs a point $p \in P \cap R'$.*

The data structure can be maintained dynamically such that points can be inserted and deleted in $O(\log n)$ time.

Basically this approximate range searching data structure works well if the query rectangle is fat; and since our algorithm we present in the next section will only query square rectangular regions, all the results in [AM00] and [AM98] apply. In fact we do not even need α to be very small, $\alpha = 1$ turns out to be OK. So the use of an approximate range searching data structure helps us to get rid of the $\log n$ factor in space and some $\log \log n$ factors for the dynamic version. But to keep presentation simple we will assume for the rest of this paper that we have an exact range searching data structure at hand. Furthermore, let us briefly cite two well known inequalities that turn out to be of use in the following analysis.

Minkowski's inequality For some $\delta > 1$ and a_i, $b_i > 0$, Minkowski's sum inequality states that

$$\left(\sum_{i=1}^{k}(a_i + b_i)^\delta\right)^{\frac{1}{\delta}} \leq \left(\sum_{i=1}^{k} a_i^\delta\right)^{\frac{1}{\delta}} + \left(\sum_{i=1}^{k} b_i^\delta\right)^{\frac{1}{\delta}}.$$

Hölder's inequality Let

$$\frac{1}{p} + \frac{1}{q} = 1$$

with p, $q > 1$. Then, for some a_i, $b_i > 0$ Hölder's inequality for sums states that

$$\sum_{i=1}^{k} a_i b_i \leq \left(\sum_{i=1}^{k} a_i^p\right)^{\frac{1}{p}} \left(\sum_{i=1}^{k} b_i^q\right)^{\frac{1}{q}}.$$

Equality holds when $b_i = c a_i^{p-1}$, for some $c > 0$.

2.2 Computing Approximate k-hop Paths for many Points

We will now focus on how to process a k-hop path query for a pair of points p and q assuming that we have already constructed the data structure for orthogonal range queries (which can be done in time $\mathrm{O}(n \log n)$).

Lemma 1. *For the optimal path π_{opt} connecting p and q we have* $\frac{|pq|^\delta}{k^{\delta-1}} \leq |\pi_{\text{opt}}| \leq |pq|^\delta$.

Proof. As we can connect p and q directly using one hop, the upper bound follows immediately. For the lower bound observe that the cheapest way to connect p and q is to divide the segment pq into k subsegments of equal length, which yields the lower bound. □

Definition 1. *We define the axis-aligned square of side-length l centered at the midpoint of a segment pq as the* frame *of p and q, $F(pq, l)$.*

Lemma 2. *The optimal path π_{opt} connecting p and q lies within the frame $F(pq, k^{(\delta-1)/\delta}|pq|)$ of p and q.*

Proof. Assume that the optimal path visits some point r outside the frame F. Note that such a path has Euclidean length at least $|pr| + |rq| > k^{(\delta-1)/\delta}|pq|$ and therefore the cost is lower bounded by $\frac{(|pr|+|rq|)^\delta}{k^{\delta-1}} > \frac{(k^{(\delta-1)/\delta}|pq|)^\delta}{k^{\delta-1}} = |pq|^\delta$ which in turn implies that doing one direct hop from p to q is even better. $\square$

We are now armed to state our algorithm to compute a k-hop path which is a $(1+\epsilon)$ approximation to the optimal k-hop path from p to q.

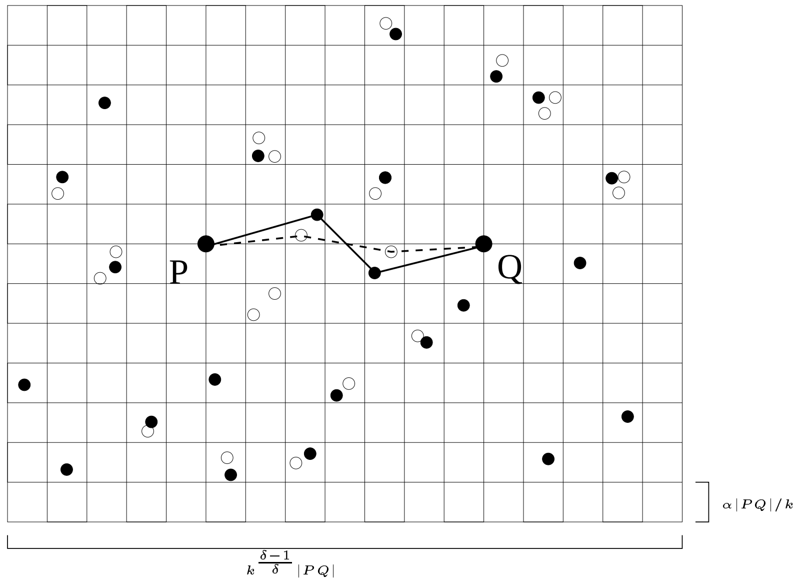

Fig. 2. 3-hop-query for P and Q: representatives for each cell are denoted as solid points, the optimal path is drawn dotted, the path computed by the algorithm solid

k-hop-Query(p,q, ϵ)

1. Put a grid of cell-width $\alpha \cdot |pq|/k$ on the frame $\mathrm{F}(pq, k^{(\delta-1)/\delta}|pq|)$ with $\alpha = \frac{1}{4\sqrt{2}} \cdot \frac{\epsilon}{\delta}$.
2. For each grid cell C perform an orthogonal range query to either certify that the cell is empty or report one point inside which will serve as a representative for C .
3. Compute the optimal k-hop path $\pi(p,q)$ with respect to all representatives and $\{p,q\}$.
4. Return $\pi(p,q)$

Please look at Figure 2.2 for a schematic drawing of how the algorithm computes the approximate k-hop path. It remains to argue about correctness and running time of our algorithm. Let us first consider its running time.

Lemma 3. *k-hop-Query(p, q, ϵ) can be implemented to return a result in time $O(\frac{\delta^2 \cdot k^{(4\delta-2)/\delta}}{\epsilon^2} \cdot T_R(n) + T_{k,\delta}(\frac{\delta^2 \cdot k^{(4\delta-2)/\delta}}{\epsilon^2}))$, where $T_R(n)$ denotes the time for one 2-dimensional range query on the original set of n points and $T_{k,\delta}(x)$ denotes the time for the exact computation of a minimal k-hop path for one pair amongst x points under the weight function $\omega(pq) = |pq|^\delta$.*

Proof. By the choice of frame and cell width, it is easy to see that in the first step of our algorithm we generate a grid of size $O((\frac{\delta \cdot k^{(2\delta-1)/\delta}}{\epsilon})^2)$ which is $O(\frac{\delta^2 \cdot k^{(4\delta-2)/\delta}}{\epsilon^2})$. For each of the cells we perform an orthogonal range query each of which takes $T_R(n)$ time. For all the representatives which we have found, we run an exact minimum k-hop path algorithm which takes $T_{k,\delta}(\frac{\delta^2 \cdot k^{(4\delta-2)/\delta}}{\epsilon^2})$. □

Let us now turn to the correctness of our algorithm, i.e. for any given ϵ, we want to show that our algorithm returns a k-hop path of weight at most $(1+\epsilon)$ times the weight of the optimal path. We will show that only using the representatives of all the grid cells there exists a path of at most this weight. In the following we assume that the optimal path π_{opt} consists of a sequence of points $p_0 p_1 \dots p_j$, $j \leq k$ and $l_i = |p_{i-1} p_i|$.

Before we get to the actual proof of this claim, we need to state a small technical lemma.

Lemma 4. *For any $\delta > 1$ and $l_i, \xi > 0$ the following inequality holds*

$$\frac{\sum_{i=1}^k (l_i + \xi)^\delta}{\sum_{i=1}^k l_i^\delta} \leq \left(\frac{\sum_{i=1}^k (l_i + \xi)}{\sum_{i=1}^k l_i} \right)^\delta$$

Proof. Note that Minkowski's inequality directly implies

$$\left(\sum_{i=1}^k (l_i + \xi)^\delta \right)^{\frac{1}{\delta}} \leq \left(\sum_{i=1}^k l_i^\delta \right)^{\frac{1}{\delta}} + \left(\sum_{i=1}^k \xi^\delta \right)^{\frac{1}{\delta}} = \left(\sum_{i=1}^k l_i^\delta \right)^{\frac{1}{\delta}} + k^{\frac{1}{\delta}} \cdot \xi$$

which in turns give

$$\left(\frac{\sum_{i=1}^k (l_i + \xi)^\delta}{\sum_{i=1}^k l_i^\delta} \right)^{\frac{1}{\delta}} \leq 1 + \frac{k^{\frac{1}{\delta}} \cdot \xi}{(\sum_{i=1}^k l_i^\delta)^{\frac{1}{\delta}}}.$$

Moreover, note that Hölder's inequality implies

$$\left(\frac{\sum_{i=1}^k l_i^\delta}{k} \right)^{\frac{1}{\delta}} \geq \frac{\sum_{i=1}^k l_i}{k}$$

for $b_i = 1$, $i = 1, \dots, k$, $p = \delta$ and $q = \frac{\delta}{\delta - 1}$, which completes the proof. □

Lemma 5. *k-hop-Query(p, q, ϵ) computes a k-hop path from p to q of weight at most $(1+\epsilon)\omega(\pi_{\text{opt}}(p,q))$ for $0 < \epsilon \leq 1$.*

Proof. Using Lemma 4, with ξ denoting the possible error incurred by taking the respective representative edge, it is sufficient to show that

$$\left(\frac{\sum_{i=1}^{k}(l_i+\xi)}{\sum_{i=1}^{k} l_i}\right)^{\delta} \leq 1+\epsilon.$$

Knowing that the absolute 'detour'[1] incurred by replacing its endpoints by the respective representatives in their grid cell is bounded by $\xi \leq 2\sqrt{2}\alpha\frac{d}{k}$ we have

$$\left(1+\frac{k\cdot 2\sqrt{2}\alpha\frac{d}{k}}{\sum_{i=1}^{k} l_i}\right)^{\delta} \leq 1+\epsilon.$$

Thus, observing that $(\sum_{i=1}^{k} l_i)/d$ is at least 1, we get following bound on α

$$\alpha \leq \frac{(1+\epsilon)^{1/\delta}-1}{2\sqrt{2}}.$$

Consider the functions $f(x)=(1+\epsilon)^x$ and $g(x)=1+\frac{\epsilon\cdot x}{2}$ in the range $x\in[0,1]$. Clearly, $f(0)=g(0)$ and as for $\epsilon\in[0,1]$,

$$\frac{f'(x)}{g'(x)}=\frac{(1+\epsilon)^x\ln(1+\epsilon)}{\epsilon/2}\geq 1$$

we have $f(x)\geq g(x)$ for $x\in[0,1]$ and therefore it suffice to choose

$$\alpha=\frac{1}{4\sqrt{2}}\cdot\frac{\epsilon}{\delta}.$$

□

2.3 Computing Optimal k-hop Paths for few Points

In our approximation algorithm we reduced the problem of computing an approximate k-hop path from p to q to one exact k-hop path computation of a small, i.e. constant number of points (only depending on k,δ and ϵ). Still, we have not provided a solution for this problem yet. In the following we will present first a generic algorithm which works for all possible δ and then quickly review the exact algorithm presented by [BSS02] which only works for the case $\delta=2$, though.

Layered Graph Construction We can consider almost the complete graph with all edge weights explicitly stored (except for too long edges, which cannot be part of the optimal solution) and then use the following construction:

---

[1] Here the analysis has to be changed slightly when using approximate range reporting data structures: for $\alpha=1$ we might get twice the 'detour' assumed here.

Lemma 6. *Given a connected graph $G(V, E)$ with $|V| = n$, $|E| = m$ with weights on the edges and one distinguished node $s \in V$, one can compute for all $p \in V - \{s\}$ the path of minimum weight using at most k edges in time $O(km)$.*

Proof. We assume that the graph G has self-loop edges (v, v) with assigned weight 0. Construct $k + 1$ copies $V^{(0)}, V^{(1)}, \ldots, V^{(k)}$ of the vertex set V and draw a directed edge $(v^{(i)}, w^{(i+1)})$ iff $(v, w) \in E$ with the same weight. Compute the distances from $s^{(0)}$ to all other nodes in this layered, acyclic graph. This takes time $\mathrm{O}(km)$ as each edge is relaxed only once. □

So in our subproblem we can use this algorithm and the property that each representative has $\mathrm{O}(\frac{\delta^2 k^2}{\epsilon^2})$ adjacent edges (all other edges are too long to be useful) to obtain the following corollary:

Corollary 1. *The subroutine of our algorithm to solve the exact k-hop problem on $O(\frac{\delta^2 \cdot k^{(4\delta-2)/\delta}}{\epsilon^2})$ points can be solved in time $O(\frac{\delta^4 \cdot k^{(7\delta-2)/\delta}}{\epsilon^4})$ for arbitrary δ, ϵ.*

Reduction to Nearest Neighbor In [BSS02] the authors presented an algorithm which for the special case $\delta = 2$ computes the optimal k-hop path in time $\mathrm{O}(kn \log n)$ by dynamic programming and an application of geometric nearest neighbor search structures to speed up the update of the dynamic programming table. Applied to our problem we get the following corollary:

Corollary 2. *The subroutine of our algorithm to solve the exact k-hop problem on $O(\frac{k^3}{\epsilon^2})$ points can be solved in time $O(\frac{k^5}{\epsilon^2} \cdot \log \frac{k}{\epsilon})$ if $\delta = 2$.*

2.4 Summary

Let us summarize our general result in the following Theorem (we give the bound for the case where an approximate nearest neighbor query data structure as mentioned in Theorem 2 is used).

Theorem 3. *We can construct a dynamic data structure allowing insertions and deletions with $O(n)$ space and $O(n \log n)$ preprocessing time such that $(1+\epsilon)$ approximate minimum k-hop path queries under the metric $\omega(p, q) = |pq|^\delta$ can be answered in time $O(\frac{\delta^2 \cdot k^{(4\delta-2)/\delta}}{\epsilon^2} \cdot \log n + \frac{\delta^4 \cdot k^{(7\delta-2)/\delta}}{\epsilon^4})$.*

The query time does not change when using exact range query data structures, only space, preprocessing and update times get slightly worse (see Theorem 1). For the special case of $\delta = 2$ we obtain a slightly improved query time of $\mathrm{O}(\frac{\delta^2 \cdot k^{(4\delta-2)/\delta}}{\epsilon^2} \cdot \log n + \frac{\delta^2 \cdot k^{(5\delta-2)/\delta}}{\epsilon^2} \cdot \log \frac{\delta k}{\epsilon}))$.

3 Precomputing Approximate k-hop Paths for Constant Query Time

In the previous section we have seen how to answer a (p, q) query in $O(\log n)$ time (considering k, δ, ϵ as constants). Standard range query data structures were the only precomputed data structures used. Now we explain how additional precomputation can further reduce the query time. We show how to precompute a linear number of k-hop paths, such that for every (p, q), a slight modification of one of these precomputed paths is a $(1+\epsilon)(1+2\psi)^2$ approximate k-hop path and such a path can be accessed in constant time. Here $\psi > 0$ is the error incurred by the use of these precomputed paths and can be chosen arbitrarily small (the size of the well-separated pair decomposition then grows, though).

3.1 The Well-Separated Pair Decomposition

We will first briefly introduce the so-called *well-separated pair decomposition* due to Callahan and Kosaraju ([CK92]).

The *split-tree* of a set P of points in $\mathbb{R}^2$ is the tree constructed by the following recursive algorithm:

SplitTree(P)

1. if size(P)=1 then return leaf(P)
2. partition P into sets P_1 and P_2 by halving its minimum enclosing box $R(P)$ along its longest dimension
3. return a node with children (SplitTree(P_1), SplitTree(P_2))

Although such a tree might have linear depth and therefore a naive construction as above takes quadratic time, Callahan and Kosaraju in [CK92] have shown how to construct such a binary tree in $O(n \log n)$ time. With every node of that tree we can conceptually associate the set A of all points contained in its subtree as well as their minimum enclosing box $R(A)$. By $r(A)$ we denote the radius of the minimum enclosing disk of $R(A)$. We will also use A to denote the node associated with the set A if we know that such a node exists.

For two sets A and B associated with two nodes of a split tree, $d(A, B)$ denotes the distance between the centers of $R(A)$ and $R(B)$ respectively. A and B are said to be *well-separated* if $d(A, B) > sr$, where r denotes the radius of

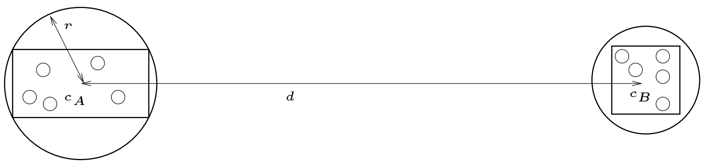

Fig. 3. Clusters A and B are 'well-separated' if $d > s \cdot r$

the larger of the two minimum enclosing balls of $R(A)$ and $R(B)$ respectively. s is called the *separation constant.*

In [CK92], Callahan and Kosaraju present an algorithm which, given a split tree of a point set P with $|P| = n$ and a separation constant s, computes in time $\mathrm{O}(n(s^2 + \log n))$ a set of $\mathrm{O}(n \cdot s^2)$ additional *blue* edges for the split tree, such that

- the point sets associated with the endpoints of a blue edge are well-separated with separation constant s.
- for any pair of leaves (a, b), there exists exactly one blue edge that connects two nodes on the paths from a and b to their lowest common ancestor $lca(a, b)$ in the split tree

The split tree together with its additional blue edges is called the *well-separated pair decomposition (WSPD).*

3.2 Using the WSPD for Precomputing Path Templates

In fact the WSPD is exactly what we need to efficiently precompute k-hop paths for all possible $\Theta(n^2)$ path queries. So we will use the following preprocessing algorithm:

1. compute a well-separated pair decomposition of the point set with $s = k^{(\delta-1)/\delta} \cdot 8\delta \cdot \frac{1}{\psi}$
2. for each *blue* edge compute a $(1 + \epsilon)$-approximation to the lightest k-hop path between the centers of the associated bounding boxes

At query time, for a given query pair (p, q), it remains to find the unique blue edge (A, B) which links a node of the path from p to $lca(p, q)$ to a node of the path from q to $lca(p, q)$. We take the precomputed k-hop path associated with this blue edge, replace its first and last node by s and t respectively and return this modified path.

In the following we will show that the returned path is indeed a $(1+\epsilon)(1+2\psi)^2$ approximation of the lightest k-hop path from p to q. Later we will also show that this path can be found in constant time. For the remainder of this section let $\pi^P_{\mathrm{opt}}(x, y)$ denote the optimal k-hop path between two points x, y not necessarily in P such that all hops have starting and end point in P (except for the first and last hop). We first start with a lemma which formalizes the intuition that the length of an optimal k-hop path does not change much when perturbing the query points slightly.

Lemma 7. *Given a set of points P and two pairs of points (a, b) and (a', b') with $d(a, b) = d$ and $d(a, a') \leq c$, $d(b, b') \leq c$ with $c \leq \frac{\psi d}{k^{(\delta-1)/\delta} \cdot 4\delta}$, then we have $\omega(\pi^P_{opt}(a', b')) \leq (1 + 2\psi)\omega(\pi^P_{opt}(a, b)$.*

Proof. Consider the path $\pi^P_{opt}(a, b) = v_0 v_1 \ldots v_k$ with $v_0 = a$, $v_k = b$ and consider its modification $\pi' = a' v_1 \ldots v_{k-1} b'$. For the analysis we will distinguish between the cases where the edge (a,v_1) ((v_{k-1},b) respectively) is 'long' or 'short'. If $(c+|av_1|)^\delta \le (1+\psi)|av_1|^\delta$, we are safe and this is the case when $|av_1|$ is *long* edge, namely $|av_1| \ge \frac{c}{(1+\psi)^{1/\delta}-1}$. So let us now consider the case that (a,v_1) is short. Clearly an upper bound on the cost of $\pi^P_{opt}(a',b')$ is given by:

$$\begin{aligned}\omega(\pi^P_{opt}(a',b')) &\le \omega(\pi') = \omega(\pi^P_{opt}(a,b)) - |av_1|^\delta + |a'v_1|^\delta - |bv_{k-1}|^\delta + |b'v_{k-1}|^\delta \\ &\le \omega(\pi^P_{opt}(a,b)) - |av_1|^\delta + (|av_1|+c)^\delta - |bv_{k-1}|^\delta + (|bv_{k-1}|+c)^\delta\end{aligned}$$

And since for $0 \le x < y, \delta \ge 1$ we have $(c+x)^\delta - x^\delta \le (c+y)^\delta - y^\delta$ and the respective edges are short, we can continue with

$$\begin{aligned}&\le \omega(\pi^P_{opt}(a,b)) + 2\cdot((\frac{c}{(1+\psi)^{1/\delta}-1}+c)^\delta - (\frac{c}{(1+\psi)^{1/\delta}-1})^\delta) \\ &= \omega(\pi^P_{opt}(a,b)) + 2c^\delta\cdot((\frac{(1+\psi)^{1/\delta}}{(1+\psi)^{1/\delta}-1})^\delta - (\frac{1}{(1+\psi)^{1/\delta}-1})^\delta) \\ &= \omega(\pi^P_{opt}(a,b)) + 2c^\delta\cdot\frac{\psi}{((1+\psi)^{1/\delta}-1)^\delta} \\ &\le \omega(\pi^P_{opt}(a,b)) + 2c^\delta\psi\cdot(\frac{2\delta}{\psi})^\delta \\ &\le \omega(\pi^P_{opt}(a,b))(1+2c^\delta\psi\cdot(\frac{2\delta}{\psi})^\delta\cdot\frac{k^{\delta-1}}{d^\delta})\end{aligned}$$

So finally it just remains to choose c such that $2\cdot(\frac{2\delta c}{d})^\delta\cdot\frac{k^{\delta-1}}{\psi^{\delta-1}} \le \psi$ which is certainly true for $c \le \frac{\psi d}{k^{(\delta-1)/\delta}\cdot 4\delta}$ □

The following corollary of the above Lemma will be used later in the proof:

Corollary 3. *Given a set of points P and two pairs of points (a,b) and (a',b') with $d(a,b) = d$ and $d(a,a') \le c$, $d(b,b') \le c$ with $c \le \frac{\psi d}{k^{(\delta-1)/\delta}\cdot 8\delta}$, then we have $\omega(\pi^P_{opt}(a',b')) \le (1+2\psi)\omega(\pi^P_{opt}(a,b))$ as well as $\omega(\pi^P_{opt}(a,b)) \le (1+2\psi)\omega(\pi^P_{opt}(a',b')$.*

Proof. Clearly the first claim holds according to Lemma 7. For the second one observe that $d' = |a'b'| \ge d - 2\cdot c$ and then apply the Lemma again. □

Applying this Corollary, it is now straightforward to see that the approximation ratio of the modified template path is $(1+2\psi)^2(1+\epsilon)$.

Lemma 8. *Given a well separated pair decomposition of a point set $P \subset \mathbb{Z}^2$ with separation constant $s = \frac{k^{(\delta-1)/\delta}\cdot 8\delta}{\psi}$, the path $\pi(p,q)$ returned for a query pair (p,q) is a $(1+2\psi)^2(1+\epsilon)$ approximate k-hop path from p to q.*

Proof. Let (A,B) be the unique cluster pair connected by a blue edge with $p \in A$, $q \in B$, c_A, c_B their respective cluster centers. By the choice of the separation parameter and Lemma 7, we know that $|pc_A|, |qc_B| \le \frac{\psi|c_A c_B|}{k^{(\delta-1)/\delta}\cdot 8\delta}$ and therefore $\omega(\pi(p,q)) \le (1+2\psi)(1+\epsilon)\omega(\pi_{opt}(c_A,c_B)) \le (1+2\psi)^2(1+\epsilon)\omega(\pi(p,q))$. □

We leave it to the reader to figure out the right choice for ψ and ϵ to obtain an arbitrary approximation quality of $(1+\phi)$, but clearly $\psi, \epsilon \in \Omega(\phi)$.

3.3 Retrieving Cluster Pairs for query points in O(1) time

In the previous paragraphs we have shown that using properties of the well-separated pair decomposition, it is possible to compute $O(n)$ 'template paths' such that for any query pair (s,t) out of the $\Omega(n^2)$ possible query pairs, there exists a good template path which we can modify to obtain a good approximation to the lightest k-hop path from s to t. Still, we have not shown yet how to determine this good template path for a given query pair (s,t) in constant time. We note that the following description does not use any special property of our original problem setting, so it may apply to other problems, where the well-separated pair decomposition can be used to encode in $O(n)$ space sufficient information to cover a query space of $\Omega(n^2)$ size.

Gridding the Cluster Pairs The idea of our approach is to round the centers c_A, c_B of a cluster pair (A,B) which is part of the WSPD to canonical grid points $\widetilde{c_A}, \widetilde{c_B}$ such that for any query pair (s,t) we can determine $\widetilde{c_A}, \widetilde{c_B}$ in constant time. Furthermore we will show that there is only a constant number of cluster pairs (A', B') which have their cluster centers rounded to the same grid positions $\widetilde{c_A}, \widetilde{c_B}$, so well-known hashing techniques can be used to store and retrieve the respective blue edge and also some additional information $I_{(A,B)}$ (in our case: the precomputed k-hop path) associated with that edge.

In the following we assume that we have already constructed a WSPD of the point set P with a separation constant $s > 4$. For any point $p \in \mathbb{Z}^2$, let $\mathrm{snap}(p,w)$ denote the closest grid-point of the grid with cell-width w originated at $(0,0)$ and let $H : \mathbb{Z}^4 \to (I \times E)^*$ denote a hash table data structure which maps pairs of integer points in the plane to a list of pairs consisting of some information type and a (blue) edge in the WSPD. Using universal hashing [CW79] this data structure has constant expected access time.

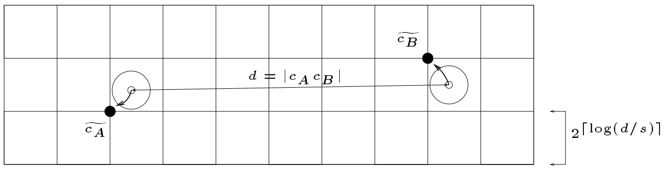

Fig. 4. Cluster centers c_A and c_B are snapped to closest grid points $\widetilde{c_A}$ and $\widetilde{c_B}$

Preprocessing

- For every blue edge connecting clusters (A,B) in the split tree
  - $c_A \leftarrow \mathrm{center}(R(A))$, $c_B \leftarrow \mathrm{center}(R(B))$
  - $w \leftarrow |c_A c_B|/s$
  - $\widetilde{w} \leftarrow 2^{\lceil \log w \rceil}$

- $\widetilde{c_A} \leftarrow \mathrm{snap}(c_A, \widetilde{w})$
- $\widetilde{c_B} \leftarrow \mathrm{snap}(c_B, \widetilde{w})$
- Append $((I_{(A,B)}, (A, B)))$ to $H[(\widetilde{c_A}, \widetilde{c_B})]$

Look at Figure 4 for a sketch of the preprocessing routine for one cluster pair (A, B). Clearly this preprocessing step takes linear time in the size of the WSPD. So given a query pair (s, t), how to retrieve the information $I_{(A,B)}$ stored with the unique cluster pair (A, B) with $s \in A$ and $t \in B$?

Query(p, q)

– $w' \leftarrow |pq|/s$
– $\widetilde{w_1} \leftarrow 2^{\lceil \log w' \rceil - 1}$
– $\widetilde{w_2} \leftarrow 2^{\lceil \log w' \rceil}$
– $\widetilde{w_3} \leftarrow 2^{\lceil \log w' \rceil + 1}$
– for grid-widths w_i, $i = 1, 2, 3$ and adjacent grid-points $\widetilde{c_p}, \widetilde{c_q}$ of p and q respectively
  - Inspect all items $(I_{(A,B)}, (A, B))$ in $H[(\widetilde{c_p}, \widetilde{c_q})]$
    * if $p \in A$ and $q \in B$ return $I_{(A,B)}$

In this description we call a grid-point $\widetilde{g}$ *adjacent* to a point p if $|\widetilde{g}p|_x, |\widetilde{g}p|_y < \frac{3}{2}\widetilde{w}$, where $|\cdot|_{x/y}$ denotes the horizontal/vertical distance. Clearly there are at most 9 *adjacent* points for any point p in a grid of width $\widetilde{w}$. In the remainder of this section we will show that this query procedure outputs the correct result (the unique $I_{(A,B)}$ with $s \in A$ and $t \in B$ such that (A, B) is blue edge in the WSPD) and requires only constant time. In the following we stick to the notation that $\widetilde{w} = 2^{\lceil \log |c_A c_B|/s \rceil}$, where c_A, c_B are the cluster centers of the cluster pair (A, B) we are looking for.

Lemma 9. *For $s > 4$, we have $\widetilde{w_i} = \widetilde{w}$ for some $i \in \{1, 2, 3\}$.*

Proof. As A and B are well-separated and $p \in A$ and $q \in B$, we know that $|c_A p|, |c_B q| < |c_A c_B|/s$. Therefore $|c_A c_B|(1 - 2/s) < |pq| < |c_A c_B|(1 + 2/s)$ and hence

$$\lceil \log \frac{|c_A c_B|(1 - 2/s)}{s} \rceil < \lceil \log \frac{|pq|}{s} \rceil < \lceil \log \frac{|c_A c_B|(1 + 2/s)}{s} \rceil$$

Using the fact that $s > 4$ we obtain

$$\lceil \log \frac{|c_A c_B|}{s} - 1 \rceil < \lceil \log \frac{|pq|}{s} \rceil < \lceil \log \frac{|c_A c_B|}{s} + 1 \rceil$$

and therefore

$$\lceil \log \frac{|c_A c_B|}{s} \rceil - 1 < \lceil \log \frac{|pq|}{s} \rceil < \lceil \log \frac{|c_A c_B|}{s} \rceil + 1$$

which proves the Lemma. □

This Lemma says that at some point the query procedure uses the correct grid-width as determined by c_A and c_B. Furthermore for any given grid-width and a pair of query points p and q, there are at most $9 \cdot 9 = 81$ pairs of adjacent grid points to inspect. We still need to argue that given the correct grid-width $\widetilde{w}$, the correct pair of grid points $(\widetilde{c_A}, \widetilde{c_B})$ is amongst these ≤ 81 possible pairs of grid points that are inspected.

Lemma 10. *For $\widetilde{w_i} = \widetilde{w}$, $\widetilde{c_A}$ and $\widetilde{c_B}$ are amongst the inspected grid-points.*

Proof. In the following we will restrict on the part to show that c_A is adjacent to p. Clearly we have $|c_A p| < \frac{|c_A c_B|}{s} \leq \widetilde{w}$, so in particular $|c_A p|_x, |c_A p|_y < \widetilde{w}$. Furthermore $|c_A \widetilde{c_A}|_x, |c_A \widetilde{c_A}|_y, \leq \widetilde{w}/2$ and hence $|\widetilde{c_A} p|_{x/y} < \frac{3}{2}\widetilde{w}$, i.e. c_A is *adjacent* to p. □

The last thing to show is that only a constant number of cluster pairs (A, B) can be rounded during the preprocessing phase to a specific pair of grid positions $(\widetilde{g}_1, \widetilde{g}_2)$ and therefore we only have to scan a list of constant size that is associated with $(\widetilde{g}_1, \widetilde{g}_2)$. Before we can prove this, we have to cite a Lemma from the original work of Callahan and Kosaraju on the WSPD [CK92].

Lemma 11 (CK92). *Let C be a d-cube and let $S = \{A_1, \ldots, A_l\}$ be a set of nodes in the split tree such that $A_i \cap A_j = \emptyset$ and $l_{\max}(p(A_i)) \geq l(C)/c$ and $R(A_i)$ overlaps C for all i. Then we have $l \leq (3c + 2)^d$.*

Here $p(A)$ denotes the parent of a node A in the split tree, $l_{\max}(A)$ the longest side of the minimum enclosing box of $R(A)$.

Lemma 12. *Consider a WSPD of a point set P with separation constant $s > 4$, grid width $\widetilde{w}$ and a pair of grid points $(\widetilde{g}_1, \widetilde{g}_2)$. The number of cluster pairs (A, B) such that c_A and c_B are rounded to $(\widetilde{g}_1, \widetilde{g}_2)$ is $O(1)$.*

Proof. In the following we will establish a lower bound on $r(p(A))$ such that we can apply Lemma 11, since $r(p(A)) \leq l_{\max}(p(A))$. As $(p(A), B)$ is not a blue edge in the WSPD we have $r(p(A)) > \frac{|c_{p(A)} c_B|}{s}$. Furthermore $|c_{p(A)} c_B| \geq |c_A c_B| - r(p(A))$ and hence using the fact that $\frac{s\widetilde{w}}{2} < |c_A c_B| \leq s\widetilde{w}$ we obtain the following lower bound

$$r(p(A) > \frac{s\widetilde{w}}{2 + 2s} = \frac{\widetilde{w}}{2 + 2/s}$$

So for $s > 4$ we obtain $l_{\max}(p(A)) > \frac{2}{5}\widetilde{w}$ Before we can apply Lemma 11, observe that if the clusters around $\widetilde{g}_1$ which are paired with some clusters around $\widetilde{g}_2$ are not disjoint (i.e. one is the parent of another), we can just refine those clusters until no cluster is the parent of another, this will only increase the number of links to 'the other side'. But applying Lemma 11 for the cube centered at $\widetilde{g}_1$ with side-length $\widetilde{w}$ and $c = 5/2$ bounds even this larger number of clusters by $(3 \cdot 5/2 + 2)^2 < 91$. As the same bound holds for the clusters around $\widetilde{g}_2$, the number of cluster pairs that might be assigned to $(\widetilde{g}_1, \widetilde{g}_2)$ is less than $91 \cdot 91 < 9000$ and therefore O(1). □

Putting everything together we get the main theorem of this section:

Theorem 4. *Given a well-separated pair decomposition of a point set P with separation constant $s > 4$. Then we can construct a data structure in space $O(n \cdot s^2)$ and construction time $O(n \cdot s^2)$ such that for any pair of points (p, q) in P we can determine the unique pair of clusters (A, B) that is part the well-separated pair decomposition with $p \in A$, $q \in B$ in constant time.*

Together with the results of the previous Section we obtain the following main result of our paper:

Theorem 5. *Given a set of points $P \subset \mathbb{Z}^2$, a distance function $\omega : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{R}^+$ of the form $\omega(p, q) = |pq|^\delta$, where $\delta \geq 1$ and $k \geq 2$ are constants, we can construct a data structure of size $O(\frac{1}{\epsilon^2} \cdot n)$ in preprocessing time $O(\frac{1}{\epsilon^4} \cdot n \log n + \frac{1}{\epsilon^6} \cdot n)$ such that for any query (p, q) from P, a $(1+\epsilon)$-approximate lightest k-hop path from p to q can be obtained in constant $O(1)$ time which does not depend on δ, ϵ, k.*

We also remark that there are techniques to maintain the well-separated pair decomposition dynamically, and so our whole construction can be made dynamic as well (see [CK95]).

4 Discussion

We have developed a data structure for constant approximate shortest path queries in a simple model for geometric graphs. Although this model is motivated by communication in radio networks, it is sufficiently simple to be of independent theoretical interest and possibly for other applications. For example, Chan, Efrat, and Har-Peled [EH98,CE01] use similar concepts to model the fuel consumption of airplanes routed between a set P of airports.

We can also further refine the model. For example, the above flight application would require more general cost functions. Here is one such generalization: If the cost of edge (p, q) is $|pq|^\delta + C_p$ for a node dependent cost offset C_p, our result remains applicable under the assumption of some bound on the offset costs. In Lemma 5 we would choose the cell representative as the node with minimum offset in the cell (this can be easily incorporated into the standard geometric range query data structures). The offset could model distance independent energy consumption like signal processing costs or it could be used to steer away traffic from devices with low battery power.

References

[AM00] S. Arya and D. M. Mount:*Approximate range searching*, Computational Geometry: Theory and Applications, (17), 135-152, 2000

[AM98] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, A. Wu: *An optimal algorithm for approximate nearest neighbor searching*, Journal of the ACM, 45(6):891-923, 1998

[BSS02] R. Beier, P. Sanders, N. Sivadasan: *Energy Optimal Routing in Radio Networks Using Geometric Data Structures* Proc. of the 29th Int. Coll. on Automata, Languages, and Programming, 2002.

[BKOS] M. de Berg, M. van Krefeld, M. Overmars, O. Schwarzkopf: *Computational Geometry: Algorithms and Applications*, Springer, 1997

[CK92] P.B. Callahan, S.R. Kosaraju: *A decomposition of multi-dimensional point-sets with applications to k-nearest-neighbors and n-body potential fields*, Proc. 24th Ann. ACM Symp. on the Theory of Computation, 1992

[CK95] P.B. Callahan, S.R. Kosaraju: *Algorithms for Dynamic Closest Pair and n-Body Potential Fields*, Proc. 6th Ann. ACM-SIAM Symp. on Discrete Algorithm, 1995

[CW79] J.L. Carter and M.N. Wegman. Universal Classes of Hash Functions. *Journal of Computer and System Sciences, 18(2):143–154, 1979*

[CE01] T. Chan and A. Efrat. Fly cheaply: On the minimum fuel consumption problem. *Journal of Algorithms*, 41(2):330–337, November 2001.

[EH98] A. Efrat, S. Har-Peled: *Fly Cheaply: On the Minimum Fuel-Consumption Problem*, Proc. 14th ACM Symp. on Computational Geometry 1998.

[MN90] K. Mehlhorn, S. Näher: *Dynamic Fractional Cascading*, Algorithmica (5), 1990, 215–241

[Pat00] D. Patel. Energy in ad-hoc networking for the picoradio. Master's thesis, UC Berkeley, 2000.

[Rap96] T. S. Rappaport. *Wireless Communication*. Prentice Hall, 1996.

[ThoZwi01] M.Thorup and U.Zwick. Approximate Distance Oracles *Proc. of 33rd Symposium on the Theory of Computation 2001.*