2009 Sixth International Symposium on Voronoi Diagrams

Approximate Shortest Path Queries in Graphs Using Voronoi Duals

Shinichi Honiden*†, Michael E. Houle†, Christian Sommer*†, and Martin Wolff†
*The University of Tokyo, Tokyo, Japan †National Institute of Informatics, Tokyo, Japan

***Abstract*—We propose an approximation method to answer point-to-point shortest path queries in undirected graphs, based on random sampling and Voronoi duals. We compute a simplification of the graph by selecting nodes independently at random with probability p. Edges are generated as the Voronoi dual of the original graph, using the selected nodes as Voronoi sites. This overlay graph allows for fast computation of approximate shortest paths for general, undirected graphs. The time–quality tradeoff decision can be made at query time. We provide bounds on the approximation ratio of the path lengths as well as experimental results. The theoretical worst-case approximation ratio is bounded by a logarithmic factor. Experiments show that our approximation method based on Voronoi duals has extremely fast preprocessing time and efficiently computes reasonably short paths.**

***Keywords*-shortest path; approximation; distance oracle; graph Voronoi diagram**

I. Introduction

We wish to answer shortest path queries for large graphs such as those stemming from transportation networks, social networks, protein interaction networks, and the web graph. We could use a classical algorithm such as Dijkstra's [1], which has worst-case running time $O(m + n \log n)$, where n denotes the number of nodes and m the number of edges. However, for large graphs, only a small portion of the graph can be considered at query time. If preprocessing is allowed, queries can be answered much more quickly. The best algorithm for precomputing everything, that is, for solving the All Pairs Shortest Path Problem, runs in time $O(n^3/\log^2 n)$ [2]. Shortest path queries could then be answered in constant time. Unfortunately, for this method the preprocessing time is prohibitive.

The goal is to mediate between these two extremes. The desired tradeoff between preprocessing time and query time depends on the needs of the application.

A. Related work

In the following, we give a brief overview of related work for shortest path and distance queries.

Theoretical. Data structures allowing for shortest path or distance queries are referred to as distance oracles. Their construction is closely related to that of graph spanners. For a pair of nodes (s, t), an approximate distance oracle is said to have stretch (α, β) if it returns a distance in the range $[d(s, t), \alpha \cdot d(s, t) + \beta]$. A girth conjecture by Erdős implies that, for general undirected graphs, distance oracles with multiplicative stretch $\alpha < 2k + 1$ need $\Omega(n^{1+1/k})$ space. An algorithm by Thorup and Zwick [3] constructs such an oracle in expected time $O(kmn^{1/k})$ with query time $O(k)$ and stretch $(2k - 1, 0)$. For constant k, except for the preprocessing time, all their bounds are tight. For planar (directed) graphs with integer weights, an algorithm by Thorup [4] constructs a $(1 + \epsilon, 0)$-stretch oracle in time $O(n \log^3 n \log(n\Delta))$, where Δ denotes the largest weight. Unfortunately, for huge non-planar graphs, these results are not practical.

Practical. The main focus of practical investigations so far has been on large road networks. There has been considerable recent progress: for the road networks of Europe or the USA, using a high-performance computer, a speed-up of several orders of magnitude compared to Dijkstra's algorithm can be achieved with a preprocessing time in the tens of minutes [5]. Unfortunately, theoretical bounds on both query time and preprocessing time are difficult to obtain. Goldberg and Harrelson [6] proposed a variant of A* search [7] in which distances are precomputed to a small set of 'landmark' vertices. Hierarchical methods [8], [9] provide an efficient framework, especially in the case of road networks. Sanders and Schultes [5], [9], [10] developed a method to compute shortest paths in 'almost constant time' with a carefully designed structure consisting of precomputed shortest paths. Their solution is tailored to perform exceptionally well for road networks, where graphs are almost planar and nodes have small constant degrees. Precomputation is time- and space-consuming; however, it is still manageable in practice, and allows for extremely fast query times.

However, even though road networks constitute the most common and popular application of shortest path query algorithms to date, other challenging applications exist. Computer networks, social networks, protein interaction networks, and the web graph exhibit different degree and structural properties, and may contain hundreds of millions or even billions of nodes. In specific cases, a user might be willing to trade preprocessing time against exactness due to the vast size of the data or due to restricted processing power. These scenarios may require the use of a fast approximation method.

B. Contribution

We propose an approximation method to answer shortest path queries in general, undirected graphs with positive edge weights, based on random sampling and graph Voronoi

978-0-7695-3781-8/09 $25.00 © 2009 IEEE
DOI 10.1109/ISVD.2009.30

duals [11], [12]. In preprocessing, each node is selected as a Voronoi site independently at random with probability p, and the Voronoi dual is computed for the selected sites (Sec. II). For $p < 1$, the resulting dual graph is expected to be smaller than the original graph. At query time, search for the shortest path from source s to target t can potentially be done faster in the Voronoi dual. We let the shortest path in the Voronoi dual guide the search for an approximate shortest path in the original graph. We prove that the expected approximation ratio is at most logarithmic in the number of nodes on the actual shortest path, and that this bound is tight (Sec. III). Our experimental results show that, in practice, the approximation is much better than the stated theoretical bound (Sec. IV).

II. Graph Voronoi Dual

In this section we present the construction of the graph Voronoi dual, and show how a shortest path in the dual can be used to find an approximation of the shortest path in the original graph. In the following, unless indicated otherwise, we consider only undirected, connected graphs. First, we introduce the notions and terminology needed in this paper.

A. Preliminaries

A *weighted* graph $G = (V, E, \omega)$ consists of a graph (V, E) together with a *weight function* $\omega : E \to \mathbb{R}$. We assume positive edge weights; that is, $\omega : E \to \mathbb{R}^+$. For the remainder of the paper, we will refer to the number of nodes and edges of the graph by $n = |V|$ and $m = |E|$, respectively.

In an edge-weighted graph $G = (V, E, \omega)$, a *path* from $s = u_0 \in V$ to $t = u_h \in V$ is a node sequence $(u_0, u_1, \ldots, u_h)$ for which $(u_i, u_{i+1}) \in E$ for all $i \in \{0, 1, \ldots h-1\}$. The *length* of a path P is the sum of its edge weights $\ell(P) := \sum_{i=0}^{h-1} \omega(u_i, u_{i+1})$. A subpath P' of a path P is a subsequence of its nodes $P' = (u_i, u_{i+1}, \ldots u_j)$, $0 \le i < j \le h$. A *simple path* is a path without repeated vertices. Let $\mathcal{P}_G(u, v)$ denote the set of paths from u to v in a graph G. The *distance* $d(u, v)$ between two nodes u, v is the length of a shortest path from u to v; that is, $d(u, v) = \min_{P \in \mathcal{P}(u,v)} \ell(P)$. If $\mathcal{P}(u, v) = \emptyset$ then $d(u, v) := \infty$. Let $SP_G(s, t)$ be an arbitrary shortest path from s to t. Analogously to the multiplicative stretch of a distance oracle, we define the *stretch* of a path P from s to $t \neq s$ as the ratio $\ell(P)/\ell(SP_G(s, t))$.

B. Graph Voronoi Diagram

The classical Voronoi diagram is a distance-based decomposition of a metric space relative to a discrete set, the Voronoi sites. Mehlhorn [11] and Erwig [12] proposed an analogous decomposition, the *Graph Voronoi Diagram*, for undirected and directed graphs respectively.

Definition 1 (Graph Voronoi Diagram [11], [12])**.** *In a graph $G = (V, E, \omega)$, the* Voronoi diagram *for a set of nodes $K = \{v_1, \ldots, v_k\} \subseteq V$ is a disjoint partition $\mathsf{Vor}_{(G,K)} := \{V_1, \ldots, V_k\}$ of V such that for each node $u \in V_i$, $d(u, v_i) \le d(u, v_j)$ for all $j \in \{1, \ldots, k\}$.*

The V_i are called *Voronoi regions*. The graph Voronoi diagram is not necessarily unique, as a node u may have the same distance to more than one Voronoi node. Let $\mathsf{vor}(u)$ denote the index i of the Voronoi region V_i containing u; that is, $\mathsf{vor}(u) = i \Leftrightarrow u \in V_i$.

Analogously to the Delaunay triangulation dual for classical Voronoi diagrams of point sets, we define the Voronoi dual for graphs.

Definition 2. *Let $G = (V, E, \omega)$ be a weighted graph and $\mathsf{Vor}_{G,K}$ its Voronoi diagram. The* Voronoi dual *is the graph $G^* = (K, E^*, \omega^*)$ with edge set $E^* := \{(v_i, v_j) : v_i, v_j \in K$ and $\exists u \in V_i \wedge \exists w \in V_j : (u, w) \in E\}$, and edge weights $\omega^*(v_i, v_j) := \min_{\substack{u \in V_i, w \in V_j \\ (u,w) \in E}} \{d(v_i, u) + \omega(u, w) + d(w, v_j)\}$.*

Figure 3 illustrates two graph Voronoi diagrams for the same (planar) graph but with different edge weights. Although the classical Voronoi dual of a non-degenerate set of points in the plane is always a triangulation, the graph Voronoi dual is not necessarily a triangulation, even for planar graphs. For example, a graph Voronoi dual may have nodes whose removal would disconnect the graph.

Erwig [12, Thm. 2] showed that the graph Voronoi diagram can be constructed with a single Dijkstra search in time $O(m + n \cdot \log n)$. A heap is used to store the shortest path distances from nodes to their closest Voronoi node. The heap is initialized to store the Voronoi nodes themselves. Thereafter, as long as there are nodes in the queue, the minimum is extracted from the heap and processed (or 'settled') by assigning to it a Voronoi region, storing the distance to its Voronoi node, and adding to or updating its neighbors in the queue. We slightly modify this construction of the Voronoi *diagram* [12, Sec. 3.1] to compute the Voronoi *dual* — that is, to also compute E^* and ω^*. Whenever a node u is settled in the Dijkstra search (and thereby assigned to a Voronoi region $V_{\mathsf{vor}(u)}$), for all its settled neighbors u' of different Voronoi regions ($\mathsf{vor}(u) \neq \mathsf{vor}(u')$), if no edge exists, we add the edge $(v_{\mathsf{vor}(u)}, v_{\mathsf{vor}(u')})$ with weight $\omega^*(v_{\mathsf{vor}(u)}, v_{\mathsf{vor}(u')}) = d(v_{\mathsf{vor}(u)}, u) + \omega(u, u') + d(u', v_{\mathsf{vor}(u')})$, and if there already is an edge in G^* representing a longer path in G from $v_{\mathsf{vor}(u)}$ to $v_{\mathsf{vor}(u')}$, we decrease its length. The final edge weight $\omega^*(v_{\mathsf{vor}(u)}, v_{\mathsf{vor}(u')})$ equals the length of a shortest path in G that crosses the Voronoi border, which may be larger than the actual distance $d(v_{\mathsf{vor}(u)}, v_{\mathsf{vor}(u')})$. This modification of the construction increases the time complexity by at most a constant factor.

Definition 3. *Given a path $P = (u_0, u_1, \ldots, u_h)$, the* Voronoi path *of P is the sequence of vertices $P^* = (v_{\mathsf{vor}(u_0)}, v_{\mathsf{vor}(u_1)}, \ldots, v_{\mathsf{vor}(u_h)})$.*

Note that the Voronoi path P^* may not necessarily be simple, as multiple consecutive occurrences of nodes $v_{\mathsf{vor}(u_i)}$ are possible in P^*. They are treated as a single occurrence, and such paths are deemed to be equivalent.

Lemma 1. *For any path $P = (u_0, \ldots, u_h)$ in an undirected graph $G = (V, E, \omega)$, the corresponding Voronoi path P^* exists and is unique.*

Proof: Omitted in this version. ∎

Definition 4. *For a path P^* in the Voronoi dual G^* of a graph G, the* Voronoi sleeve *is the subgraph of G induced by the nodes in the union of all Voronoi regions V_i for which v_i lies on P^*, $\mathsf{Sleeve}_{(G,G^*)}(P^*) := G\left[\bigcup_{v_i \in P^*} V_i\right]$.*

With the definitions at hand we can now state the approximation method.

C. Approximation Algorithm

Given a graph G and its Voronoi dual G^* we answer (approximate) shortest path queries between source s and target t using the following algorithm. The algorithm first searches for a shortest path $SP_{G^*}(v_{\mathsf{vor}(s)}, v_{\mathsf{vor}(t)})$ in the smaller Voronoi dual G^*. This path determines the subgraph $\mathcal{S} = \mathsf{Sleeve}(SP_{G^*}(v_{\mathsf{vor}(s)}, v_{\mathsf{vor}(t)}))$, whose shortest path $SP_{\mathcal{S}}(s,t)$ approximates the shortest path $SP_G(s,t)$ in G. The shortest path in the Voronoi dual 'guides' the Dijkstra search in the original graph.

Algorithm 1 (Construction)**.** *Input: Graph $G = (V, E, \omega)$, Sampling Rate $p \in [0, 1]$.*
Output: Voronoi dual G^* with Voronoi nodes selected independently at random with probability p.

1) *Random sampling: Generate the set of Voronoi nodes by selecting each node of V independently at random: $\forall v \in V, Pr[v \in K] = p$.*
2) *Compute a Voronoi dual $G^* = (K, E^*, \omega^*)$ using the modified version of Erwig's algorithm [12, sec. 3.1].*
3) *Return G^*.*

Lemma 2. *For a graph $G = (V, E)$ with $n := |V|$ and $m := |E|$, Algorithm 1 takes time $O(m + n \log n)$.*

Proof: See [12, sec. 3.1]. ∎

Algorithm 2 (Query)**.** *Input: Graph G, Voronoi dual G^*, Source s, Target t.*
Output: an approximate shortest path P from s to t.

1) *Find Voronoi source $v_{\mathsf{vor}(s)}$ from s and Voronoi target $v_{\mathsf{vor}(t)}$ from t. If thereby a shortest path $SP_G(s,t)$ has been found, return it.*
2) *Compute a shortest path from $v_{\mathsf{vor}(s)}$ to $v_{\mathsf{vor}(t)}$ in the Voronoi dual G^*: $SP_{G^*}(v_{\mathsf{vor}(s)}, v_{\mathsf{vor}(t)})$.*
3) *Compute the Voronoi sleeve $\mathcal{S} := \mathsf{Sleeve}(SP_{G^*}(v_{\mathsf{vor}(s)}, v_{\mathsf{vor}(t)}))$.*
4) *Compute a shortest path from s to t in the Voronoi sleeve, $SP_{\mathcal{S}}(s,t)$.*
5) *Return $P = SP_{\mathcal{S}}(s,t)$.*

The running time of Algorithm 2 depends on G and p. Let N^* and M^* denote the random variables measuring the number of nodes and edges of the Voronoi dual. Clearly $E[N^*] = p \cdot n$. The expected query time *without* refinement (computing the shortest path in the Voronoi sleeve) is at most $O(N^* \log N^* + M^*)$. The time for the refinement step depends on the size of the Voronoi sleeve. The analysis will show that the refinement step is not necessary for the approximation ratio to hold for long distance queries; however, it makes a practical difference for the quality of paths. For $p = O(n^{-2/3})$, $E[N^*] = O(n^{1/3})$, and thus we can afford to compute all-pairs shortest path distances in the Voronoi dual G^* in overall linear expected time. This allows for constant-time approximate distance queries.

In the next section, we prove that the expected path length approximation ratio is logarithmic in the number of edges of an exact shortest path.

Theorem 1. *For shortest paths having h edges, Algorithm 2, given a graph and its Voronoi dual with sampling rate p (constructed by Algorithm 1), has expected approximation ratio $O(\log_{1/(1-p)} h)$.*

III. Proof of Theorem 1

The path $SP_{\mathcal{S}}(s,t)$ found by the algorithm is an approximation, since it is possible that no actual shortest path $SP_G(s,t)$ lies entirely within the Voronoi sleeve $\mathcal{S}$. We explain how this is possible, and give an upper bound on the expected length $\ell(SP_{\mathcal{S}}(s,t))$. For this purpose, we prove relationships between the lengths of simple paths P and their corresponding Voronoi paths P^*. The stretch of a path P^* depends on the number and distribution of Voronoi nodes on the path P. In particular, the stretch depends linearly on the largest interval between two Voronoi nodes on the path.

Definition 5. *For a path $P = (u_0, u_1, \ldots, u_h)$ in a graph $G = (V, E, \omega)$, and a set of Voronoi nodes $K \subseteq V$, two Voronoi nodes v_i, v_j on P are called* consecutive *if the subpath between v_i and v_j does not contain another Voronoi node. The* gap *g between two consecutive Voronoi nodes on the path is defined as the number of edges of this subpath. The* largest gap *of a path is the maximum over all gaps between two consecutive Voronoi nodes on the path.*

To simplify the analysis, we initially assume that s and t are Voronoi nodes. Later, we will relax this restriction. We wish to prove that the stretch is at most the size of the largest gap $\bar{h}$ between two Voronoi nodes on the path $SP_G(s,t)$. For the analysis we fix a shortest path $SP_G(s,t) = (s, u_1, u_2, \ldots, u_{h-1}, t)$. If the corresponding Voronoi path $(SP_G(s,t))^*$ is a shortest path from s to t in the Voronoi dual, then the Voronoi sleeve $\mathcal{S}$ also contains $SP_G(s,t)$. Figure 1 gives an example for which $(SP_G(s,t))^*$ is not a shortest path in the dual.

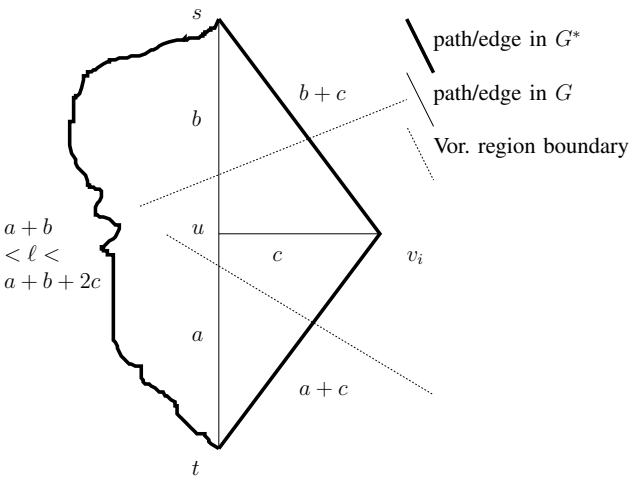

Figure 1. s, t, and v_i are Voronoi nodes. The shortest path from s to t leads through u, which is in v_i's Voronoi region (if $c < a$ and $c < b$), and paths in the Voronoi dual pass through v_i. If $\ell < a + b + 2c$, the shortest path in the Voronoi dual SP_{G^*} takes the left-hand route, and the Voronoi sleeve $\mathcal{S}$ does not contain u.

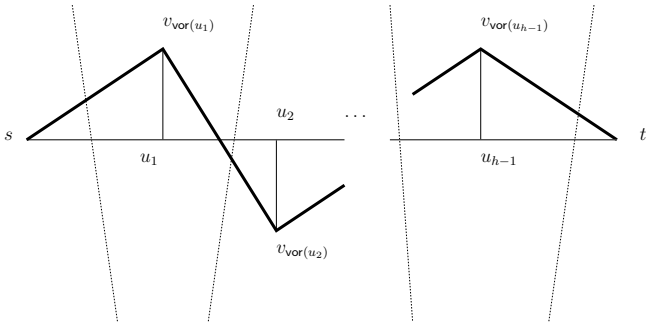

Figure 2. The shortest path between two Voronoi nodes s and t with $h-1$ intermediate nodes $u_1, \dots, u_{h-1}$. The distance between two Voronoi nodes that are adjacent in the Voronoi dual is at most $\omega^*(v_{\mathsf{vor}(u_k)}, v_{\mathsf{vor}(u_{k+1})}) \leq d(v_{\mathsf{vor}(u_k)}, u_k) + \omega(u_k, u_{k+1}) + d(u_{k+1}, v_{\mathsf{vor}(u_{k+1})})$.

In Lemma 3, for any simple path P, we give a worst-case bound on the length of the corresponding Voronoi path. P^* can have maximal stretch if there is no Voronoi node among the intermediate nodes and the corresponding Voronoi nodes have maximal distance (while still satisfying the Voronoi condition).

Lemma 3. *Given a simple path $P = (s, u_1, \dots, u_{h-1}, t)$ between two Voronoi nodes $s = u_0$ and $t = u_h$ with h edges and length $\ell(P)$, the corresponding Voronoi path P^* in the Voronoi dual G^* has at most length $\ell(P^*) \leq h \cdot \ell(P)$. This upper bound is tight.*

Proof: The path contains $h-1$ intermediate nodes and h edges and therefore passes through at most $h+1$ different Voronoi regions. Out of these, at most $h-1$ regions are 'interfering' regions, meaning that the original shortest path does not lead through the corresponding Voronoi nodes but the shortest Voronoi path does. The path length $\ell(P)$ in the original graph is the sum of the edge weights $\ell(P) := d(s,t) = \sum_{k=0}^{h-1} \omega(u_k, u_{k+1})$. The length $d^*(v_{\mathsf{vor}(u_k)}, v_{\mathsf{vor}(u_{k+1})})$ of an edge between two Voronoi nodes on the path P^* can be bounded as follows (see Figure 2):

$$\begin{aligned} d^*(v_{\mathsf{vor}(u_k)}, v_{\mathsf{vor}(u_{k+1})}) &\leq d(v_{\mathsf{vor}(u_k)}, u_k) \\ &\quad + \omega(u_k, u_{k+1}) + d(u_{k+1}, v_{\mathsf{vor}(u_{k+1})}) \end{aligned}$$

From the Voronoi condition, we observe that $\forall j : d(u_k, v_{\mathsf{vor}(u_k)}) \leq d(u_k, v_{\mathsf{vor}(u_j)})$. Due to the assumption that s and t are also Voronoi nodes, this also holds for source and target. That is,

$$\begin{aligned} d(u_k, v_{\mathsf{vor}(u_k)}) &\leq d(s, u_k) \\ d(u_k, v_{\mathsf{vor}(u_k)}) &\leq d(u_k, t) \end{aligned}$$

This yields:

$$\begin{aligned} \ell(P^*) &\leq d^*(s,t) \\ &= d^*(s, v_{\mathsf{vor}(u_1)}) \\ &\quad + \sum_{k=1}^{h-2} \Big[d(v_{\mathsf{vor}(u_k)}, u_k) + \omega(u_k, u_{k+1}) \\ &\quad + d(u_{k+1}, v_{\mathsf{vor}(u_{k+1})}) \Big] + d^*(v_{\mathsf{vor}(u_{h-1})}, t) \\ &\leq \omega(s, u_1) + d(u_1, v_{\mathsf{vor}(u_1)}) \\ &\quad + \sum_{k=1}^{h-2} \Big[d(v_{\mathsf{vor}(u_k)}, u_k) + d(u_{k+1}, v_{\mathsf{vor}(u_{k+1})}) \Big] \\ &\quad + \sum_{k=1}^{h-2} \omega(u_k, u_{k+1}) \\ &\quad + d(v_{\mathsf{vor}(u_{h-1})}, u_{h-1}) + \omega(u_{h-1}, t) \\ &\leq d(s,t) + \sum_{k=1}^{h-1} \Big[d(s, u_k) + d(u_k, t) \Big] = h \cdot \ell(P) \end{aligned}$$

There exist constructions for which the bound can be shown to be tight. For example, for any choice of $a > \epsilon > 0$, the edge weights of G may be chosen such that $d(u_k, v_{\mathsf{vor}(u_k)}) = a - \epsilon$, $\omega(u_k, u_{k+1}) = \epsilon$, and $\omega(s, u_1) = \omega(u_{h-1}, t) = a$. Path P has length $2a + (h-2)\epsilon$, and the Voronoi path P^* has length $2a + (h-2)\epsilon + 2(h-1) \cdot (a - \epsilon)$. As $\epsilon \to 0$, the ratio $\ell(P^*)/\ell(P) \to h$. ∎

If in addition to the endpoints there are Voronoi nodes on the shortest path, the maximum stretch is guaranteed to be smaller than the number of edges on the shortest path. In the following lemma, we prove that the maximum stretch is proportional to the largest gap between Voronoi nodes on the path. The proof is a simple composition of Lemma 3, and is supported by the illustration in Figure 2.

Lemma 4. *Let $P = (v_i, u_1, \dots, u_{h-1}, v_j)$ be a simple path of length $\ell(P)$ between two Voronoi nodes $v_i = u_0$ and $v_j =$*

u_h. *Let $\bar{h}$ denote the largest gap of P. The corresponding Voronoi path P^* in the Voronoi dual G^* has at most length $\ell(P^*) \leq \bar{h} \cdot \ell(P)$. This upper bound is tight.*

Proof: Suppose there are $2 + \nu$ Voronoi nodes $u_k = v_{\mathsf{vor}(u_k)}$ on the path. The remaining $h-1-\nu$ nodes are non-Voronoi nodes. We cut the path P into subpaths P_k between Voronoi nodes. Let h_k denote the number of edges between two consecutive Voronoi nodes, which is the number of edges of P_k. The Voronoi path is composed of $1 + \nu$ segments P_k between Voronoi nodes ($\sum_{k=0}^{\nu} \ell(P_k) = P$, $\sum_{k=0}^{\nu} h_k = h$, $\forall k : h_k \leq \bar{h}$). Composition of Lemma 3 leads to the following bound on the path length:

$$\sum_{k=0}^{\nu} h_k \ell(P_k) \leq \sum_{k=0}^{\nu} \max_{\kappa \in \{0,\ldots,\nu\}} h_\kappa \ell(P_k) \leq \bar{h} \cdot \ell(P).$$

Tightness can be shown with the same example as in the proof of Lemma 3. ∎

Lemma 5 gives an upper bound on the expected size of the largest gap.

Lemma 5. *In a path of length $h-1$, where each node has been selected as a Voronoi node independently at random with probability p, the longest sequence of non-Voronoi nodes is of expected length at most $O(\log_{1/(1-p)} h)$.*

Proof: The path can be seen as a sequence of coin tosses, for which we want to bound the expected length of the longest sequence of tails. This problem is known as the Longest Success-Run [13, Ch. 8.5]. We wish to bound the expectation of the maximum of N independent geometric random variables with probability p and sum $h-1-N$ (N itself being a random variable).

To derive a bound on the expectation, we observe that by dropping the sum condition, and by taking the maximum over $h \geq N$ random variables, the maximum value obtained can only increase. The expectation of the maximum of h geometric random variables with probability p is known to be at most $O(\log_{1/(1-p)} h)$ [14, eq. (2.12)]. ∎

We now combine Lemmas 3, 4, and 5 to prove Theorem 1. Consider first the case where s and t are both Voronoi nodes. Let $\bar{h}$ denote the largest gap of some shortest path $SP_G(s,t)$. Lemma 4 implies that the corresponding Voronoi path $(SP_G(s,t))^*$ has length at most $\bar{h} \cdot \ell(SP_G(s,t))$. Trivially, the shortest path in the Voronoi dual is of length no more than that of the Voronoi path; that is, $\ell((SP_G(s,t))^*) \geq \ell(SP_{G^*}(s,t))$. The path $SP_{G^*}(s,t)$ in the Voronoi dual corresponds to a path P' of the same length in the Voronoi sleeve $\mathsf{Sleeve}(SP_{G^*}(s,t))$. Therefore, $\ell(SP_{\mathcal{S}}(s,t)) \leq \ell(P') = \ell(SP_{G^*}(s,t)) \leq \ell((SP_G(s,t))^*) \leq \bar{h} \cdot \ell(SP_G(s,t))$. Recall that nodes are independently selected as Voronoi nodes with sampling rate p. For a shortest path with h edges, the expected largest gap $\bar{h}$ is at most $O(\log_{1/(1-p)} h)$ by Lemma 5.

For the case where either s or t (or both) are not Voronoi nodes, if the path returned by Algorithm 2 has been found in Step 1, it is optimal, and the result holds trivially. For the remainder of the proof we assume that the shortest path has not been found in Step 1. In this case, the path returned is at most as long as the shortest path P_{vor} in G from s to t having $SP_{\mathsf{Sleeve}(SP_{G^*}(v_{\mathsf{vor}(s)}, v_{\mathsf{vor}(t)}))}(v_{\mathsf{vor}(s)}, v_{\mathsf{vor}(t)})$ as a subpath. In the following, we derive an upper bound on $\ell(P_{\mathsf{vor}})$ with respect to the number of edges on the shortest path between s and t, denoted by h'. We have that $\ell(P_{\mathsf{vor}}) \leq d(s, v_{\mathsf{vor}(s)}) + d^*(v_{\mathsf{vor}(s)}, v_{\mathsf{vor}(t)}) + d(v_{\mathsf{vor}(t)}, t)$. Since the shortest path from s to t has not already been found directly in Step 1, it must be true that both $d(s, v_{\mathsf{vor}(s)}) \leq d(s,t)$ and $d(s, v_{\mathsf{vor}(s)}) \leq d(s,t)$. It remains to bound the distance between $v_{\mathsf{vor}(s)}$ and $v_{\mathsf{vor}(t)}$ in the dual graph.

Observe that augmenting the graph G with one edge $(u, v_{\mathsf{vor}(u)})$ of weight $d(u, v_{\mathsf{vor}(u)})$ for each non-Voronoi node $u \in V \setminus K$ affects neither the Voronoi diagram nor the Voronoi dual, since the nodes on the shortest path from $v_{\mathsf{vor}(u)}$ to u cannot be interfered with by another Voronoi node.

In the augmented primal graph, by the triangle inequality, we have that $d(v_{\mathsf{vor}(s)}, v_{\mathsf{vor}(t)}) \leq d(v_{\mathsf{vor}(s)}, s) + d(s,t) + d(t, v_{\mathsf{vor}(t)}) \leq 3d(s,t)$ using a path with at most $1 + h' + 1$ edges. Therefore, the expected distance $d^*(v_{\mathsf{vor}(s)}, v_{\mathsf{vor}(t)})$ is also bounded by $O(\log h') \cdot 3d(s,t)$. The bound for P_{vor} follows directly.

This concludes the proof of Theorem 1.

IV. Experiments

In the following, we provide an experimental evaluation for our implementation of the Voronoi shortest path approximation method. The preprocessing and query times are compared with those of Dijkstra's algorithm and with those of related but exact methods.

A. Algorithms

Benchmarking. We measure the performance of the methods against the bidirectional version of Dijkstra's algorithm, in terms of the ratio of the number of nodes settled by Dijkstra's algorithm over the number of nodes settled by the Voronoi method. This ratio, which we will refer to as the *speed-up* of the method, can be used to evaluate the performance of Steps 1, 2, and 4 of Algorithm 2. In addition, we count the number of marked regions to account for Step 3.

The use of the Voronoi sleeve in Steps 3 and 4 of Algorithm 2 leads to practical improvements in accuracy; however, the example in Figure 1 shows that for general graphs the worst-case stretch does not improve. For all the experiments, we evaluate the method once using the refinement step and once with these Voronoi sleeve steps omitted. For the second type of queries, the reported distance is the sum of the distances from the query source to the

Voronoi source, from the Voronoi source to the Voronoi target, and from the Voronoi target to the query target, as computed in Steps 1 and 2 of Algorithm 2.

1) Voronoi method: Our method using the Voronoi dual can be parameterized using the sampling probability p, the value of which determines the trade-off between approximation quality and speed-up. For the evaluation, we consider three values of the probability — $p = 1/2$, $p = n^{-1/2}$, and $p = n^{-2/3}$ — that produce Voronoi nodesets of expected sizes $n/2$, $\sqrt{n}$, and $\sqrt[3]{n}$ respectively. The variants are referred to as VORHALF, VORROOT, and VORCUBERT.

2) Other methods: Sanders and Schultes [5, Table 1] provide a detailed overview of methods for accelerated point-to-point shortest path queries in road networks. Bauer et al. [22, p.13] list another set of methods and compare their performance on several transportation networks. We select some of the fastest methods for comparison with our algorithm. Unless stated otherwise, we will use the naming conventions of [5], [22] to refer to these methods.

- Highway Hierarchies (HH) [10] are based on the observation that a certain class of edges (the 'highway' edges) tend to have greater representation among the portion of the shortest paths that are not in the vicinity of either the source or target. A recursive computation of these edges, paired with a contraction step, leads to a hierarchy of graphs that enables an impressive speed-up at query time. HH+dist denotes a variant of HH where all higher levels with at most $O(\sqrt{n})$ nodes are replaced by a single distance table. HH+dist+A* is HH combined with A* search and implemented with distance tables [15]. Highway Node Routing (HNR) [9] is another variant of the Highway Hierarchies strategy.
- In the same spirit as HH, Transit Node Routing (TNR) [16] identifies a set of nodes (called 'transit' nodes) that often occur on shortest paths. A table storing the distances between all pairs of these nodes allows any shortest path distance to be computed with a small number of table look-ups. Two variants are listed: TNR-eco with economical space consumption, and TNR-gen with generous space consumption.
- The Arc-Flag method [23] computes a partition of the graph and then, for each component and for each shortest path ending in that component, it labels the first edge. A variant of this method, SHARC [24], incorporates techniques developed for Highway Hierarchies.
- Contraction Hierarchies (CHHNR) [8] is an extension of highway hierarchies in which the graph is further simplified using contraction operations. Many variants have been proposed; we consider only the variant with the fastest preprocessing time, $\text{CHHNR}_{\text{EDS1235}}$, and the variant with the best speed-up, $\text{CHHNR}_{\text{EVSQWL}}$. The CHASE method [22] integrates the Contraction Hierarchies and Arc-Flag methods.
- A method based on A* search by Goldberg and Harrelson [6], which we will refer to as simply A*, is one of the first methods with reasonable preprocessing time and good speed-up.
- ALT-m16 [25] is a variant of ALT [26], which in turn is a combination of A*, Landmarks, and speed-up techniques based on the triangle inequality. CALT-m16 and CALT-a64 [22] are two variants of a method that combines ALT and Contraction Hierarchies.

B. Data sets

For the sake of comparison, we consider transportation networks that were used by Sanders and Schultes [5] and Bauer et al. [22], [24] in their evaluations. In addition, to demonstrate that our method is effective for more general graphs, we run experiments with a social network, a citation graph, a router network, and protein interaction networks as data sets. The node degrees of these graphs seem to follow a power-law distribution.

1) Road networks: The road network of Western Europe has been made available for scientific use by the company PTV AG. It covers 14 countries and, with its massive size of 18,010,173 nodes and 42,560,279 directed edges, it serves as an important benchmark for shortest path queries. In order to apply the Voronoi method, we convert the graph into an undirected form. There are two different edge weightings, one representing geographical distances and the other representing driving time. We conduct experiments for both.

2) Public transportation: We also conduct experiments for three European public transportation networks: (1) long railway connections in Europe, with 1,586,862 nodes and 2,402,352 directed edges, (2) the bus network of the Rhein-Main-Verkehrsverbund RMV, with 2,278,066 nodes and 3,417,084 directed edges, and (3) the bus network of the Verkehrsverbund Berlin Brandenburg VBB, with 2,600,818 nodes and 3,901,212 directed edges. The graphs considered by [22], [24] differ slightly from those used for experimentation with the Voronoi method. The numbers of nodes and edges of the RMV and VBB input graphs are nearly identical; however, the long railway graph used in our experimentation has 33% more nodes and edges than in [22], [24]. Again, for the Voronoi experimentation, the graphs were converted into an undirected form.

3) Social networks: We extracted the DBLP computer science bibliography [17] co-author graph from an official XML version downloaded on 24 August 2008. In the graph, two authors are connected by an edge if they have at least one joint publication. This yielded an undirected graph, from which we selected the largest connected component. The final graph is unweighted and consists of 511,163 nodes and 1,871,070 edges.

4) Router topology: CAIDA maintains data on the router-level topology of a portion of the Internet [18]. After

cleaning we obtained an undirected, unweighted graph with 190,914 nodes and 607,610 edges.

5) Citation graph: The citations for 27,400 publications in the high energy physics research literature were used as a data set in the KDD Cup 2003 competition [19]. From these citations, we constructed an undirected, unweighted graph with 352,542 edges.

6) Protein interactions: The Database of Interacting Proteins [20] catalogs experimentally determined interactions among proteins. We extracted the largest connected component, consisting of 19,928 nodes and 82,406 edges. BioGRID is a general repository for interaction data sets [21] from which we extracted the largest connected component, consisting of 4,039 nodes and 43,854 edges.

C. Experimental setting

Our implementation is written in C++ and executed on one core of a 2x2.66 GHz Dual-Core Intel Xeon Desktop with 6 GB 800 MHz DDR2 FB-DIMM running Mac OS X 10.5.6.

Every graph was preprocessed $1,000$ times using different random seeds (250 times for the European road networks). For these runs we report the mean value and standard deviation of the execution time in seconds. After preprocessing, we performed 100 shortest path queries for random (s, t) pairs. For these queries, we provide the mean values and standard deviations of the speed-up relative to the bidirectional version of Dijkstra's algorithm, and of the multiplicative stretch relative to a shortest path.

D. Results and Interpretation

Running times, speed-ups, and approximation qualities for the Voronoi method are listed in Table I, for all data sets. The performances of the other methods are listed in Table II as originally summarized in [5], [8], [22].

Preprocessing For the Voronoi method, as Lemma 2 predicts, the preprocessing cost is extremely low for all three values of p. For the non-planar graphs, the greatest preprocessing times were observed for the largest value, $p = 1/2$. This likely reflects the logarithmic cost of the heap operations associated with the computation of Voronoi regions. At the start of the Dijkstra search, the heap is initialized with all neighbors of the graph Voronoi nodes. When p is large, the initial heap size is a large proportion of the total number of nodes, and the cost of the heap operations becomes significant. On the other hand, when p and the average node degree are both small, the heap evolves smoothly with its size remaining small.

Speed-up For road networks VORHALF achieves moderate speed-ups of approximately 2, which likely reflects the fact that the expected number of nodes of the Voronoi dual is half that of the original graph. For the power-law graphs, probability $p = 1/2$ does not lead to a significant speed-up. One reason for this might be that the Voronoi dual for each of these graphs is quite dense and, as a consequence, the Dijkstra search in the dual explores many nodes until it can find the destination. For the smaller probabilities, larger speed-ups can be observed, but the performance gain is significantly smaller than the speed-ups obtained for almost planar networks. There, the speed-up seems proportional to $1/p$. As expected, if for small values of p the sleeve is used to refine the path, the speed-up decreases drastically due to the large size of this subgraph.

Stretch The Voronoi method achieved stretch values that were surprisingly consistent among different data sets, with most values under 2 and very close to optimal for the road networks. Figure 4 shows the approximate path length versus the shortest path length, with and without the sleeve refinement steps. The theoretical worst-case logarithmic dependency on the number of edges cannot be observed in the experimental results. Refinement using the sleeve substantially improves the stretch in practice, although the theoretical performance is not affected.

V. CONCLUSION

We have presented a simple and general method based on Voronoi duals to efficiently support shortest path queries in undirected graphs with very low preprocessing overheads and competitive query times, at the cost of exactness. The method was shown to be effective on a variety of graph types while remaining a reasonable alternative to existing exact methods specifically designed for transportation networks. The results of our experiments also demonstrate that the approximation ratio in practice is significantly better than the tight theoretical worst-case bound proved in the main theorem of this paper. The maximal distortion of paths in the graph Voronoi dual depends on the distance between nodes in the original graph, unlike Delaunay triangulations of the Euclidean plane, which have constant distortion [27], [28].

An interesting topic for future research would be an expected-case analysis for weighted graphs from a variety of distributions.

It remains open as to whether the Voronoi method presented in this paper can be extended to handle directed graphs. The nature of the Voronoi dual within a directed graph is inherently different from the dual within an undirected graph. The need for path connectivity suggests the construction of two Voronoi diagrams, one where reachability paths are oriented outward from Voronoi nodes and another where reachability paths are oriented inward. As the respective Voronoi regions may not coincide [12], it is not straightforward to define a single dual structure whose shortest path lengths approximate those of the original graph.

Another natural extension is the computation of a hierarchical structure of Voronoi duals, where the Voronoi nodes are chosen through recursive sampling. At a given level of the hierarchy, shortest path queries within the Voronoi dual

would be resolved by a recursive call one level higher in the structure. This is planned for future work.

ACKNOWLEDGMENTS

We thank Georg Troxler (staila technologies) for providing us with a machine for our experiments, and Daniel Delling for supplying the public transportation graphs. The third author thanks Xiang-Yang Li, Elad Verbin, and Uri Zwick for helpful comments and interesting discussions.

REFERENCES

[1] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numerische Math.*, vol. 1, pp. 269–271, 1959.

[2] T. M. Chan, "More algorithms for all-pairs shortest paths in weighted graphs," in *Symposium on Theory of Computing*, 2007, pp. 590–598.

[3] M. Thorup and U. Zwick, "Approximate distance oracles," *J. ACM*, vol. 52, no. 1, pp. 1–24, 2005.

[4] M. Thorup, "Compact oracles for reachability and approximate distances in planar digraphs," *J. ACM*, vol. 51, no. 6, pp. 993–1024, 2004.

[5] P. Sanders and D. Schultes, "Engineering fast route planning algorithms," in *Workshop on Experimental Algorithms*, 2007, pp. 23–36.

[6] A. V. Goldberg and C. Harrelson, "Computing the shortest path: A* search meets graph theory," in *Symposium on Discrete Algorithms*, 2005, pp. 156–165.

[7] J. E. Doran, "An approach to automatic problem-solving," *Machine Intelligence*, vol. 1, pp. 105–124, 1967.

[8] R. Geisberger, P. Sanders, D. Schultes, and D. Delling, "Contraction hierarchies: Faster and simpler hierarchical routing in road networks," in *Workshop on Experimental Algorithms*, 2008, pp. 319–333.

[9] D. Schultes and P. Sanders, "Dynamic highway-node routing," in *Workshop on Experimental Algorithms*, 2007, pp. 66–79.

[10] P. Sanders and D. Schultes, "Engineering highway hierarchies," in *European Symposium on Algorithms*, 2006, pp. 804–816.

[11] K. Mehlhorn, "A faster approximation algorithm for the Steiner problem in graphs," *Information Processing Letters*, vol. 27, no. 3, pp. 125–128, 1988.

[12] M. Erwig, "The graph Voronoi diagram with applications," *Networks*, vol. 36, no. 3, pp. 156–163, 2000.

[13] P. Embrechts, T. Mikosch, and C. Klüppelberg, *Modelling extremal events: for insurance and finance*. Springer-Verlag, 1997.

[14] W. Szpankowski and V. Rego, "Yet another application of a binomial recurrence. Order statistics," *Computing*, vol. 43, no. 4, pp. 401–410, 1990.

[15] D. Delling, P. Sanders, D. Schultes, and D. Wagner, "Highway hierarchies star," in *9th DIMACS Implementation Challenge*, 2006.

[16] H. Bast, S. Funke, D. Matijevic, P. Sanders, and D. Schultes, "In transit to constant time shortest-path queries in road networks," in *Workshop on Algorithm Engineering and Experiments*, 2007.

[17] M. Ley, "The DBLP computer science bibliography: Evolution, research issues, perspectives," in *Symposium on String Processing and Information Retrieval*, 2002, pp. 1–10.

[18] Cooperative Association for Internet Data Analysis, "Router-level topology measurements," Online at http://www.caida.org/tools/measurement/skitter/router_topology/, 2003.

[19] L. Getoor, T. E. Senator, P. Domingos, and C. Faloutsos, Eds., *Knowledge Discovery and Data Mining (SIGKDD)*, 2003.

[20] L. Salwinski, C. S. Miller, A. J. Smith, F. K. Pettit, J. U. Bowie, and D. Eisenberg, "DIP, the database of interacting proteins," *Nucleic Acids Research*, vol. 30, no. 1, pp. 303–305, 2002.

[21] C. Stark, B. Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, and M. Tyers, "BioGRID: a general repository for interaction datasets," *Nucleic Acids Research*, vol. 34, no. 1, pp. 535–539, 2006.

[22] R. Bauer, D. Delling, P. Sanders, D. Schieferdecker, D. Schultes, and D. Wagner, "Combining hierarchical and goal-directed speed-up techniques for Dijkstra's algorithm," in *Workshop on Experimental Algorithms*, 2008, pp. 303–318.

[23] U. Lauther, "An extremely fast, exact algorithm for finding shortest paths in static networks with geographical background," in *Geoinformation und Mobilität*, vol. 22, 2004, pp. 219–230.

[24] R. Bauer and D. Delling, "SHARC: Fast and robust unidirectional routing," in *Workshop on Algorithm Engineering and Experiments*, 2008, pp. 13–26.

[25] D. Delling and D. Wagner, "Landmark-based routing in dynamic graphs," in *Workshop on Experimental Algorithms*, 2007, pp. 52–65.

[26] A. V. Goldberg and R. F. F. Werneck, "Computing point-to-point shortest paths from external memory," in *Workshop on Algorithm Engineering and Experiments*, 2005, pp. 26–40.

[27] D. P. Dobkin, S. J. Friedman, and K. J. Supowit, "Delaunay graphs are almost as good as complete graphs," *Discrete & Comp. Geom.*, vol. 5, pp. 399–407, 1990.

[28] J. M. Keil and C. A. Gutwin, "Classes of graphs which approximate the complete Euclidean graph," *Discrete & Comp. Geom.*, vol. 7, pp. 13–28, 1992.

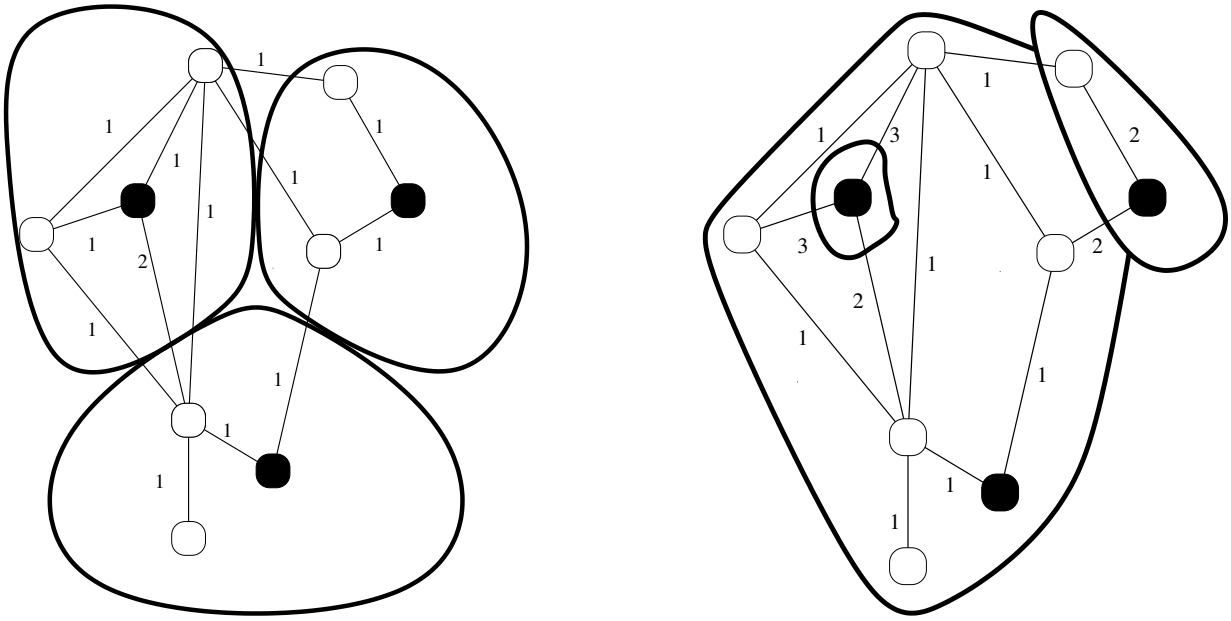

Figure 3. Two graph Voronoi diagrams for the same planar graph but with different edge weights. Voronoi nodes are black and the remaining nodes are white. Even though the graphs are structurally equivalent, the corresponding graph Voronoi diagrams are not.

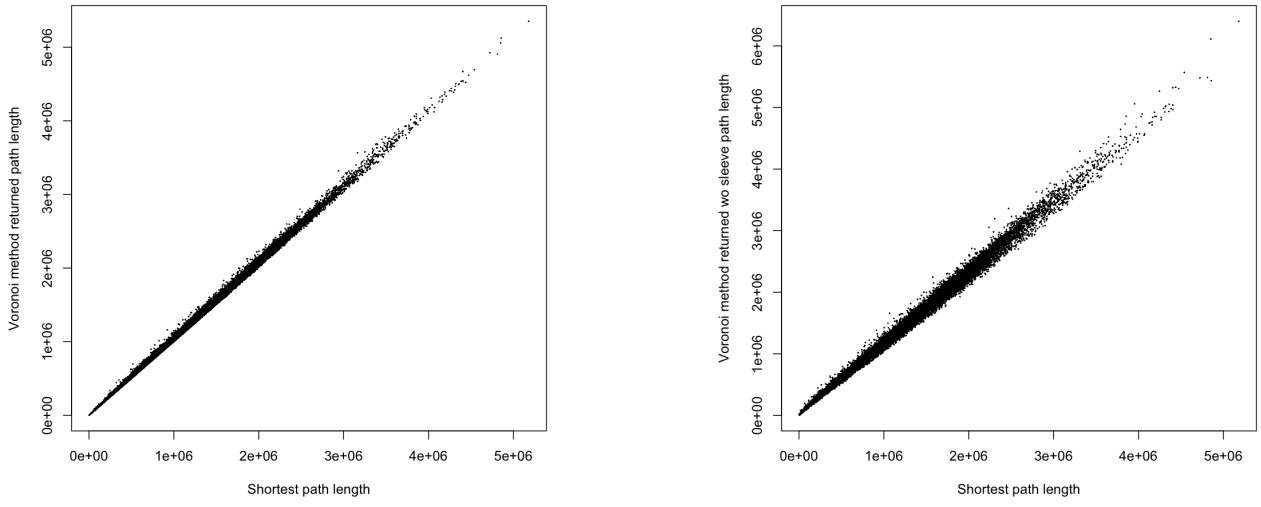

Figure 4. Approximate path length versus actual shortest path length for VORROOT on the European road network, distance metric. Left: using sleeve. Right: with sleeve steps omitted. The theoretical worst-case logarithmic dependency on the number of edges cannot be observed in the experimental results. Refinement using the sleeve substantially improves the stretch in practice, although the theoretical performance is not affected.

method	preprocessing $[s]$	without sleeve		with sleeve	
		speed-up	stretch	speed-up	stretch
PTV European road network, driving time, 18,010,173 nodes, 42,560,279 edges					
VORHALF	31.7686±4.4436	2.6061± 0.0734	1.0394±0.0131	2.5878± 0.0750	1.0111±0.0062
VORROOT	40.5296±3.6423	3,518.0645± 725.2776	1.6613±0.2078	4.9991± 4.9017	1.1291±0.0783
VORCUBERT	31.3372±2.8181	39,918.4988±14,207.5395	1.5544±0.4292	1.5863± 1.1123	1.0405±0.0597
PTV European road network, geographical distance, 18,010,173 nodes, 42,560,279 edges					
VORHALF	29.8365±4.3576	2.6266± 0.0558	1.0307±0.0095	2.5800± 0.0627	1.0139±0.0057
VORROOT	34.2785±3.0609	3,672.4070± 511.1418	1.1821±0.0960	5.9212± 7.9921	1.0390±0.0249
VORCUBERT	22.5531±2.0284	42,266.6442±13,530.5983	1.2882±0.5384	1.6383± 1.4232	1.0141±0.0291
Public transportation, long distance railway, 1,586,862 nodes, 2,402,352 edges					
VORHALF	2.0499±0.1998	1.9511± 0.1231	1.0180±0.0227	1.8972± 0.1367	1.0080±0.0143
VORROOT	1.9086±0.0946	363.8390± 153.4644	1.3813±0.2848	2.8527± 3.3113	1.0829±0.0971
VORCUBERT	1.7633±0.0860	2,116.0373± 1,251.1773	1.5167±0.6610	1.2599± 0.5990	1.0247±0.0658
Public transportation, RMV, 2,278,066 nodes, 3,417,084 edges					
VORHALF	3.7714±0.4064	1.9892± 0.1813	1.0290±0.0255	1.9315± 0.1766	1.0104±0.0131
VORROOT	3.7455±0.2158	789.2912± 328.2714	1.2972±0.2591	3.1802± 5.4237	1.0644±0.0864
VORCUBERT	3.4120±0.1633	5,973.7950± 3,748.1389	1.3522±0.6003	1.3089± 0.9703	1.0204±0.0583
Public transportation, VBB, 2,600,818 nodes, 3,901,212 edges					
VORHALF	4.1409±0.4180	1.9881± 0.6476	1.0335±0.0248	1.9313± 0.5172	1.0075±0.0097
VORROOT	4.0242±0.2914	866.8917± 405.4821	1.4042±0.2516	3.7864± 7.6010	1.0834±0.1000
VORCUBERT	3.7145±0.2333	7,373.2971± 4,742.2783	1.4375±3.3690	1.3427± 1.2759	1.0244±0.0660
DBLP co-authorship, 511,163 nodes, 1,871,070 edges					
VORHALF	0.9145±0.0431	1.3576± 1.4690	1.2093±0.1805	1.3447± 1.4364	1.1419±0.1468
VORROOT	0.8376±0.0430	37.7082± 53.2992	1.9323±0.3591	11.4432±14.8387	1.3954±0.2850
VORCUBERT	0.6041±0.0312	143.8757± 208.7946	2.0033±0.3630	9.9616±12.5412	1.2881±0.2406
CAIDA router topology, 190,914 nodes, 607,610 edges					
VORHALF	0.3050±0.0154	1.3164± 1.1720	1.1810±0.1703	1.2972± 1.1074	1.1283±0.1359
VORROOT	0.1793±0.0092	42.4832± 54.6527	1.7845±0.3533	7.8865± 8.8062	1.2345±0.2175
VORCUBERT	0.1562±0.0081	135.5521± 188.9479	1.8314±0.3755	6.0451± 7.1000	1.1621±0.1837
High energy physics citations, 27,400 nodes, 352,542 edges					
VORHALF	0.1764±0.0100	1.6620± 1.2240	1.3179±0.2909	1.6452± 1.1544	1.2107±0.2323
VORROOT	0.0611±0.0043	40.1114± 21.9262	1.9918±0.4695	11.5248± 7.9582	1.3390±0.3286
VORCUBERT	0.0461±0.0032	101.9210± 58.6233	2.0330±0.4852	9.0423± 7.5795	1.2325±0.2750
Database of Interacting Proteins, 19,928 nodes, 82,406 edges					
VORHALF	0.0117±0.0007	2.2044± 1.0637	1.1887±0.2188	2.1248± 1.0093	1.1183±0.1778
VORROOT	0.0108±0.0007	57.7343± 45.7341	1.8214±0.4084	9.1154± 6.0720	1.3216±0.3030
VORCUBERT	0.0096±0.0006	134.4816± 106.4737	1.9277±0.4444	6.2541± 3.8117	1.2644±0.2703
BioGRID, 4,039 nodes, 43,854 edges					
VORHALF	0.0035±0.0002	1.5086± 0.8003	1.2581±0.2718	1.3722± 0.6858	1.1334±0.1973
VORROOT	0.0025±0.0001	10.7295± 7.9563	1.8676±0.5737	3.0394± 1.9172	1.2753±0.3354
VORCUBERT	0.0024±0.0001	18.6805± 14.7570	1.9412±0.6250	2.7906± 1.7177	1.2308±0.3137

Table I
EXPERIMENTAL RESULTS FOR THE VORONOI METHOD.

PTV European road network, driving time			
		prep. $[s]$	speed-up
CHHNR$_{\text{EDS1235}}$	[8]	602	≈8,505
A*	[6]	780	28
HH	[10]	780	4,002
HH+dist	[10]	900	8,320
HH+dist+A*	[15]	1,320	11,496
HNR	[9]	1,440	4,079
CHHNR$_{\text{EVSQWL}}$	[8]	1,914	≈10,874
TNR-eco	[16]	2,760	471,881
TNR-gen	[16]	9,840	1,129,143

		long distance rail		RMV		VBB	
$\|V\|$		1,192,736		2,277,812		2,599,953	
$\|E\|$		1,789,088		3,416,552		3,899,807	
		prep. $[s]$	speed-up	prep. $[s]$	speed-up	prep. $[s]$	speed-up
CALT-a64	[22]	87	291.84	191	267.11	123	459.30
CALT-m16	[22]	158	182.71	377	159.62	174	281.23
ALT-m16	[25]	291	20.30	556	18.91	604	23.04
CHHNR	[8]	286	1,620.62	2,584	2,077.69	1,636	3,124.59
CHASE	[22]	536	2,660.93	2,863	4,649.26	2,008	10,398.64
SHARC	[24]	12,540	81.04			36,120	118.10

Table II

Road networks: THE TABLE ON THE LEFT IS EXCERPTED FROM SANDERS AND SCHULTES [5, TABLE 1] EXCEPT FOR CHHNR VALUES, WHICH ARE FROM [8, TABLE 1]. PREPROCESSING TIMES ARE CONVERTED FROM MINUTES TO SECONDS TO EASE COMPARISON WITH OUR METHOD. **Public transportation networks**: THE TABLE ON THE RIGHT IS EXCERPTED FROM BAUER ET AL. [22, P.13]. SHARC IS EVALUATED IN [24, P.10]. THE SPEED-UP IS COMPUTED ACCORDING TO THE NUMBER OF SETTLED NODES. MACHINES USED (EXCEPT FOR A*): 2.0 OR 2.6 GHZ PROCESSOR, 8 OR 16 GB RAM, C++ IMPLEMENTATION.