Additive Spanners and Distance Oracles in Quadratic Time

Mathias Bæk Tejs Knudsen*

University of Copenhagen, Copenhagen, Denmark
mathias@tejs.dk

Abstract
Let $G$ be an unweighted, undirected graph. An additive $k$-spanner of $G$ is a subgraph $H$ that approximates all distances between pairs of nodes up to an additive error of $+k$, that is, it satisfies $d_H(u, v) \leq d_G(u, v) + k$ for all nodes $u, v$, where $d$ is the shortest path distance. We give a deterministic algorithm that constructs an additive $O(1)$-spanner with $O(n^{3/3})$ edges in $O(n^3)$ time. This should be compared with the randomized Monte Carlo algorithm by Woodruff [ICALP 2010] giving an additive 6-spanner with $O(n^{4/3} \log^3 n)$ edges in expected time $O(n^2 \log^2 n)$.

An $(\alpha, \beta)$-approximate distance oracle for $G$ is a data structure that supports the following distance queries between pairs of nodes in $G$. Given two nodes $u, v$ it can in constant time compute a distance estimate $\tilde{d}$ that satisfies $d \leq \tilde{d} \leq \alpha d + \beta$ where $d$ is the distance between $u$ and $v$ in $G$. Sommer [ICALP 2016] gave a randomized Monte Carlo $(2,1)$-distance oracle of size $O(n^{5/3} \log n)$ in expected time $O(n^2 \log n)$. As an application of the additive $O(1)$-spanner we improve the construction by Sommer [ICALP 2016] and give a Las Vegas $(2,1)$-distance oracle of size $O(n^{5/3})$ in time $O(n^2)$. This also implies an algorithm that in $O(n^3)$ time gives approximate distance for all pairs of nodes in $G$ improving on the $O(n^2 \log n)$ algorithm by Baswana and Kavitha [SICOMP 2010].

1998 ACM Subject Classification G.2.2 Graph Theory

Keywords and phrases graph algorithms, data structures, additive spanners, distance oracles

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.64

1 Introduction
Let $G = (V, E)$ be an unweighted, undirected graph on $n$ nodes and $m$ edges. A subgraph $H$ of $G$ is an additive $k$-spanner if the following holds for every pair $u, v$ of nodes in $G$:
\[ d_H(u, v) \leq d_G(u, v) + k, \]
where $d_H(u, v)$ and $d_G(u, v)$ is the distance between $u$ and $v$ in $H$ and $G$ respectively. This paper will only consider additive spanners and not multiplicative or mixed spanners, so we will simply say that $H$ is a $k$-spanner when we mean that $H$ is an additive $k$-spanner.

In this paper we consider algorithms constructing $k$-spanners, and there are therefore three interesting parameters: The distortion $k$, the running time of the algorithm, and the size of the spanner created. Elkin and Peleg [19] showed how to construct 2-spanners with $O(n^{3/2})$ edges in $O(n^{5/2})$ time, and Baswana et al [9] gave an algorithm that constructs 6-spanners with $O(n^{4/3})$ edges in $O(n^{2/3}m)$ time.

* Research partly supported by Advanced Grant DFF-0602-02499B from the Danish Council for Independent Research under the Sapere Aude research career programme and by the FNU project AlgoDisc — Discrete Mathematics, Algorithms, and Data Structures.

Additive Spanners and Distance Oracles in Quadratic Time

Table 1 A summary of the performance of selected algorithms that creates a $k$-spanner $H$ from a graph on $n$ nodes. It shows the additive distortion, $k$, and an upper bound on the number of edges in $H$ as well as the running time of the algorithm that constructs $H$.

<table>
<thead>
<tr>
<th>$k$</th>
<th>Number of Edges</th>
<th>Running Time</th>
<th>Comment</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$O(n^{3/2})$</td>
<td>$O(n^{3/2})$</td>
<td>Deterministic</td>
<td>[19]</td>
</tr>
<tr>
<td>2</td>
<td>$O(n^{3/2} \log^{1/2} n)$</td>
<td>$O(n^2 \log^2 n)$</td>
<td>Deterministic</td>
<td>[18]</td>
</tr>
<tr>
<td>2</td>
<td>$O(n^{3/2})$</td>
<td>$O(n^2)$</td>
<td>Deterministic</td>
<td>Theorem 5</td>
</tr>
<tr>
<td>6</td>
<td>$O(n^{4/3})$</td>
<td>$O(n^{2/3} m)$</td>
<td>Deterministic</td>
<td>[9]</td>
</tr>
<tr>
<td>6</td>
<td>$O(n^{4/3} \log^3 n)$</td>
<td>$O(n^2 \log^2 n)$</td>
<td>Randomized Monte Carlo</td>
<td>[33]</td>
</tr>
<tr>
<td>8</td>
<td>$O(n^{4/3})$</td>
<td>$O(n^2)$</td>
<td>Deterministic</td>
<td>Theorem 9</td>
</tr>
</tbody>
</table>

The running time of these algorithms can be improved if we allow the $k$-spanners to be larger by a poly log $n$ factor. Dor, Halperin and Zwick [18] showed that we can construct 2-spanners with $O(n^{3/2} \log^{1/2} n)$ edges in $O(n^2 \log^2 n)$ time, and Woodruff [33] gave an algorithm to construct 6-spanners with $O(n^{4/3} \log^3 n)$ edges in $O(n^2 \log^2 n)$ time. The construction of Woodruff is furthermore randomized Monte Carlo. These results are summarized in Table 1.

These improvements to the running time fit into the following paradigm. For a fixed $k$ the authors find algorithms that produce spanners that are almost as small as the best known construction of $k$-spanners and have near-quadratic running time. We reverse this way of looking at the problem. We are now trying to find algorithms that yield spanners that are exactly as small as the best known constructions for any $k = O(1)$, i.e. $O(n^{4/3})$, and at the same time we want the algorithm to run as fast as possible. All known algorithms for creating $O(1)$-spanners that have close to optimal size run in time $\Omega(n^2)$.

So a natural question is to ask if there exists a $k = O(1)$ and an algorithm that constructs a $k$-spanner with $O(n^{4/3})$ edges in $O(n^2)$ time. In fact Sommer [28] mentioned at his talk at ICALP 2016 that the main obstacle towards getting a better running time for constructing the distance oracle he presented is the lack of such an algorithm. In his case the distortion $k = O(1)$ is only factored into the running time and not the distortion of oracle. Therefore, it does not matter what $k$ is as long as it is constant.

We show that it is possible to attain this goal by giving an algorithm that constructs 8-spanners deterministically with $O(n^{4/3})$ edges in $O(n^2)$ time. Comparing this with the algorithm by Woodruff [33] this gets rid of the $\log^3 n$ factor on the number of edges and a factor of $\log^2 n$ in the running time. Furthermore, the algorithm is deterministic and not randomized Monte Carlo. The price of these improvements is that the distortion is larger than 6. We note that there are no lower bounds ruling out the possibility of a 4-spanner with $O(n^{4/3})$ edges. For the application to the distance oracle by Sommer [28], the distortion is unimportant as long as it is constant. We also show how to construct 2-spanners with $O(n^{3/2})$ edges in $O(n^2)$ time. For a comparison to previous work see Table 1.

Related work. Elkin and Peleg [19] showed that any graph on $n$ nodes has a 2-spanner with $O(n^{3/2})$ edges, Chechik [15] showed that it has a 4-spanner with $O(n^{7/5} \log^{1/5} n)$ edges, and Baswana et al [9] showed that it has a 6-spanner with $O(n^{4/3})$ edges. These results are

---

1. For instance the algorithm by Baswana et al [9] gives a 6-spanner with $O(n^{4/3})$ edges and is therefore only interesting when $m = \Omega(n^{4/3})$, in which case the running time is $\Theta(n^{2/3} m) = \Omega(n^2)$.
2. Aingworth et al [5] earlier showed the same result up to logarithmic factors on the size of the spanner.
We improve the construction time and the size to Theorem 9 is obtained using this clustering and a careful modification of the path-buying technique. Previous algorithms that construct \( k \)-spanners by Woodruff \cite{33} the number of neighbourhoods that need to be hit is so large complemented by a negative result of Abboud and Bodwin \cite{1}. A consequence of their result is that for any \( k = O(1) \) there exists a graph on \( n \) nodes such that any \( k \)-spanner of this graph has at least \( n^{4/3-o(1)} \) edges.

Another negative result comes from Erdős’s girth conjecture \cite{20}. It states that for any constant \( k \) there exists graphs with \( n \) nodes and \( \Omega(n^{1+1/k}) \) edges where the girth is \( 2k + 2 \). This conjecture has been proved for \( k = 2, 3, 5 \) \cite{31, 11}. In particular if the conjecture is true this implies that there exists graphs for which any \( (2k - 1) \)-spanner must have at least \( \Omega(n^{1+1/k}) \) edges. Woodruff \cite{32} proved that whether the conjecture is true or not, there exists a graph on \( n \) nodes such that any \( (2k - 1) \)-spanner of the graph has at least \( \Omega(k^{-1}n^{1+1/k}) \) edges.

There are also upper and lower bounds when we allow the distortion \( k \) to depend on \( n \), see \cite{14, 13, 15, 22}. In this paper, however, we are only interested in the case where \( k = O(1) \). The upper and lower bounds for \( k = O(1) \) are summarized in Table 2.

<table>
<thead>
<tr>
<th>( k )</th>
<th>Upper Bound</th>
<th>Lower Bound</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 &amp; 3</td>
<td>( O(n^{7/2}) )</td>
<td>( \Omega(n^{7/2}) )</td>
<td>\cite{19}/[31]</td>
</tr>
<tr>
<td>4 &amp; 5</td>
<td>( O(n^{7/3} \log^{1/3} n) )</td>
<td>( \Omega(n^{4/3}) )</td>
<td>\cite{15}/[11]</td>
</tr>
<tr>
<td>( \geq 6 )</td>
<td>( O(n^{4/3}) )</td>
<td>( n^{4/3-o(1)} )</td>
<td>\cite{9}/[1]</td>
</tr>
</tbody>
</table>

**Techniques.** Previous algorithms that construct \( k \)-spanners in \( O(n^2) \) time all relied on constructing a hitting set for some set of neighbourhoods. In \cite{18} this is done deterministically via a dominating set algorithm, and in \cite{33} this is done via sampling. This approach will inherently come with the cost of a \( \log n \) factor. Furthermore, in the construction of 6-spanners by Woodruff \cite{33} the number of neighbourhoods that need to be hit is so large that it seems impossible with current techniques to modify the algorithm to be Las Vegas. To avoid this we instead use a clustering approach described in Section 2. The algorithm in Theorem 9 is obtained using this clustering and a careful modification of the path-buying algorithm of \cite{9}.

**Approximate Distance Oracles and All Pairs Almost Shortest Paths.** Given an undirected an unweighted graph \( G \) an \((\alpha, \beta)\)-approximate distance oracle for \( G \) is a data structure that supports the following query. Given two nodes \( u, v \) it can compute a distance estimate \( \hat{d} \) that satisfies \( d \leq \hat{d} \leq \alpha d + \beta \) where \( d \) is the distance between \( u \) and \( v \) in \( G \). For work on approximate distance oracles see e.g. \cite{2, 3, 4, 6, 7, 8, 10, 12, 16, 17, 23, 24, 26, 27, 29, 30, 34}. Sommer \cite{28} gave a randomized Monte Carlo \((2,1)\)-distance oracle that can be constructed in \( O(n^2 \log n) \) time, has size \( O(n^{5/3} \log n) \) and can answer queries in \( O(1) \) time. We improve the construction time and the size to \( O(n^2) \) and \( O(n^{5/3}) \) respectively, and our construction is randomized Las Vegas. As a corollary we can compute an estimate \( \hat{d}(u, v) \) for all pairs of nodes in \( G \) satisfying \( d_G(u, v) \leq \hat{d}(u, v) \leq 2d_G(u, v) + 1 \) in time \( O(n^2) \). This improves upon the \( O(n^2 \log n) \) algorithm by Baswana and Kavitha \cite{8}.

**Preliminaries.** For a graph \( G \) and two nodes \( u, v \) we denote the distance from \( u \) to \( v \) in \( G \) by \( d_G(u, v) \). All graphs considered in this paper are unweighted, and unless otherwise specified they are undirected as well. For an undirected graph \( G \) and a node \( u \) the neighbourhood of \( u \) is the set of nodes adjacent to \( u \) and is denoted by \( \Gamma_G(u) \).
Overview. In Section 2 we introduce the clustering we use when constructing the spanners. In Section 3 we show how to create an 8-spanner with $O(n^{2/3})$ edges in $O(n^2)$ time and thereby prove Theorem 9. In Section 4 we provide the details on how to give an improved $(2,1)$-distance oracle.

2 Clustering

Our construction of additive spanners uses clustering techniques, and we present our clustering framework below. Let $G = (V,E)$ be a graph with $n$ vertices and $m$ edges. We let $t$ be a parameter that can depend on our needs. For a sequence $u_1, \ldots, u_\ell$ of nodes we define the clusters $C_i, i \in \{1, \ldots, \ell\}$ by

$$C_i = (\Gamma_G(u_i) \cup \{u_i\}) \setminus (C_1 \cup \ldots \cup C_{i-1}).$$

Furthermore we also define graphs $G_0, G_1, \ldots, G_\ell$ in the following way. We let $G_0 = G$, and for $i > 0$ we let $G_i$ be the subgraph of $G$ defined in the following way. The nodes of $G_i$ are the same as the nodes of $G$. An edge $(u,v)$ from $G$ is contained in $G_i$ unless both endpoints $u$ and $v$ are contained in $C_1 \cup \ldots \cup C_i$. From each node $u_i$ we let $T_i$ be a BFS tree in $G_{i-1}$ rooted at $u_i$.

Definition 1. A sequence $u_1, \ldots, u_\ell$ is called a $t$-clustering if the following requirements are satisfied.

- The node $u_i$ maximizes the size of $(\Gamma_G(u_i) \cup \{u_i\}) \setminus (C_1 \cup \ldots \cup C_{i-1}).$
- Every cluster $C_i$ contains at least $t$ nodes.
- For every node $v$ we have $|\Gamma_G(v) \cup \{v\}| \setminus (C_1 \cup \ldots \cup C_{i-1})| < t.$

We say that a node $v$ is clustered if $v \in C_1 \cup \ldots \cup C_\ell$ and unclustered otherwise. We note that since every cluster $C_i$ contains at least $t$ nodes and the clusters are disjoint we have $\ell \leq \frac{n}{t}$.

Lemma 2. Let $u_1, \ldots, u_\ell$ be a $t$-clustering of a graph $G = (V,E)$. For every $i = 1, 2, \ldots, \ell$ the number of edges in $G_{i-1}$ is at most $n|C_i|$. The number of edges in $G_\ell$ is less than $nt$.

Proof. The set of edges in $G_{i-1}$ can be written as

$$\{\{u,v\} \mid u \in V, v \in (\Gamma_G(u) \setminus (C_1 \cup \ldots \cup C_{i-1})) \}. $$

Therefore, the number of edges in $G_{i-1}$ is bounded by

$$\sum_{v \in V} |(\Gamma_G(v)) \setminus (C_1 \cup \ldots \cup C_{i-1})|. $$

(1)

Since every term in (1) is bounded by $|C_i|$, we conclude that the number of edges in $G_i$ is at most $n|C_i|$. In the same manner we see that the number of edges in $G_\ell$ is bounded by the sum $\sum_{v \in V} |(\Gamma_G(v)) \setminus (C_1 \cup \ldots \cup C_\ell)|$, which is clearly less than $nt$.

Lemma 3. Let $u_1, \ldots, u_\ell$ be a $t$-clustering of $G = (V,E)$ and let $u, v \in V$ be a pair of nodes. Assume that some shortest path from $u$ to $v$ in $G$ is not contained in $G_\ell$ from Lemma 2. Then there exists an index $i \in \{1, 2, \ldots, \ell\}$ such that

$$d_{T_i}(u_i, u) + d_{T_i}(u_i, v) \leq d_G(u, v) + 2.$$
Proof. Consider a shortest path \( p \) from \( u \) to \( v \) that is not contained in \( G_t \) and let \( w \) be a clustered node on \( p \) such that \( w \in C_i \). We choose \( w \) such that \( i \) is smallest possible. By choosing \( i \) smallest possible \( p \) is contained in \( G_{i-1} \). Furthermore since the distance from \( w \) to \( u_i \) is at most 1 we see that
\[
d_{G_{i-1}}(u, u) + d_{G_{i-1}}(u, v) \leq d_{G_{i-1}}(w, u) + d_{G_{i-1}}(w, v) + 2 = d_G(u, v) + 2.
\]
Since \( T_i \) is a is shortest path tree in \( G_{i-1} \) the conclusion follows.

Lemma 4. Given a graph \( G \) and a parameter \( t > 0 \) we can construct a \( t \)-clustering \( u_1, \ldots, u_\ell \), the corresponding BFS trees \( T_1, \ldots, T_\ell \) and \( G_t \) in \( O(n^2) \) time.

Proof. The algorithm will work by finding the nodes \( u_1, \ldots, u_\ell \) consecutively, i.e. first \( u_1 \), then \( u_2 \) and so on. The algorithm will maintain a graph \( G' \). In the beginning of the algorithm we have \( G' = G_0 \), and after we add \( u_i \) we will alter \( G' \) such that \( G' = G_i \). The total cost of altering all \( G' \) will be \( O(m) = O(n^2) \).

We find \( u_1 \) by looking at all nodes in \( G' = G_{i-1} \) and count the number of neighbours not in \( C_1 \cup \ldots \cup C_{i-1} \). Since \( G_{i-1} \) has at most \( n |C_i| \) edges this takes \( O(n |C_i|) \) time. Then the algorithm finds a BFS tree from \( u_i \) in \( G_{i-1} \) in \( O(n |C_i|) \) time. Hence the total time used by the algorithm is:
\[
O \left( m + \sum_{i=1}^{\ell} n |C_i| \right) = O(n^2).
\]

3 Constructing \( O(1) \)-Spanners

In this section we present our construction of an \( n \)-spanner with \( O(n^{4/3}) \) edges in \( O(n^2) \) time. As a warmup we show how we can use the clustering from Section 2 to give a \( n \)-spanner with \( O(n^{3/2}) \) edges in \( O(n^2) \) time.

Theorem 5. There exists an algorithm that given a graph \( G \) with \( n \) nodes constructs a \( 2 \)-spanner of \( G \) with \( \leq 2n^{3/2} \) edges in \( O(n^2) \) time.

Proof. Let \( t = \sqrt{n} \) and construct a \( t \)-clustering \( u_1, \ldots, u_\ell \) with Lemma 4. Let \( H = T_1 \cup \ldots \cup T_\ell \cup G_t \). The number of edges in \( H \) is at most \( n \ell + nt \leq 2n \sqrt{n} \) by Lemma 2 and the fact that \( \ell \leq t \).

Now we just need to prove that \( H \) is a \( 2 \)-spanner. Let \( u,v \) be arbitrary nodes and let \( p \) be a shortest path from \( u \) to \( v \) in \( G \). We wish to prove that
\[
d_H(u, v) \leq d_G(u, v) + 2.
\]

If \( p \) is contained in \( G_t \) then (2) is obviously true. Otherwise there exists an index \( i \) such that \( d_{T_i}(u, v) \leq d_G(u, v) + 2 \) by Lemma 3, and (2) is true since \( T_i \subset H \).

Next we turn to showing how to create an \( 8 \)-spanner \( H \) with \( O(n^{4/3}) \) edges in \( O(n^2) \) time. The idea is the following. We start by creating a \( t \)-clustering \( u_1, \ldots, u_\ell \) with \( t = n^{1/3} \) and \( \ell \leq n^{2/3} \). Using the BFS trees \( T_1, \ldots, T_\ell \) along with Lemma 3 we can then get an additive 2-approximation of \( d_G(u_i, u_j) \) for all pairs of indices \( i, j \), which we will call \( \delta_{i,j} \). The calculation of the BFS trees in \( O(n^2) \) time relies on an idea similar to one in [5]. The BFS trees also gives us a path from \( u_i \) to \( u_j \) that is at most 2 longer than the shortest path. If we add all these shortest paths to our spanner along with \( G_t \) and the neighbours in \( C_i \) of each
We now define \( \delta_{i,j} \) we can calculate the right hand side of (3) in \( O(\ell) \) time since we are taking the minimum over \( \ell \) different values. So in \( O(\ell^3) \) time the algorithm calculates \( \delta_{i,j} \) for all pairs of indices \( i, j \). Since \( \ell \leq n^{2/3} \) this is within the \( O(n^2) \) time bound. As a consequence of Lemma 3 we get that \( \delta_{i,j} \) is a good approximation of \( d_G(u_i, u_j) \), more precisely:

\[
d_G(u_i, u_j) \leq \delta_{i,j} \leq d_G(u_i, u_j) + 2. \tag{4}\]

We now define \( T'_k \) to be the tree obtained from \( T_k \) by contracting each edge in \( G_{\ell} \). Since an edge is contained in \( G_{\ell} \) iff at least one of its endpoints is unclustered we can construct \( T'_k \) from \( T_k \) in \( O(n) \) time. The algorithm does so for all \( i \in \{1, 2, \ldots, \ell\} \) in \( O(n \ell) = O(n^{5/3}) \) time. We note that the shortest path between two nodes \( u, v \) in \( T'_k \) contains exactly the edges on the shortest path between \( u, v \) in \( T_k \) excluding the edges that are contained in \( G_{\ell} \).

The algorithm initializes \( \Delta_{i,j} = \infty \) for all pairs of indices \( i, j \) with \( i \neq j \) and let \( \Delta_{i,i} = 0 \) for all \( i \). We will maintain that \( \Delta_{i,j} \) is an upper bound on \( d_H(u_i, u_j) \) throughout the algorithm. Now the algorithm goes through all pairs \( u_i, u_j \) and adds a path of length at most 2 longer than a shortest path between the nodes if needed. Specifically, we do the following:

Let \( L \) be an upper bound on the number of nodes of the path \( p \) from \( u_i \) to \( u_j \) in \( T'_k \) on line 6. Then Algorithm 1 can implemented in \( O(\ell^3 + \ell^2 L) \) time. Hence we just need to prove that \( L = O(\ell) \) in order to conclude that it can be implemented in \( O(\ell^3) = O(n^2) \) time. This follows from the fact that \( p \) is an almost shortest path and the following reasoning. If \( p \) contained \( > C \ell \) nodes for some sufficiently large constant \( C \) it would contain more than \( C \) nodes of the same color. Since nodes of the same color have distance at most 2 in \( G \) this would imply that there was a much shorter path from \( u \) to \( v \) in \( G \) contradicting (4) if \( C \) was chosen large enough. The details with \( C = 5 \) are given in the following lemma:

\begin{lemma}
\textbf{Lemma 6.} The path \( p \) contains no nodes of color 0, and at most 5 nodes of each color \( \neq 0 \).
\end{lemma}
Algorithm 1

For each pair of indices $i, j \in \{1, 2, \ldots, \ell\}$:
  
  For all $k \in \{1, 2, \ldots, \ell\}$:
    
    Set $\Delta_{i,j} := \min \{\Delta_{i,j}, \Delta_{i,k} + \Delta_{k,j}\}$.
    
  If $\Delta_{i,j} > \delta_{i,j} + 2$ do:
    
    Find a $k \in \{1, 2, \ldots, \ell\}$ such that $d_{T_k}(u_k, u_i) + d_{T_k}(u_k, u_j) = \delta_{i,j}$.
    
    Find the path $p$ from $u_i$ to $u_j$ in $T_k^\prime$.
    
    Add all edges from $p$ to $H$.
    
    Write $p = (w_0, w_1, w_2, \ldots, w_{s-1})$.
    
    For all $x \in \{0, 1, 2, \ldots, s - 1\}$:
      
      Set $y := d_{T_k}(u_i, w_x)$.
      
      Set $\Delta_{i,c(w_x)} := \min \{\Delta_{i,c(w_x)}, y + 1\}$
      
      Set $\Delta_{i,(w_x),j} := \min \{\Delta_{i,(w_x),j}, (\delta_{i,j} - y) + 1\}$

Proof. Obviously $p$ does not contain a node with color 0, since all its incident edges would be contained in $G_\ell$ and hence not in $T_k^\prime$. Now assume for the sake of contradiction that $p$ contains 6 nodes of some color $r \neq 0$. When traversing $p$ from $u_i$ to $u_j$ let $\alpha$ and $\beta$ be the first and the last node of color $r$ respectively. The distance from $\alpha$ to $\beta$ when following $p$ must be at least 5 by assumption. On the other hand $\alpha$ and $\beta$ have distance at most 2 in $G$. So there exists a path in $G$ from $u_i$ to $u_j$ that is at least 3 edges shorter than $p$. This contradicts (4). Hence the assumption was wrong and $p$ contains at most 5 nodes of each color $\neq 0$. ▶

Since there are $\ell$ different colors $\neq 0$ the path $p$ contains at most $5\ell$ nodes and the running time of Algorithm 1 is $O(n^2)$. So now we just need to prove that $H$ is an 8-spanner and that $H$ has at most $O(n^{k/3})$ edges. We start by proving that $H$ is an 8-spanner. Here we will utilize that the $\Delta_{i,j}$ is an upper bound on the distance from $u_i$ to $u_j$ in $H$. Furthermore, Algorithm 1 guarantees that $\Delta_{i,j} \leq \delta_{i,j} + 2$. Together with (4) this gives that

$$d_H(u_i, u_j) \leq d_G(u_i, u_j) + 4. \quad (5)$$

**Lemma 7.** The subgraph $H$ of $G$ is an additive 8-spanner of $G$.

Proof. Assume for the sake of contradiction that $H$ is not an additive 8-spanner and let $u, v$ be a pair of nodes with shortest possible distance in $G$ such that:

$$d_H(u, v) > d_G(u, v) + 8. \quad (6)$$

Say that $d_G(u, v) = D$ and let $p = (w_0, w_1, \ldots, w_D)$ be a shortest path from $u$ to $v$ in $G$ where $w_0 = u$ and $w_D = v$. Since the pair $(u, v)$ has the smallest possible distance in $G$ such that (6) holds and $d_G(w_1, v) = D - 1$ we have $d_H(w_1, v) \leq (D - 1) + 8$. In particular the edge $(u, w_1)$ is not in $H$ as it would contradict (6). Hence $u$ cannot be unclustered, as all the edges incident to an unclustered node is contained in $G_\ell$ and therefore $H$. With the same reasoning we conclude that $v$ is clustered. Let the colors of $u$ and $v$ be $i$ and $j$ respectively. The distances from $u$ and $v$ to $u_i$ and $u_j$ respectively are at most 1. Combining this insight with (5) we get:

$$d_H(u, v) \leq d_H(u_i, u_j) + 2 \leq d_G(u_i, u_j) + 6 \leq d_G(u, v) + 8.$$ 

But this contradicts the assumption (6). Hence the assumption was wrong and $H$ is an additive 8-spanner of $G$. ▶
Lastly, we need to prove that $H$ contains no more than $O(n^{4/3})$ edges. Informally, we argue the following way. Whenever the $s - 1$ edges of $p$ are added to $H$ on line 7 of Algorithm 1 there are $\Omega(s)$ different colors on $p$. For each color $r$ on $p$ we then argue that either $\Delta_{i,r}$ or $\Delta_{r,j}$ are made smaller on line 11 or 12 of Algorithm 1. Lastly, we argue that $\Delta_{i,j}$ can only be updated $O(1)$ times, and since there are $t^2 \leq n^{4/3}$ variables $\Delta_{i,j}$ this implies that Algorithm 1 only adds $O(n^{4/3})$ edges to $H$. This intuition is formalized in Lemma 8 bellow:

Lemma 8. Algorithm 1 adds no more than $25t^2$ edges to $H$.

Proof. Say that the algorithm adds the edges from the path $p = (w_0, w_1, \ldots, w_{s-1})$ on line 7 of Algorithm 1 where $w_0 = u_i$, $w_{s-1} = u_j$. First we note that since $d_G(u_i, u_j) \geq \delta_{i,j} - 2$ by (4) we have that $d_G(u_i, w_s) \geq y - 2$ for every $x \in \{0, 1, \ldots, s - 1\}$, where we consider $y$ to be a function of $x$ defined by $y = d^T(u_i, w_x)$ as on line 10. Now fix $x$ and let $r = c(w_x)$. Then there is an edge between $w_x$ and $u_r$ and therefore $d_G(u_i, u_r) \geq y - 3$, i.e. $y + 1 \leq d_G(u_i, u_r) + 4$.

So if Algorithm 1 decreases $\Delta_{i,r}$ on line 11 we have $\Delta_{i,r} \leq d_G(u_i, u_r) + 4$ after it is decreased. Since $\Delta_{i,r}$ is an upper bound on $d_H(u_i, u_r)$ and therefore also an upper bound on $d_G(u_i, u_r)$ we see that $\Delta_{i,r}$ can be decreased at most 5 times for each choice of $i, r$. By symmetry we see that we can also decrease $\Delta_{r,j}$ on line 12 at most 5 times. Since there are $t^2$ pairs of indices the algorithm can change the values of $\Delta_{i,r}$ or $\Delta_{r,j}$ on line 11 and 12 of Algorithm 1 at most $5t^2$ times.

Let $r$ be a color on $p$. After the execution of lines 9-12 we have

$$\Delta_{i,r} + \Delta_{r,j} \leq \delta_{i,j} + 2.$$ 

Due to the execution of lines 2 and 3 this was not the case before. Hence either $\Delta_{i,r}$ or $\Delta_{r,j}$ were updated. By Lemma 6 there are at least $\frac{t}{5}$ colors on $p$, so if the algorithm adds $A$ edges in total it makes at least $\frac{A}{5}$ updates of upper bounds $\Delta_{i,r}$ or $\Delta_{r,j}$. Since there can be at most $5t^2$ such updates we conclude that $\frac{A}{5} \leq 5t^2$ and that Algorithm 1 adds no more than $25t^2$ edges.

To summarize, the algorithm presented in this section runs in $O(n^2)$ time and gives an additive 8-spanner with no more than $26n^{4/3} + n = O(n^{4/3})$ edges. We have made no attempt to optimize the constant in the $O$-notation. Hence we get:

Theorem 9. There exists an algorithm that given a graph $G$ with $n$ nodes constructs an 8-spanner of $G$ with $O(n^{4/3})$ edges in $O(n^2)$ time.

4 Distance Oracles

In the following we show how to modify the construction by Sommer [28] to obtain a $(2,1)$-distance oracle of size $O(n^{5/3})$ that can be constructed in expected $O(n^2)$ time.

Let $G$ be a given graph, and $H$ an 8-spanner of $G$ constructed by Theorem 9. $H$ is constructed in $O(n^2)$ time and has $O(n^{4/3})$ edges. During the construction we use only $O(n^{5/3})$ space.

Let $u_1, u_2, \ldots, u_t$ be a $n^{1/3}$-clustering of $G$. Using Lemma 4 we obtain $T_1, \ldots, T_t$ and $G_t$ in $O(n^2)$ time. For each node $v$ we define four portals $p_1(v), p_2(v), p_3(v), p_4(v)$. We define $p_1(v) = u_i$, where $u_i$ is chosen such that the distance between $v$ and $u_i$ in $T_i$ is minimized. In case of ties we choose the node $u_i$ with the lowest index $i$. The node $p_{j+1}(v)$ for $j = 1, 2, 3$ is chosen depending on $p_j(v)$. If $p_j(v) = u_i$ we let $p_{j+1}(v) = u_i$. Otherwise $p_j(v) = u_i$ for some index $i$. We let $p_{j+1}(v) = u_{i'}$ where $u_{i'}$ is chosen among $u_1, u_2, \ldots, u_{i-1}$ such that the
distance between \( u_i \) and \( v \) in \( T_i \) is minimized. In case of ties we choose the node \( u_i \) with the lowest index \( i' \). The portals for all nodes can be found in \( O(n^{5/3}) \) time.

We will use the following lemma by Pătraşcu and Roditty [23] to construct a \((2,1)\)-distance oracle for \( G_t \), that uses space \( O(n^{5/3}) \).

**Lemma 10** ([23]). For any unweighted, undirected graph, there exists a distance oracle of size \( O(n^{5/3}) \) that, given any nodes \( u \) and \( v \) at distance \( d \), returns a distance of at most \( 2d + 1 \) in constant time. The distance oracle can be constructed in expected time \( O(mn^{2/3}) \).

In the proof in [23] they only claim a running time of \( O(mn^{2/3} + n^{7/3}) \), however, this can be fixed to give the correct running time of \( O(mn^{2/3}) \) [25]. By [23, Claim 9] it is easy to see how to get a running time of \( O(mn^{2/3} + n^2) \) which suffice for our purposes.

We are now ready to define the distance oracle. For each \( i = 1, 2, \ldots, \ell \) we store the distances \( d_T(u_i, v) \) and \( d_H(u_i, v) \) for all nodes \( v \). The distances \( d_H(u_i, v) \) can be calculated using a BFS in time \( O(\ell n^{2/3}) = O(n^2) \). For each node \( v \) we store its portals \( p_j(v), j = 1, 2, 3, 4 \). We augment this distance oracle with the Pătraşcu-Roditty distance oracle from Lemma 10 for \( G_t \).

We now show how to use the distance oracle to obtain approximate distances for a query \( u, v \). We let \( \delta_{PR}(u, v) \) be the approximate distance in \( G_t \) returned by the Pătraşcu-Roditty distance oracle. We define \( \delta_j(u, v) \) in the following way. Let \( p_j(u) = u_i \). Then \( \delta_j(u, v) = d_T(u_i, u) + \min\{d_T(u_i, v), d_H(u_i, v)\} \). The distance returned by the distance oracle is the minimum of \( \delta_{PR}(u, v) \), \( \delta_j(u, v) \) and \( \delta_j(v, u) \) for \( j = 1, 2, 3, 4 \).

We will now argue that if the the distance between \( u \) and \( v \) is \( d \), then the distance oracle returns a distance between \( d \) and \( 2d + 1 \). The distance returned is obviously at least \( d \), so we just need to show that it is at most \( 2d + 1 \). Consider a shortest path between \( u \) and \( v \) in \( G \). If there is at most one node on the shortest path which is incident to a node \( u_i \) in the clustering then the shortest path is contained in \( G_i \), and therefore:

\[
\delta_{PR}(u, v) \leq 2d_{G_i}(u, v) + 1 = 2d + 1.
\]

So assume that there exists an edge on the shortest path not in \( G_i \). Let \( i \) be the smallest index such that there is an edge \((z, t)\) on the shortest path with \( z, t \in C_i \cup \ldots \cup C_{i'} \). Say that \( z \) is closer to \( u \) than to \( v \) in \( G \). Assume that \( z \in C_i \) and \( t \in C_{i'} \) for some index \( i' \leq i \) (the case where \( z \in C_i \) and \( t \in C_{i} \) is handled symmetrically). Since the shortest path is contained in \( G_{i-1} \) and \( G_{i'-1} \) we have that

\[
d_T(u_i, u) + d_{T_{i'}}(u_{i'}, v) \leq (d_G(u, z) + 1) + (d_G(t, v) + 1) = d + 1,
\]

and therefore:

\[
\min\{d_T(u_i, u), d_{T_{i'}}(u_{i'}, v)\} \leq \frac{d + 1}{2}.
\]

Assume that \( d_T(u_i, u) \leq \frac{d + 1}{2} \). The other case is handled similarly. Say that \( p_j(u) = u_{k_j} \) for \( j = 1, 2, 3, 4 \). First assume that \( k_j > i \) for all \( j = 1, 2, 3, 4 \). Then we conclude that \( d_{T_{k_j}}(p_1(u), u) \leq d_T(u_i, u) - 4 \). The distance returned by the distance oracle is at most

\[
\delta_1(u, v) \leq d_{T_{k_j}}(p_1(u), u) + d_H(p_1(u), v)
\]

\[
\leq d_{T_{k_j}}(p_1(u), u) + d_G(p_1(u), v) + 8
\]

\[
\leq 2d_{T_{k_j}}(p_1(u), u) + d_G(u, v) + 8
\]

\[
\leq 2(d_T(u_i, v) - 4) + d + 8 \leq 2d + 1.
\]
Additive Spanners and Distance Oracles in Quadratic Time

Now assume that $k_j \leq i$ for some $j \in \{1, 2, 3, 4\}$ and let $j$ be the smallest index such that $k_j \leq i$. By definition we have that $d_{G_{k_j}}(p_j(u), u) \leq d_T(u_i, u)$. Furthermore the shortest path is contained in $G_{k_{j-1}}$ and therefore $d_{G_{k_j}}(p_j(u), v) \leq d_{G_{k_j}}(p_j(u), u) + d_G(u, v)$. The distance returned is at most

$$\delta_j(u, v) \leq d_{G_{k_j}}(p_j(u), u) + d_{G_{k_j}}(p_j(u), v) \leq 2d_{G_{k_j}}(p_j(u), u) + d \leq 2d + 1.$$

We conclude that the distance returned by the distance oracle is always between $d$ and $2d + 1$. The result is summarized in Theorem 11.

**Theorem 11.** For any unweighted, undirected graph, there exists a distance oracle of size $O(n^{5/3})$ that, given any nodes $u$ and $v$ at distance $d$, returns a distance of at most $2d + 1$ in constant time. The distance oracle can be constructed in expected time $O(n^2)$.

**Acknowledgements.** The author would like to thank Christian Sommer for helpful discussions on the application of the 8-spanner to his construction of distance oracles.

References


Liam Roditty. personal communication.


Additive Spanners and Distance Oracles in Quadratic Time


