I/O-Efficiently Pruning Dense Spanners

Joachim Gudmundsson[1] and Jan Vahrenhold[2]

[1] Department of Mathematics and Computing Science, TU Eindhoven, 5600 MB, Eindhoven, The Netherlands.
h.j.gudmundsson@tue.nl

[2] Westfälische Wilhelms-Universität Münster, Institut für Informatik, 48149 Münster, Germany.
jan@math.uni-muenster.de

Abstract. Given a geometric graph $G = (S, E)$ in $\mathbb{R}^d$ with constant dilation t, and a positive constant ε, we show how to construct a $(1+\varepsilon)$-spanner of G with $\mathcal{O}(|S|)$ edges using $\mathcal{O}(\mathrm{sort}(|E|))$ I/Os.

1 Introduction

Complete graphs represent ideal communication networks but they are expensive to build; sparse spanners represent low cost alternatives. Spanners for complete Euclidean graphs as well as for arbitrary weighted graphs find applications in robotics, network topology design, distributed systems, design of parallel machines, and many other areas. Consider a set S of n points in the Euclidean space $\mathbb{R}^d$. Throughout this paper, we will assume that d is constant. A Euclidean network on S can be modeled as an undirected graph G with vertex set S and with edges $e = (u, v)$ of weight $|uv|$. If G is a Euclidean graph, then $\delta_G(p, q)$ denotes the Euclidean length of a shortest path in G between p and q. Hence, G is a *t-spanner* for S if $\delta_G(p, q) \le t|pq|$ for any two points p and q of S. The minimum value t such that G is a t-spanner for S is called the *dilation* of G. A subgraph G' of G is a t'-spanner of G, if $\delta_{G'}(p, q) \le t' \cdot \delta_G(p, q)$ for any two points p and q of S.

Many algorithms are known that compute t-spanners with $\mathcal{O}(|S|)$ edges that have additional properties such as bounded degree, small spanner diameter, low weight, and fault-tolerance; see the survey [4].

For the analysis in this paper we use the standard two-level I/O model [1] which defines the following parameters:

$$N = \#\text{ of objects in the problem instance,}$$
$$M = \#\text{ of objects fitting in internal memory,}$$
$$B = \#\text{ of objects per disk block,}$$

where $N \gg M$ and $1 \le B \le M/2$. An *input/output operation* (or simply *I/O*) consists of reading a block of contiguous elements from disk into internal memory or writing a block from internal memory to disk. Computations can only be performed on objects in internal memory. This model of computation captures the characteristics of working with massive data sets that are too large to fit into main memory and thus are stored on disk. In the two-level I/O model, we measure the efficiency of an algorithm by the number of I/Os it performs, the amount of disk space it uses (in units of disk blocks), and the internal memory computation time. Aggarwal and Vitter [1] developed matching upper and lower I/O bounds for sorting and permuting: sorting N items in external memory requires $\Theta(\frac{N}{B} \log_{M/B} \frac{N}{B})$ I/Os while scanning N items in external memory obviously can be done in $\Theta(\frac{N}{B})$ I/Os. The upper bounds for sorting and for scanning N items are often abbreviated as $\mathcal{O}(\mathrm{sort}(N)) = \mathcal{O}(\frac{N}{B} \log_{M/B} \frac{N}{B})$ and as $\mathcal{O}(\mathrm{scan}(N)) = \mathcal{O}(\frac{N}{B})$. I/O-efficient algorithms have been developed for several problem domains, including computational geometry, graph theory, and string processing. Recent surveys can be found in [2, 7].

In this paper we consider the problem of I/O-efficiently *pruning* a given t-spanner, even if it has a super-linear number of edges. Given a geometric graph $G = (S, E)$ in $\mathbb{R}^d$ with constant dilation t, and a positive constant ε, we show how to I/O-efficiently construct a $(1+\varepsilon)$-spanner of G with only $\mathcal{O}(|S|)$ edges using $\mathcal{O}(\mathrm{sort}(|E|))$ I/Os. This bound matches the (internal memory) complexity of the algorithm in [6].

While building a sparse spanner is asymptotically faster than pruning a dense spanner, the latter technique allows to specifically designate edges that should participate and edges that are not allowed in the sparse spanner to be constructed.

2 Pruning Dense Spanners

We are now ready to sketch our algorithm for I/O-efficiently pruning a dense t-spanner $G = (S, E)$. Our algorithm is similar to the internal memory algorithm by Gudmundsson *et al.* [6]. They showed that one can use the well-separated pair decomposition (WSPD) by Callahan and Kosaraju [3].

First we present three lemmas that demonstrate that a tree can be labeled I/O-efficiently in a hierarchical manner.

Lemma 1. *Given a tree T with N nodes, we can label all leaves in left-to-right order in $\mathcal{O}(\mathrm{sort}(N))$ I/Os.*

Lemma 2. *Given a tree T with N nodes whose leaves are labeled in left-to-right order, we can, in $\mathcal{O}(\mathrm{sort}(N))$ I/Os, label each internal node v with an interval $[l_v, r_v]$, $l_v, r_v \in \mathbb{N}$, such that the following holds:*

1. *Each leaf in the subtree rooted at v is labeled with some integer $\ell(v) \in [l_v, r_v]$.*
2. *There exists at least one leaf in the subtree rooted at v that is labeled with an integer $\ell(v) \in [l_v, r_v]$.*
3. *The interval $[l_v, r_v]$ is the minimal interval having this property.*

Lemma 3. *Given a unique relabeling of the vertices of a geometric graph $G = (S, E)$, we can relabel the edges in E such that each edge $e = (v, w) \in E$ is labeled $(\ell(v), \ell(w))$ where $\ell(v), \ell(w) \in [1 \dots |S|]$ are the unique labels assigned to v and w. Given the set E of edges and a tree storing the labeled vertices in its leaves, we can relabel all edges in $\mathcal{O}(\mathrm{sort}(|E|))$ I/Os.*

Now, assume that we are given a t-spanner $G = (S, E)$. Compute a WSPD $\{A_i, B_i\}$, $1 \le i \le m$, for S, with separation ratio $s = 4(1 + (1 + \varepsilon)t)/\varepsilon$ and $m = \mathcal{O}(|S|)$. Let $G' = (S, E')$ be the graph that contains for each i, exactly one (arbitrary) edge (x_i, y_i) of E with $x_i \in A_i$ and $y_i \in B_i$, provided such an edge exists. It holds that G' is a $(1 + \varepsilon)$-spanner of G with m edges [6]. Our algorithm first computes a well-separated pair decomposition $\{A_i, B_i\}$ with separation ratio $s = 4(1+(1+\varepsilon)t)/\varepsilon$, using the algorithm of Govindarajan *et al.* [5] and spending an overall number of $\mathcal{O}(\mathrm{sort}(|S|))$ I/Os. The well-separated pair decomposition is represented by a split tree having $\mathcal{O}(|S|)$ leaves which is laid out on disk in $\mathcal{O}(|S|/B)$ disk blocks. We then use Lemma 1 to label all *vertices* stored in the leaves from left to right and to label each leaf v with the minimal interval containing the labels of the points stored with v, that is we assign to each vertex v of the graph an unique integer $\ell(v) \in [1 \dots |S|]$. Finally, we perform a labeling of the internal nodes that fulfills the requirements of Lemma 2. By Lemma 1 and Lemma 2 the complexity computing this labeling is $\mathcal{O}(\mathrm{sort}(|S|))$.

The algorithm of Gudmundsson *et al.* [6] prunes a dense spanner by only keeping one edge connecting the two components of each well-separated pair $\{A_i, B_i\}$ considered. We can restate this pruning processes as a special case of the range-reporting problem. We first identify each edge $e = (v, w)$ in the original spanner with a point $p_e := (\ell(v), \ell(w)) \in [1 \dots |S|]^2$ (see Lemma 3).

Lemma 4. *Let T be a split tree for $G = (S, E)$ whose nodes have been labeled with intervals according to Lemma 2 and let a and b two nodes of T that correspond to a well-separated pair $\{A_i, B_i\}$. An edge $e = (v, w) \in E$ connects two vertices $v \in A_j$ and $w \in B_i$ if and only if $\ell(v) \in [l_a, r_a]$ and $\ell(w) \in [l_b, r_b]$.*

Let the set $\mathcal{E}$ be defined as $\mathcal{E} := \{(\ell(v), \ell(w)) \in [1 \dots |S|]^2 \mid (v, w) \in E\}$. The above lemma allows us to perform the pruning algorithm for each well-separated pair $\{A_i, B_i\}$ corresponding to two nodes a and b in the split tree by performing an orthogonal range reporting query with query range $[l_a, r_a] \times [l_b, r_b]$ on the set $\mathcal{E}$ while reporting exactly one point. Except for the edge corresponding to the point reported, all edges connecting points in A_i and B_i can be pruned, and this implies that the pruned spanner consists exactly of all edges corresponding to the results of all range queries.

What remains to show is that all range reporting queries can be performed I/O-efficiently. First of all, note that constructing the set $\mathcal{E}$ from the set E of edges can be done in $\mathcal{O}(\mathrm{sort}(|E|))$ I/Os, see Lemma 3. In a similar way, we can construct the query ranges $[l_a, r_a] \times [l_b, r_b]$ for all pairs $\{A_i, B_i\}$ in the well-separated pair decomposition: We extract the labels of all nodes in the split tree and use two successive sort-merge steps to generate the set $\mathcal{Q}$ of $\mathcal{O}(|S|)$ query ranges in $\mathcal{O}(\mathrm{sort}(|S|))$ I/Os.

Lemma 5. *Given a set $\mathcal{Q}$ of orthogonal range queries on a set $\mathcal{E}$ of points in the plane where $|\mathcal{Q}| \in \mathcal{O}(|\mathcal{E}|)$, we can process all queries in $\mathcal{O}(\mathrm{sort}(|\mathcal{E}|))$ I/Os while at the same time reporting no more than one answer per query.*

We use the above result to process a dataset of size $\mathcal{O}(|E|)$ and a query set of size $\mathcal{O}(|S|)$, and thus we obtain an answer set of size $\mathcal{O}(|S|)$ spending no more than $\mathcal{O}(\mathrm{sort}(|E|))$ I/Os.

Theorem 1. *Given geometric graph $G = (S, E)$ which is a t-spanner for S for some constant $t > 1$ and given a constant $\varepsilon > 0$, we can compute a $(1+\varepsilon)$-spanner $G' = (S, E')$ of G with $E' \subseteq E$ and $|E'| \in \mathcal{O}(|S|)$ spending $\mathcal{O}(\mathrm{sort}(|E|))$ I/Os.*

References

1. A. Aggarwal and J. S. Vitter. The input/output complexity of sorting and related problems. *Communications of the ACM*, 31(9):1116–1127, Sept. 1988.
2. L. A. Arge. External memory data structures. In J. Abello, P. M. Pardalos, and M. G. C. Resende, editors, *Handbook of Massive Data Sets*. Kluwer, 2002. 313-357.
3. P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM*, 42:67–90, 1995.
4. D. Eppstein. Spanning trees and spanners. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 425–461. Elsevier Science Publishers, Amsterdam, 2000.
5. S. Govindarajan, T. Lukovszki, A. Maheswari, and N. Zeh. I/O-efficient well-separated pair decomposition and its application. In *Proc. 8th European Symposium on Algorithms*, volume 1879 of *Lecture Notes in Computer Science*, pages 220–231. Springer-Verlag, 2000.
6. J. Gudmundsson, C. Levcopoulos, G. Narasimhan, and M. Smid. Approximate distance oracles for geometric graph. In *Proc. 13th ACM-SIAM Symposium on Discrete Algorithms*, pages 828–837, 2002.
7. J. S. Vitter. External memory algorithms and data structures: Dealing with massive data. *ACM Computing Surveys*, 33(2):209–271, June 2001.