Local Routing in a Tree Metric 1-Spanner

Milutin Brankovic[1], **Joachim Gudmundsson**[2], **and André van Renssen**[3]

1 **University of Sydney, Australia**
`milutin.brankovic3@gmail.com`
2 **University of Sydney, Australia**
`joachim.gudmundsson@sydney.edu.au`
3 **University of Sydney, Australia**
`andre.vanrenssen@sydney.edu.au`

Abstract

Solomon and Elkin [5] constructed a shortcutting scheme for weighted trees which results in a 1-spanner for the tree metric induced by the input tree. The spanner has logarithmic lightness, logarithmic diameter, a linear number of edges and bounded degree (provided the input tree has bounded degree). This spanner has been applied in a series of papers devoted to designing bounded degree, low-diameter, low-weight $(1+\epsilon)$-spanners in Euclidean and doubling metrics. In this paper, we present a simple local routing algorithm for this tree metric spanner. The algorithm has a routing ratio of 1, is guaranteed to terminate after $O(\log n)$ hops and requires $O(\Delta \log n)$ bits of storage per vertex where Δ is the maximum degree of the tree on which the spanner is constructed.

1 Introduction

Let T be a weighted tree. The tree metric induced by T, denoted M_T, is the complete graph on the vertices of T where the weight of each edge (u, v) is the weight of the path connecting u and v in T. For $t \geq 1$, a t-spanner for a set of points V with a distance function d is a subgraph H of the complete graph on V such that every pair of distinct points $u, v \in V$ is connected by a path in H of total weight at most $t \cdot d(u, v)$. We refer to such paths as t-spanner paths. A t-spanner has diameter Λ if every pair of points is connected by a t-spanner path consisting of at most Λ edges. Typically, t-spanners are designed to be sparse, often with a linear number of edges. The lightness of a graph is the ratio of its weight to the weight of its minimum spanning tree. Solomon and Elkin [5] define a 1-spanner for tree metrics. Given an n vertex weighted tree of maximum degree Δ, the 1-spanner has $O(n)$ edges, $O(\log n)$ diameter, $O(\log n)$ lightness and maximum degree bounded by $\Delta + k$ (k is an adjustable parameter considered to be a constant for our purposes). While being an interesting construction in its own right, this tree metric 1-spanner has been used in a series of papers as a tool for reducing the diameter of various Euclidean and doubling metric spanner constructions [1, 2, 4, 5].

A local routing algorithm for a weighted graph G is a method by which a message can be sent from any vertex in G to a given destination vertex. The successor to each vertex u on the path traversed by the routing algorithm must be determined using only knowledge of the destination vertex, the neighbourhood of u and possibly some extra information stored at u. In some settings, the routing algorithm may modify the message header to provide extra information for future routing decisions. However, the routing algorithm presented in this paper does not require a modifiable header. Given two vertices u, v, we define $d_{route}(u, v)$ to be the length of the routing path traversed when routing from u to v. The routing ratio of the routing algorithm is defined to be $\max_{u,v \in V(G)} \frac{d_{route}(u,v)}{d_G(u,v)}$ where $d_G(u, v)$ denotes the

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

length of the shortest path from u to v in G. We define the diameter of a local routing algorithm to be an upper bound on the number of edges traversed when routing between any two vertices. In this paper, we present a simple local routing algorithm for the tree metric spanner from [5] with routing ratio 1 and diameter $O(\log n)$.

2 Shortcutting Scheme

In this section we describe the tree metric 1-spanner for which our routing algorithm is defined. The spanner is due to Solomon and Elkin [5]. For brevity, we only give a high level overview of the construction. Full details can be found in [5] and will also appear in the full version of this paper.

Given a rooted weighted tree T on n vertices, an integer k, $k \geq 1$, is chosen. While the construction is defined for any k in the range $1 \leq k \leq n-2$, we choose $k = O(1)$. Next, at most $k+1$ vertices are selected from $V(T)$. Let us denote this set by C_T. The method by which these vertices are selected is deterministic. Denote by $T \setminus C_T$ the forest resulting from the removal of the vertices C_T (and their incident edges) from T. Next, the procedure adds the edges of the complete graph on C_T to the spanner. If the forest $T \setminus C_T$ is non-empty, the procedure is recursively applied to each tree in $T \setminus C_T$. We define *canonical subtrees* to be the trees on which the recursive procedure is called during the course of the construction of the spanner. For a canonical subtree T', we say $C_{T'}$ is a set of cut vertices. Note that for every vertex v, there is a canonical subtree T^v for which $v \in C_{T^v}$. We say T^v is the canonical subtree of v.

We note that the spanner defined by Solomon and Elkin [5] actually differs slightly from what is presented here in that rather than including the edges of the complete graph on sets of cut vertices, a certain spanner with $O(k)$ edges and $O(\alpha(k))$ diameter (α denotes the inverse Ackermann function) is used instead. However, the spanner resulting from the use of the complete graph has higher weight and degree only by a factor of k while being far easier to work with for the purpose of routing.

Let G denote the graph resulting from the construction described above. The following is established by Solomon and Elkin [5] for the version of the spanner defined in the original paper. It is easy to see the properties also hold for the version of the spanner described here.

▶ **Theorem 1.** *The graph G satisfies the following:*

1. *G is a $O(\log n)$ diameter 1-spanner for M_T.*
2. *$wt(G) = O(\log n) \cdot wt(T)$.*
3. *For any canonical subtree T', each tree in $T' \setminus C_{T'}$ has at most $2 \cdot |T'|/k$ vertices.*
4. *The maximum degree of G is at most $O(1) + \Delta$ where Δ is the maximum degree of T.*

Note that property 3 implies that the recursion depth of the spanner construction algorithm is $O(\log n)$.

3 Routing Algorithm

In this section, we describe a local routing algorithm for the spanner described above. The routing algorithm presented in this section requires that the vertices of the spanner store certain information which we specify below. We make use of the labelling scheme of Santoro and Khatib [3].

Let $rank(v)$ denote the rank of v in a post-order traversal of T. We define

$$L(v) := \min\{rank(w) : w \text{ is a descendant of } v\}.$$

Let N_v be the set of neighbours of v in G. Each vertex v of G stores the following information:

1. $rank(v)$ and $L(v)$.
2. The depth of v, i.e, the hop distance of v from the root of T.
3. $rank(w)$ and $L(w)$ for each $w \in N_v$.
4. The depth of w for each $w \in N_v$.

▶ **Lemma 2.** *In the labelling scheme outlined above, each vertex of G stores $O((\Delta+k)\log n)$ bits of information.*

We note that this labelling scheme enables us to determine if a given vertex is an ancestor or descendant of another. Indeed, a vertex u is an ancestor of a vertex v if and only if $L(u) \leq rank(v) \leq rank(u)$ and u is a descendant of v if and only if $L(v) \leq rank(u) \leq rank(v)$.

This test is also used in the tree routing algorithm of Santoro and Khatib [3]. In our routing algorithm, we use this test to determine the neighbours of the current vertex in the tree spanner which are actually on the original path to the destination. We must limit our routing steps to these vertices to ensure a routing ratio of 1. When then use additional criteria to make the best choice from the feasible routing steps to ensure the diameter of the routing algorithm is $O(\log n)$.

Given a current vertex u and a destination vertex v, the algorithm executes the routing steps of one of the cases defined below. We assume that at each stage of the algorithm, the integers $rank(v)$ and $L(v)$ are known.

For convenience of analysis, in each case we specify two routing steps. For ease of exposition, we consider a vertex u to be both a descendant and ancestor of itself.

Case 0: u and v are joined by an edge. Route to v.

Case 1: u is an ancestor of v in T. Let X be the set of vertices in C_{T^u} which are ancestors of v. Let x be the deepest element of X. Route first to x and then to the child of x which is an ancestor of v.

Case 2: u is a descendant of v in T. Let X be the set of vertices in C_{T^u} which are descendants of v and ancestors of u. Let x be the highest vertex in X. Route first to x and then to its parent.

Case 3: u is not an ancestor or descendant of v. Let X be the set of vertices in C_{T^u} which are ancestors of v and not ancestors of u. If $X \neq \emptyset$, we define x to be the deepest vertex in X and define x' to be the child of x which is an ancestor of v. Let Y be the set of vertices in C_{T^u} which are ancestors of u but not ancestors of v. We define y to be the highest vertex in Y.

Case 3 a): X is empty. Route first to y and then to the parent of y.

Case 3 b): X is non-empty. Route first to x and then to x'.

▶ **Theorem 3.** *Let u and v be vertices of G. Let $\delta_T(u,v)$ denote the length of the path from u to v in T. The routing algorithm described above is guaranteed to terminate after a finite number of steps and the length of the path traversed is exactly $\delta_T(u,v)$.*

Next we show that routing paths consist of $O(\log n)$ edges. In order to do this, we must define *canonical sequences.* First, we assign an integer sequence $S_{T'}$ to each canonical subtree T'. These sequences are defined inductively as follows. The original tree T is assigned the empty sequence. Let T' be a canonical subtree and suppose T' has already been assigned the sequence $S_{T'}$. Each canonical subtree $T_j \in T' \setminus C_{T'} = \{T_1, ..., T_p\}$ is assigned the sequence obtained by appending j to $S_{T'}$. Given a vertex v of G, we define its canonical sequence to be $S_v = S_{T^v}$. Note that if for two vertices u and v, S_u is a prefix of S_v, then T^u contains T^v. Note also that $S_u = S_v$ if and only if $T^u = T^v$ and so u and v are joined by an edge if $S_u = S_v$, by definition of the spanner.

▶ **Lemma 4.** *Let u and v be vertices of T such that u is either an ancestor or a descendant of v. Let u' be the vertex reached after executing the routing steps of either Case 1 or Case 2 when routing from u to v. Then the following statements hold:*

1. *If S_u is a prefix of S_v, then $|S_{u'}| > |S_u|$. Moreover, either $S_{u'} = S_v$ or S'_u is a prefix of S_v.*
2. *If S_v is a prefix of S_u, then $|S_{u'}| < |S_u|$. Moreover, either $S_{u'} = S_v$ or S_v is a prefix of $S_{u'}$.*
3. *Suppose S_u and S_v share a common prefix S of length $m < \min\{|S_u|, |S_v|\}$. Then $|S_{u'}| < |S_u|$. Moreover, either $S_{u'} = S$ or S is a prefix of $S_{u'}$*

Since the spanner construction algorithm has logarithmic depth, we see that the length of a canonical sequence is at most $O(\log n)$. Using this observation and Lemma 4, it is not difficult to show the following.

▶ **Lemma 5.** *Suppose u and v in G are such that u is an ancestor or descendant of v in T. Then, when routing from u to v, the routing algorithm reaches v after traversing $O(\log n)$ edges.*

Consider the case where u is neither an ancestor nor a descendant of v. The following lemma shows that in this case, the algorithm either routes to a vertex on the path from $lca(u,v)$ to v, where $lca(u,v)$ is the lowest common ancestor of u and v, or it executes the routing steps that would be executed if the algorithm were routing from u to $lca(u,v)$.

▶ **Lemma 6.** *Let u and v be vertices of G such that $lca(u,v) \notin \{u,v\}$. Suppose that the set X as defined in Case 3 is empty so that the algorithm executes the routing steps of Case 3 a) when routing from u to v. Then the algorithm performs the routing steps which would be performed if the destination was $lca(u,v)$ rather than v.*

Lemmas 5 and 6 imply the following:

▶ **Theorem 7.** *Let u and v be vertices in G. The routing algorithm reaches v when routing from u after traversing at most $O(\log n)$ edges.*

4 Concluding Remarks

We have demonstrated that a slightly modified version of the tree metric 1-spanner of Solomon and Elkin [5] supports a $O(\log n)$ diameter local routing algorithm with routing ratio 1. The tree metric spanner has been used in the literature as a tool to reduce the diameter of various spanner constructions while either preserving or incurring minimal penalties in other desirable properties of the spanner such as number of edges, degree, diameter and weight. We leave it as future work to use this local routing algorithm as a basis for local routing algorithms on some of the aforementioned Euclidean and doubling metric spanners.

References

1 S. Arya, G. Das, D. M. Mount, J. S. Salowe, and M. Smid. Euclidean spanners: Short, thin, and lanky. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, STOC '95, pages 489–498, 1995.

2 T. Chan, M. Li, L. Ning, and S. Solomon. New doubling spanners: Better and simpler. *SIAM Journal on Computing*, 44(1):37–53, 2015.

3 N. Santoro and R. Khatib. Labelling and implicit routing in networks. *The Computer Journal*, 28(1):5–8, 1985.

4 S. Solomon. From hierarchical partitions to hierarchical covers: Optimal fault-tolerant spanners for doubling metrics. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, page 363–372, 2014.

5 S. Solomon and M. Elkin. Balancing degree, diameter, and weight in Euclidean spanners. *SIAM Journal on Discrete Mathematics*, 28(3):1173–1198, 2014.