Collective Tree Spanners and Routing in AT-free Related Graphs

Feodor F. Dragan[1], Chenyu Yan[1], and Derek G. Corneil[2]

[1] Department of Computer Science, Kent State University, Kent, Ohio, USA
{dragan,cyan}@cs.kent.edu
[2] Department of Computer Science, University of Toronto, Toronto, Ontario, Canada
dgc@cs.toronto.edu

Abstract. In this paper we study collective additive tree spanners for families of graphs that either contain or are contained in AT-free graphs. We say that a graph $G = (V, E)$ *admits a system of μ collective additive tree r-spanners* if there is a system $\mathcal{T}(G)$ of at most μ spanning trees of G such that for any two vertices x, y of G a spanning tree $T \in \mathcal{T}(G)$ exists such that $d_T(x, y) \leq d_G(x, y) + r$. Among other results, we show that AT-free graphs have a system of two collective additive tree 2-spanners (whereas there are trapezoid graphs that do not admit any additive tree 2-spanner). Furthermore, based on this collection of trees, we derive a compact and efficient routing scheme for those graphs. Also, any DSP-graph (there exists a dominating shortest path) admits one additive tree 4-spanner, a system of two collective additive tree 3-spanners and a system of five collective additive tree 2-spanners.

1 Introduction

Given a graph $G = (V, E)$, a spanning subgraph H is called a *spanner* if H provides a "good" approximation of the distances in G. More formally, for $t \geq 1$, H is called a *multiplicative t–spanner* of G [1, 14, 13] if $d_H(u, v) \leq t \cdot d_G(u, v)$ for all $u, v \in V$. If $r \geq 0$ and $d_H(u, v) \leq d_G(u, v) + r$ for all $u, v \in V$, then H is called an *additive r–spanner* of G [8]. The parameters t and r are called, respectively, the *multiplicative* and the *additive stretch factors*. Clearly, every additive r-spanner of G is a multiplicative $(r + 1)$-spanner of G (but not vice versa). In this paper, we continue the approach taken in [4] of studying *collective tree spanners*. We say that a graph $G = (V, E)$ *admits a system of μ collective additive tree r-spanners* if there is a system $\mathcal{T}(G)$ of at most μ spanning trees of G such that for any two vertices x, y of G a spanning tree $T \in \mathcal{T}(G)$ exists such that $d_T(x, y) \leq d_G(x, y) + r$ (a multiplicative variant of this notion can be defined analogously). Clearly, if G admits a system of μ collective additive tree r-spanners, then G admits an additive r-spanner with at most $\mu \times (n - 1)$ edges (take the union of all those trees), and if $\mu = 1$ then G admits an additive tree r-spanner. Note also that any graph on n vertices admits a system of at most $n - 1$ collective additive tree 0-spanners (take $n - 1$ Breadth-First-Search–trees rooted at different vertices of G). In particular, we examine the problem of finding small

J. Hromkovič, M. Nagl, and B. Westfechtel (Eds.): WG 2004, LNCS 3353, pp. 68–80, 2004.

systems of collective additive tree r-spanners for small values of r on classes of graphs that are related to the well known *asteroidal triple-free (AT-free) graphs*, notably the restricted families: permutation graph and trapezoid graphs and the generalizations: DSP-graphs and graphs with bounded asteroidal number.

Once one has determined a system of collective additive tree spanners, it is interesting to see how such a system can be used to design compact and efficient routing schemes for the given graph. Following [12], one can give the following formal definition. A family $\Re$ of graphs is said to have an $l(n)$-bit *routing labeling scheme* if there is a function L labeling the vertices of each n-vertex graph in $\Re$ with distinct labels of up to $l(n)$ bits, and there exists an efficient algorithm, called the *routing decision*, that given the label of a source vertex v and the label of the destination vertex (the header of the packet), decides in time polynomial in the length of the given labels and using only those two labels, whether this packet has already reached its destination, and if not, to which neighbor of v to forward the packet. The quality of a routing scheme is measured in terms of its *additive stretch*, called *deviation*, (or *multiplicative stretch*, called *delay*), namely, the maximum surplus (or ratio) between the length of a route, produced by the scheme for some pair of vertices, and their distance.

1.1 Our Results

After introducing the notation and definitions used throughout the paper, we examine various families of graphs related to AT-free graphs from the perspective of determining whether they have a small constant number of collective additive tree r-spanners for small constant r. In Section 2 we show that AT-free graphs have a system of two collective additive tree 2-spanners, permutation graphs have a single additive tree 2-spanner but there are trapezoid graphs that do not admit any additive tree 2-spanner (thereby disproving a conjecture of [15]). All of these tree spanners can be easily constructed in linear time. For families that strictly contain AT-free graphs, we prove that any DSP-graph admits one additive tree 4-spanner, a system of two collective additive tree 3-spanners and a system of five collectible additive tree 2-spanners. Furthermore, any graph G with asteroidal number $\mathsf{an}(G)$ admits a system of $\mathsf{an}(G)(\mathsf{an}(G)-1)/2$ collective additive tree 4-spanners and a system of $\mathsf{an}(G)(\mathsf{an}(G)-1)$ collective additive tree 3-spanners. In Section 3, we show how the system of two collective additive tree 2-spanners for AT-free graphs can be used to derive a compact and efficient routing scheme. In particular we will show that any AT-free graph with diameter $\mathcal{D}$ and maximum vertex degree Δ admits a $(3\log_2 \mathcal{D}+6\log_2 \Delta+O(1))$-bit routing labeling scheme of deviation at most 2. Moreover, the scheme is computable in linear time, and the routing decision is made in constant time per vertex.

1.2 Basic Notions and Notation

All graphs occurring in this paper are connected, finite, undirected, loopless and without multiple edges. In a graph $G = (V, E)$ the *length* of a path from a vertex v to a vertex u is the number of edges in the path. The *distance* $d_G(u, v)$ between the vertices u and v is the length of a shortest path connecting u and

v. The *eccentricity* $ecc(v)$ of a vertex v of G is $\max_{u \in V} d_G(u, v)$. The diameter $diam(G)$ of G is $\max_{v \in V} ecc(v)$. The *ith neighborhood* of a vertex v of G is the set $N_i(v) := \{u \in V : d_G(v, u) = i\}$. For a vertex v of G, the sets $N(v) := N_1(v)$ and $N[v] := N(v) \cup \{v\}$ are called the *open neighborhood* and the *closed neighborhood* of v, respectively. For a set $S \subseteq V$, by $N[S] := \bigcup_{v \in S} N[v]$ we denote the *closed neighborhood* of S and by $N(S) := N[S] \setminus S$ the *open neighborhood* of S. A set $D \subseteq V$ is called a *dominating set* of a graph $G = (V, E)$ if $N[D] = V$.

An independent set of three vertices such that each pair is joined by a path that avoids the neighborhood of the third is called an *asteroidal triple*. A graph G is an *AT-free graph* if it does not contain any asteroidal triples [2]. In [7], the notion of asteroidal triple was generalized. An independent set $A \subseteq V$ of a graph $G = (V, E)$ is called an *asteroidal set* of G if for each $a \in A$ the vertices of $A \setminus \{a\}$ are contained in one connected component of $G - N[a]$, the graph obtained from G by removing vertices of $N[a]$. The maximum cardinality of an asteroidal set of G is denoted by $\mathsf{an}(G)$, and called the *asteroidal number* of G. The class of *graphs of bounded asteroidal number* extends naturally the class of AT-free graphs; AT-free graphs are exactly the graphs with asteroidal number at most two.

Let P be a shortest path of G. If every vertex z of G belongs to the neighborhood $N[P]$ of P, then we say that P is a *dominating shortest path* of G. A graph G is called a *Dominating-Shortest-Path–graph* (or *DSP–graph*, for short), if it has a dominating shortest path. By the Dominating Pair Theorem given in [2], any AT-free graph is a DSP-graph.

The class of AT-free graphs contains many intersection families of graphs, among them the permutation graphs, the trapezoid graphs and the cocomparability graphs. These three families of graphs can be defined as follows. Consider two parallel lines (upper and lower) in the plane. Assume that each line contains n points, labeled 1 to n, and each two points with the same label define a segment with that label. The intersection graph of such a set of segments between two parallel lines is called a *permutation graph*. Assume now that each line contains n intervals, labeled 1 to n, and each two intervals with the same label define a trapezoid with that label (a trapezoid can degenerate to a triangle or to a segment). The intersection graph of such a set of trapezoids between two parallel lines is called a *trapezoid graph*. Clearly, every permutation graph is a trapezoid graph, but not vice versa. The class of cocomparability graphs (which contains all trapezoid graphs as a subclass) can be defined as the intersection graphs of continuous function diagrams, but for this paper it would be more convenient to define them via the existence of a special vertex ordering. A graph G is a *cocomparability graph* if it admits a vertex ordering $\sigma = [v_1, v_2, \ldots, v_n]$, called a *cocomparability ordering*, such that for any $i < j < k$, if v_i is adjacent to v_k then v_j must be adjacent to v_i or to v_k. According to [11], such an ordering of a cocomparability graph can be constructed in linear time. Note also that, given a permutation graph G, a *permutation model* (i.e., a set of segments between two parallel lines, defining G) can be found in linear time [11]. A *trapezoid model* for a trapezoid graph can be found in $O(n^2)$ time [9].

2 Collective Additive Tree Spanners

2.1 AT-free Graphs

It is known [15] that any AT-free graph admits one additive tree 3-spanner. In this subsection we show that any AT-free graph admits a system of two collective additive tree 2-spanners.

As a consequence of the Dominating Pair Theorem given in [2], any AT-free graph has a dominating shortest path which can be found in linear time by $2 \times LexBFS$ [3]. The $2 \times LexBFS$ method first starts a *lexicographic breadth-first search* (*LexBFS*) from an arbitrary vertex x of G and then starts a second LexBFS from the vertex x_0 last visited by the first LexBFS. Let x_l be the vertex of G last visited by the second LexBFS. As shown in [3], every shortest path $(x_0, x_1, \ldots, x_l)$, connecting x_0 and x_l, is a dominating shortest path of G. Next we demonstrate how to use such a dominating shortest path in an AT-free graph to show that every AT-free graph admits a system of two collective additive tree 2-spanners. We will need the following result from [6].

Lemma 1. *[6] Let $P := (x_0, x_1, \ldots, x_l)$ be a dominating shortest path of an AT-free graph $G = (V, E)$ constructed by $2 \times LexBFS$. Then, for every $i = 1, 2, \ldots, l$, every vertex $z \in N_i(x_0)$ is adjacent to x_i or x_{i-1}.*

Using this lemma, we construct a first spanning tree $T_1 = (V, E_1)$ for an AT-free graph $G = (V, E)$ as follows: put into initially empty E_1 all edges of the path $P := (x_0, x_1, \ldots, x_l)$, and then for each vertex $z \in N_i(x_0)$, put edge zx_{i-1} into E_1, if z is adjacent to x_{i-1} in G, and put edge zx_i into E_1, otherwise. We call this spanning tree the *caterpillar-tree* of G (with *spine* P). According to [15], this caterpillar-tree gives already an additive tree 3-spanner for the AT-free graph G. To get a collective additive stretch factor 2 for G, we construct a second spanning tree $T_2 = (V, E_2)$ for G as follows. Set $L_i := N_i(x_0)$ for each $i = 1, 2, \ldots, l$.

```
set E_2 := {all edges of the path P := (x_0, x_1, ..., x_l)};
set dev(x_i) := 0 for each vertex x_i of the path P;
for i = 1 to l do
   for each vertex z ∈ L_i \ {x_i} do
      among all neighbors of z in L_{i-1} choose a neighbor w with minimum
         deviation dev(w);
      add edge zw to E_2 and set dev(z) := dev(w) + 1;
   enddo
enddo.
```

We call spanning tree T_2 the *cactus-tree* of G (with *stem* P). It is evident, by construction, that the cactus-tree T_2 is a special kind of *breadth-first-search–tree* of G. The value $dev(z)$ (called the *deviation of z from stem P*) gives the distance in T_2 between vertex z and path P. In Figure 1 we show an AT-free graph G along with its caterpillar-tree T_1 and cactus-tree T_2.

Lemma 2. *Spanning trees $\{T_1, T_2\}$ are collective additive tree 2-spanners of AT-free graph G.*

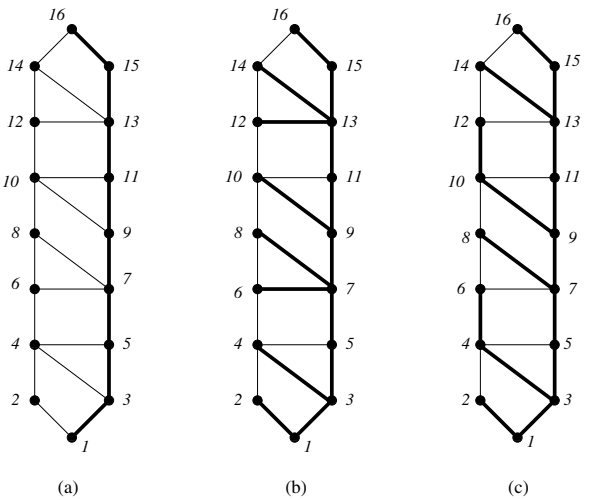

Fig. 1. (a) An AT-free graph G with a dominating path P, (b) the caterpillar-tree T_1 of G and (c) the cactus-tree T_2 of G

Proof. Consider two arbitrary vertices $x \in L_i$ and $y \in L_j$ $(y \neq x)$ of G, where $j \leq i$. If $i = j$, i.e., both x and y lie in the same layer $L_i = L_j$, then the distance in T_1 between x and y is at most 3, since in the worst case one of them is adjacent to x_i in T_1 and the second to x_{i-1}. Thus, $d_{T_1}(x, y) \leq 3 \leq d_G(x, y) + 2$ holds when $i = j$, and therefore, we may assume that $i > j$.

We know that $d_G(x, y) \geq i - j$. By the construction of the caterpillar-tree T_1, we have $d_{T_1}(y, x_j) \leq 2$ and $d_{T_1}(x, x_{i-1}) \leq 2$. Hence, $d_{T_1}(x, y) \leq d_{T_1}(x, x_{i-1}) + d_{T_1}(x_{i-1}, x_j) + d_{T_1}(y, x_j) \leq 2 + i - 1 - j + 2 \leq d_G(x, y) + 3$, and equality $d_{T_1}(x, y) = d_G(x, y) + 3$ holds if and only if $d_G(x, y) = i - j$, vertex x is adjacent to x_i in T_1 (and thus in G, vertex x is not adjacent to x_{i-1}) and vertex y is adjacent to x_{j-1} in T_1 but does not coincide with x_j. We will show that in this case in the cactus-tree T_2, $d_{T_2}(x, y) \leq d_G(x, y) + 2$.

Consider in G a shortest path $(y = y_0, y_1, \ldots, y_{i-j} = x)$ connecting vertices y and x. Clearly, $y_k \in L_{j+k}$ for each $k = 0, 1, \ldots, i - j - 1$, and since y_k is a neighbor of y_{k+1} in layer L_{j+k}, by construction of T_2, we have $dev(y_0) = 1$ and $dev(y_{k+1}) \leq dev(y_k) + 1 \leq k + 2$. Hence, the deviation of vertex x is at most $i - j + 1$. That is, there is a path in T_2 between x and a stem vertex x_s $(j - 1 \leq s \leq i - 2)$ of length $i - s$. The latter implies the existence in T_2 of a path of length $i - j + 1$ between vertices x and x_{j-1}. Therefore, $d_{T_2}(x, y) \leq d_{T_2}(x, x_{j-1}) + 1 = i - j + 1 + 1 = d_G(x, y) + 2$. □

From this lemma we immediately conclude.

Theorem 1. *Any AT-free graph admits a system of two collective additive tree 2-spanners, constructable in linear time.*

In the next subsection, we will show that to get a collective additive stretch factor 2 for some AT-free graphs, one needs at least two spanning trees. There-

fore, the result given in Theorem 1 is best possible. Furthermore, to achieve a collective additive stretch factor 1 or 0 for some AT-free graphs, one needs $\Omega(n)$ spanning trees.

2.2 Permutation Graphs and Trapezoid Graphs

It is known [10] that any permutation graph admits a multiplicative tree 3-spanner. In this subsection, we show that any permutation graph admits an additive tree 2-spanner and any system of collective additive tree 1–spanners must have $\Omega(n)$ spanning trees for some permutation graphs. Here also we disprove a conjecture given in [15], that any cocomparability graph admits an additive tree 2-spanner. We show that there exists even a trapezoid graph which does not admit any additive tree 2-spanner.

Let $G = (V, E)$ be a permutation graph given together with a permutation model. In what follows, "u.p." and "l.p." refer to a vertex's point on the upper and lower, respectively, line of the permutation model. Construct BFS-layers $(\{L_0, L_1, \cdots\})$ and the spine $\{x_1, x_2, \cdots\}$ of G as follows (the process continues until $L_i = \emptyset$).

set x_0 := the vertex whose u.p. is as far left as possible;
set $L_0 := \{x_0\}$;
set L_1 := {vertices whose l.p.s are to the left of the l.p. of x_0};
set x_1 := the vertex in L_1 with the u.p. as far right as possible;
set L_2 := {vertices whose u.p.s are between the u.p.s of x_0 and x_1} $\setminus L_1$;
set x_2 := the vertex in L_2 with the l.p. as far right as possible;
for $i = 3$ to n do
  if i is odd then
    set L_i := {vertices with l.p. between the l.p.s of x_{i-3} and x_{i-1}} $\setminus L_{i-1}$;
    set x_i := the vertex in L_i with the u.p. as far right as possible;
  else
    set L_i := {vertices whose u.p.s are between the u.p.s of x_{i-3} and x_{i-1}} $\setminus L_{i-1}$;
    set x_i := the vertex in L_i with the l.p. as far right as possible;
enddo.

It is straightforward to show that for all $i \geq 0$, if $y \in L_{i+1}$ then $x_i y \in E$. Now by forming T to include all edges from x_i to L_{i+1} for appropriate i, we can conclude: (The details will be in the journal version of the paper.)

Theorem 2. *Every permutation graph admits an additive tree 2-spanner, constructable in linear time.*

In the journal version we show that there exists a trapezoid graph which does not admit any additive tree 2-spanner, thereby disproving a conjecture from [15] that any cocomparability graph admits an additive tree 2-spanner. We show also that there are bipartite permutation graphs on $2n$ vertices for which any system of collective additive tree 1–spanners will need to have at least $\Omega(n)$ spanning trees.

2.3 DSP-graphs

It follows from a result in [15] that any DSP-graph admits one additive tree 4-spanner. In this subsection we show that any DSP-graph admits a system of two collective additive tree 3-spanners and a system of five collective additive tree 2-spanners.

Let $G = (V, E)$ be a DSP-graph and $P := (v = x_0, x_1, \ldots, x_l = u)$ be a dominating shortest path of G. We will build five spanning trees $\{T_1, T_2, T_3, T_4, T_5\}$ for G, all containing the edges of P, in such a way that for any two vertices $x, y \in V$, there will be a tree $T' \in \{T_1, T_2, T_3, T_4, T_5\}$ with $d_{T'}(x, y) \leq d_G(x, y) + 2$.

Our first three trees T_1, T_2, T_3 are very similar to the trees constructed for AT-free graphs. The tree $T_1 = (V, E_1)$ is constructed as follows. Add to initially empty set E_1 all edges of path P. Then, for each vertex $z \in V \setminus P$ choose an arbitrary neighbor w_z in P and add edge zw_z to E_1. The tree T_1 is an analog of the caterpillar-tree constructed for an AT-free graph. The second and third trees are analogs of the cactus-tree considered for an AT-free graph. The tree $T_2 = (V, E_2)$ is a special breadth-first-search-tree T_v with vertex v as the root, the tree $T_3 = (V, E_3)$ is a special breadth-first-search-tree T_u with vertex u as the root. For construction of T_2 we can use the algorithm given in Subsection 2.1 with one additional line at the end: for each $z \in N_{l+1}(v)$, add edge zu to E_2 and set $dev(z) := 1$. T_3 is constructed similarly; we simply reverse the order of vertices of P and consider u instead of v and E_3 instead of E_2. The detailed algorithm will appear in the journal version.

Our tree $T_4 = (V, E_4)$ is a generalization of the tree T_2 and is constructed as follows.

```
set E_4 := {all edges of the path P := (v = x_0, x_1, ..., x_l = u)};
set dev(x_i) := 0 for each vertex x_i of the path P;
for i = 1 to l do
   for each vertex z ∈ N_i(v) \ {x_i} do case
      case (z is adjacent to x_{i-1} in G)
         add edge z x_{i-1} to E_4 and set dev(z) := 1;
      case (z is adjacent to x_i in G)
         add edge z x_i to E_4 and set dev(z) := 1;
      case (z is adjacent to a vertex w ∈ N_i(v) which is adjacent to x_{i-1})
         choose such a w and add edge zw to E_4 and set dev(z) := 2;
      otherwise /* none of above */
         among all neighbors of z in N_{i-1}(v) choose a neighbor w with
            minimum deviation dev(w) (break ties arbitrarily);
         add edge zw to E_4 and set dev(z) := dev(w) + 1;
   endcase
enddo
for each z ∈ N_{l+1}(v), add edge zu to E_4 and set dev(z) := 1.
```

It is an easy exercise to show by induction that for any vertex $z \in N_i(v)$, the vertex of P closest to z in T_4 is either x_s or x_{s-1} with $s = i - dev(z) + 1$. Moreover, the length of the path of T_4 between z and P is $dev(z)$. Our last tree $T_5 = (V, E_5)$ is a version of the tree T_4, constructed downwards.

set $E_5 := \{$all edges of the path $P := (v = x_0, x_1, \ldots, x_l = u)\}$;
set $dev(x_i) := 0$ for each vertex x_i of the path P and $dev(z) := \infty$ for any $z \in V \setminus P$;
for $i = l - 1$ down to 1 do
  for each vertex $z \in N_i(v) \setminus \{x_i\}$ do case
    case (z is adjacent to x_{i+1} in G)
      add edge zx_{i+1} to E_5 and set $dev(z) := 1$;
    case (z is adjacent to x_i in G)
      add edge zx_i to E_5 and set $dev(z) := 1$;
    case (z is adjacent to a vertex $w \in N_i(v)$ which is adjacent to x_{i+1})
      choose such a w and add edge zw to E_5 and set $dev(z) := 2$;
    otherwise
      if z has neighbors in $N_{i+1}(v)$ then
        among all neighbors of z in $N_{i+1}(v)$ choose a neighbor w with
          minimum deviation $dev(w)$ (break ties arbitrarily);
        if $dev(w) < \infty$ then add edge zw to E_5 and set $dev(z) := dev(w) + 1$;
  endcase
enddo
for each vertex z with $dev(z)$ still ∞ do
  let $z \in N_i(v)$;
  if $i = l$ and z is adjacent to x_l then add edge zx_l to E_5 and set $dev(z) := 1$;
  else add edge zx_{i-1} to E_5;
  /* this edge exists in G since P is a dominating path
    and z is adjacent in G neither to x_{i+1} nor x_i */
enddo.

Again, it is easy to see that for any vertex $z \in N_i(v)$ with finite deviation $dev(z)$, the vertex of P closest to z in T_5 is either x_s or x_{s+1} with $s = i+dev(z)-1$. The length of the path of T_5 between z and P is $dev(z)$.

We are ready to present the main result of this subsection.

Lemma 3. *Let G be a DSP-graph with a dominating shortest path $P := (v = x_0, x_1, \ldots, x_l = u)$ and spanning trees T_1, T_2, T_3, T_4, T_5 constructed starting from P as described above. Then, for any two vertices $x, y \in V$:*

1. *$d_{T_1}(x, y) \leq d_G(x, y) + 4$;*
2. *there is a tree $T' \in \{T_1, T_2\}$ such that $d_{T'}(x, y) \leq d_G(x, y) + 3$;*
3. *there is a tree $T'' \in \{T_1, T_2, T_3, T_4, T_5\}$ such that $d_{T''}(x, y) \leq d_G(x, y) + 2$.*

Proof. The proof of this lemma is quite technical and will appear in the journal version of the paper. □

Theorem 3. *Any DSP-graph admits one additive tree 4-spanner, a system of two collective additive tree 3-spanners and a system of five collective additive tree 2-spanners. Moreover, given a dominating shortest path of G, all trees are constructable in linear time.*

Note that there is a DSP-graph for which two trees are necessary to get a collective additive stretch factor 3. However, it is an open question whether to achieve a collective additive stretch factor 2, one really needs five spanning trees.

2.4 Graphs with Bounded Asteroidal Number

It is known [7] that any graph G with asteroidal number $\mathsf{an}(G)$ admits an additive tree $(3\mathsf{an}(G)-1)$-spanner. In this subsection we show that any graph with asteroidal number $\mathsf{an}(G)$ admits a system of $\mathsf{an}(G)(\mathsf{an}(G)-1)/2$ collective additive tree 4-spanners and a system of $\mathsf{an}(G)(\mathsf{an}(G)-1)$ collective additive tree 3-spanners.

In what follows we will use the following definitions and results from [7]. An asteroidal set A of a graph G is *repulsive* if for every vertex $v \in V \setminus N[A]$, not all vertices of A are contained in one connected component of $G - N[v]$. According to [7], any graph has a repulsive asteroidal set. A set $D \subseteq V$ in a graph G is said to be a *dominating target*, if $D \cup S$ is a dominating set in G for every set $S \subseteq V$ for which the subgraph of G induced by $D \cup S$ is connected. It is shown in [7] that any graph G has a dominating target D with $D \leq \mathsf{an}(G)$. Furthermore, every repulsive asteroidal set of G is such a dominating target of G.

In the journal version of the paper, we prove the following stronger version of the result above. Let $D \subseteq V$ be a repulsive asteroidal set of a graph $G = (V, E)$.

Lemma 4. *For every $x, y \in V$, there exist $a, b \in D$ such that $x, y \in N[P]$ for any path P of G between a and b.*

Consider two arbitrary vertices a, b of D and a shortest path $P(a,b) := (a = x_0, x_1, \ldots, x_l = b)$ connecting a and b in G. We can build two spanning trees $T_1(a,b)$ and $T_2(a,b)$ for G, both containing the edges of $P(a,b)$, in such a way that for any two vertices $x, y \in N[P(a,b)]$, $d_{T_1(a,b)}(x,y) \leq d_G(x,y) + 4$ and $\min\{d_{T_1(a,b)}(x,y), d_{T_2(a,b)}(x,y)\} \leq d_G(x,y) + 3$.

Our trees $T_1(a,b)$ and $T_2(a,b)$ are very similar to the trees constructed for AT-free graphs. The tree $T_1(a,b) = (V, E_1)$ is constructed as follows. Add to initially empty set E_1 all edges of path $P(a,b)$. Then, for each vertex $z \in N(P(a,b))$ choose an arbitrary neighbor w in $P(a,b)$ and add edge zw to E_1. The obtained subtree of G (which covers so far only vertices from $N[P(a,b)]$) extends to a spanning tree $T_1(a,b)$ arbitrarily. The tree $T_1(a,b)$ is an analog of the caterpillar-tree constructed for an AT-free graph. The second tree is an analog of the cactus-tree considered for an AT-free graph. The tree $T_2(a,b) = (V, E_2)$ is a special breadth-first-search-tree T_a with vertex a as the root. The detailed algorithm for constructing $T_2(a,b)$ and the proof of the following lemma will appear in the journal version.

Lemma 5. *For any two vertices $x, y \in N[P(a,b)]$:*

1. *$d_{T_1(a,b)}(x,y) \leq d_G(x,y) + 4$;*
2. *there is a tree $T' \in \{T_1(a,b), T_2(a,b)\}$ such that $d_{T'}(x,y) \leq d_G(x,y) + 3$.*

If we construct trees $T_1(a,b)$ and $T_2(a,b)$ for each pair of vertices $a, b \in D$, from Lemma 4 and Lemma 5, we get (recall that $|D| \leq \mathsf{an}(G)$):

Theorem 4. *Any graph G with asteroidal number $\mathsf{an}(G)$ admits a system of $\mathsf{an}(G)(\mathsf{an}(G)-1)/2$ collective additive tree 4-spanners and a system of $\mathsf{an}(G)(\mathsf{an}(G)-1)$ collective additive tree 3-spanners.*

Corollary 1. *Any graph G with asteroidal number bounded by a constant admits a system of a constant number of collective additive tree 3-spanners. Moreover, given a repulsive asteroidal set of G, all trees are constructable in total linear time.*

3 Routing Labeling Schemes in AT-free Graphs

In this section, we use the results obtained above to design compact and efficient routing labeling schemes for AT-free graphs. For DSP-graphs and graphs with bounded asteroidal number, corresponding routing labeling schemes are described in the journal version of the paper. We will show that any AT-free graph with diameter $\mathcal{D} := diam(G)$ and maximum vertex degree Δ admits a $(3\log_2 \mathcal{D} + 6\log_2 \Delta + O(1))$-bit routing labeling scheme of deviation at most 2. Moreover, the scheme is computable in linear time, and the routing decision is made in constant time per vertex.

It is worth mentioning that any AT-free graph admits a $(\log_2 \mathcal{D} + 1)$-bit distance labeling scheme of deviation at most 2 (see [5]). That is, there is a function L labeling the vertices of each AT-free graph G with (not necessarily distinct) labels of up to $\log_2 \mathcal{D}+1$ bits such that given two labels $L(v), L(u)$ of two vertices v, u of G, it is possible to compute in constant time, by merely inspecting the labels of u and v, a value $\widehat{d}(u,v)$ such that $0 \leq \widehat{d}(u,v) - d_G(u,v) \leq 2$. To the best of our knowledge, the method of [5] cannot be used (at least directly) to design a routing labeling scheme for AT-free graphs.

Labels. In subsection 2.1, we showed that any AT-free graph $G = (V, E)$ admits a system of two collective additive tree 2-spanners. During the construction of the cactus-tree T_2 for G, each vertex $z \in V$ received a deviation number $dev(z)$ which is the distance in T_2 between z and the stem $P := (x_0, x_1, \ldots, x_l)$ of T_2. To simplify the routing decision, it will be useful to construct one more spanning tree $T' = (V, E')$ for G. Let $P := (x_0, x_1, \ldots, x_l)$ be the dominating path of G described in Lemma 1.

set $E' :=$ {all edges of the path $P := (x_0, x_1, \ldots, x_l)$};
set $dev'(x_i) := 0$ for each x_i of the path P and $dev'(z) := l + 1$ for any $z \in V \setminus P$;
for each $z \in N_l(x_0)$ which is adjacent to x_l, set $dev'(z) := 1$ and add edge zx_l to E';
for $i = l - 1$ down to 1 do
  for each vertex $z \in N_i(x_0) \setminus \{x_i\}$ do
    if z is adjacent to x_i in G then add edge zx_i to E' and set $dev'(z) := 1$;
    else if z has neighbors in $N_{i+1}(x_0)$ then
      among all neighbors of z in $N_{i+1}(x_0)$, choose a neighbor w with
        minimum deviation $dev'(w)$ (break ties arbitrarily);
      if $dev'(w) < l + 1$ then add edge zw to E' and set $dev'(z) := dev'(w) + 1$;
  enddo
enddo
for each vertex z with $dev'(z)$ still $l + 1$ do
  let $z \in N_i(x_0)$;
  add edge zx_{i-1} to E';
enddo.

We name tree T' the *willow-tree* of G. As a result of its construction, each vertex $z \in V$ received a second deviation number $dev'(z)$, which is either $l+1$ or the distance in T' between z and the path $P := (x_0, x_1, \ldots, x_l)$ of T'.

Now we are ready to describe the routing labels of the vertices of G. For each vertex $x_i \in P$ $(i = 0, 1, \ldots, l)$, we have

$$Label(x_i) := (b(x_i), level(x_i), port_{up}(x_i), port_{down}(x_i)),$$

where

- $b(x_i) := 1$, a bit indicating that x_i belongs to P;
- $level(x_i)$ $(= i)$ is the index of x_i in P, i.e., the distance $d_G(x_i, x_0)$;
- $port_{up}(x_i)$ is the port number at vertex x_i of the edge x_ix_{i+1} (if $i = l$, $port_{up}(x_i) := \mathsf{nil}$);
- $port_{down}(x_i)$ is the port number at vertex x_i of the edge x_ix_{i-1} (if $i = 0$, $port_{down}(x_i) := \mathsf{nil}$).

For each vertex $z \in V \setminus P$, we have

$$Label(z) := (b(z), level(z), a_v(z), port_{v-in}(z), port_{v-out}(z), a_h(z), port_{h-in}(z),$$

$$port_{h-out}(z), dev(z), port_{down}(z), dev'(z), port_{up}(z)),$$

where

- $b(z) := 0$, a bit indicating that z does not belong to P;
- $level(z)$ is the distance $d_G(z, x_0)$;
- $a_v(z)$ is a bit indicating whether z is adjacent to $x_{label(z)-1}$;
- $port_{v-in}(z)$ is the port number at vertex $x_{label(z)-1}$ of the edge $x_{label(z)-1}z$ (if z and $x_{label(z)-1}$ are not adjacent in G, then $port_{v-in}(z) := \mathsf{nil}$);
- $port_{v-out}(z)$ is the port number at vertex z of the edge $zx_{label(z)-1}$ (if z and $x_{label(z)-1}$ are not adjacent in G, then $port_{v-out}(z) := \mathsf{nil}$);
- $a_h(z)$ is a bit indicating whether z is adjacent to $x_{label(z)}$;
- $port_{h-in}(z)$ is the port number at vertex $x_{label(z)}$ of the edge $x_{label(z)}z$ (if z and $x_{label(z)}$ are not adjacent in G, then $port_{h-in}(z) := \mathsf{nil}$);
- $port_{h-out}(z)$ is the port number at vertex z of the edge $zx_{label(z)}$ (if z and $x_{label(z)}$ are not adjacent in G, then $port_{h-out}(z) := \mathsf{nil}$);
- $dev(z)$ is the deviation of z in tree T_2;
- $port_{down}(z)$ is the port number at vertex z of the edge zw, where w is the father of z in T_2;
- $dev'(z)$ is the deviation of z in tree T';
- $port_{up}(z)$ is the port number at vertex z of the edge zw, where w is the father of z in T' (if $dev'(z) = l+1$, $port_{up}(z) := \mathsf{nil}$).

Clearly, the label size of each vertex of G is at most $3\lceil \log_2 l \rceil + 6\lceil \log_2 \Delta \rceil + 3 \leq 3\log_2 \mathcal{D} + 6\log_2 \Delta + O(1)$ bits.

Routing Decision. The routing decision algorithm is obvious. Suppose that a packet with the header (address of destination) $Label(y)$ arrives at vertex x.

The vertex x can use the following constant time algorithm to decide where to submit the packet. Note that each vertex v of G is uniquely identified by its label $Label(v)$.

function routing_decision_AT-free($Label(x), Label(y)$)

```
if Label(x) = Label(y) then return "packet reached its destination";
else do case
  case (b(x) = 1)
  /* x belongs to P and routing is performed on the caterpillar-tree T1 of G */
    do case
      case (level(x) > level(y))
        send packet via port_down(x);
      case (level(x) < level(y))
        if b(y) = 1 then send packet via port_up(x);
        else if level(y) = level(x) + 1 and a_v(y) = 1 then
                send packet via port_{v-in}(y);
             else send packet via port_up(x);
      case (level(x) = level(y))
        if a_h(y) = 1 then send packet via port_{h-in}(y);
        else send packet via port_down(x);
    endcase;
  /* now x does not belong to P */
  case (level(x) > level(y))
    do case
      case (a_v(x) = 1)
        send packet via port_{v-out}(x); /* routing is performed on T1 */
      case (b(y) = 1 or b(y) = 0 and a_h(y) = 1)
        send packet via port_{h-out}(x); /* routing is performed on T1 */
      otherwise /* here we have d_{T1}(x, y) = level(x) - level(y) + 3 */
        if dev(x) ≤ level(x) - level(y) + 1 then send packet via port_down(x);
             /* the cactus-tree T2 of G is used for routing */
        else send packet via port_{h-out}(x); /* routing is performed on T1 */
    endcase;
  case (level(x) < level(y))
    do case
      case (a_h(x) = 1)
        send packet via port_{h-out}(x); /* routing is performed on T1 */
      case (b(y) = 1 or b(y) = 0 and a_v(y) = 1)
        send packet via port_{v-out}(x); /* routing is performed on T1 */
      otherwise /* here we have d_{T1}(x, y) = level(y) - level(x) + 3 */
        if dev'(x) ≤ level(y) - level(x) + 1 then send packet via port_up(x);
             /* the willow-tree T' of G is used for routing */
        else send packet via port_{v-out}(x); /* routing is performed on T1 */
    endcase;
  case (level(x) = level(y)) /* routing is performed on T1 */
    if a_h(x) = 1 then send packet via port_{h-out}(x);
    else send packet via port_{v-out}(x);
endcase.
```

Thus, we have the following result.

Theorem 5. *Every AT-free graph of diameter $\mathcal{D} := diam(G)$ and of maximum vertex degree Δ admits a $(3\log_2 \mathcal{D} + 6\log_2 \Delta + O(1))$-bit routing labeling scheme of deviation at most 2. Moreover, the scheme is computable in linear time, and the routing decision is made in constant time per vertex.*

Acknowledgements

DGC wishes to thank the Natural Sciences and Engineering Research Council of Canada for financial assistance and the Department of Computer Science at Kent State University for hosting his visit there.

References

1. L.P. Chew, There are planar graphs almost as good as the complete graph, *J. of Computer and System Sciences,* 39 (1989), 205–219.
2. D.G. Corneil, S. Olariu and L. Stewart, Asteroidal Triple–free Graphs, *SIAM J. Discrete Math.*, 10 (1997), 399–430.
3. D.G. Corneil, S. Olariu and L. Stewart, Linear time algorithms for dominating pairs in asteroidal triple–free graphs, *SIAM J. on Computing,* 28 (1999), 1284–1297.
4. F.F. Dragan, C. Yan and I. Lomonosov, Collective tree spanners of graphs, *Algorithm Theory - SWAT 2004, 9th Scandinavian Workshop on Algorithm Theory,* Humlebaek, Denmark, July 8-10, 2004, *Lecture Notes in Computer Science*, Vol. 3111, pp. 64 - 76.
5. C. Gavoille, M. Katz, N.A. Katz, C. Paul, and D. Peleg, Approximate Distance Labeling Schemes, in *Proceedings of the 9th Annual European Symposium on Algorithms" (ESA'01)*, Aarhus, Denmark, August 28-31, 2001, *Lecture Notes in Computer Science 2161*, Springer, 2001, pp. 476-487.
6. T. Kloks, D. Kratsch and H. Müller, Approximating the bandwidth for asteroidal triple-free graphs, *J. Algorithms,* 32 (1999), 41–57.
7. T. Kloks, D. Kratsch and H. Müller, On the structure of graphs with bounded asteroidal number, *Graphs and Combinatorics,* 17 (2001), 295–306.
8. A.L. Liestman and T. Shermer, Additive graph spanners, *Networks,* 23 (1993), 343–364.
9. T.H. Ma and J.P. Spinrad, On the two-chain subgraph cover and related problems, *J. of Algorithms*, 17 (1994), 251–268.
10. M.S. Madanlal, G. Venkatesan, and C. Pandu Rangan, Tree 3-spanners on interval, permutation and regular bipartite graphs, *Inform. Process. Lett.,* 59 (1996), 97–102.
11. R.M. McConnell and J.P. Spinrad, Linear-time transitive orientation, In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms,* New Orleans, Louisiana, 5-7 January 1997, pp. 19-25.
12. D. Peleg, Distributed Computing: A Locality-Sensitive Approach, *SIAM Monographs on Discrete Math. Appl.*, SIAM, Philadelphia, 2000.
13. D. Peleg, and A.A. Schäffer, Graph Spanners, *J. Graph Theory*, 13 (1989), 99-116.
14. D. Peleg and J.D. Ullman, An optimal synchronizer for the hypercube, *in Proc. 6th ACM Symposium on Principles of Distributed Computing,* Vancouver, 1987, 77–85.
15. E. Prisner, D. Kratsch, H.-O. Le, H. Müller, and D. Wagner, Additive tree spanners, *SIAM Journal on Discrete Mathematics*, 17 (2003), 332–340.