Succinct Greedy Geometric Routing in the Euclidean Plane*

Michael T. Goodrich and Darren Strash

Computer Science Department, University of California, Irvine, USA

Abstract. We show that greedy geometric routing schemes exist for the Euclidean metric in $\mathbf{R}^2$, for 3-connected planar graphs, with coordinates that can be represented *succinctly*, that is, with $O(\log n)$ bits, where n is the number of vertices in the graph.

1 Introduction

Geometric routing algorithms perform message passing using geometric information stored at the nodes and edges of a network.

Greedy Geometric Routing. Perhaps the simplest routing rule is the *greedy* one:

- If a node v receives a message M intended for a destination $w \neq v$, then v should forward M to a neighbor that is closer to w than v is.

This rule can be applied in any metric space, of course, but simple and natural metric spaces are preferred over cumbersome or artificial ones. Thus, we are interested in greedy routing schemes that assign network nodes to virtual coordinates in a natural metric space.

Interest in greedy geometric routing in fixed-dimensional Euclidean spaces has expanded greatly since the work by Papadimitriou and Ratajczak [8], who showed that any 3-connected planar graph can be embedded in $\mathbf{R}^3$ so as to support greedy geometric routing. Indeed, their conjecture that such embeddings are possible in $\mathbf{R}^2$ spawned a host of additional papers (e.g., see [1,2,3,6,7]). Leighton and Moitra [5] settled this conjecture by giving an algorithm to produce a greedy embedding of any 3-connected planar graph in $\mathbf{R}^2$, and a similar result was independently found by Angelini *et al.* [1]. Greedy embeddings in $\mathbf{R}^2$ were previously known for graphs containing Delaunay triangulations [6], and existentially (but not algorithmically) for triangulations [2].

Succinct Geometric Routing. In spite of their theoretical elegance, these results settling the Papadimitriou-Ratajczak conjecture have an unfortunate drawback, in that the virtual coordinates of nodes in these solutions require $\Omega(n \log n)$ bits each in the worst case. These space inefficiencies reduce the applicability of these results for greedy geometric routing, since one could alternatively keep routing tables of size $O(n \log n)$ bits

* This work was supported by NSF grants 0724806, 0713046, 0830403, and ONR grant N00014-08-1-1015.

Y. Dong, D.-Z. Du, and O. Ibarra (Eds.): ISAAC 2009, LNCS 5878, pp. 781–791, 2009.

at each network node to support message passing. Indeed, such routing tables would allow for network nodes to be identified using labels of only $O(\log n)$ bits each, which would significantly cut down on the space, bandwidth, and packet header size needed to communicate the destination for each packet being routed. Thus, for a solution to be effectively solving the routing problem using a greedy geometric routing scheme, we desire that it be *succinct*, that is, it should use $O(\log n)$ bits per virtual coordinate. Succinct greedy geometric routing schemes are known for fixed-dimensional hyperbolic spaces [3,7], but we are unaware of any prior work on succinct greedy geometric routing in fixed-dimensional Euclidean spaces.

Our Results. We provide a succinct greedy geometric routing scheme for 3-connected planar graphs in $\mathbf{R}^2$. At the heart of our scheme is a new greedy embedding for 3-connected planar graphs in $\mathbf{R}^2$ which exploits the tree-like topology of a spanning (Christmas cactus) subgraph. Our embedding allows us to form a coordinate system which uses $O(\log n)$ bits per vertex, and allows distance comparisons to be done just using our coordinate representations consistently with the Euclidean metric.

2 Finite-Length Coordinate Systems

Let us begin by formally defining what we mean by a coordinate system, and how that differs, for instance, from a simple compression scheme. Let Σ be an alphabet, and let Σ^* a set of finite-length strings over Σ. We define a *coordinate system* f for a space S:

1. f is a map, $f : \Sigma^* \to S$, which assigns character strings to points of S.
2. f may be *parameterized*: the assignment of strings to points may depend on a fixed set of parameters.
3. f is *oblivious*: the value of f on any given $x \in \Sigma^*$ must depend only on f's parameters and x itself. It cannot rely on any other character strings in Σ^*, points in S, or other values of f.

If f is lacking property 3, we prefer to think of f as a *compression scheme*.

3 Greedy Routing in Christmas Cactus Graphs

Our method is a non-trivial adaptation of the Leighton and Moitra scheme [5], so we begin by reviewing some of the ideas from their work.

A graph G is said to be a *Christmas cactus graph* if: (1) each edge of G is in at most one cycle, (2) G is connected, and (3) removing any vertex disconnects G into at most two components. For ease of discussion, we consider any edge in a Christmas cactus graph that is not in a simple cycle to be a simple cycle itself (a 2-cycle); hence, every edge in is in exactly one simple cycle. The *dual tree* of a Christmas cactus graph G is a tree containing a vertex for each simple cycle in G with an edge between two vertices if their corresponding cycles in G share a vertex. Rooting the dual tree at an arbitrary vertex creates what we call a *depth tree*.

Having a depth tree allows us to apply the rooted tree terminology to cycles in G. In particular: *root*, *depth*, *parent*, *child*, *ancestor*, and *descendant* all retain their familiar

definitions. We define the *depth* of a node v to be the minimum depth of any cycle containing v. The unique node that a cycle C shares with its parent is called the *primary node* of C. Node v is a *descendant* of a cycle C if v is in a cycle that is a descendant of C and v is not the primary node of C. Node v is a *descendant* of node u if removing neighbors of u with depth less than or equal to u leaves u and v in the same component.

Greedy Routing with a Christmas Cactus Graph Embedding. Working level by level in a depth tree, Leighton and Moitra [5] embed the cycles of a Christmas cactus graph on semi-circles of increasing radii, centered at the origin. Within the embedding we say that vertex u is above vertex v if u is embedded farther from the origin than v, and we say that u is to the left of v if u is embedded in the positive angular direction relative to v. We can define below and right similarly. These comparisons naturally give rise to directions of movement between adjacent vertices in the embedding: up, down, left, and right.

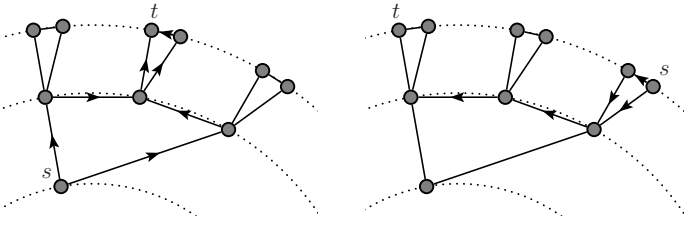

Fig. 1. Arrows indicate valid greedy hops. (a) Descendants of s can be reached by a simple path of up and right hops, up and left hops, or a combination of the two. (b) If t is not a descendant of s, then we route down and (left or right) in the direction of t until we reach an ancestor of t.

Routing from start vertex s to a terminal vertex t in a Christmas cactus graph embedding can be broken down into two cases: (1) t is a descendant of s, and (2) t is not a descendant of s.

1. As shown in Fig. 1(a), if t is a descendant of s, then we can route to t by a simple path of up and right hops, up and left hops, or a combination of the two.
2. As shown in Fig. 1(b), if t is not a descendant of s, then we route to the least common (cycle) ancestor of s and t. Suppose, without loss of generality, that t is to the left of s, then we can reach this cycle by a sequence of down and left hops. Once on the cycle, we can move left until we reach an ancestor of t. Now we are back in case 1.

This routing scheme immediately gives rise to a simple succinct compression scheme for 3-connected planar graphs, which we discuss in the full version of this paper.

4 Toward a Succinct Greedy Embedding

Given a 3-connected planar graph, we can find a spanning Christmas cactus subgraph in polynomial time [5]. Therefore, we restrict our attention to Christmas cactus graphs.

Our results apply to 3-connected planar graphs with little or no modification. In this section, we construct a novel greedy embedding scheme for any Christmas cactus graph in $\mathbf{R}^2$. We then build a coordinate system from our embedding and show that the coordinates can be represented using $O(\log^2 n)$ bits. In the next section, we show how to achieve an optimal $O(\log n)$-bit representation.

Heavy Path Decompositions. We begin by applying the Sleator and Tarjan [9] *heavy path decomposition* to the depth tree T for G.

Definition 1. *Let T be a rooted tree. For each node v in T, let $n_T(v)$ denote the number of descendants of v in T, including v. For each edge $e = (v, \text{parent}(v))$ in T, label e as a heavy edge if $n_T(v) > n_T(\text{parent}(v))/2$. Otherwise, label e as a light edge. Connected components of heavy edges form paths, called heavy paths. Vertices that are incident only to light edges are considered to be zero-length heavy paths. We call this the* ***heavy path decomposition*** *of T.*

For ease of discussion, we again apply the terminology from nodes in T to cycles in G. A cycle in G is on a heavy path H if its dual node in T is on H. Let H be a heavy path in T. We say that $\text{head}(H)$ is the cycle in H that has minimum depth, we define $\text{tail}(H)$ similarly. Let C_1 and C_2 be two cycles such that $C_1 = \text{parent}(C_2)$ and let $\{p\} = V(C_1) \cap V(C_2)$. If C_1 and C_2 are on the same heavy path then we call p a *turnpike*. If C_1 and C_2 are on different heavy paths (where $C_1 = \text{tail}(H_1)$ and $C_2 = \text{head}(H_2)$) then we call p an *off-ramp* for H_1 and the vertices $v \in V(C_2) \setminus \{p\}$ *on-ramps* for H_2.

An Overview of Our Embedding Strategy. Like Leighton and Moitra [5], we lay the cycles from our Christmas cactus graph on concentric semi-circles of radius $1 = R_0 < R_1 < R_2 \ldots$; however, our embedding has the following distinct differences: we have $\Theta(n \log n)$ semi-circles instead of $O(n)$ semi-circles, on-ramps to heavy paths are embedded on special semi-circles which we call *super levels*, turnpikes are placed in a predefined position when cycles are embedded, and the radii of semi-circles can be computed without knowing the topology of the particular Christmas cactus graph being embedded.

To make our embedding scheme amenable to a proof by induction, we modify the input Christmas cactus graph. After constructing a greedy embedding of this modified graph, we use it to prove that we have a greedy embedding for the original graph.

Modifying the Input Christmas Cactus Graph. Given a Christmas cactus graph G on n vertices, we choose a depth tree T of G, and compute the heavy path decomposition of T. For a cycle C on a heavy path H, we define $\text{relativeDepth}(C)$ to be $\text{depth}(C) - \text{depth}(\text{head}(H))$. For each C_1, $C_2 = \text{child}(C_1)$ forming a light edge in T, let $\{p\} = V(C_1) \cap V(C_2)$. Split p into two vertices p_1 and p_2 each on their own cycle, and connect p_1 to p_2 with a path of $n - 1 - \text{relativeDepth}(C_1)$ edges. The new graph G' is also a Christmas cactus graph, and our new depth tree T' looks like T stretched out so that heads of heavy paths (from T) are at depths that are multiples of n. We continue to call the paths copied from T heavy paths (though they do not form a heavy path decomposition of T'), and the newly inserted edges are *dummy* edges.

Embedding the Modified Christmas Cactus Graph in $\mathbf{R}^2$. Given a Christmas cactus graph G on n vertices, run the modification procedure described above and get G' and T'. We embed G' in phases, and prove by induction that at the end of each phase we have a greedy embedding of an induced subgraph of G'.

Lemma 1 (Leighton and Moitra [5]). *If points* $c = (0, 1 + z)$, $b = (-\sin\beta, \cos\beta)$, *and* $a = (-(1+\epsilon)\sin(\beta-\alpha), (1+\epsilon)\cos(\beta-\alpha))$ *are subject to the constraints* $0 < \alpha \leq \pi/2$, $0 < \beta \leq \pi/2$, $0 < \epsilon \leq (1-\cos\beta)/6$, $0 \leq z \leq \epsilon$, *and* $\sin\alpha \leq \frac{\epsilon(1-\cos\beta)}{2(1+\epsilon)}$ *then* $d(a,c) - d(b,c) \geq \epsilon^2 > 0$.

We begin by embedding the root cycle, $C = (v_0, \ldots, v_{k-1})$, of T'. We trace out a semi-circle of radius $R_0 = 1$ centered at the origin and divide the perimeter of this semi-circle into $2n + 1$ equal arcs. We allow vertices to be placed at the leftmost point of each arc, numbering these positions 0 to $2n$. We place vertices $v_0, \ldots, v_{k-1}$ clockwise into any k distinct positions, reserving position n for C's turnpike. If C does not have a turnpike, as is the case if C is a dummy edge or the tail of a heavy path, then position n remains empty. The embedding of C is greedy (proof omitted here).

Inductive Step: Suppose we have a greedy embedding all cycles in T' up to depth i, call this induced subgraph G'_i. We show that the embedding can be extended to a greedy embedding of G'_{i+1}. Our proof relies on two values derived from the embedding of G'_i.

Definition 2. *Let* s, t *be any two distinct vertices in* G'_i *and fix* $n_{s,t}$ *to be a neighbor of* s *such that* $d(s,t) > d(n_{s,t}, t)$. *We define* $\delta(G'_i) = \min_{s,t}\{d(s,t) - d(n_{s,t}, t)\}$.

We refer to the difference $d(s,t) - d(n_{s,t}, t)$ as the *delta value* for distance-decreasing paths from s to t through $n_{s,t}$.

Definition 3. *Let* $\beta(G'_i)$ *to be the minimum (non-zero) angle that any two vertices in the embedding of* G'_i *form with the origin.*

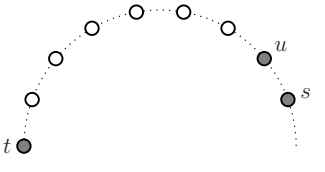

Fig. 2. s, u and t form a lower bound for $\delta(G'_0)$

Since we do not specify exact placement of all vertices, we cannot compute $\delta(G'_0)$ and $\beta(G'_0)$ exactly. We instead compute positive underestimates, δ_0 and β_0, by considering hypothetical vertex placements, and by invoking the following lemma.

Lemma 2. *Let* s *and* u *be two neighboring vertices embedded in the plane. If there exists a vertex* t *that is simultaneously closest to the perpendicular bisector of* su *(on the* u *side), and farthest from the line* su, *then the delta value for* s *to* t *through* u *is the smallest for any choice of* t.

Applying the above lemma to all hypothetical s, u, and t placements for the embedding of G'_0 leads to the underestimate $\delta_0 = 2-\sqrt{2+2\cos\frac{\pi}{2n+1}} < d(s,t)-d(u,t) \leq \delta(G'_0)$ where s, u, and t are shown in Fig. 2. Trivially, $\beta_0 = \frac{\pi}{2n+1} \leq \beta(G'_0)$.

We now show how to obtain a greedy embedding of G'_{i+1}, given a greedy embedding of G'_i and values δ_i and β_i.

Let $\epsilon_i = \min\{\delta_i/3, R_i\frac{1-\cos\frac{2}{3}\beta_i}{6}\}$. Trace out a semi-circle of radius $R_{i+1} = R_i + \epsilon_i$ centered at the origin. Each cycle at depth $i+1$ of T' has the form $C = (v, x_1, \ldots, x_m)$ where v, the primary node of C, has already been embedded on the ith semi-circle. We embed vertices x_1 to x_m in two subphases:

Subphase 1 We first embed vertex x_1 from each C. Choose an orientation for C so that x_1 is not a turnpike.[1] We place x_1 where the ray beginning at the origin and passing through v meets semi-circle $i+1$. We now show that distance decreasing paths exist between all pairs of vertices embedded thus far.

Distance decreasing paths between vertices in G'_i are preserved by the induction hypothesis. For t placed during this subphase: t has a neighbor v embedded on semi-circle i. If $s = v$ then s's neighbor t is strictly closer to t. Otherwise if $s \in G'_i$ then since t is within distance $\delta_i/3$ of v, then s's neighbor u that is closer to v is also closer to t. If s was placed during this subphase then s is within distance $R_i\frac{1-\cos\frac{2}{3}\beta_i}{6}$ from its neighbor v, and the perpendicular bisector of sv contains s on one side and every other vertex placed on the other side. Therefore s's neighbor v is closer to t.

The next subphase requires new underestimates, which we call δ_i^1 and β_i^1. By construction, $\beta_i^1 = \beta_i$. No s–t paths within G'_i decrease the delta value. Paths from $s \in G'_i$ to t placed in this subphase have delta value at least $\delta_i/3$ by design. For paths from s placed in this subphase, s's neighbor v is the closest vertex to the perpendicular bisector of sv on the v side. If we translate v along the perpendicular bisector of sv to a distance of R_{i+1} from sv, this hypothetical point allows us to invoke Lemma 2 to get an underestimate for the delta value of all paths beginning with s. Therefore, our new underestimate is: $\delta_i^1 = \min\{\delta_i/3, \sqrt{R_{i+1}^2 + \epsilon_i^2} - R_{i+1}\}$.

Subphase 2 We now finish embedding each cycle $C = (v, x_1, \ldots x_m)$. Let the value $\alpha = \min\{\beta_i^1/3,\ \delta_i^1/(3R_{i+1})\}$, s.t. $\sin\alpha \leq \frac{\epsilon_i(1-\cos\frac{2}{3}\beta_i^1)}{2(1+\epsilon_i)}$. Trace out an arc of length $R_{i+1}\alpha$ from the embedding of x_1, clockwise along semi-circle $i+1$. We evenly divide this arc into $2n+1$ positions, numbered 0 to $2n$. Position 0 is already filled by x_1. We embed vertices in clockwise order around the arc in $m-1$ distinct positions; reserving position n for C's turnpike. If there is no such node, position n remains empty.

This completes the embedding of G'_{i+1}. We show that the embedding of G'_{i+1} is greedy. We only need to consider distance decreasing paths that involve a vertex placed during this subphase. For t placed during this subphase, t is within distance $\delta_i^1/3$ from an x_1, therefore, all previously placed $s \neq x_1$ have a neighbor u that is closer to t. If

[1] For the case where C is a 2-cycle and x_1 is a turnpike we insert a temporary placeholder vertex p into C with edges to v and x_1, and treat p as the new x_1. We can later remove this placeholder by the triangle inequality.

$s = x_1$ the s's neighbor closer to t is x_2. Finally, for s placed during this subphase, let the cycle that s is on be $C = (v, x_1, \ldots, x_m)$. For $s = x_i \neq x_m$, since $\alpha \leq \beta_i^1/3$, the interior of the sector formed by x_1, x_m and the origin is empty, therefore t is either on the x_{i-1} side of the perpendicular bisector to $x_{i-1}x_i$ or on the x_{i+1} side of the perpendicular bisector to $x_i x_{i+1}$. If $s = x_m$ If t is embedded to the left s, the closer neighbor is x_{m-1}. Otherwise, applying Lemma 1, our choice of $\sin\alpha \leq \frac{\epsilon_i(1-\cos\frac{2}{3}\beta_i^1)}{2(1+\epsilon_i)}$ forces the perpendicular bisector to sv to have s on one side, and all nodes to the right of s on the other side. All cases are considered, so the embedding of G'_{i+1} is greedy.

To complete the inductive proof, we must compute δ_{i+1} and β_{i+1}. Trivially, $\beta_{i+1} = \frac{\alpha}{2n} \leq \beta(G'_{i+1})$. Distance decreasing paths between vertices placed before this subphase will not update the delta value. Therefore, we only evaluate paths with s or t embedded during this subphase. By design, paths from s previously placed to t placed during this subphase have a delta value $\geq \delta_i^1/3$. Distance-decreasing paths from s placed in this subphase to $t \in G'_{i+1}$ take two different directions. If s's neighbor u which is closer to t is on semi-circle $i+1$ then points that are closest to the perpendicular bisector to su are along the perimeter of the sector formed by s, u, and the origin. The point closest to the perpendicular bisector is where the first semi-circle intersects the sector. We translate this point down $R_{i+1} + 2$ units along the perpendicular bisector, and we have an underestimate for the delta value for any path beginning with a left/right edge. If s's neighbor that is closer to t is on the ith semi-circle, then a down edge is followed. To finish, we evaluate down edges su added during the second subphase. The closest vertex to the perpendicular bisector to su on the u side is either u, or the vertex placed in the next clockwise position the $i+1$th semi-circle. Translating this point $2R_{i+1}$ units away from su along the perpendicular bisector gives us the an underestimate for paths beginning with su.

This completes the proof for the greedy embedding of G'. To obtain a greedy embedding for G, we repeatedly collapse dummy edges in G' until we get G. When we collapse an edge (p, x_1), where p is the primary node for the 2-cycle, we collapse the edge to vertex p in the embedding. Collapsing in the other direction may break distance decreasing paths through p and neighbors of p embedded on the same semi-circle as p. After collapsing all such dummy edges, we have a greedy embedding of G.

We call the levels where the on-ramps to heavy paths are embedded *super levels*, and all other levels are *baby levels*. There are $n-1$ baby levels between consecutive super levels and, since any path from root to leaf in a depth tree travels through $O(\log n)$ different heavy paths, there are $O(\log n)$ super levels.

Our Coordinate System. Let v be a vertex in G. We define $\mathrm{level}(v)$ to be the number of baby levels between v and the previous super level (zero if v is on a super level) and $\mathrm{cycle}(v)$ to be the position, 0 to $2n$, where v is placed when its cycle is embedded. These values can be assigned to vertices without performing the embedding procedure.

Let s be v's ancestor on the first super level. The path from s to v passes through $O(\log(n))$ heavy paths, entering each heavy path at an on-ramp, and leaving at an off-ramp. We define v's coordinate to be a $O(\log n)$-tuple consisting of the collection of $(\mathrm{level}(\cdot), \mathrm{cycle}(\cdot))$ pairs for each off-ramp where a change in heavy paths occurs on the path from s to v, and the pair $(\mathrm{level}(v), \mathrm{cycle}(v))$, which is either an off-ramp or a turnpike. Using the coordinate for v and the parameter n, we can compute the Euclidean

coordinates for all the turnpikes and off-ramps on the path from s to v, including the coordinate for v. Thus, we have defined a coordinate system for the Euclidean plane.

Using a straightforward encoding scheme, each level-cycle pair is encoded using $O(\log n)$ bits. Since a coordinate contains $O(\log n)$ of these pairs, we encode each coordinate using $O(\log^2 n)$ bits.

Greedy Routing with Coordinate Representations. By design, the routing scheme discussed in Sect. 3 is greedy for our embedding. We develop a comparison rule using the potential number of edges that may be traversed on a specific path from s to t.

Let s_i be the vertex between super levels i and $i+1$, whose level-cycle pair is in position i of s's coordinate. We define t_i similarly. Let $\mathrm{superlevel}(s)$ be the position that contains the level-cycle pair for s itself. Let h be the smallest integer such that s_h and t_h differ. Using the level-cycle pairs for s_h and t_h, we can compute the level-cycle pair for the off-ramps on the least common ancestor C that diverge toward s and t, which we call s_C and t_C. That is, if $\mathrm{level}(s_h) = \mathrm{level}(t_h)$ then $s_C = s_h$ and $t_C = t_h$. Otherwise, assume without loss of generality that $\mathrm{level}(s_h) < \mathrm{level}(t_h)$, then s_C's pair is $(\mathrm{level}(s_h), \mathrm{cycle}(s_h))$ and t_C is a turnpike with the pair $(\mathrm{level}(s_h), n)$.

We define l, r, d, u be the potential number of left, right, down, and up edges that may be traversed from s to t. Values d and u are simply the number of semi-circles passed through by down and up hops, respectively. That is,

$$d = (\mathrm{superlevel}(s) \cdot n + \mathrm{level}(s)) - (hn + \mathrm{level}(s_C))$$
$$u = (\mathrm{superlevel}(t) \cdot n + \mathrm{level}(t)) - (hn + \mathrm{level}(t_C)).$$

If $\mathrm{cycle}(t_C) < \mathrm{cycle}(s_C)$, then we count the maximum number left edges on the path from s to t_C, and the maximum number of right edges from t_C to t. That is,

$$l = \begin{cases} \mathrm{cycle}(s) + 2n(d-1) + \mathrm{cycle}(s_C) - \mathrm{cycle}(t_C) & \text{if } s \neq s_C, \\ \mathrm{cycle}(s_C) - \mathrm{cycle}(t_C) & \text{if } s = s_C. \end{cases}$$

$$r = \begin{cases} 2n(u-1) + \mathrm{cycle}(t) & \text{if } t \neq t_C, \\ 0 & \text{if } t = t_C. \end{cases}$$

If $\mathrm{cycle}(t_C) \geq \mathrm{cycle}(s_C)$, then we count the maximum number of right edges on the path from s to t_C, and the maximum number of right edges from t_C to t. That is,

$$l = 0$$
$$r = r_1 + r_2, \text{ where}$$
$$r_1 = \begin{cases} 2n - \mathrm{cycle}(s) + 2n(d-1) + \mathrm{cycle}(t_C) - \mathrm{cycle}(s_C) & \text{if } s \neq s_C, \\ \mathrm{cycle}(t_C) - \mathrm{cycle}(s_C) & \text{if } s = s_C. \end{cases}$$

$$r_2 = \begin{cases} 2n(u-1) + \mathrm{cycle}(t) & \text{if } t \neq t_C, \\ 0 & \text{if } t = t_C. \end{cases}$$

Our comparison rule is:

$$D(s,t) = l + r + (2n+1)u + d.$$

Following the routing scheme from Sect. 3, any move we make toward the goal will decrease $D(\cdot,\cdot)$, and any move away from the goal will increase $D(\cdot,\cdot)$. Therefore, we can use this comparison rule to perform greedy routing in our embedding efficiently, and comparisons made while routing will evaluate consistently with the corresponding Euclidean coordinates under the L_2 metric.

5 An Optimal Succinct Greedy Embedding

Conceptually, the $\mathrm{level}(\cdot)$ and $\mathrm{cycle}(\cdot)$ values used in the previous section are encoded as integers whose binary representation corresponds to a path from root to a leaf in a full binary tree with n leaves. Instead of encoding with a static $O(\log n)$ bits per integer, we will modify our embedding procedure so we can further exploit the heavy path decomposition of the dual tree T, using *weight-balanced binary trees* [4].

Definition 4. *A **weight-balanced binary tree** is a binary tree which stores weighted items from a total order in its leaves. If item i has weight w_i, and all items have a combined weight of W then item i is stored at depth $O(\log W/w_i)$. An inorder listing of the leaves outputs the items in order.*

Encoding the Level Values. Suppose we have a depth tree T for G, and a heavy path decomposition of T. Let C be a simple cycle in G on some heavy path H and let C_{next} be the next cycle on the heavy path H, if it exists. Let $n(C)$ be the number of vertex descendants of C in G. We define a weight function $\gamma(\cdot)$ on the cycles in G as follows:

$$\gamma(C) = \begin{cases} n(C) & \text{if } C = \mathrm{tail}(H), \\ n(C) - n(C_{\mathrm{next}}) & \text{if } C \neq \mathrm{tail}(H). \end{cases}$$

For each heavy path H, create a weight-balanced binary tree B_H containing each cycle C in H as an item with weight $\gamma(C)$, and impose a total order so that cycles are in their path order from $\mathrm{head}(H)$ to $\mathrm{tail}(H)$.

Let v be a vertex whose coordinate we wish to encode, and suppose v is located between super levels l and $l+1$. Let v_i be the vertex whose level-cycle pair is in position i of v's coordinate. Let v_i be contained in cycle C_i (such that v_i is not C_i's primary node) on heavy path H_i. The code for $\mathrm{level}(v_i)$ is a bit-string representing the path from root to the leaf for C_i in the weight-balanced binary tree B_{H_i}. Let W_i be the combined weight of the items in B_{H_i}. Since C_i is at a depth of $O(\log W_i/\gamma(C_i))$, this is length of the code. Thus, the level values in v's coordinate are encoded with $O(\sum_{0\leq i\leq l} \log W_i/\gamma(C_i))$ bits total. By design, this sum telescopes to $O(\log n)$ bits.

Encoding the Cycle Values. For a node v in G we define a weight function $\mu(v)$ to be the number of descendants of v in G.

Let $C = (p, x_1, x_2, \ldots, x_m)$ be a cycle in G, where p is the primary node of C. Let x_h be the turnpike that connects C to the next cycle on the heavy path, if it exists. Let x_i have weight $\mu(x_i)$ and impose a total order so $x_j < x_k$ if $j < k$. For each cycle C, we create a weight-balanced binary tree B_C containing nodes x_1 to x_m as follows. We first create two weight-balanced binary trees B_C^1 and B_C^2 where B_C^1 contains x_j

for $j < h$ and B_C^2 contains x_k for $k > h$. If no such x_h exists, then choose an integer $1 \leq k \leq m$ and insert items x_j for $j < k$ into B_C^1 and insert the remaining items into B_C^2. We form our single weight-balanced binary tree B_C in two steps: (1) create a tree B_C^3 with B_C^1 as a left subtree and a node for x_h as a right subtree, and (2) form B_C with B_C^3 as a left subtree and B_C^2 as a right subtree. We build B_C in this way to ensure that every turnpike is given the same path within its tree, and hence the same cycle code and value.

The code for $\mathrm{cycle}(v_i)$ is a bit-string representing the path from root to the leaf for v_i in the weight-balanced binary tree B_{C_i}. Let W_i be the combined weight of the items in B_{C_i}. Since v_i is at a depth of $O(\log W_i/\mu(v_i))$, this is length of the code. Thus, the cycle values in v's coordinate are encoded with $O(\sum_{0 \leq i \leq l} \log W_i/\mu(v_i))$ bits total. By design, this sum telescopes to $O(\log n)$ bits.

Interpreting the Codes. Let c be the smallest integer constant such that item i stored in the weight-balanced binary tree is at depth $\leq c \log W/w_i$. We can treat the position of i in the weight-balanced binary tree as a position in a full binary tree of height $c \log n$. We interpret this code to be the number of tree nodes preceding i in an in-order traversal of the full binary tree. Using our codes as described, we require $2n^c - 2$ baby levels between each super level and $8n^c - 1$ cycle positions.

An Overview of the Optimal Embedding. Let T be the depth tree for our Christmas cactus graph G. We create weight-balanced binary trees on the heavy paths in T and on each of the cycles in G, giving us the level and cycle codes for every vertex. We adjust the graph modification procedure so that adjacent cycles on heavy paths are spaced out according to the level codes. That is, adjacent cycles on the same heavy path have heavy dummy edges (dummy edges that are considered to be on the heavy path) inserted between them so that they are placed on the appropriate baby levels. For cycles on different heavy paths, we insert dummy edges to pad out to the next superlevel, and heavy dummy edges to pad out to the appropriate baby level.

We embed the modified graph analogously to our $O(\log^2 n)$ embedding, except that the cycle codes dictate vertex placements. We augment our coordinate system to store the level value for elements on the root cycle, otherwise it is not possible to compute the corresponding Euclidean point from our succinct representation. The same comparison rule applies to our new coordinate system, with little change to account for the new range of cycle values. Using this embedding scheme and coordinate system we achieve optimal $O(\log n)$ bits per coordinate.

References

1. Angelini, P., Frati, F., Grilli, L.: An algorithm to construct greedy drawings of triangulations. In: Tollis, I.G., Patrignani, M. (eds.) GD 2008. LNCS, vol. 5417, pp. 26–37. Springer, Heidelberg (2009)
2. Dhandapani, R.: Greedy drawings of triangulations. In: Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms (SODA 2008), pp. 102–111. SIAM, Philadelphia (2008)
3. Eppstein, D., Goodrich, M.T.: Succinct greedy graph drawing in the hyperbolic plane. In: Tollis, I.G., Patrignani, M. (eds.) GD 2008. LNCS, vol. 5417, pp. 14–25. Springer, Heidelberg (2009)

4. Knuth, D.E.: Optimum binary search trees. Acta Informatica 1, 14–25 (1971)
5. Leighton, T., Moitra, A.: Some results on greedy embeddings in metric spaces. In: Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008), pp. 337–346. IEEE Press, Los Alamitos (2008)
6. Lillis, K.M., Pemmaraju, S.V.: On the efficiency of a local iterative algorithm to compute delaunay realizations. In: McGeoch, C.C. (ed.) WEA 2008. LNCS, vol. 5038, pp. 69–86. Springer, Heidelberg (2008)
7. Muhammad, R.B.: A distributed geometric routing algorithm for ad hoc wireless networks. In: Proceedings of the 4th International Conference on Information Technology (ITNG 2007), pp. 961–963. IEEE Press, Los Alamitos (2007)
8. Papadimitriou, C.H., Ratajczak, D.: On a conjecture related to geometric routing. Theoretical Computer Science 344(1), 3–14 (2005)
9. Sleator, D.D., Tarjan, R.E.: A data structure for dynamic trees. Journal of Computer and System Sciences 26(3), 362–391 (1983)