Distributed Routing in Tree Networks with Few Landmarks

Ioannis Z. Emiris[1], Euripides Markou[1], and Aris Pagourtzis[2]

[1] Dept. Informatics & Telecoms, National University of Athens, 15784 Greece,
{emiris,emarkou}@di.uoa.gr
Research partly supported by PYTHAGORAS, project 70/3/7392 under the EPEAEK program funded by the Greek Ministry of Educational Affairs and EU.
[2] School of Elec. and Comp. Eng., National Technical University of Athens, 15780 Greece, pagour@cs.ntua.gr
Research partly supported by the General Secretariat of Research and Technology of Greece through PENED 2003 programme, contract nr. 03ED285, co-funded by the European Social Fund (75%) and national resources (25%).

Abstract. We consider the problem of finding a short path between any two nodes of a network when no global information is available, nor any oracle to help in routing. A mobile agent, situated in a starting node, has to walk to a target node traversing a path of minimum length. All information about adjacencies is distributed to certain nodes called landmarks. We wish to minimize the total memory requirements as well as keep the memory requirements per landmark to reasonable levels. We propose a landmark selection and information distribution scheme with overall memory requirement linear in the number of nodes, and constant memory consumption per non-landmark node. We prove that a navigation algorithm using this scheme attains a constant stretch factor overhead in tree topologies, compared to an optimal landmark-based routing algorithm that obeys certain restrictions. The flexibility of our approach allows for various trade-offs, such as between the number of landmarks and the size of the region assigned to each landmark.

1 Introduction

Our motivation is to model navigation in unknown or time-dependent environments, and to face questions of robustness in network routing by avoiding to store global information in a single node, or in some external oracle. Our work should find application in distributed computation and in sensor networks.

Efficient routing in a network has been studied extensively under several scenarios with respect to the kind and amount of available topology information. Several models have been proposed in order to design efficient routing schemes with relatively low memory requirements; cf. [15].

Hierarchical routing has been proposed in [20] where the problem of efficiency-memory tradeoffs for routing schemes was first raised. The authors proposed the general approach of hierarchically clustering a network into k levels and using the resulting structure for routing. The total memory used is $O(n^{1+1/k}\log n)$ while bounds were derived on the increase of average path length due to the reduction of routing information. However, in order to apply the method of [20], one needs to make some fairly strong assumptions regarding the existence of a certain partition of the network. Several variations and improvements were studied later [21, 25]. In [28], a landmark hierarchy has been proposed where the nodes inside the radius of some landmarks have routing information of how to send a message there. The authors give some results comparing average cases for space requirements and stretch factors. A hierarchy within a geometric setting has been proposed in [7]. In [18], the problem of minimizing the number of landmarks was studied so that the distances to the landmarks uniquely determine a mobile agent's position.

Another approach stores limited information about the network in every node. Explicit and implicit algorithms have been proposed [26, 10–12]. In the former kind, names and labels are arbitrary and some detailed routing information for all destinations is maintained per node. In implicit solutions [13, 27], names and labels are assigned according to a scheme so that the information implicit in the labelling can be used to choose the neighbor to which a message should be sent. Results about name-independent routing schemes are found in [4, 23, 3].

Lower bounds for the space-efficiency tradeoff of routing schemes were studied in [24, 8, 9, 16, 6, 22, 14]. In particular, in [24], it has been shown that no routing strategy can guarantee for every graph a routing scheme with a stretch factor $O(k)$ and $o(n^{1+1/k})$ bits of total memory. Other lower and upper bound results for space-stretch tradeoff can be found in [1, 2].

A related problem is the k-center problem, where one has to select k centers and to partition the nodes among them so that no more than L nodes be assigned per center, and the distance between a node and its center be minimized. There exist constant-ratio approximation algorithms for this problem, e.g. in the works cited below. Although this literature is very rich, we do not apply their algorithms in a black-box manner and, instead, we propose our own distribution scheme. The reason is that such algorithms either do not put a limit to the region size [17], or, whenever they do, as in the capacitated $k-$center problem [5, 19], they do not require that every region be connected; thus, a path from a node to its center can cross into another region. However, these two requirements are essential in reducing memory load. Our scheme on trees guarantees both. Another reason for preferring the term "landmark" to "center" is that the former suggests that landmarks are visible within their region and beyond. With tree networks this is true because there is a natural direction (namely, upwards) so as to reach a landmark.

Formally, given parameter L, we define certain nodes as landmarks and partition the nodes to corresponding regions so that no more than L nodes are assigned per landmark. A mobile agent MA with limited storage and computing capabilities, follows a simple algorithm using information stored at nodes it traverses in order to reach quickly a target node. One faces three simultaneous (and contradictory) tasks while distributing information about the graph:

– only a limited number of landmarks should contain information, while every other node may know only its label,
– the total memory needed should be minimized,
– the stretch factor (travelled / optimal distance) of the routing scheme should be minimized.

In this paper we propose Tree-Landmarks Routing (TLR), which is a distributed routing scheme appropriate for tree networks the nodes of which have labels assigned in a DFS manner. We store additional information (apart from node labels) only to a certain number of nodes. We also assume that every node knows the port leading to its parent. We prove the efficiency of our approach, namely that it stores a linear total amount of information to landmarks. We also show that TLR yields a constant overhead on the stretch factor, compared to that of an optimal landmark-based routing scheme that obeys the same upper and lower bounds on the size of regions. Our algorithm is flexible thus allowing for various trade-offs: most importantly between the number of landmarks and the region size. In particular we prove that, given an upper bound L on the region size and a parameter, TLR partitions a given tree network to $O(\sqrt{\frac{\delta}{\alpha}}\frac{n}{\sqrt{L}})$ regions of size at most αL, where δ is the fan-out of the tree, for any $\alpha \geq 1$. We expect that our approach generalizes to arbitrary graphs; we offer indications of this generalization.

The rest of the paper is organized as follows. The next section formalizes the problem and introduces all necessary notions. Section 3 focuses on tree networks, describes our routing scheme and proves its properties. Section 4 shows that our scheme achieves a low stretch factor and constant stretch overhead compared to routing schemes that follow similar assumptions; a second overhead measuring model is also examined. We conclude in Section 5 by suggesting possible extensions of our approach to arbitrary graphs; we also state some open questions.

2 Model and Preliminaries

Let us describe the general framework before concentrating on trees. A mobile agent MA, situated in a starting node s of a graph $G = (V, E)$, has to walk to a target node t of G traversing a minimum number of edges (or a walk of a minimum weight). The goal is to distribute information about the graph topology to a small subset of nodes so that MA can find quickly its way towards any target node.

More specifically, in a precomputation phase, among the n nodes of G, k of them are selected as *landmarks* and every node v of the graph is associated with exactly one landmark $m = lm(v)$ so that the following property is respected: The selection of landmarks is such that no landmark has more than L associated nodes. After the selection of landmarks and the association of nodes to landmarks, we distribute the information about the graph as follows.

First, in every node v of G, the next node in the path $\langle v \rightarrow lm(v)\rangle$ is stored (for the case of trees, only the port label leading to the parent of a node is necessary). Notice that the nodes have to be assigned to landmarks in such a way that, going from the node to its landmark only nodes assigned to the same landmark are encountered. Otherwise the above limited information could lead to unwanted delays, or even worse, could make MA enter a loop, thus preventing her from reaching the landmark.

Second, in every landmark m, there is a table containing information about the path $\langle m \rightarrow u\rangle$, for every node u such that $m = lm(u)$. These nodes constitute the *region* of landmark m, denoted as $region(m)$. In addition, a landmark may contain information about paths leading to some other landmarks together with indication about which nodes belong to the regions of those landmarks.

We say that two landmarks m_1, m_2 are *adjacent* or *neighbors* if there are adjacent nodes u, v so that: $m_1 = lm(u)$, $m_2 = lm(v)$.

We will mainly deal with rooted trees, so let us introduce some appropriate notation. For two nodes u, v in a rooted tree T, their *least common ancestor* $lca(u, v)$ is defined in the usual way, as the common ancestor of u and v that has the greatest depth in T. We define the *least common landmark* of u and v, denoted by $lclm(u, v)$, as the common ancestor of u and v that is a landmark and that has the greatest depth in T. It can be shown that $lclm(u, v) = lm(lca(u, v))$.

The agent initially knows only s, t. A routing from s to t could be the following: MA follows a walk from s to $lm(s)$, then from $lm(s)$ to $lm(t)$ and, finally, from $lm(t)$ to t. In the first phase MA reaches $lm(s)$ using only ports leading from nodes to their parents. In the second phase it reaches $lm(t)$ using only information stored in landmarks. In the third phase it only uses information stored in $lm(t)$ to find t. For the second-phase routing it makes sense to define a hierarchy of landmarks, or use of an implicit hierarchy, as in trees.

We assume that the labels have been assigned to nodes in a DFS manner and that each node knows its label and the port leading to its parent. We also assume that MA has limited memory and computing capabilities.

We would like to have a simple and efficient routing scheme, at the same time respecting bounds on the allowed memory load of landmarks and simple nodes. We measure the efficiency of our method by means of the stretch factor and the stretch overhead. Below we define these notions formally.

Let $w_A(s, t)$ be the walk from s to t followed by a routing scheme A and let $p(s, t)$ be the shortest path from s to t. The *stretch factor* $\mathcal{SF}_{A,G}$ of routing scheme A on graph G is the maximum ratio between $|w_A(s, t)|$ (the length of $w_A(s, t)$) and $|p(s, t)|$ taken over all pairs (s, t) of nodes of G.

We also use another natural measure of performance which we call *stretch overhead* of routing scheme A on graph G and denote it as $\mathcal{H}_{A,G}$: It is the maximum ratio between $|w_A(s, t)|$ and $|w_{A^*}(s, t)|$, taken over all pairs (s, t) and routing schemes A^* that follow similar restrictions as A. In particular, we are interested in routing schemes which also use landmarks, obey the same bounds on region size as A, and follow the same routing scenario, that is, MA travels from s to t by first traveling from s to $lm(s)$, then from $lm(s)$ to $lm(t)$, and finally from $lm(t)$ to t. Note that the stretch overhead can also be defined as the maximum ratio between the stretch factors $\mathcal{SF}_{A,G}$ and $\mathcal{SF}_{A^*,G}$, taken over all similarly restricted routing schemes A^*.

3 Distributed routing in arbitrary trees

In this section we present a distributed routing scheme, which we call Tree-Landmarks Routing (TLR). This scheme consists of three algorithms: an algorithm for selecting landmarks in arbitrary trees, an algorithm for distributing routing information to the nodes, and an agent navigation algorithm.

3.1 Landmark selection algorithm

Given a tree T and an upper bound L on the region size, the algorithm presented below selects landmarks in T and assigns regions of T to the landmarks, in such a way that the size of each region is between $L' = \sqrt{L/\delta}$ and L, where δ is the maximum fan-out of tree T. Consequently, the number of landmarks is at most n/L', that is, at most $\sqrt{\delta L}$ times the best possible (n/L) with respect to the upper bound L.

```
**Algorithm Landmark-Selection(tree $T$, region size $L$)**
select a root $r_0$ for $T$ arbitrarily
set $R := \{r_0\}$
set $L' := \lfloor\sqrt{\frac{L}{\delta}}\rfloor$ (* the choice of $L'$ will be justified in Prop. 1; we assume $L > \delta$ *)
  (* landmark selection – assignment of regions to landmarks *)
while $R \neq \emptyset$ do
    pick an element $r$ from $R$
    add $r$ to the set of landmarks and remove it from $R$
    traverse subtree $T(r)$ rooted at $r$ in BFS manner
    until $L'$ nodes have been visited or there are no nodes left in $T(r)$
    assign visited nodes of $T(r)$ to $region(r)$
    add unvisited nodes that are adjacent to $region(r)$ to $R$
end while
  (* merging of non-full regions with parent regions *)
for each $r$ such that $|region(r)| < L'$ do
    let $r'$ be the landmark of $parent(r)$        (* $parent(r) \in region(r')$ *)
    assign all nodes in $region(r)$ to $r'$
    remove $r$ from the set of landmarks
end for
```

The algorithm builds regions of size L' starting from the root of the tree and traversing the tree in a BFS manner. When all nodes have been visited, the algorithm assigns each region of size smaller than L' to the landmark of its parent region. We will show in Prop. 1 that the selection of L' guarantees that the size of any augmented region does not exceed L. We will also give an upper bound on the number of landmarks selected by our algorithm. We first state a simple lemma.

Lemma 1. *The region of nodes assigned to a landmark r by algorithm Landmark-Selection forms a subtree rooted at r.*

Proposition 1. *Algorithm Landmark-Selection partitions a tree into regions of size at most L. The total number of regions is $O(\sqrt{\delta}\frac{n}{\sqrt{L}})$, where δ is the maximum fan-out of tree T.*

Proof. Let the *border* $b(S)$ of a region S be the set of nodes in this region that have children outside the region. Due to the fact that regions are formed by using BFS traversal, the border of S may contain: a) leaves of S and b) at most one internal node of S, namely the parent of the leaf that was included in S last.

A region S of size L' can be augmented by small (i.e. of size $< L'$) regions rooted at nodes that are adjacent to nodes in the border of S. We call these regions *child regions* of S and their landmarks *child landmarks* of S. Let us now give an upper bound on the number of child landmarks.

Clearly, the number of child landmarks of S can be at most $|b(S)|\delta$. We distinguish between two cases:

- The root of S does not belong to $b(S)$. In this case the number of child landmarks of S is at most $(L'-1)\delta$.
- The root of S belongs to $b(S)$ (note that this can only happen if $L' \leq \delta$). In this case S is a tree of height 1, consisting of its root and $L'-1$ leaves. Therefore, the root may have at most $\delta-(L'-1)$ children outside S; together with the children of leaves of S we get an upper bound of $(L'-1)\delta + (\delta - (L'-1)) = L'(\delta-1)+1$ on the number of child landmarks of S.

The above give an unconditional upper bound of $L'\delta - \min(L'-1,\delta) \leq L'\delta$ on the number of child landmarks of S.

An upper bound on the size of S' (S after augmentation) is given by

$$\begin{aligned} |S'| &\leq (\#\text{ child landmarks of } S)(L'-1) + L' \\ &\leq L'^2\delta - L'(\delta-1) \leq L'^2\delta \leq L \end{aligned}$$

where the last inequality holds because $L' = \lfloor\sqrt{L/\delta}\rfloor$.

Note that $L' \geq (\sqrt{\frac{L}{\delta}})/2$ since $L > \delta$. Therefore, the number of regions (also of landmarks) is at most $\frac{n}{L'} \leq \frac{2n}{\sqrt{\frac{L}{\delta}}} = O(\sqrt{\delta}\frac{n}{\sqrt{L}})$.

Region size relaxation. We can relax the maximum size of a region by means of a parameter α; in particular, for any $\alpha > 1$ we may specify αL as the maximum region size. Then, the upper bound on the number of landmarks will become $\sqrt{\frac{\delta}{\alpha}}\frac{n}{\sqrt{L}}$, that is, it will be reduced by $\sqrt{\alpha}$. Notice also that by setting $\alpha = L\delta$

we can guarantee an optimal (with respect to an 'ideal' region size L) number of landmarks n/L, at a cost of allowing the size of a region to increase up to δL^2.
Remark: We have assumed that $L > \delta$. For cases in which $L \leq \delta$ it is not clear what would be a reasonable landmark selection strategy. For example, consider a star with n nodes. For $L < n$, it turns out that there would be at most one region with more than one nodes (the one containing the center), and several regions consisting of a single landmark (recall that we insist in keeping regions connected). Therefore, the number of landmarks would be $\Omega(n)$, which would lead to poor memory efficiency (not better than that of any standard routing method).

3.2 Memory Requirements

This section details the routing information assignment scheme. The information distribution algorithm is as follows:

Starting from the root, traverse the tree in a DFS manner and assign labels to nodes. Also, assign port labels to edges: for every node u assign port labels $1, \ldots, d(u)$ to its outgoing edges, where $d(u)$ is the degree of u. In particular, assign port label 1 to the edge leading from u to its parent.

Store the information of the graph topology as follows:

– **Type 1 info (all nodes):** At every node u only its label has been stored; port 1 leads to $parent(u)$. Note that if u is not a landmark this port is the one that leads to the next node in the path $\langle u \rightarrow lm(u) \rangle$; if u is a landmark, this port leads to the next node in the path to its parent landmark.
– **Type 2 info (landmarks only):** At every landmark m, for each node v in the region of m store the label of v (for simplicity we will call it also v), the label of v's parent, and the port j which leads from $parent(v)$ to v. More precisely, the entry for v is $[v : parent(v), j]$; if v is a child of m then the entry is $[v : m, j]$.
– **Type 3 info (landmarks only):** At every landmark m, for any child landmark m' of m store $[m' : parent(m'), j \mid R_{m'}]$, where j is the port leading from $parent(m')$ to m', and $R_{m'}$ is the node with the greatest label among all nodes in the subtree rooted at m'.

In the next section we will describe how the above information can be used by an agent in order to efficiently find its way from any node s to any node t. We now show the space requirements of the above scheme in a real RAM model, where labels consume $O(1)$ space.

Proposition 2. *The above routing table assignment scheme requires $O(\delta L)$ space for each landmark. The entire space needed is $O(n)$.*

Proof. The space needed at a landmark is as follows:
- $O(1)$ for type 1 info, as above.
- $O(L)$ for type 2 info (since there are at most L nodes associated with a landmark and for each one a constant number of labels needs to be stored),

- $O(\delta L)$ for type 3 info (since there are at most δL child landmarks of a landmark m and for each of them a constant number of labels needs to be stored).

Finally, the total space needed for the information stored in the tree is $O(n)$ because:
- for type 1 info we need to store $O(n)$ port numbers in total,
- for type 2 info we need to store $O(|region(m)|)$ labels at each landmark m, which gives a total of $O(n)$ labels
- for type 3 info we need to store $O(\#\text{ child landmarks of } m)$ labels at each landmark m, which gives a total of $O(\#\text{ landmarks}) = O(\sqrt{\frac{\delta}{L}n})$ labels, by Proposition 1.

3.3 Navigation Algorithm

We now describe a navigation algorithm that MA can use to find its way from node s to node t. In the following we denote by u the current node where MA is located (initially $u = s$); we describe the appropriate action of MA, depending on the kind of u and the routing information obtained so far. We use a boolean variable *direction* which takes values 'up' or 'down', showing MA should move upwards or downwards. Initially *direction* is set to 'up'.

– **Case a.** If u is not a landmark and direction is 'up', then go to $parent(u)$.
– **Case b.** If u is not a landmark and direction is 'down', then follow the first port in the list of ports obtained in previous steps (see below), and remove this port from the list. (In this case MA moves towards destination t — if not already there — or towards a landmark m' containing t in its subtree.)
– **Case c.** If u is a landmark, look at type 2 info table in order to find t. If t is found, set *direction* to 'down', and use table information to construct a sequence $ports(u \to t)$ of ports leading from u to t. Store $ports(u \to t)$ to local (agent's) memory, and follow the first port in $ports(u \to t)$ (removing this port from the sequence).
– **Case d.** If u is a landmark but t is not found within type 2 info table, look at type 3 info table for entries $[m' : v, j \mid R_{m'}]$ such that $m' \le t \le R_{m'}$. If such an entry is found, then set *direction* to 'down', and use type 2 info table to construct a sequence $ports(u \to v)$ of ports leading from u to v; append port j to $ports(u \to v)$. Store $ports(u \to v) \mid j$ to local (agent's) memory, and follow $ports(u \to v) \mid j$ to eventually reach m'.
– **Case e.** If u is a landmark but t is not found in tables of type 2 or type 3 info then move towards the parent of u.

Below we show the correctness of TLR, in particular, that a mobile agent situated at node s will manage to reach node t following this scheme.

Proposition 3. *Given a graph G, a target node t and an agent MA situated at some node s of G, if G is preprocessed by using landmark selection and information distribution algorithms of TLR then MA will eventually reach t by using the Navigation Algorithm of TLR.*

Proof. We will show that the agent follows 4 particular walks in a row: $\langle s \rightarrow lm(s)\rangle$, $\langle lm(s) \rightarrow lclm(s,t)\rangle$, $\langle lclm(s,t) \rightarrow lm(t)\rangle$, and $\langle lm(t) \rightarrow t\rangle$. Note that some of these may be of zero length.

If s is a landmark then MA is already in $lm(s)$, otherwise it moves up (case a) until it reaches $lm(s)$.

If t is 'visible' from $lm(s)$ (that is, belongs to the subtree rooted at $lm(s)$) then MA is already in $lclm(s,t)$, otherwise it moves upwards (case e and then case a repeatedly, possibly repeating this pattern several times) until it reaches a landmark m such that t belongs to the subtree of m (at least one such landmark exists: the root). Clearly, $m = lclm(s,t)$.

If $t \in region(m)$ then MA is already in $lm(t)$. Otherwise, it starts moving down (case d and then case b repeatedly) to reach the child landmark of m that contains t in its subtree. This process is possibly repeated several times until MA reaches $lm(t)$.

It then follows the path implied by the routing information of $lm(t)$ (case c and then case b repeatedly) in order to reach t.

All the calculations needed by MA in order to determine each time the port by which it has to leave its current position can be done in time linear in the number of nodes.

4 Efficiency of TLR scheme

In this section we give estimations for the efficiency of Tree-Landmarks Routing (TLR) in terms of the stretch factor and the stretch overhead achieved.

4.1 Stretch factor analysis

In the following we establish a stretch factor for TLR in trees which is proportional to the region height.

Proposition 4. *TLR achieves stretch factor $\mathcal{SF}_{TLR,T} = 2h + 1$ on a tree T, where h is the maximum height of a region of T.*

Proof. Consider a pair of nodes s, t. The walk followed by MA under TLR scheme can be analyzed in the following paths (in this order): $\langle s \rightarrow lca(s,t)\rangle$, $\langle lca(s,t) \rightarrow lclm(s,t)\rangle$, $\langle lclm(s,t) \rightarrow lca(s,t)\rangle$, and $\langle lca(s,t) \rightarrow t\rangle$. Clearly, all these paths are shortest paths since they are simple. Therefore,

$$\begin{aligned}\mathcal{SF}_{TLR,T} &= \max_{s,t} \frac{|p(s, lca(s,t))| + 2|p(lca(s,t), lclm(s,t))| + |p(lca(s,t),t)|}{|p(s,t)|} \\ &\leq 1 + \max_{s,t} \frac{2|p(lca(s,t), lclm(s,t))|}{|p(s,t)|}\end{aligned}$$

because the shortest path $p(s,t)$ between s and t consists of shortest paths $p(s, lca(s,t))$ and $p(lca(s,t),t)$.

The distance $|p(lca(s,t), lclm(s,t))|$ is at most h, since $lclm(s,t)$ is the landmark of $lca(s,t)$. Therefore, $\mathcal{SF}_{TLR,T} \leq 2h+1$. On the other hand, there is a case in which $|p(lca(s,t), lclm(s,t))| = 2h$ and $|p(s,t)| = 1$. Namely, whenever s is parent of t (therefore $s = lca(s,t)$), and s and t are in different regions (t is a child landmark of $lm(s)$).

Taking into account that h is bounded by the maximum region size we obtain the following.

Corollary 1. *Tree-Landmarks Routing achieves a stretch factor of $O(L)$.*

4.2 Stretch overhead with upper and lower bounds on region size

Next we establish a constant stretch overhead for TLR in trees where all regions have size between L' and L. This restriction is necessary in order to have a fair comparison to an optimal routing scheme. Notice that, for any routing scheme under consideration, since each region size is between L' and L, the number of landmarks is between n/L and n/L'. We first prove the result for line graphs and full δ-ary trees and then sketch a proof for arbitrary trees.

Proposition 5. *TLR achieves stretch overhead 2 when applied to line graphs.*

Proof. TLR, when applied to a line, produces regions which are also lines. All these regions have size L', except possibly for one or two 'leaf' regions which are merged with their 'parent' regions. Therefore the maximum height of a region is at most $2L'$.

As mentioned above, any routing scheme A^* that we would like to compare to TLR will produce regions of size $\geq L'$. Clearly, the stretch factor for one of these regions alone will be $\geq L'$ (consider two neighbor nodes, as far from the landmark as possible). Therefore, the stretch overhead is at most 2.

We will now show that this is also the case for full (i.e. complete) δ-ary trees.

Proposition 6. *TLR achieves stretch overhead 2 when applied to full δ-ary trees.*

Proof. Let $h_\delta(r)$ denote the height of a complete δ-ary tree with r nodes; note that $h_\delta(r) = \Theta(\log_\delta r)$. During its last stage, Landmark-Selection augments a region R by combining it with regions that have the following properties: their height is smaller than $h_\delta(L')$ and their landmarks are adjacent to nodes of R that are in distance $h_\delta(L')$ or $h_\delta(L') - 1$ from the landmark of R. Therefore, the maximum height of a region after augmentation is at most $2h_\delta(L')$.

As mentioned above, A^* makes regions of size $\geq L'$. Clearly, such a region cannot have height smaller than $h_\delta(L')$, which implies that the stretch overhead of TLR is at most 2.

Next, we sketch how to extend this result to arbitrary trees.

Proposition 7. *TLR achieves stretch overhead* 4 *when applied to an arbitrary tree.*

Proof. Let h denote the maximum height of a region R of a tree T produced by Landmark-Selection algorithm in its first phase (that is, before merging). As argued in the proof of Prop. 6 the maximum region height at the end of Landmark-Selection will be at most $2h$. Consider any partitioning of T into connected regions of size at least L'; it can be shown that nodes of R cannot be accommodated into regions that all have height smaller than $\frac{h}{2}$. Therefore, there exists at least one region of height $\frac{h}{2}$, leading to a stretch overhead of 4.

4.3 Stretch overhead with upper bounds on region size and number of landmarks

We now consider a different definition of stretch overhead. Namely, we allow comparison with any routing scheme that respects the same upper bound L on the region size and the same upper bound on the number of landmarks n/L'.

Propositions 5 and 6 can be adapted to show that TLR achieves optimal (within a factor of 2) stretch overhead in the extreme cases of line graphs and full trees. This is because the bound on the number of landmarks guarantees that there must be at least one region of size $\geq L'$. However, this does not hold for trees that are not complete, as shown by the following example.

Example of $\Theta(\frac{L}{\delta \log_\delta^2 L}^{1/4})$ stretch overhead. Consider a graph consisting of a line segment S with L' nodes, followed by t complete δ-ary trees, $T_1, \ldots, T_t$, each with L' nodes; the root of T_1 is adjacent to a leaf of S, and the root of T_i, $2 \leq i \leq t$, is adjacent to a leaf of T_{i-1}. Algorithm Landmark-Selection will divide such a graph to $t+1$ regions, namely S and the T_i trees. Therefore, the stretch factor will be $\Theta(L')$ due to the height of S.

On the other hand, it is possible, using the same number of landmarks (i.e. $t+1$), to split S into t regions of height L'/t and construct one region containing all T_i's, provided that $tL' \leq L$. The maximum region height of the new partition is $h^* = \max(L'/t, t \log_\delta(L'))$; this is minimized for $t = \sqrt{L'/\log_\delta L'}$ giving $h^* = \sqrt{L' \log_\delta L'}$. Hence, the stretch overhead is equal to $L'/h^* = \sqrt{\frac{L'}{\log_\delta L'}} = \Theta(\frac{L}{\delta \log_\delta^2 L}^{1/4})$.

5 Conclusions

In this work we have proposed Tree-Landmarks Routing (TLR), which is a distributed routing scheme appropriate for tree networks. Given an upper bound L on region size, TLR partitions a tree T to $O(\sqrt{\delta}\frac{n}{\sqrt{L}})$ regions of size at most L, achieving a reasonable stretch factor and an optimal (within a constant) stretch overhead, while storing linear amount of information to landmarks and constant information per node to non-landmark nodes.

The optimal stretch overhead of TLR is achieved for the model with both upper and lower bounds on the size of regions. An interesting open question is whether we can devise a routing scheme with constant or even logarithmic stretch overhead for arbitrary trees under the model with upper-bounded region size and number of landmarks.

A second question is whether we can reduce the number of regions to e.g. $O(\frac{n}{L})$ without affecting the efficiency of the scheme. A possible modification of our landmark-selection algorithm toward this direction is the following: instead of merging regions of size $< L'$ only with parent regions, we do the same with all regions of size $< L$ as long as no region size becomes $> L$, in a bottom-up manner. This makes larger regions, resulting to fewer landmarks. In practice, we expect that for large trees most regions will be of size close to L and only a few regions will have size L'. However, to quantify these claims we would need extra hypotheses on the topology of our tree. On the downside, such a modification shall increase the stretch factor.

Another important issue is whether TLR can be extended to work for arbitrary graphs, and what the efficiency will be. A possible approach might be the following: given a graph G, specify a BFS spanning tree T for G, rooted at a minimum eccentricity node; then apply TLR to tree T. Clearly, this would lead to the same trade-off, as that for trees, between the region size and the number of landmarks and the same memory requirements. On the other hand, it seems that the stretch factor and overhead can become arbitrarily large. Therefore, a very interesting question is whether there exist routing schemes with low stretch overhead for arbitrary graphs, under any of the two overhead measuring models.

Acknowledgements. We thank Leonidas Guibas for introducing us to sensor-networks problems and related questions on distributed routing, thus leading to this paper. We also thank Christodoulos Fragoudakis, Valia Mitsou, and Elias Tsigaridas for useful discussions. Finally we thank the anonymous referees for their helpful suggestions.

References

1. I. Abraham, C. Gavoille, and D. Malkhi. On space-stretch trade-offs for compact routing schemes. Research Report RR-1374-05, LaBRI, France, November 2005.
2. I. Abraham, C. Gavoille, and D. Malkhi. Compact routing for graphs excluding a fixed minor. In Proc. 19th International Symposium on Distributed Computing (DISC), LNCS 3724 pp. 442–456, September 2005.
3. I. Abraham and D. Malkhi. Name Independent Routing for Growth Bounded Networks. In Proc. 17th ACM Symp. Parall. Algorithms and Architectures (SPAA '05), pp. 49–55, 2005.
4. B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg. Compact distributed data structures for adaptive network routing, Proc. of 21st ACM Symp. on Theory of Computing, pp. 230–240, May 1989.
5. J. Bar-Ilan, G. Kortsarz, and D. Peleg. How to allocate network centers. J. Algorithms 15 (1993), pp. 385–415.

6. H. Buhrman, J.-H. Hoepman, and P. Vitanyi. Optimal routing tables. In Proc. 15th ACM Symp. on Principles of Distributed Computing, pp. 134–142, May 1996.
7. Q. Fang, J. Gao, L. Guibas, V. de Silva, L. Zhang. GLIDER: Gradient Landmark-Based Distributed Routing for Sensor Networks. In Proc. 24th Conf. of IEEE Com. Soc. (INFOCOM'05), 2005.
8. P. Fraigniaud and C. Gavoille. Memory requirement for universal routing schemes. In Proc. 14th ACM Symp. on Principles of Distributed Computing, pp. 223–230, August 1995.
9. P. Fraigniaud and C. Gavoille. Local memory requirement of universal routing schemes. In Proc. 8th ACM Symp. on Parallel Algorithms and Architectures, pp. 183–188, June, 1996.
10. G.N. Frederickson, R. Janardan. Separator-Based Strategies for Efficient Message Routing. In Proc. 27th IEEE Symp. on Foundations of Computer Science, 1986 pp. 428–437.
11. G.N. Frederickson, R. Janardan. Designing networks with compact routing tables. Algorithmica 3: 171–190, 1988.
12. G.N. Frederickson, R. Janardan. Efficient message routing in planar networks. SIAM Journal on Computing 18: 843–857, 1989.
13. P. Fraigniaud, C. Gavoille. Routing in Trees. In Proc. 28th International Colloquium, ICALP 2001 Crete, Greece, LNCS 2076, 2001, pp. 757–772.
14. C. Gavoille and M. Gengler. Space-efficiency of routing schemes of stretch factor three. In Proc. 4th Int. Colloq. on Structural Information & Communication Complexity, pp. 162–175. Carleton Scientific, 1997.
15. C. Gavoille and D. Peleg. Compact and localized distributed data structures. Distributed Computing, Volume 16, Issue 2-3, Sep 2003, Pages 111–120.
16. C. Gavoille and S. Perennes. Memory requirement for routing in distributed networks. In Proc. 15th ACM Symp. on Principles of Distributed Computing, pp. 125–133, May 1996.
17. D. Hochbaum and D. B. Shmoys. A best possible heuristic for the $k-$center problem. Mathematics of Operations Research, Vol 10:180–184, 1985.
18. S. Khuller, B. Raghavachari, and A. Rosenfeld. Landmarks in Graphs. Discrete Applied Mathematics, Vol 70 (3), pp. 217–229 (1996).
19. S. Khuller and Y.J. Sussmann, The capacitated k-center problem. SIAM J Discrete Math 13 (2000), 403-418.
20. L. Kleinrock, F. Kamoun. Hierarchical routing for large networks: performance evaluation and optimization. Computer Networks 1: 155–174, 1977.
21. L. Kleinrock, F. Kamoun. Optimal clustering structures for hierarchical topological design of large computer networks. Computer Networks 10: 221–248, 1980.
22. E. Kranakis and D. Krizanc. Lower bounds for compact routing. In Proc. 13th Symp. on Theoretical Aspects of Computer Science, LNCS 1046, pp. 529–540, February 1996.
23. D. Peleg. Distance-dependent distributed directories. Information and Computation, pp. 270-298, 1993.
24. D. Peleg and E. Upfal. A tradeoff between size and efficiency for routing tables. J. ACM, 36:510–530, 1989.
25. R. Perlman. Hierarchical networks and the subnetwork partition problem. In Proc. 5th Conf. on System Sciences, 1982.
26. N. Santoro, R. Khatib. Labelling and implicit routing in networks. The Computer Journal 28: 5–8, 1985.

27. M. Thorup, U. Zwick. Compact routing schemes. In Proc. 13th annual ACM symposium on Parallel algorithms and architectures, pp. 1–10, 2001, Crete Island, Greece.
28. P.F. Tsuchiya. The landmark hierarchy: A new hierarchy for routing in very large networks. Computer Communication Review 18, 4 (August 1988), pp. 35–42.