Nordic Journal of Computing 14(2008), 202–215.

TRACKING MOBILE USERS IN CELLULAR NETWORKS USING TIMING INFORMATION*

EVANGELOS KRANAKIS†
School of Computer Science, Carleton University
Ottawa, ON, K1S 5B6, Canada
kranakis@scs.carleton.ca

DANNY KRIZANC
Department of Mathematics and Computer Science
Wesleyan University, Middletown CT 06459, USA
dkrizanc@wesleyan.edu

SUNIL SHENDE
Department of Computer Science,
Rutgers University, Camden, NJ, 08102, USA
shende@camden.rutgers.edu

Abstract. We consider the problem of tracking a mobile user moving through a cellular network using node queries that provide information as to last time the user visited the node. Queries are executed in rounds, each round may involve queries to many nodes simultaneously, and the user may move at the same time that queries are performed. The cost measures considered are the number of rounds required to find the user and the total number of queries made during the execution of the algorithm. We present a number of algorithms for general networks, as well as upper and lower bounds for specific network topologies.

ACM CCS Categories and Subject Descriptors: F.2.3 [Analysis of Algorithms and Problem Complexity]: Tradeoffs between Complexity Measures

Key words: cellular networks, query, round, timing information, tracking

1. Introduction

Cellular networks face the task of tracking mobile users (typically, cell-phone subscribers) traversing the nodes of a given network. To track a mobile user a central server queries in rounds many nodes (typically, base stations) of the network simultaneously until it locates the user. In each round any number of nodes can be queried and it is up to the tracking algorithm to optimize this search. The tracking problem has been considered extensively in the literature. (See the discussion of related work below.) For the most part, the solutions involve some compromise

*This is an expanded and revised version of a paper that appeared in the proceedings of SIROCCO 2003, Carleton Scientific, 2003, J. Sibeyn, ed., pages 223–234.

†Research supported in part by NSERC (Natural Sciences and Engineering Research Council of Canada) and MITACS (Mathematics of Information Technology and Complex Systems) grants.

Received December 2007; accepted February 2009.

between maintaining a central database of the precise location of all mobile users at any time or querying the nodes of the network whenever a user's location must be determined.

The algorithms that query the network assume that nodes provide only one bit of information, i.e., the presence or non-presence of the user at the location queried. In such a model, if no further information is provided, any algorithm that finds a user with certainty must eventually query all of the nodes that could conceivably contain the user. In this paper, we consider a new "query model" whereby the nodes maintain a list of users that they have serviced recently and when queried respond with the time that a user last visited the node. Under this assumption, we show query algorithms for user tracking which significantly improve upon the worst-case complexity of search strategies that use only the standard single bit response model. Note that we do not assume that other information concerning a user's behavior is available (e.g., the probability of being at a particular node) but such information could be combined with our techniques to obtain even better results.

1.1 The model

A cellular network is modeled by a synchronous network. This is a graph $G = (V, E)$: V is the set of cells or nodes, and E the set of edges between nodes. Let $n = |V|$ be the number of vertices, $m = |E|$ the number of edges, and d the maximum degree of any node. In addition, there is a global clock maintaining synchrony over the entire network. Originally (i.e., at time 0), the user occupies a certain node s, known to a central query algorithm. This could be, for instance, the node at which the user was last found during a search operation. At each time unit the user may either stay at its current node or move to an adjacent node. At a later time t, a *search algorithm* begins searching for the user by querying nodes of the network. The search algorithm has access to the whole network and can query any number of nodes at the same time. The algorithm succeeds when it queries the node that contains the user. In all instances the search ends when the user is found in any of the nodes queried. The network is synchronous: this means that the queries are executed simultaneously and the user, the nodes and the search algorithm use identical clocks. Our complexity measures are (1) the number of rounds r taken by the search algorithm to find the user and (2) the total number of queries q used in all rounds. Note that a round takes a unit of time while the user can make one unit move between two successive rounds.

An important point about the model is that the search algorithm may query any node of the graph, while, at the same time, the user is moving along vertices of the graph. In particular when, at a given time instance, the search algorithm has determined that the position of the user is a certain node u, at the next time step the search algorithm must query not only the node u but also all its neighbors in order to locate the user with certainty.

A query to a node is of the form: "When did the user last visit this node?" The answer to the query is an integer k indicating that the user last visited the node k time units ago. If the user has never visited the node this is indicated by some

special value, e.g., $k = -1$. A number of variations on this model are possible. If the node only returns 0 to indicate the user is not present or 1 to indicate the user is present we get the model most often studied in the literature. Models in between these two might be one in which the node is able to remember a user's last visit for limited amount of time or for a limited number of users that are not present. A model stronger than the one considered here might include information of the form, "the time the user was last seen as well as the cell the user went to upon leaving". Note that for constant degree networks, this model would be equivalent to the one considered here as one could always query a node and all of its neighbors with only constant overhead. We will consider the first model above and we refer to it as the *standard timed model*.

The timing information model as well as the global clock assumption employed in the paper is not inconsistent with usage in networks of currently established wireless phone providers. In addition, [5] assume a synchronous model with a global clock.

1.2 Contributions of the paper

We study the problem of designing search algorithms that minimize both the number of rounds required to find the user and the total number of queries used over all rounds. In Section 2, we present three upper bounds that work on arbitrary topologies. The efficiency of the first depends only on the degree of the graph, the second uses separator properties of the graph and the third depends on the cutwidth of the graph. These upper bounds can be adapted to get improved results for specific topologies. We consider lines, trees, meshes and cliques, as well as arbitrary planar topologies. Finally, in Section 3, we give lower bounds which show some of the upper bounds are tight for specific cases. The upper and lower bounds on the efficiency of tracking algorithms for different network topologies with n nodes and for the standard timed model are summarized in Table I. In all cases, the search algorithm begins search t time units after the last known position of the user and $t \leq n$, while for the lower bound in the case of arbitrary graphs we also assume that $t \leq \delta$. Note that n is the number of vertices, h the height of the tree, d is the max degree and δ is the diameter of the network.

Throughout the paper we use techniques from (planar) separators and cutwidth in order to show that simple algorithms provide tight bounds for the cases considered.

1.3 Related work

Searching is related to rendezvous search games (see [1]) and in recent years there has been relevant algorithmic work from the mobile agent community (e.g., see [7]) whereby search is accomplished with the aid of mobile agents in a network. A related version of our problem of tracking mobile users has been investigated by the graph theory community as a *cops and robbers* search game under various models (e.g., see [10] and [9]). Closely related is also the search number of a graph which goes back to the work of [16]: which is the minimum number of searchers needed to guarantee that a moving target will be captured by the searchers. (It is

T I: Upper and lower bounds on the number of queries as a function of the number r of rounds given that the search algorithm begins search t time units after the last known position of the user and $t \leq n$, for various topologies of size n, where d is the max degree of the network and h the height of the tree. Note that for a clique, n is upper and lower bound regardless of the number of rounds.

Graph	# of Rounds	Lower Bound	Upper Bound:
Clique	Any	n	n
Line	r	$\Omega(r(\min\{n,t\})^{1/r})$	$O(r(\min\{n,t\})^{1/r})$
Mesh	r	$\Omega(\min\{n,t\}^{\frac{2^r}{2^r-1}})$	$O(r(\min\{n,2t\})^{\frac{2^r}{2^r-1}})$
Tree	h	$d+1$	$O(hd)$
Planar	$O(\log n)$	$d+1$	$O(d\sqrt{n}\log n)$
Arbitrary	r	$\max\{d+1, rt^{1/r}\}$	$O(rd^{\lceil t/r\rceil+1})$

always possible to solve the problem by placing a searcher at every vertex of the graph–though inefficient). More recent is the work of [13] and [11] on various relevant search numbers. A survey of graph theoretic perspectives on searching and sweeping can be found in [2].

The problem of tracking users in cellular networks considered in the current paper originates in research of the wireless networks community. As discussed by [4] a key requirement for enabling location services is user location and tracking. [3] consider the problem of concurrent on-line tracking of mobile users for the purpose of providing/locating services, while [6] considers topology based tracking strategies. Managing the database of mobile user locations in a distributed manner for accomplishing an efficient location strategy is an important issue investigated in [20]. Another class of models assumes that the search is aided by knowledge of the probability distribution of the location of the mobile in the network. In this respect, probabilistic location update is considered in [12], conference calls with delay constraints in [5] and [18], while [17] provides a framework for location uncertainty. An information-theoretic approach whereby the inherent location uncertainty is being used for tracking users can be found in [8] while a tracking strategy incorporating a distance-based strategy with a timer, that counts the time duration since the last location update is proposed in [15].

For the most part, solutions cited above involve maintaining a central database of the precise location of all users at any time or querying the nodes of the network whenever a user's location must be determined. Moreover, the query algorithms only assume that nodes can provide only the presence or non-presence of the user at the location queried. The approach of the current paper is to expand this model by having the nodes store additional timing information of the "last time the mobile visited the node"; the advantage is that a user can be found faster without having to query all of the nodes of the network.

2. Upper bounds

In this section we present a variety of algorithms that provide upper bounds for arbitrary graphs, planar graphs, meshes, trees, and lines.

2.1 Arbitrary graphs

We present three algorithms that work on arbitrary topologies. The efficiency of the first is limited by the maximum degree d of the underlying network.

T 1. *Suppose that a user begins moving at time* 0 *from a known location s of an arbitrary connected graph. There is a search algorithm starting at time t that will find the user in at most r rounds with a total of at most $O(rd^{\lceil t/r\rceil+1})$ queries.*

P . Beginning with the start node s, in each round the search algorithm queries all the nodes in a neighborhood $N_{\lceil t/r\rceil+1}(u)$ of nodes at distance at most $\lceil t/r\rceil + 1$ from a given node u. Initially, $u := s$. The node is updated with that node in $N_{\lceil t/r\rceil+1}(u)$ that has seen the user most recently. At the end of the ith round the user will move to a neighbor of a previously occupied node during the $i-1$ round, while the search algorithm is at a distance of at most $t - i\lceil t/r\rceil$ from the user. So in r rounds the search algorithm will be at a distance of at most $t - r\lceil t/r\rceil \le 0$ from the user and the user will be located. In each round at most $O(d^{\lceil t/r\rceil+1})$ nodes are queried. This completes the proof of Theorem 1. □

Our second algorithm depends on the separator properties of a graph. An edge separator of graph G is a set S of edges of the graph that separates the graph into two or more parts of at most $n/2$ nodes each. We define the *edge separator function* of the graph G as follows: for any integer k,

$$F_G(k) = \max_{(\text{induced subgraph } G_k)} \min_{(S \text{ edge separator of } G_k)} |S|, \tag{1}$$

where the max is taken over all induced subgraphs G_k of G with at most k nodes, and the min over all edge separators S of G_k. We prove the following theorem.

T 2. *Suppose that a user begins moving at time* 0 *from a known location s of an arbitrary connected graph. There is an algorithm that will find the user in at most $O(\log n)$ rounds with a total of at most*

$$2\sum_{i=0}^{\lceil \log n\rceil}\sum_{j=0}^{i} F_G(n/2^j)$$

queries, where F_G is the edge separator function of the graph G.

P . Let the graph be $G = (V, E)$. In the first round we consider an edge separator with node set S_1 of the graph G of size at most $2F_G(n)$ (i.e, at most 2 nodes per edge in the separator). The search algorithm queries the nodes in S_1 and determines in which component of the graph minus the separator the user lies. In

the next round the search algorithm queries the set of nodes in S_2, where S_2 is the set of nodes in an edge separator of the component previously determined, along with the nodes of S_1 (since, in the meantime, the user may have escaped to a node belonging to S_1). We iterate this process and we observe that if S_i is the new set of nodes queried in the i-th round then the number of queries in the i-th round is at most $|S_1| + |S_2| + \cdots + |S_i|$. This implies that the total number of queries in all rounds is at most

$$\sum_{i=0}^{\lceil \log n \rceil} \sum_{j=0}^{i} |S_j|.$$

The upper bound stated in the theorem now follows easily from the definition (see Equation 1) of the edge separator function of the graph. This completes the proof of Theorem 2. □

We note that a natural generalization of Theorem 2 exists where the graph is separated into l or more pieces of size at most n/l, resulting in an algorithm using $O(\log_l n)$ rounds and query count given by the corresponding sum of separator sizes. An example of this approach is given in Theorem 4.

The *cutwidth* of a graph G, denoted by $c(G)$, is the smallest integer k such that the n vertices of the graph can be arranged in a linear order $v_1, v_2, \ldots, v_n$ in such a way that for all $i < n$ there are at most k edges with one endpoint in $v_1, \ldots, v_i$ and the other in $v_{i+1}, \ldots, v_n$ (see [19]).

T 3. *Suppose that a user begins moving at time* 0 *from a known location* s *of an arbitrary connected graph. There is an algorithm that will find the user in at most* $O(\log n)$ *rounds with a total of at most* $O(c(G) \log n)$ *queries, where* $c(G)$ *is the cutwidth of the graph* G.

P . Let the cutwidth of the graph G be $c := c(G)$. Therefore there is a linear order $v_1, v_2, \ldots, v_n$ of the vertices of the graph such that for all $i < n$ there are at most c edges with one endpoint in $v_1, \ldots, v_i$ and the other in $v_{i+1}, \ldots, v_n$. At any round of the algorithm, we maintain an interval v_a to v_b in which we are certain the user lies. Initially, $v_a = v_1$ and $v_b = v_n$. For such an order of the vertices define the sets $A = \{v_a, \ldots, v_{\lfloor (a+b)/2 \rfloor}\}$, $B = \{v_{\lfloor (a+b)/2 \rfloor + 1}, \ldots, v_b\}$.

The search algorithm queries the set S of nodes consisting of the endpoints of edges between (i) $v_1, \ldots, v_{a-1}$ and $v_a, \ldots, v_b$, (ii) $v_a, \ldots, v_b$ and $v_{b+1}, \ldots, v_n$ and (iii) A and B, plus the node s. Note that this consists of at most $6c + 1$ nodes. If the user is among these nodes then we are done. Else, let $u \in S$ be the node queried by the search algorithm that saw the user last. If $u \in A$ (respectively, B) then in the next round we can be certain the user will be located in A (respectively, in B). (Note that the sets (i) and (ii) above insure that the user does not escape during the current round.) The search continues inductively on an interval of at most half the size. This implies that the user will be located within $O(\log n)$ rounds using $O(c)$ queries per round. This completes the proof of Theorem 3. □

We note that a natural generalization of Theorem 3 exists where the linear order is divided into l approximately equal parts. The resulting algorithm runs in $O(\log_l n)$

rounds and uses $O(lc(G)\log_l n)$ queries. An example of this approach is given in Theorem 8.

2.2 Planar graphs

As a corollary to Theorem 2 we get the following result for planar graphs with max degree d.

C 1. *Suppose that a user begins moving at time* 0 *from a known location s of a planar graph of maximum degree d. There is an algorithm that will find the user in at most $O(\log n)$ rounds with a total of at most $O(d\sqrt{n}\log n)$ queries.*

P . We use the planar separator theorem of [14]: Any n-node planar graph of maximum degree d has an edge separator of size $O(d\sqrt{n})$. The result follows immediately from Theorem 2. This completes the proof of Corollary 1. □

2.3 Mesh

In this section we give upper bounds for the mesh that improve upon the obvious application of the general algorithms. We prove the following theorem.

T 4. *Suppose that a user begins moving at time* 0 *from a known location s of an $n \times n$ mesh. For any constant r number of rounds there is a search algorithm that starting at time t will find the user in at most $O(r\min\{n, 2t\}^{\frac{2^r}{2^r-1}})$ queries.*

P . We consider two cases depending on the relative sizes of $t + r, n$.

Case $t + r \geq n$. By querying all of the nodes one can find the user in one round using n^2 queries. We show how to achieve $O(n^{4/3})$ queries in two rounds and then show how to generalize this result to get the claimed result. On round one we query all nodes on every $n^{2/3}$rd row and column and their neighbors, a total of less than $6n^{4/3}$ queries. At this point the search algorithm can deduce which of the resulting $n^{2/3} \times n^{2/3}$ submeshes the user is in by following its progress from the submesh containing s and observing which boundaries it crosses when. If all queries are null then the submesh containing s is the correct one. One more round of $n^{4/3}$ queries suffices.

To generalize to r rounds, divide them into two sets of rounds, the first round and the remaining $r - 1$ rounds and apply the solution for fewer rounds inductively. On round one we query all nodes on every

$$n^{1-1/(2^r-1)} = n^{(2^r-2)/(2^r-1)}$$

row and column (and neighbors), a total of at most $6n^{2^r/(2^r-1)}$ queries. At this point we can deduce on which of the resulting

$$n^{(2^r-2)/(2^r-1)} \times n^{(2^r-2)/(2^r-1)}$$

submeshes the user is in. The remaining $r - 1$ rounds are now executed inductively on an $m \times m$ submesh containing s, where $m = n^{(2^r-2)/(2^r-1)}$.

Case $t + r < n$. In r rounds the agent cannot reach a point more than $t + r$ hops away from s. We consider a square mesh with s as its center and with side $2(t + r)$. Clearly the agent will not leave this mesh during r rounds. We now run the algorithm for the previous case with $n = 2(t + r)$. (Note that if s is near the edge of the mesh some portion of this mesh may not exist. In this case we run the algorithm on a "virtual" mesh that includes nonexistent nodes outside of the actual mesh. Queries to "virtual" nodes are ignored.) The user is then found in $r(2(r + t)^{\frac{2^r}{2^r-1}})$ queries.

Combining the two cases together we get an upper bound of

$$O\left(r \min\{n, 2t\}^{\frac{2^r}{2^r-1}}\right)$$

on the number of queries.

This completes the proof of Theorem 4. □

Applying Theorem 2 directly to the mesh results in an algorithm that uses $O(\log n)$ rounds and $O(n \log n)$ queries. By a slight adjustment we can remove the factor of $\log n$.

T 5. *Suppose that a user begins moving at time* 0 *from a known location* s *of an* $n \times n$ *mesh. There is a search algorithm that starting at time* t *will find the user in at most* $O(\log \min\{n, t\})$ *rounds and at most* $O(\min\{n, t\})$ *queries.*

P . We consider two cases depending on the relative sizes of $t + \log t, n$.

Case $t + \log t \geq n$. We note that a $k \times l$ mesh has an edge separator of size $O(\min\{k, l\})$ and the resulting graphs after removing the separator are meshes with sides $\min\{k, l\}$ and $\max\{k, l\}/2$. Applying an algorithm analogous to that of Theorem 2 but at each round rather than querying the separator from the previous rounds, we enlarge the mesh by $\log n$ in order to be certain the user does not escape, we get an algorithm that after $O(\log n)$ rounds and $O(n)$ queries has reduced the range of the user to a mesh of maximum side $O(\log n)$. One round of $O(\log^2 n)$ queries completes the search.

Case $t + \log t < n$. This can be handled easily by considering a mesh of side $2(t + \log t)$ which is centered at s.

This completes the proof of Theorem 5. □

2.4 Tree

In this section we give upper bounds for trees. Direct application of the results for arbitrary graphs provide upper bounds but these can be improved somewhat by taking advantage of the tree topology.

The algorithm of Theorem 1 when adapted to trees of maximum degree d yields the following theorem.

T 6. *Suppose that a user begins moving at time* 0 *from a known location* s *of a tree of maximum degree* d *and height* h. *There is an algorithm that will find the user in at most* h *rounds with a total of at most* $O(hd)$ *queries.*

P . The search algorithm starts by querying the root of the tree and its neighbors. If none of these nodes has seen the user then it must be located in the same subtree rooted at that child of the root as that in which the start node s is located. Otherwise, we consider the node v that has seen the user most recently. If v is the root then the user has been found. If v is a child of the root then the user must be located in the subtree rooted at v. In this last case, we can detach the subtree rooted at v and iterate the search inductively. In any case, there is a total of at most h rounds and in each round the search algorithm makes at most $d + 1$ queries. This completes the proof of Theorem 6. □

The algorithm from Theorem 2 when applied to a constant degree tree results in an algorithm that requires $O(\log^3 n)$ queries to find a user in $O(\log n)$ rounds. This can be improved as we see in the following theorem.

T 7. *Suppose that a user begins moving at time* 0 *from a known location* s *of a tree of maximum degree* d. *There is an algorithm that will find the user in at most* $O(\log_{\frac{d}{d-1}} n)$ *rounds with a total of at most* $O(\log^2_{\frac{d}{d-1}} n)$ *queries.*

P . We use the fact that a tree of degree d has a single edge that splits the tree into two pieces neither of which is smaller than n/d. Proceeding as in Theorem 2 using this separator rather than a perfect separator, the theorem follows. (Note that on each round all previous separator edges must be queried.) This completes the proof of Theorem 7. □

2.5 Line

In this section we give upper bounds for the line. Applying Theorem 3 yields an algorithm that finds a user in $O(\log n)$ rounds using $O(\log n)$ queries. This algorithm can be generalized to yield an optimal algorithm for any number of rounds less than $\log n$. In section 3 we show our upper bound is tight. We can show the following theorem.

T 8. *Suppose that a user begins moving at time* 0 *from a known location* s *of a line of length* n. *There is a search algorithm that starting at time* t *will find the user in* r *rounds with a total of* $O(r \cdot \min\{t, n\}^{1/r})$ *queries.*

P . We consider two cases depending on the relative sizes of $2t + 2r + 1, n$.

Case $2t + 2r + 1 < n$. Assume the user started at node 0 and that the line consists of the nodes $0, \pm 1, \pm 2, \ldots, \pm n/2$. If $t < n/2$, then at time t it can be at any of the $2t + 1$ nodes of the closed interval $[-t, t]$.

The search algorithm is executed in rounds and each round involves querying a certain number of nodes. In the first round the algorithm queries the set of nodes $\{kt^{(r-1)/r} : |k| \leq t^{1/r}\} \cup \{-t, t\}$ and determines an interval, say I_1, of size at most $t^{(r-1)/r}$ which is most recently occupied by the user. By induction, assume that in the i-th round the algorithm has determined an interval, say I_i, of size at most $t^{(r-i)/r}$ which is most recently occupied by the user. Set $m := t^{(r-i)/r}$. In the $i + 1$-st round the algorithm queries a set of $m^{1/(r-i)}$ equally spaced nodes of the interval I_i

at distance $t^{(r-i-1)/r}$ from each other. Since $m^{1/(r-i)} = t^{1/r}$ the number of queries in this new round is $O(t^{1/r})$. The user is found at the r-th round by querying all the nodes of the subinterval I_r. It follows that r rounds with $O(t^{1/r})$ queries per round are sufficient to locate the user.

Case $2t + 2r + 1 \geq n$. If $2t + 2r + 1 \geq n$ then at time t the user may be located at any node of the line and a similar proof will work.

This completes the proof of Theorem 8. □

3. Lower bounds

In this section we present lower bounds that show some of our upper bounds are tight.

3.1 Clique

The following general lower bound is straightforward:

T 9. *Suppose that a user begins moving at time* 0 *from a known location s of an arbitrary graph. Let* $N_s(t)$ *be the set of nodes reachable from s in t steps. Any search algorithm starting at time* $t > 0$ *requires at least* $N_s(t)$ *queries in order to find a node in a single round.* □

The above shows that the obvious algorithm of querying all nodes of a clique in a single round is optimal.

T 10. *Searching for a user on a clique requires n queries in a single round.* □

3.2 Tree

In this section, we give a lower bound for bounded degree trees. The lower bound implies the algorithm of Theorem 8 is optimal.

T 11. *Suppose that a user begins moving at time* 0 *from a known location s of a tree of size n and maximal degree d. Any search algorithm starting at time t and finding the user in r rounds requires* $\Omega(r \cdot \frac{\min\{t,n\}^{1/r}}{d})$ *queries.*

P . Consider the case where $t < n/2$ and consider the greater than t nodes within distance t of s. By an *uncertainty subtree* we mean a subtree any of whose nodes may be occupied by the user based on the current knowledge of a search algorithm, i.e., the results of its queries.

C 1. *If prior to the execution of the queries at a given round there is an uncertainty subtree of size at least L and if during a round a total of q queries are made to nodes of this subtree then after the execution of this round there is an uncertainty subtree of size at least* $\frac{L}{qd}$.

P (C 1). Suppose that q nodes, in the given subtree are queried. This divides the subtree into qd subtrees one of which is of size at least $\frac{L}{qd}$. An adversary can always answer the queries so as to leave the search algorithm with an uncertainty subtree of size at least $\frac{L}{qd}$. For example, the queries may be answered consistent with the user walking directly to the first non-queried node in the large subtree and waiting there. This completes the proof of Claim 1.

Now suppose that an algorithm is executed for r rounds and that at round $i \leq r$ we made exactly q_i queries. In view of Claim 1, at the end of the last round r, the uncertainty subtree must be of size at least

$$\frac{t}{(q_1 * d)(q_2 * d) \cdots (q_r * d)}$$

and the total number of queries is $q_1 + q_2 + \cdots + q_r$. Hence we have to solve the following optimization problem:

$$\begin{array}{ll} \text{minimize} & q_1 + q_2 + \cdots + q_r, \\ \text{subject to} & (q_1 * d)(q_2 * d) \cdots (q_r * d) \geq t. \end{array}$$

It is easy to see that this is minimized when $q_1 = q_2 = \cdots = q_r$, in which case the total number of queries is at least

$$\Omega\left(r \cdot \frac{t^{1/r}}{d}\right).$$

If $t \geq n/2$ then at time t the user may be located at any node of the tree and a similar proof will work. This completes the proof of Theorem 11. □

3.3 Mesh

We now show that the upper bound of Theorem 4 is tight for any constant number of rounds. Namely we prove the following result.

T 12. *Suppose that a user begins moving at time* 0 *from a known location s of an* $n \times n$ *mesh. For any constant r, any search algorithm starting at time t and finding the user in r rounds requires* $\Omega(\min\{n, t\}^{\frac{2^r}{2^r-1}})$ *queries.*

P . We prove the result for $t = 2rn$ and s being the center of the mesh. The necessary adjustments when either $t > 2rn$ or $t < 2rn$ and for arbitrary s are straightforward.

We require the following definitions. For $0 < c \leq n$, an $n \times n$ *c-mesh* is formed by taking an $n \times n$ mesh, choosing up to $n - c$ nodes and removing them along with all of the edges in the rows and columns corresponding to these nodes. A row or column is said to be *clean* if no nodes (and therefore edges) were removed from it in forming the c-mesh. By considering the greater than or equal to c clean rows and columns of the mesh, it is easy to see that an $n \times n$ c-mesh has a connected subgraph with $\Omega(c^2)$ nodes and diameter less or equal to $2n$. Furthermore, if we

remove c' nodes and corresponding edges in their rows and columns, from a c-mesh, the result is a $(c - c')$-mesh.

We prove the following claim concerning c-meshes.

C 2. *Suppose that a user begins moving at time* 0 *from a known location* s *of an* $n \times n$ *c-mesh. For any constant* r*, any search algorithm starting at time* $t = 2rn$ *and finding the user in* r *rounds requires* $\Omega(c^{\frac{2^r}{2^r-1}})$ *queries.*

P . We prove the claim by induction on r. The case $r = 1$ follows from Theorem 9. Assume the claim holds for some constant number of rounds, $r - 1$. Divide the c clean rows and columns of the c-mesh into blocks of length $c^{\frac{2^r-2}{2^r-1}}$. The connected subgraph formed by the intersection (along with connecting edges and vertices) of the ith block of rows and the jth block of columns forms a $c^{\frac{2^r-2}{2^r-1}}$-mesh and there are $c^{\frac{2}{2^r-1}}$ such meshes.

Let q be the number of queries in the first round of the search. Without loss of generality we can assume that $q < \frac{c^{\frac{2^r}{2^r-1}}}{2}$ (otherwise the proof of the claim is complete). By an averaging argument, there is a $c^{\frac{2^r-2}{2^r-1}}$-mesh, A, that receives fewer than $\frac{c^{\frac{2^r-2}{2^r-1}}}{2}$ queries. We answer all queries of the first round consistent with the user having started at time 0 and having walked directly to the nearest non-queried node in A and having waited there until at total of $2n$ steps have passed. For queries inside A we report never having seen the user. We remove all of the nodes of A corresponding to queries along with the edges in their rows and columns. The search algorithm is now left with the problem of finding the user in $r-1$ rounds, starting at time $2(r-1)n$ on a $\frac{c^{\frac{2^r-2}{2^r-1}}}{2}$-mesh. By induction, this requires $\Omega\left(c^{\frac{2^r}{2^r-1}}/2^{\frac{2^{r-1}}{2^{r-1}-1}}\right)$ queries where r is constant. This completes the proof of Claim 2. □

Taking $c = n$ in the claim completes the proof of Theorem 12. □

3.4 Arbitrary graphs

A lower bound for arbitrary graphs follows easily from the lower bound for the tree.

T 13. *Suppose that a user begins moving at time* 0 *from a known location* s *of an arbitrary graph with* n *nodes of maximal degree* d*. Any search algorithm starting at time* t *and finding the user in* r *rounds requires* $\Omega(\max\{d+1, r\frac{\min\{t,n\}^{1/r}}{d}\}$ *queries.*

P . The lower bound $d+1$ follows from the fact that a node of degree d and all its neighbors may have to be queried in order to locate the user. The lower bound $r\frac{\min\{t,n\}^{1/r}}{d}$ follows from the lower bound for the tree by considering a spanning tree of the graph. This completes the proof of Theorem 13. □

4. Conclusions

In this paper we considered tradeoffs on the number of rounds and queries for tracking a user in a network where nodes respond to queries with the time a user last visited the node. We gave several tracking algorithms for arbitrary networks and specific topologies, like, planar graphs, lines, meshes, and trees. We also considered lower bounds for all these types of networks and showed our algorithms are optimal in several cases. Several interesting problems remain, in addition to tightening our bounds. These include variants of the query/answer model we studied here, that are either more or less restrictive in the information they provide. Additional twists concern modeling the behavior of possibly faulty information, e.g., either an upper bound on the number of nodes that may be faulty or nodes may give incorrect answers to queries with some probability. By using a parameter that characterizes the relative speed of the mobiles with respect to the server query response time can provide interesting problems for further investigation in Mobile Agent settings. Finally, an interesting question concerns the complexity of the proposed strategies since they affect the performance of the central server.

Acknowledgements

Many thanks to Paolo Penna for useful conversations on the subject and to the anonymous referees for their suggestions.

References

[1] A , S. G , S. 2003. *The Theory of Search Games and Rendezvous*. Kluwer Academic Publishers, Norwell, Massachusetts.
[2] A , B . 2006. Searching and Sweeping Graphs: A Brief Survey. *Le Matematiche (Catania) 59*, 5–73.
[3] A , B P , D . 1991. Concurrent Online Tracking of Mobile Users. In *SIGCOMM*, 221–233.
[4] B , P P , V N. 1999. User Location and Tracking in an In-Building Radio Network. Tech. Report MSR-TR-99-12, Microsoft Research (MSR).
[5] B -N , A. M , G. 2004. Establishing Wireless Conference Calls under Delay Constraints. *Journal of Algorithms 51*, 145–169.
[6] B -N , A , K , I , N , M . 1996. Topology-Based Tracking Strategies for Personal Communication Networks. *MONET 1*, 1, 49–56.
[7] B , L., F , P., F , P., S , N. 2003. Election and Rendezvous of Anonymous Mobile Agents in Anonymous Networks with Sense of Direction. In *Proceedings of the 9th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, 17–32.
[8] B , A. D , S. K. 2002. LeZi-Update: An Information-Theoretic Framework for Personal Mobility Tracking in PCS Networks. *Wireless Networks 8*, 121–135.
[9] C , N E. C , E L. 2006. Cops, Robber, and Alarms. *Ars Combinatoria 81*, 283–296.
[10] C , N E. N , R J. 2000. Cops, Robber, and Photo Radar. *Ars Combinatoria 56*, 97–104.
[11] F , F. V., F , P., T , D. M. 2004. The Price of Connectedness in Expansions. Tech. Report LSI-04-28-R, Departament de Llenguatges i Sistemes Informatics, Universitat Politecnica de Catalunya, Barcelona, Spain.

[12] J , D G J , W S . 1998. Probabilistic Location Update for Advanced Cellular Mobile Networks. *IEEE Communications Letters 2*, 8–10.

[13] L , A. 1993. Recontamination does not help to search a graph. *Journal of the ACM 40*, 2, 224–245.

[14] L , R. J. T , R. E. 1979. A Separator Theorem for Planar Graphs. *SIAM J. Appl. Math. 36*, 177–199.

[15] N , Z. 2003. Tracking Mobile Users with Uncertain Parameters. *Wireless Networks 9*, 637–646.

[16] P , T. D. 1978. The Search Number of a Connected Graph. In *Proc. 9th Southeastern Conf. on Combinatorics, Graph Theory, and Computing, Utilitas Math. Publ., Winnipeg*, 549–554.

[17] R , C Y , R . 1997. Location Uncertainty in Mobile Networks: a theoretical framework. *IEEE Communications Magazine 35*, 2, 94–101.

[18] R , C Y , R D. 1995. Minimizing the average cost of paging under delay constraints. *Wireless Networks 1*, 2, 211–219.

[19] T , D. M., S , M. J., B , H. L. 2001. A Polynomial Time Algorithm for the Cutwidth of Bounded Degree Graphs with Small Treewidth. In *9th Annual European Symposium on Algorithms ESA*, Volume 2161 of *LNCS*, 380–390.

[20] W , J Z . 1993. A fully distributed location registration strategy for universal personal communication systems. *IEEE Journal on Selected Areas in Communications 11*, 6 (Aug.), 850–860.