A Theoretical Study of Optimization Techniques Used in Registration Area Based Location Management: Models and Online Algorithms

Sandeep K. S. Gupta, Goran Konjevod, Georgios Varsamopoulos
Department of Computer Science and Engineering
Arizona State University
Tempe, AZ 85287-5406

{sandeep.gupta,goran.konjevod,georgios.varsamapoulos}@asu.edu

ABSTRACT

We describe a study of optimization techniques which have been proposed in the literature for registration area based location management from the perspective of optimal online algorithms. We show that most of these optimization techniques such as forwarding pointers and overlapped registration areas can be modeled using metrical task systems. Some of these models are simple while others are somewhat intricate. These representations directly imply the applicability of existing results on metrical task systems to algorithm design for location management. However, this also means that general lower bounds known for online metrical task systems carry over to our formulations of location management. We also discuss some restricted models in which much better (and simpler) algorithms are possible.

Categories and Subject Descriptors

C.1.3 [**Computer Systems Organization**]: Other Architecture Style—*Cellular architecture (e.g. mobile)*; C.2.4 [**Computer-Communication Network**]: Distributed Systems—*Distributed Databases*; F.1.2 [**Theory of Computation**]: Modes of Computation—*Online Computation*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Computation on Discrete Structures; H.2 [**Database Management**]: Miscellaneous; H.3.4 [**Information Storage and Retreival**]: Systems and Software—*Performance Evaluation (efficiency and effectiveness)*

General Terms

Algorithms, Performance, Theory

Keywords

On-line algorithm, Location management, Registration area, Metrical task systems

1. INTRODUCTION

We use the term *location management* to describe the task of tracking the locations of mobile users with wireless terminals in a network consisting of a fixed set of base stations (cells). The two basic operations in a location management scheme are called *update* and *search*. Mobile users update their locations regularly or when necessary. The network searches for a mobile user in order to deliver a call. In the process of searching, the network uses the information provided by the mobile terminal through updates. As it provides a mechanism for recording and querying locations of mobile units in the network, location management is essential to providing a ubiquitous computing environment for mobile users, for example establishing calls in Personal Communication Services (PCS) networks such as GSM or routing messages to mobile units in the mobility-enabled networks such as Internet with Mobile IP or IPv6. The challenge of location management lies in the inherent trade-off between the cost of two basic operations. Techniques which try to optimize one tend to increase the cost of the other operation. For example, if a mobile updates its location more often then it can be found more quickly. This reduces the search cost but increases the update cost. On the other hand, fewer updates typically means that a greater area must be searched for the mobile. Further, different mobile users of the network may have wildly varying call and mobility patterns. In order to find the perfect balance between the two operations, an ideal location management technique must be adaptable to the call and mobility pattern of each user.

A common technique for balancing the cost of update and search is the use of *registration areas* (RA) to track the user. The geographical area is divided into several zones, called registration areas, each of which consists of several cells. The system tracks a mobile terminal's registration area instead of its cell. Whenever a mobile terminal crosses from one registration area to another it reports its new location to the system. To establish a call to a mobile terminal, the system pages all the cells in the registration area to find the current cell of the mobile terminal. A database called *location register* (LR) is associated with each registration area to keep information about the mobiles currently registered in that registration area. Common standards for location management (e.g. IS-41 and GSM) use registration areas along with a two-level hierarchy of location registers. The hierarchy consists of home location registers (HLR) and visitor location registers (VLR). Location information of a mobile node is kept at both its current location registrar, i.e. its VLR, and its permanent home location registrar. This facilitates locating non-local mobiles during call delivery: the home location register of the mobile unit is contacted to find its current location. Hence, the two-level hierarchy scheme increases the location update cost while reducing the location search/call delivery cost.

Many techniques have been proposed in the literature in order to make registration area based location management more efficient and adaptive to call and mobility pattern of the user. Examples include forwarding pointers[19], location caching[21, 26], and overlapping registration areas[20, 33]. All of these techniques involve dynamic decision making. For example, when using forwarding pointers the system must decide dynamically how long the forwarding pointer chain should be, based on the current call and mobility rate of the user. In order to deal with this problem, most research so far uses a stochastic mobility model to represent the mobility pattern of the user and assume that this mobility pattern remains the same over time, thus allowing the location management scheme to be tailored to each user. However, there is currently no consensus as to what is the best way to infer and represent the mobility pattern of a user. In this paper, we examine on-line location management techniques from the perspective of their competitiveness with respect to off-line algorithms. Through several examples, we show how most of the registration-based location management models studied so far can be represented using metrical task systems. Some of these models are simple while others are somewhat intricate. These representations directly imply the applicability of existing results on metrical task systems to algorithm design for location management. We briefly describe positive and negative results on metrical task systems, and then describe some restricted models for location management in which even better online algorithms are possible. We see the purpose of our paper as the introduction of a realistic problem to the theoretical community in the hope of stimulating research on a family of online algorithms problems that may be of interest to practitioners in mobile computing.

2. PROBLEM STATEMENT AND PREVIOUS WORK

2.1 Registration areas and location management

The service area of a cellular network is represented by a finite set C of cells $C = \{c_1, c_2, \ldots, c_n\}$. A *registration area* (RA) is a subset $R \subset C$. We assume that C is covered by registration areas, $C = \cup_i R_i$. (In real-life situations, RAs are usually geographically contiguous.) Define the *availability vector* $A(c)$ for a cell c as the set of all registration areas that contain c. For systems with non-overlapping RAs, for every cell c, $A(c)$'s only component refers to the unique registration area available at c. When RAs overlap, an availability vector may contain more than one registration area. Define a *region* to be any maximal set $g \subseteq C$ of cells whose availability vectors are all equal. The notion of availability naturally extends to regions, $A(g) = A(c)$, for any $c \in g$.

We represent the motion of a user who travels within the network as a sequence of regions $p = g_1 g_2 \ldots g_k$. The user starts in the region g_1 and stops in g_k. While the user moves from region to region, the availability of RAs changes. The user must be registered at all times, which necessitates the selection of an available RA for each region the user visits. A user registers at most once per region, that is, once the user selects an RA and registers, the user doesn't re-register until leaving the region. After moving to a new region, the user may continue to be registered with the previous RA if the RA is still available in the current region, or may switch to a different RA among the available ones. In this case a registration is performed. If the user moves to a region where the previous RA is not available, then a new RA must be chosen. We formalize the problem of finding the optimal registration sequence as follows. Consider the region path p partitioned into disjoint subpaths $p_1, p_2, \ldots, p_m$, $(p_i = g_{i_1} g_{i_2} \ldots g_{i_j})$, such that $p_1 p_2 \ldots p_m = p$. We say that a registration sequence $S = R_1 R_2 ... R_m$ is *consistent* with the path p if and only if there exists a partitioning $p_1 p_2 \ldots p_m$ of path p such that $\forall g_i \in p_j \;\; R_j \in A(g_i)$. Then, the *deterministic optimal registration problem* is: given a sequence of regions and their availability vectors, find a consistent registration sequence of minimum length.

Many models of location management have been proposed [2, 4, 15, 27, 35], which suggest various way of performing location tracking (update) and call delivery (search). Nevertheless, only the *Registration Area* (RA) model is used in existing PCS networks like GSM, IS-41 and Mobile IP. This is so mainly because the RA model is simple, has centralized control in per-user aspect (which is simple and helps with service billing), and it has shown good scalability thus far. Nevertheless, this model is starting to show inadequacies with the rapidly growing market of PCS customers and new solutions are becoming necessary.

Most of the proposed schemes are enhancements to the RA model, which try to either reduce the search cost (ie the cost of a search operation) or update cost (ie the cost of an update operation, or the frequency of updates). As demonstrated in several works, there is a tradeoff between update and search costs. A strategy which tries to reduce one cost tends to increase the other, and vice versa. For example, if a mobile updates its location more often (increasing update cost) it can be searched more easily (decreasing search cost). Some techniques that reduce the search cost are *Location Caching* [21, 26], in either the form of *lazy caching* or *eager caching*. Caching reduces the average case cost, without reducing the worst case cost. A form of eager caching is *Profile Replication* [18, 22, 25, 28, 30], which reduces the inherent cost of search. Another search cost reducing technique is *Prediction* [14, 36]

Efforts have been made to reduce the update cost: *Forwarding Pointers* [19], which reduces the cost of an update operation, and *Look-ahead registration* [32], which reduces the frequency of updates, are two examples of per-user techniques. Apart from per-user schemes, there are system-wide enhancements that can be applied to help reduce update cost. *Multi-layered Organization* [7, 20] is a global technique that is based on hierarchically dividing the coverage area so the spatial locality is exploited at its best. Multi-layered hierarchical schemes show very good average-case performance. *RA Overlapping* [20, 33] is a global technique that tries to smooth the effect of frequent registration operations at registration area boundaries. *Mobile RAs* [8] is another technique that suggests that RAs should move along with the users to reduce the frequency of update operations.

Varsamopoulos and Gupta [34] describe a deterministic version of the location management problem with overlapping registration areas. They present a generalization of Dijkstra's shortest path algorithm that computes an optimal solution to the registration problem by dynamic programming, given the network and trajectory of the mobile. In the same paper, a stochastic approach to optimal registration is discussed for the situation where the complete trajectory of the user is not known in advance, but only the user's mobility pattern in the form of a random walk model. The *stochastic optimal registration problem* is to determine a registration area that minimizes the expected minimum registration sequence length over the random distribution on paths of given length (in the number of edges, that is, time-steps) generated by the random walk model.

2.2 Metrical task systems

A *metrical task system* processes a sequence $T_1, T_2, \ldots, T_k$ of *tasks*. For each i, processing the task T_i incurs a cost $c(T_i, S_j)$ that depends on the *state* S_{j_i} into which the system is configured before the processing. The states form a finite metric space and n denotes their number. When assigned a new task T_{i+1}, the system may make a *transition* and reconfigure into a new state $S_{j_{i+1}}$ before processing the task. The cost of processing the new task T_{i+1} will, of course, depend on the state $S_{j_{i+1}}$ the system enters, but resources may also be expended in the transition to the new state. Formally, a cost $c(T_i, S_j)$ is associated with processing the task T_i in state S_j for every i and j, and a cost $c(S_i, S_j)$ is associated with the transition between S_i and S_j for every i and j. The name of the problem arises from the assumption that the transition costs form a distance function or metric, that is, they satisfy symmetry ($c(S_i, S_j) = c(S_j, S_i)$) and triangle inequality ($c(S_i, S_j) + c(S_j + S_k) \geq c(S_i, S_k)$).

A *schedule* for processing a sequence of tasks $T_1, T_2, \ldots, T_k$ is simply a sequence of states $S_0, S_{i_1}, S_{i_2}, \ldots, S_{i_k}$; the *cost* of the schedule is the sum

$$\sum_{j=1,k} c(S_{i_{j-1}}, S_{i_j}) + c(T_i, S_{i_j})$$

of all processing and transition costs.

The *competitive ratio* of a metrical task system algorithm (and more generally, of any on-line algorithm) is the maximum ratio of the cost of a schedule found by the algorithm and an optimal schedule found by an off-line algorithm (one that knows the task sequence in advance). Thus a bound on the competitive ratio of an algorithm is a bound on its worst-case performance. For a randomized algorithm, the competitive ratio is defined to be the expectation of the worst-case ratio of a schedule found by the algorithm to an optimal schedule found by an off-line algorithm.

Metrical task systems were introduced by Borodin et al. [10] who gave the first positive and negative results on the competitive ratio. For years, most of the positive results on metrical task systems and the k-server problem were stated for instances defined on special underlying metric spaces [9, 10, 12, 13, 17, 23, 24]. Recently, improvements have been found for more general metric spaces, all based on a general method for approximating general metric spaces by tree metrics [5, 6, 11]. The currently best algorithm for general metrical task systems is due to Fiat and Mendel [16] and has a competitive ratio polylogarithmic in the number of states in the system. This guarantee also holds for our location management models. However, lower bounds are known for metrical task systems also: for *no* metric space does there exist an algorithm that achieves a competitive ratio better than $\Omega(\log \log n)$. The conjectured lower bound is in fact $\Omega(\log n)$. Thus modeling location management as a metrical task system has its disadvantages.

There are some previous work on location management from theoretical standpoint. Bar-noy and Kessler [3] study a model in which there are no registration areas, but instead a subset of cells are denoted as *reporting centers* and a user registers only when entering a reporting center. The search entails checking all cells in the proximity of the last center to which the user reported. A subsequent paper [4] studies different strategies for update in a similar setting: the time-based, number of movements-based and distance-based strategies. The paper of Awerbuch and Peleg [1] describes an efficient algorithm for maintaining a distributed directory server for location management that uses a graph-theoretic notion, *regional matching*. The fundamental difference between these approaches and ours is that we are viewing the location management problems in the context of optimal online algorithms.

3. PERFORMANCE IN UN-RESTRICTED MODEL

In this section we prove non-existence of a competitive algorithm when no restriction is placed on either the mobility pattern or the structure of the network.

THEOREM 3.1. *Given any integer $k > 1$, there exists an input model for which any online registration algorithm is at best $k-1$ competitive.*

PROOF. Let there be k registration areas which intersect and create a total of k regions such that

1. each region has $k-1$ available RAs, and
2. each RA is available in $k-1$ regions.

Note that for each RA there exists a region that the RA does not cover.

We take *any* online registration algorithm and we let it chose an initial RA. Then we move the user to a region where the selected RA is unavailable – we know there is always such a region. The algorithm then makes a new choice among the available RAs and we move the user to a region where the new RA is not available. Continuing like this, for m user moves we make a registration sequence of m updates. On the other hand, an optimal off-line solution for those m moves is to group the regions of the user's path in groups of $k-1$ regions each and chose an RA that contains those $k-1$ regions – we know we can always find such a RA. This yields a registration sequence of $\frac{m}{k-1}$ updates.

This makes the algorithm's competitiveness lower bound to be $(k-1)$-competitive. □

However, since k can be arbitrary large, there is no bound to the competitiveness of the algorithm.

COROLLARY 3.2. *There is no competitive online algorithm for the stochastic case of the registration sequence problem.*

4. OPTIMAL REGISTRATION AS A METRICAL TASK SYSTEM

Several formulations of the location management problem can be easily stated in terms of metrical task systems.

4.1 Simple location management: user optimization

Consider, for example, the variant in which the user is charged a fixed cost for each registration and is not charged for continuing a registration. One way to represent this model as a metrical task system is to consider each separate mobile user as a "system" of its own. The state consists of the pair (location cell, registration area). Each task the system is given is of the form "go to the given new cell and, if necessary, register with one of the areas available there". The cost of processing the task is defined to be infinite if the current region is not the same as the new requested cell, so as to force the system to move to the same cell as the user. The cost of processing the task is zero if the newly requested cell is the same as the current cell. The cost of a transition between two states is zero if the registration doesn't change (the same registration is used at both locations) and equal to the registration cost if the registration does change. Notice that the registration cost in this model may vary between locations and registration areas.

This formulation is simple and the number of states in the system is not too large, being no more than the product of the number of cells and the maximum number of registration areas available at any location. A nice fact about this formulation is that it is inherently distributed; the computation necessary to determine the next registration may be localized within a network center, the current registration area or the current cell, or even distributed and performed by each mobile separately.

4.2 Simple location management: network optimization

The previous section, as well as the original formulation of the problem [34] have the goal of minimizing the cost incurred by the user. It is assumed that the network service providers set the registration costs but the rationale behind the prices is not considered. The problem of finding a good price for registration may also be formulated as a metrical task system. However, the number of states will be exponential in the number of users and so this formulation may not be very useful. The state of the system is determined by the location and registration of each mobile user. In each step a user may announce relocation and/or re-registration—each such possible announcement defines a task. Other tasks are of the form "find a specific user". The objective of the metrical task system is to perform all the tasks, that is to keep everybody registered and resolve all search requests, but minimize the overall cost (equal to the sum of all search and update costs minus the registration costs charged to the users). Each user may be assumed to follow a rational registration policy, but the model does not depend on this assumption. The number of states is equal to the number of ways the users may be located in the network and have valid registrations. By disaggregating users' actions (so that each task concerns only a single user), the degree of the state graph may be lowered and made polynomial.

4.3 Pointer forwarding

More general cases of the location management problem that include pointer forwarding can also be formulated using metrical task systems. In this setting, an *update* may be deferred upon re-registration and instead a pointer from the old to the new registration area used to locate the user in subsequent search operations. Longer pointer chains may be used as well; the optimal maximum length of a pointer chain will be determined by the frequency and complexity of a search operation as well as the overhead incurred in storing and traversing pointer chains. Information about existing pointer chains will be necessary in addition to the cell and registration to determine the state of the system. Two kinds of tasks exist now: search and relocation. The cost of a search is the cost of following the pointer chain to the correct registration area plus the cost to search the registration area. The cost of a relocation is (as before) infinite unless we are in a state associated with the correct cell, in which case the cost is zero (this will force the system to move to a state corresponding to the user's current cell). Transitions between states model most of the other features. We list a few typical transitions. If only the cell changes, and the registration is available at the new cell, no cost is incurred. If the registration changes (and the new registration is available at the new cell) and the chain of pointers is trivial before and after the transition, the cost of the transition equals the cost of registration. If the cell changes and a pointer is added to the chain, the cost equals that of updating the last element of the existing pointer chain. This cost should equal the cost of updating the pointer chain if the chain is shortened by one link (and this is in fact the only such restriction our model places on the problem). Some other transition costs may be explicitly defined, others are induced by these like distances between nodes of an incomplete graph.

This formulation of the location management problem is surprisingly complete, but it does have a drawback: the number of states is exponential in the maximum allowed pointer chain length. However, under the reasonable assumption of symmetry in pointer update cost (updating the pointer chain costs the same regardless of which cell is added or removed from the chain), the number of states is lowered to a polynomial in the number of cells.

4.4 Multiple registrations

The mobile user may be allowed to register in more than one area at any time. For example, if the user is registered in two areas, no re-registration will be necessary until the user leaves the region accessible to **both** areas. This generalization of the optimal registration problem is as easily modeled by a metrical task system as the original problem. More elaborate tiered schemes may be proposed, for example, a user may maintain a *primary* registration and several *secondary* ones. In such a scheme, independent costs would be incurred when switching a registration from primary to secondary (or vice-versa), or when abandoning a (presumably, secondary) registration in favor of another. This brings us to the description of a second generalization of the optimal registration problem.

4.5 Preemptive and look-ahead registration

If some information about a user's trajectory is known, the user may make a preregistration in a new area before moving out of the current one. Since this registration is "optional" (that is, not absolutely necessary at the moment), some negotiation may be possible and costs may be reduced as compared to the situation in which the registration is needed the very moment the user asks for it. Registrations that must be performed immediately are called *hard*, the others *soft*.

In the metrical task system for location management with multiple hard and soft registration, a state is described by the user's location and the set of his current registrations. Thus to keep the number of states reasonably small, some bound should be placed on the number of concurrent registrations allowed. However, costs assigned to hard and soft registrations will naturally control this parameter even without an explicitly imposed upper bound. Indeed, in the real world, a user may prefer to maintain one or two preregistrations, but more than this will rarely be worthwhile.

5. BETTER ALGORITHMS FOR RESTRICTED MODELS

In this section we present simple online algorithms in several restricted models.

5.1 Simple location management with bounded incidence

Consider the simple location management problem: user optimization with overlapping registration areas, but no other features. In addition, assume that no more than k registration areas are available at any location in the network. Then following these two rules is sufficient to achieve an expected competitive ratio of k:

(a) Do not register until forced to.

(b) When forced to register, pick a random registration area among the ones available.

THEOREM 5.1. *An algorithm that follows rules* **(a)** *and* **(b)** *achieves an expected competitive ratio of k, where k is the maximum number of registration areas available at any single location in the network.*

The main ingredient in the proof is the observation that if an algorithm is to have competitive ratio worse than k, there must exist long (longer than k) subsequences of α in which the algorithm keeps changing the registration while there is a single registration valid for the whole subsequence.

PROOF. Let a denote the average length of a *run* (maximal constant subsequence) in the optimal registration sequence. Let the optimal sequence consist of ℓ constant subsequences $\sigma_1, \ldots, \sigma_\ell$. For each i, denote by t_i the length of the subsequence σ_i. The cost of the optimal sequence equals ℓ, the number of different registrations required, and so $\ell < T/k$ (otherwise *any* algorithm provides a competitive ratio no worse than k). Further, $a = (t_1 + \cdots + t_\ell)/\ell = T/\ell$. These two facts imply that $k < a$, that is, the average run length in the optimal sequence is greater than k.

Suppose now the optimal cost is 1, that is, there is a single registration area valid for all visited locations. Denote as above the number of steps by T. The expected cost of the randomized algorithm is then no more than

$$\begin{aligned}
&1 \cdot \frac{1}{k} + 2 \cdot \frac{1}{k} \cdot \frac{k-1}{k} + \cdots + (T-1)\frac{1}{k} \cdot (\frac{k-1}{k})^{T-2} \\
&+ T(\frac{k-1}{k})^{T-1} \\
=&\frac{1}{k}\sum_{i=0}^{T-1}(\frac{k-1}{k})^i + \frac{1}{k}\frac{k-1}{k}\sum_{i=0}^{T-2}(\frac{k-1}{k})^i + \cdots + \\
&\frac{1}{k}(\frac{k-1}{k})^{T-1} \cdot 1 + T(\frac{k-1}{k})^{T-1} \\
\le&1 + 1 \cdot \frac{k-1}{k} + 1 \cdot (\frac{k-1}{k})^2 + \cdots 1 \cdot (\frac{k-1}{k})^{T-1} + \\
&T(\frac{k-1}{k})^{T-1} \\
\le&k + T(\frac{k-1}{k})^{T-1}.
\end{aligned}$$

To see that the final expression above is $O(k)$, suppose $T = ck$ for some $c \geq 2$. Then $((k-1)/k)^T < (1/e)^c < (1/2)^c < 1/c$ and so $T((k-1)/k)^T < k$.

Now the proof can be completed by partitioning the given sequence into subsequences where the optimal solution is constant. For each of these subsequences the expected cost of the randomized algorithm is no more than k and the cost incurred by the optimal algorithm exactly 1, so the competitive ratio is at most k. $\square$

In fact, the same (but deterministic) performance guarantee is achieved by the following deterministic "marking" algorithm. Use the word *phase* to denote a maximal sequence for which a single registration is sufficient. At the beginning of each phase, the algorithm "unmarks" all the registration areas available at the current location and selects an arbitrary one. In each subsequent step, if an unmarked registration area cannot be used, the algorithm "marks" it. When a new registration is needed, the algorithm selects any unmarked area. The phase ends when all areas are marked. At the beginning of each phase at most k areas are unmarked. In the worst case, the algorithm will have to register in each step in a phase, and the optimal sequence will only use one single registration in a phase. However, the optimal sequence must register at least once per phase and so the competitive ratio of the algorithm is at most k.

5.2 Sliding Window Algorithm for Location Caching

Consider the Home Agent (HA) based location management scheme employed by Mobile IP. To find (search for) a mobile host (MH), a corresponding host (CH) has to contact the mobile host's HA. On the other hand, a mobile host updates it's location information at the HA whenever it's point of attachment changes. Caching the location information at a frequently communicating CH is an optimization technique proposed in the literature. A sliding window protocol has been proposed in [29] and its average case performance is analyzed in [37]. Here we analyze the competitive ratio of this online decision algorithm. This analysis is motivated by analysis of caching scheme in [31].

Define a *schedule* to be a sequence of search operations (s) and location updates (u). A s denotes a search operation initiated by the CH to find location of the MH. An u denotes an update operation performed by the MH. For example, u, s, u, s, s, u, u is a schedule. Although, these operations can occur concurrently, here we assume that they have been serialized (e.g. HA may serialize concurrent requests by processing updates before location find operation).

The basic idea of the sliding window scheme is to cache the location information at a CH when it communicates with the MH more often than the MH moves. Since the MH sees all the operation, it can dynamically decide whether or not the location information should be cached at the CH. Hence, when using location caching, the CH may or may not have cached location of the MH at the time when an operation is issued. In the SW_k (sliding window with window size k) algorithm the MH maintains a sliding window of k operation. Whenever it finds that the number of searches in the window is greater than the number of updates the MH initiates location caching at the CH by sending it an update. On the other hand, when the number of updates becomes greater than the number of searches, then the MH can stop caching of its location at CH. From MH's perspective, the cost of an update operation is 1 when the CH is caching the information and 0 otherwise (Note that MH has to always update the HA and so this cost is the incremental cost). Similarly, from the CH's perspective, the cost of a search operation is 0 when the CH is caching the location information of the MH and 1 otherwise. For a schedule ω, its cost $Cost(\omega)$ is the sum of the cost of each operation in ω. An optimal location strategy is one which minimize the cost of all possible schedules. We use $Cost_{SW_k}()$ to denote the cost of schedule when using SW_k and $Cost_{OPT}()$ to denote the optimal cost.

THEOREM 5.2. *The SW_k algorithm is $k+2$ competitive.*

PROOF. Consider a schedule ω and divide it into maximal blocks consisting of similar operations: $B_1, B_2, \ldots, B_r$. Let N_ω be the number of update operations preceding a search operation in the schedule ω. Obviously,

$$Cost_{OPT}(\omega) = N_\omega.$$

Now lets analyze the maximum cost of SW_k. Note that there are at most $N_\omega + 1$ search blocks or update blocks in ω. Assume that $k = 2l+1$. Now cost of each update block is at most $l+1$ since by the $l+1$-th update the sliding window is guaranteed to have more updates than searches and the MH can inform the CH that it won't be sending it anymore updates with the $l+1$-st update message. Similarly, the cost of each search block is at most $l+2$ since, by $l+1$-th search operation in the block the sliding window is guaranteed to have more search operations than updates and the mobile can inform the CH (at additional cost of 1 message) that it will now be sending it any future updates so the CH can start using the MH's cached location information. Hence, $Cost_{SW_k}(\omega) \leq (N_\omega + 1)(l+1) + (N_\omega + 1)(l+2) = (k+2)N_\omega + (k+2)$.

□

5.3 Dynamic Update Schemes

In [4], three dynamic update schemes have been proposed: 1) distance based, 2) time based, and 3) movement based. In distance based scheme a mobile sends registration whenever it has moved a distance of d cells away from the cell in which it last registered. In time based scheme, the mobile registers periodically every T time units and in the movement based scheme a mobile registers whenever it has crossed M cell boundaries. It is easy to see that none of these are competitive with respect to optimal offline scheme.

We analyze here a simple random update scheme RU_p, $0 \leq p \leq 1$. In this scheme a mobile, whenever it changes its location, sends a location update to its HA with a probability p ($q = 1-p$). Note that RU_0 is the "never update" scheme and RU_1 is the "always update" scheme. To search for a mobile host, the system does an expanding ring search centered at the last known location of the mobile. The cost of the search is taken to be the number of rings needed to be searched in order to find the mobile.

Similar to last section, we define a *schedule* to be a sequence of search operation (s) and location change (or relocation) operation (c). A s denotes a search operation initiated by any CH to find the current location of the MH. A c denotes a relocation of MH from one cell to another cell of the system.

In order to compute the expected competitive ratio of this scheme we make the following assumptions:

1. The location changes of the MH are Poisson distributed with parameter λ_u.
2. The searches for the MH are Poisson distributed with parameter λ_s.
3. The cost of an update operation is c_u.
4. The cost of the first search operation immediately following a move is $c_s * k$, where k is the distance between the user's current cell and its last reported cell and c_s is the search cost for searching a ring of cells centered around any cell in the system. Note that this is the incremental search cost. As a consequence the search cost is zero for a search immediately following a location update. Furthermore we assume that a search results in implicit update of the mobile users current location. Hence, the (incremental) cost of a search operation immediately following another search operation is zero.

Let $\Theta = \frac{\lambda_u}{\lambda_u + \lambda_s}$. Note that, since the Poisson distribution is memoryless, at any point in time Θ is the probability that the next operation in the schedule is a location update operation, and $1-\Theta$ is the probability that the next operation is a search.

THEOREM 5.3. *The RU_p scheme has an expected competitive ratio of $\frac{\Theta}{1-\Theta}(p + q\frac{c_s}{c_u})$, under assumptions 1 - 4.*

PROOF. Consider a schedule ω and divide it into maximal blocks consisting of similar operation: $B_1, B_2, \ldots, B_r$. Let N_ω be the number of relocation operations preceding a search operation in the schedule ω. Obviously,

$$Cost_{OPT}(\omega) = N_\omega c_u.$$

Now lets analyze the maximum cost of RU_p. The expected length of a relocation block in a schedule is $\frac{\Theta}{1-\Theta}$ and so the expected number of updates per relocation block is $\frac{p\Theta}{1-\Theta}$ with cost of $\frac{pc_u\Theta}{1-\Theta}$. The cost of a search operation is at most Mc_s, where M is the random variable which denotes the number of moves without any updates just preceding the search operation. Since the probability of a move with no location update is $q\Theta$, the expected value of M, $E[M]$ is

$\frac{q\Theta}{1-q\Theta}$, which follows from the fact that M is geometrically distributed with parameter $(1 - q\Theta)$ and starts at 0, i.e. $Pr\{M = k\} = q\Theta^k(1 - q\Theta)$, $k = 0, 1, \ldots$. Hence the expected cost of any schedule is less than $(N_\omega + 1)(\frac{pc_u\Theta}{1-\Theta} + \frac{qc_s\Theta}{1-q\Theta}) \leq (N_\omega + 1)(\frac{c_u\Theta}{1-\Theta})(p + q\frac{c_s}{c_u})$. $\square$

6. CONCLUSIONS

We described some possible approaches to studying location management problems as examples of online computation. For the most attractive model, both the positive and negative results translate directly from those on metrical task systems, and unfortunately, the negative results are quite strong so not much improvement seems possible here. However, we believe that, in the spirit of our last section, more restricted models may be useful in both a theoretical and a practical sense.

Acknowledgments. We thank an anonymous referee for helpful comments, and the observation that a simple deterministic algorithm will work for bounded incidence. This work is supported in part by NSF grant ANI-0196156.

7. REFERENCES

[1] B. Awerbuch and D. Peleg. Online tracking of mobile users. *J. ACM*, 42(5):1021–1058, 1995.

[2] B. R. Badrinath, T. Imielinski, and A. Virmani. Locating Strategies for Personal Communication Networks. Workshop on Networking of Personal Communications Applications, Dec. 1992.

[3] A. Bar-Noy and I. Kessler. Tracking mobile users in wireless networks. *IEEE Transactions on Information Theory*, 38:1877–1886, 1993.

[4] A. Bar-Noy, I. Kessler, and M. Sidi. Mobile Users: To Update or not to Update. *Wireless Networks*, 1(2):175–186, 1995.

[5] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 184–193, October 1996.

[6] Y. Bartal. On approximating arbitrary metrics by tree metrics. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 161–168, 1998.

[7] Y. Bejerano and I. Cidon. An Efficient Mobility Management Strategy for Personal Communication Systems. In *Proc. Fourth Annual IEEE/ACM Int'l Conf on Mobile Computing and Networking (MobiCom'98)*, pages 215–222, Dallas, TX, Apr. 1998.

[8] Y. Bejerano and I. Cidon. Efficient location management based on moving location areas. In *Proceedings of the IEEE Infocom*, 2001.

[9] A. Blum, P. Raghavan, and B. Schieber. Navigation in unfamiliar terrain. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 494–504, 1991.

[10] A. Borodin, N. Linial, and M. E. Saks. An optimal on-line algorithm for metrical task system. *J. ACM*, 39(4):745–763, 1992.

[11] M. Charikar, C. Chekuri, A. Goel, S. Guha, and S. Plotkin. Approximating a finite metric by a small number of tree metrics. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 379–388, 1998.

[12] M. Chrobak, H. Karloff, T. H. Payne, and S. Vishwanathan. New results on server problems. *SIAM J. Discrete Math.*, 4:172–181, 1991.

[13] M. Chrobak and L. L. Larmore. An optimal online algorithm for k servers on trees. *SIAM J. Comput.*, 20:144–148, 1991.

[14] S. K. Das and S. K. Sen. Adaptive Location Prediction Strategies Based on a Hierarchical Network Model in Cellular Mobile Environment. In Proceedings of Second International Mobile Computing Conference (IMCC), Mar. 1996.

[15] S. Dolev, D. K. Pradhan, and L. J. Welch. Modified tree structure for location management in mobile environments. In *IEEE INFOCOM '95*, volume 19, pages 335–345, 1996.

[16] A. Fiat and M. Mendel. Better algorithms for unfair metrical task systems and applications. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 725–734, 2000.

[17] A. Fiat, Y. Rabani, Y. Ravid, and B. Schieber. A deterministic $o(k^3)$-competitive algorithm for the circle. Algorithmica, 11:572–578, 1994.

[18] J. S. M. Ho and I. F. Akyildiz. Mobile User Location Update and Paging under Delay Constraints. *ACM-Baltzer Journal for Wireless Networks*, 1(4), Dec. 1995.

[19] J. S. M. Ho and I. F. Akyildiz. Local anchor scheme for reducing signaling costs in personal communications networks. *IEEE/ACM Transactions on Networking*, 4(5):709–725, Oct. 1996.

[20] J. S. M. Ho and I. F. Akyildiz. Dynamic Hierarchical Location Management in PCS Networks. *IEEE/ACM Transactions on Networking*, 5(5):646–660, Oct. 1997.

[21] R. Jain, Y.-B. Lin, C. N. Ho, and S. Mohan. A caching strategy to reduce network impacts of pcs. *IEEE Journal on Selected Areas in Communications*, 12(8):1434–1445, 1994.

[22] J. Jannick, D. Lam, N. S. andJ. Widom, and D. C. Cox. Efficient and Flexible Location Management Techniques for Wirless Communication Systems. *ACM-Baltzer Journal of Wireless Networks*, 3(5):361–374, 1997.

[23] E. Koutsoupias and C. Papadimitriou. On the k-server conjecture. *J. ACM*, 42(5):971–983, 1995.

[24] E. Koutsoupias and C. Papadimitriou. The 2-evader problem. *Information Processing Letters*, 57(5):249–252, 2000.

[25] D. Lam, Y. Cui, D. C. Cox, and J. Widom. A Location Management Technique to Support Lifelong Numbering in Personal Communication Services. Proceedings of the IEEE Global Telecommunications Conference, Nov. 1997.

[26] Y.-B. Lin. Determining User Locations for Personal Communications Services Networks. Technical Report, Bell Communications, Nov. 1993.

[27] L. J. Ng, R. W. Ronaldson and A. D. Malyan Distributed Architectures and Databases for Intelligent Personal Communication Networks. Proceedings of the ICWC, June 1992.

[28] R. Prakash and M. Singhal. Dynamic Hashing + Quorum = Efficient Location Management for Mobile Computing Systems. In Proceedings of ACM Symposium on Principles of Distributed Computing (PODC), p.291, 1997.

[29] S. Rajagopalan and B. R. Badrinath. An Adaptive Location Management Strategy for Mobile IP. MobiCom, 1995.

[30] N. Shivakumar, J. Jannick, and J. Widom. Per-User Profile Replication in Mobile Environments: Algorithms, Analysis and Simulation Results. *ACM-Baltzer Journal of Wireless Networks*, 2(2):129–140, 1995.

[31] A. P. Sistla, O. Wolfson, and Y. Huang. Minimization of communication cost through caching in mobile environments. *IEEE Trans. on Parallel and Distributed Systems*, 4:378–390, 1998.

[32] I.-F. Tsai and R.-H. Jan. The lookahead strategy for distance-based location tracking in wireless cellular networks. *Mobile Computing and Communications Review*, 3(4):27–38, 1999.

[33] G. Varsamopoulos and S. K. S. Gupta. On dynamically adapting registration areas to user mobility patterns in pcs networks. In *Proceedings of International Workshop on Collaboration and Mobile Computing (IWCMC '99)*, Sept. 1999.

[34] G. Varsamopoulos and S. K. S. Gupta. On finding optimal registration in presence of overlapping registration areas. Proceedings of Dial-M, 2001.

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

[36] H.-K. Wu, M.-H. Jin, and J.-T. Horng. Personal paging area design based on mobile's moving behaviors. In *IEEE INFOCOM 2001*, 2001.

[37] R. Yates, C. Rose, S. Rajagopalan, and B. R. Badrinath. Analysis of a mobile-assisted adaptive location management strategy. *Mobile Networks and Applications (MONET)*, 1:105–112, 1996.