Competitive On-Line Paging Strategies for Mobile Users Under Delay Constraints

[Extended Abstract]

Amotz Bar-Noy
Computer&Information Science Department
Brooklyn College
2900 Bedford Ave., Brooklyn, NY 11210
amotz@sci.brooklyn.cuny.edu

Yishay Mansour*
School of Computer Science
Tel Aviv University
Tel Aviv 69978, Israel
mansour@cs.tau.ac.il

ABSTRACT

A mobile user is roaming in a zone of n cells in a cellular network system. When a call for the mobile arrives, the system pages the mobile in these cells since it never reports its location unless it leaves the zone. A delay constraint paging strategy must find the mobile after at most $1 \leq D \leq n$ paging rounds each pages a subset of the n cells. The goal is to minimize the number of paged cells until the mobile is found. Optimal solutions are known for the off-line case, for which an a priori probability of a mobile residing in any one of the cells is known. In this paper we address the on-line case. An on-line paging strategy makes its decisions based only on past locations of the mobile while trying to learn its future locations.

We present deterministic and randomized on-line algorithms for various values of D (number of paging rounds) as a function of n (number of cells) and evaluate them using competitive analysis. In particular, we present a constant competitive on-line algorithm for the two extreme cases of $D = 2$ and $D = n$. The former is the first nontrivial delay constraint case and the latter is the case for which there are no delay constraints. We then show that the constant competitiveness can be attained already for $D \geq \log_2 n$. All of the above algorithms are deterministic. Our randomized on-line algorithm achieves a near optimal performance for all values of D. This algorithm is based on solutions to the best expert problem.

Categories and Subject Descriptors

F.2.2 [**ANALYSIS OF ALGORITHMS AND PROBLEM COMPLEXITY**]: Nonnumerical Algorithms and Problems—*Computations on discrete structures*

*This research was supported in part by a grant from the Israel Science Foundation.

PODC'04, July 25–28, 2004, St. Johns, Newfoundland, Canada.

General Terms

Algorithms

Keywords

Best Experts, Competitive Analysis, Location Management, Mobile Computing

1. INTRODUCTION

In the last decade, we have witnessed two revolutions: availability of information and availability of people. The Internet makes it possible to access information independent of the physical distance. Cellular phone systems make it possible to talk with people even if they are not residing in predetermined locations (as is the case with conventional phone systems). Both the Internet and cellular phone systems require new methods for searching specific information or specific people. For the Internet we see many search engines that improve by far the accessibility of the information. However, we do not see many sophisticated methods for cellular phone systems.

We consider cellular systems with many cells and mobile users (mobiles) that are roaming among the cells ([15, 19]). If a mobile reports its new location whenever it enters a new cell, then the system would know its exact location at any time and therefore finding (paging) this mobile becomes a trivial task. However, future cellular networks are expected to have more cells (micro-cells). As a result, a mobile might cross boundaries of cells very frequently, making it infeasible to report its new location each time it enters a new cell. This is mostly due to the scarcity of up-link wireless communication and the short life of hand-held device batteries. Indeed, many existing location management schemes (reporting and paging strategies) allow mobiles to report less often. For example, a common location management scheme partitions the cells into zones each with a few dozens of cells and a mobile reports its new location only when it crosses boundaries between zones (e.g., [13, 12]). When a call to a mobile is coming, the system need to locate the cell where the mobile currently resides by paging many cells (sometimes all the cells in a particular zone). Although the choice of a location management scheme to minimize the overall use of wireless bandwidth depends on many parameters, the common feature of such schemes is that the system must page a set of cells when a mobile needs to be found.

Suppose that a particular mobile is roaming in a zone of n cells and that it is possible to page any subset of these cells in a unit of time and find out if this mobile is located in one of the cells paged. The trivial solution would page all the n cells at once and clearly use the highest possible amount of wireless bandwidth but would require the lowest possible time. The papers [16, 21, 24] describe how to tradeoff bandwidth for time assuming an a priori knowledge of the likelihood of finding a mobile in any one of the cells. The authors design an efficient paging strategy that uses at most D units of time (D rounds of paging), $1 \leq D \leq n$, and minimizes the expected number of cells paged until the mobile is located. Given the parameters n and D, a paging strategy can be viewed as a *schedule* that partitions the n cells into D disjoint sets. Naturally, we would like to minimize both the number of paging rounds and the overall number of cells paged until the mobile is found. These are our two main criteria in evaluating the efficiency of a specific paging strategy. The first corresponds to the delay incurred until the mobile is found which is important to the mobiles and the second corresponds to the amount of wireless bandwidth used which is crucial to the system.

Previous schedules where *off-line* in their nature. They assumed an a priori knowledge of probabilities for the locations of the mobiles. This paper addresses the *on-line* case in which such probabilities are not known in advance. We further assume that the time elapsed between two successive paging events is large enough to avoid dependencies between the whereabouts of the mobile during these paging events. This is a reasonable assumption for "quiet" mobiles (those who do not use the system frequently) or for fast moving mobiles. Moreover, these assumptions enable us to focus on the paging strategy alone to better understand its nature and difficulties.

We are looking for a "good" paging strategy for a sequence of T paging events. We distinguish between on-line algorithms and off-line algorithms by restricting the on-line algorithm to select for any paging event a schedule based only on past locations of the mobiles. We consider *static* off-line algorithms, that use the same schedule for all the paging events. An optimal static off-line algorithm can be computed given the frequencies of locations for a mobile based on its locations in all the T paging events. Considering a *dynamic* off-line algorithm, one that knows the future locations, is not interesting either theoretically or practically. Theoretically, since it always has minimal cost because it pages only one cell. Practically, because this would require a huge encoding, growing linearly with T, which seems impossible to implement. Therefore, our competitive ratio would be the worst case ratio between the on-line cost and the optimal static off-line cost. Note that since we restrict ourself to static off-line schedules, the on-line algorithm may outperform it, however, in the worst case the competitive ratio would be at least 1.

Our work draws interesting connections with the *best expert problem*, which has been extensively studied in the computational learning theory literature (see [20, 11, 4, 17]). In this problem, there are several experts, and in each time step each expert suggests an action. An on-line algorithm needs to select an action based on one (or more) of the experts. After the on-line algorithm performs its action, it observes the *loss* of all the experts. The objective is to minimize the total loss. A specific performance measure is to compare the on-line algorithm total loss to that of the best expert whose total loss is minimal among all experts. This is done in an adversarial setting, where the adversary controls the loss of each expert, and no assumptions, stochastic or otherwise, are made regarding an expert's future behavior given its past performance. In our setting, we can view any possible static off-line schedule as an expert (there are exponentially many). Since the known bound depends only logarithmically on the number of experts, the overall bound is polynomial. The main difficulty in using directly the best expert algorithms is computational, since we need to keep track on an exponential number of experts.

Prior art and related work: Modelling uncertainty of mobile locations as a probability distribution vector is studied in e.g., [23]. The paper discusses a framework for measuring uncertainty. A similar technique for modelling uncertainty of mobile locations through k-Markov model and algorithms for finding mobiles is studied in [9]. Papers [16, 21, 24, 18] describe the optimal off-line solution for one paging event for given n and D. Using dynamic programming, they show how to find a D-round paging strategy that locates a mobile with minimum expected number of cells paged. Using relaxation to a continuous model, paper [24, 18] studies how to minimize the expected number of paged cells given a bound on the expected number of rounds. Papers [1, 27, 28] present sub-optimal heuristics that are computationally more efficient than the dynamic programming optimal algorithm. The problem of off-line paging more than one user for a conference call is studied in [6, 7, 14].

The combined cost of reporting and paging is studied by many papers (see the survey [2]). The main issue in this line of research is for mobiles to reduce the overall wireless cost by reporting their new locations according to some rules. The combined cost of backbone search, reporting, and paging is studied in a different model by [3, 5, 8, 22]. The effect of queuing on paging delay when search requests arrive to the system according to some random process and each request is to search for a single user is studied by [25, 16].

The paper [9] also deals with on-line performance but for the combined cost of paging and reporting and with an analysis that depends on some mobility patterns. Moreover, the solutions rely on the time elapsed between reporting and paging events.

The best expert problem has been extensively studied in the computational learning theory community and tight bounds have been derived on its behavior [20, 11, 4]. In a nutshell, one can reach almost the loss of the best expert, assuming that it is allowed to output a linear combination of the experts (or alternatively allowed to use randomization). An extension of the best expert algorithm to a randomized online algorithm appear in [17, 10].

Our Contribution: Our first contribution is the introduction of a model to evaluate on-line paging strategies. Our main objective is to show that natural algorithms with a correct setting of parameters yield constant competitive performance. We then develop both deterministic and randomized on-line algorithms. The deterministic algorithms are analyzed in two stages. We first compare the on-line strategy with a near-optimal off-line strategy and then compare the near-optimal strategy with the optimal strategy. The final competitive ratio is the product of the two ratios computed in these two stages. The randomized algorithms are based on solutions to the best experts problem.

We present the following deterministic algorithms. We describe a 6-competitive greedy algorithm for $D = 2$. Unfortunately, we show that a natural generalization of this algorithm to the $D = 3$ case fails. For $D = n$, we show a $2 - \epsilon$ competitive algorithm and prove that no deterministic on-line algorithm could perform better. For $D = \log_2 n$, we show a 3-competitive algorithm. The same algorithm and bound holds for $D \geq \log_2 n$, since we compare the performance to that of the off-line schedule with $D = n$. We generalize the algorithm for $D = \log_2 n$ into a $h+1$-competitive algorithm for the $D \approx \log_h(n)$, which implies an $O(n^{1/D})$-competitive algorithm for a general $D \leq \log_2 n$.

We present the following randomized algorithms. We show a randomized on-line algorithm whose competitiveness is $1 + \epsilon$. However, this competitive factor is attained for a "long" sequence of paging events. We describe another algorithm for $D = 2$ that converges faster. This improved algorithm generalizes to any D with a running time that is polynomial in n but exponential in D.

We note that the three cases $D = 2$, $D = n$, and $D = \log_2 n$ are very important. $D = 2$ is the first nontrivial case. When $D = n$ there are no delay constraints. The $D \geq \log_2 n$ case is important since we achieve a small constant competitive ratio while reducing the number of rounds from n to $\log n$. We leave open the problem of finding "good" competitive deterministic algorithms for all values of D between 2 and $\log n$.

Paper organization: Due to page limitations, we omitted some of the proofs. Section 2 describes the off-line model and the on-line model. Sections 3, 4, and 5, respectively, present the near-optimal off-line algorithms, the competitive deterministic algorithms, and the competitive randomized algorithms. Some open problems are discussed in Section 6.

2. MODEL

We first describe the off-line model in details. This enables us to come with a clearer description of the on-line environment. Assume a system with n cells denoted by $1, \ldots, n$. Mobiles are roaming in the system and a vector distribution p of length n is given, where p_i is the probability that the mobile is located in cell i. Assume that the mobile always *exists*, and hence $\sum_{i=1}^{n} p_i = 1$. Without loss of generality, $p_1 \geq p_2 \geq \cdots \geq p_n$. Given a subset S of $[1, n]$, let $p(S) = \sum_{i \in S} p_i$. For $S = [1, k]$ let $p(S) = q_k = \sum_{i=1}^{k} p_i$.

Paging a mobile is conducted in rounds. In each round, a subset of the cells is paged until a subset that contains the actual location of the mobile is found. The goal is to minimize the expected number of cells paged under the constraint that the paging must be complete in at most D rounds for a given parameter $1 \leq D \leq n$. In other words, the goal is to find an ordered partition of the cells into D disjoint subsets to minimize the expected number of paged cells. We call this partition a *schedule*.

Given an ordered partition $\mathcal{S}$ of the cells, let $cost(\mathcal{S})$ be the expected number of paged cells given the distribution p. Denote by $OPT(D)$ the cost of the optimal schedule for D paging rounds.

Note that by definition, some of the subsets in the partition may be empty. However, it is always better to page at least one cell in each round. The following observation relates the location probabilities and the order of paging the cells.

Observation 1. *The paging is conducted following the non-decreasing order of probabilities. That is, for any pair of rounds $1 \leq i < j \leq D$, all the cells that are paged in round j are associated with smaller or equal probabilities than all the cells that are paged in round i.*

To gain some intuition, let us consider a few special cases for the value of D. For $D = 1$, all the cells must be paged in the first and only round. Trivially, the cost of this strategy is n. For $D = 2$, the goal is to find a subset of the cells to be paged in the first round. Then in the second round, if the mobile is not found, the rest of the cells must be paged. By Observation 1, the goal is to find a number $1 \leq k \leq n$ such that in the first round only the cells $[1, k]$ are paged. Denote by T_k the schedule that pages the cells $[1, k]$ in the first round and recall that $q_k = p(T_k)$. It follows that

$$\begin{aligned} cost(T_k) &= q_k k + (1 - q_k)n \\ &= n - q_k(n - k) \\ &= k + (1 - q_k)(n - k) \ . \end{aligned} \tag{1}$$

For $D = n$, the paging is implied by Observation 1. Simply, the optimal schedule OPT pages cell i in round i for $1 \leq i \leq n$. In this case the cost is

$$cost(OPT) = OPT(n) = \sum_{i=1}^{n} i p_i \ . \tag{2}$$

In general, for any $1 \leq D \leq n$, let $\mathcal{S} = \{S_1, \ldots, S_D\}$ be an ordered partition of $[1, n]$, and let n_i be the size of the subset S_i. Then

$$cost(\mathcal{S}) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} n_j \right) p(S_i) = \sum_{i=1}^{D} \left(\sum_{j=i}^{D} p(S_j) \right) n_i \ . \tag{3}$$

The above definitions are for one paging event with a given set of probabilities. Suppose instead that a sequence of T paging events is given where at each time t, for $1 \leq t \leq T$, the mobile may be located in any one of the n cells. Let ℓ_t be the location of the mobile in time t and $m_i(t)$ be the number of times the mobile was in cell i during the time interval $[1, t]$. An algorithm is *on-line*, if at each time t, it decides on a schedule based only on the mobile locations at times $t' < t$. An algorithm is *static off-line* (off-line in short) if it is given the entire location sequence (or equivalently, it is given a frequency vector p, where $p_i = m_i(T)/T$) and outputs a single schedule that is used for all paging events. We compare an on-line algorithm which is a sequence of T schedules to an optimal off-line schedule, and the competitive ratio would be the worst case ratio between the on-line and the off-line costs. We do not make any stochastic assumptions regarding the locations of the mobile.

3. OFF-LINE SCHEDULES

In this section, we describe near-optimal off-line schedules for various values of D and compute their approximation factors when compared with the optimal off-line schedule. This is our first stage in proving competitiveness of on-line algorithms. In the second stage, we compute the competitive ratio of an on-line algorithm when compared with a near-optimal schedule. The competitive ratio of the on-line algorithm compared to the optimal schedule is the product of the two previously computed ratios. The reason we

cannot directly compare our on-line algorithm to the optimal off-line schedule is that this schedule is found via a dynamic programming and therefore does not have a simple structure. On the other hand, our near-optimal off-line algorithms are very simple and structured enabling us to compare them with our on-line algorithms.

In the first subsection, we address the $D = 2$ case and discuss several natural greedy strategies that are based on the properties of the optimal off-line schedule. We then describe the simple solution for the other extreme of $D = n$ when there are no delay constraints. In the last subsection, we address the entire range of D. For the $D = \log_2 n$ case, we show a constant approximation, which implies a constant approximation for any $D \geq \log_2 n$. We also derive cruder relationships for the range $3 \leq D \leq \log_2 n$. This is because except for the case $D = 2$, the value for $OPT(n)$ serves us as the value of $OPT(D)$ when computing approximation factors of off-line schedules and competitive factors of on-line strategies.

3.1 Two paging rounds

Assume that $D = 2$. In this case, a schedule is simply a partition of the n cells into two sets. Recall that by Observation 1, the first set must be of type $[1, k]$ for some $1 \leq k < n$. The following lemma shows a simple formula to determine the optimal partition.

Lemma 2. *For $D = 2$ paging rounds, the optimal subset S_{opt} of cells that are paged in the first round is $[1, k]$ such that $p_k > q_{k-1}/(n-k)$ and $p_{k+1} \leq q_k/(n-k-1)$.*

PROOF. Assume that it is known that in the first round the optimal strategy pages cells $[1, k]$, and we would like to see if it is worth to page cell $k + 1$ as well. By Equation 1, it follows that if

$$kq_k + (1 - q_k)n > q_{k+1}(k + 1) + (1 - q_{k+1})n$$

then the optimal strategy should page cell $k+1$ as well. This implies that,

$$p_{k+1} \quad > \quad \frac{q_k}{n-k-1}$$

Since p_k decreases with k, q_k increases with k, and $1/(n - k - 1)$ increases with k, there is a single crossing point and we proved the following theorem. □

Based on the above lemma, we derive some useful properties of the optimal partition.

Corollary 3. *For $D = 2$ paging rounds, the optimal subset S_{opt} of cells that are paged in the first round has the following properties:*

1. *If $p_k \geq 2/n$ then $k \in S_{opt}$.*
2. *$|S_{opt}| \leq n/2$.*
3. *$p(S_{opt}) \geq 1/2 - 2/n$.*
4. *If $p_{k+1} \leq 1/2n$ then $k + 1 \notin S_{opt}$.*

PROOF. For property (1), note that there are at most $n/2$ cells for which $p_k \geq 2/n$. Therefore, $q_k/(n-k) \leq 1/(n/2) = 2/n \leq p_k$, for $p_k \geq 2/n$ and the claim follows by Lemma 2.

For property (2), by Theorem 2, $p_k > q_k/(n - k)$ if cell $k = \lceil n/2 \rceil$ is paged. Since, $q_k \geq kp_k$, it follows that $p_k > (k/(n - k))p_k$ which can hold only for $k < n/2$.

For property (3), assume that $S = [1, k - 1]$ and therefore $p(S_{opt}) = q_{k-1}$. Since k is not paged, Theorem 2 implies that $p_k(n - k) \leq q_k$. By the order of the probabilities, it follows that $p([k + 1, n]) \leq p_k(n - k)$ and hence $p([k + 1, n]) \leq q_k$. Since $q_k + p([k + 1, n]) = 1$, it follows that $q_k \geq 1/2$. As a result, $p(S) + p_k \geq 1/2$. The claim follows from property (1).

For property (4), note that $q_k \geq 1/2$, since $1 - q_k \leq n(2/n) = 1/2$. Since $q_k/(n - k) > 1/2n$, by Theorem 2 we have that $p_{k+1} \notin S_{opt}$. □

The above corollary gives rise to three types of greedy schedules. Recall that the goal is to find the best schedule T_k that pages the cells $[1, k]$ in the first round. In other words, we are looking for a set S of cells to be paged in the first round.

`Threshold cell probability:` Add to S all the cells whose probabilities are greater or equal to x for some parameter $x > 0$. For a given x, define S_x to be this set: $S_x = \{i | p_i \geq x\}$. In the optimal strategy, by the first property of Corollary 3, we know that $x \leq 2/n$.

`Threshold number locations:` Take $S = [1, k]$ (i.e., strategy T_k) for some value of k that can be a function of n and the distribution vector p. In the optimal strategy, by the second property of Corollary 3, we know that $k < n/2$.

`Threshold round probability:` Add p_i to S if $p(S \cup \{i\}) \leq r$ for some $0 \leq r \leq 1$. For a given r, define R_r to be this set: $R_r = \{i | q_i \leq r \text{ and } q_{i+1} > r\}$. In the optimal strategy, by the third property of Corollary 3, we know that r must be greater or equal to $1/2 - 2/n$.

The following claims demonstrate that each of the three greedy schedules gives a very different quality of approximation. On one hand `Threshold cell probability` has 2 approximation for the threshold $x = 2/n$. On the other hand both `Threshold number locations` and `Threshold round probability` do not have a constant approximation factor. Our on-line algorithm will be compared to the Threshold cell probability schedule.

Claim 4. *For the `Threshold cell probability` schedule, $cost(S_{2/n}) \leq 2cost(S_{opt})$ where $S_x = \{i | p_i \geq x\}$.*

PROOF. By Corollary 3 we know that $S_{2/n} \subset S_{opt}$. It is easy to see that the worst case competitive ratio is when $S_{2/n} = \emptyset$, in which case $cost(S_{2/n}) = n$. For the optimal we have,

$$\begin{aligned} cost(S_{opt}) &= |S_{opt}| + (1 - p(S_{opt}))(n - |S_{opt}|) \\ &= n - p(S_{opt})(n - |S_{opt}|) \\ &\geq n - |S_{opt}|\frac{2}{n}(n - |S_{opt}|) \end{aligned}$$

where we used the fact that all the locations in $S_{opt} \setminus S_{2/n}$ have probability at most $2/n$. Now, consider the following ratio:

$$\begin{aligned} \frac{cost(OPT)}{cost(S_{2/n})} &\geq \frac{n - |S_{opt}|\frac{2}{n}(n - |S_{opt}|)}{n} \\ &\geq 1 - 2\frac{|S_{opt}|}{n} + 2(\frac{|S_{opt}|}{n})^2 \end{aligned}$$

The minimum ratio is at $|S_{opt}|/n = 1/2$ which gives a competitive ratio of 2. □

Claim 5. *For the* `Threshold number locations` *schedule with parameter* $k = \sqrt{n}$, $cost(S_{opt})\sqrt{n} \geq cost(T_{\sqrt{n}})$ *Also, for any value of the parameter* k, $cost(T_k) = \Omega(\sqrt{n}\, cost(S_{opt}))$.

PROOF. If $|S_{opt}| \geq \sqrt{n}$, we are done since $cost(T_{\sqrt{n}}) \leq n$. If $|S_{opt}| \leq \sqrt{n}$, we have $S_{opt} \subset T_{\sqrt{n}}$. This implies that the probability of reaching the second round is larger with S_{opt} than $T_{\sqrt{n}}$. Since $|S_{opt}|\sqrt{n} \geq |T_{\sqrt{n}}| = \sqrt{n}$ we are done.

For the lower bound consider two scenarios, depending on the value of k. The scenario first, for $k \geq \sqrt{n}$, has almost probability one on a single location. The optimal cost is 1 and the cost of T_k is k. The second scenario, for $k < \sqrt{n}$ has probability $1/2k$ assign to $2k$ locations. The optimal has cost $2k$ and T_k has cost $k + (1/2)(n-k) > n/2$, and the ration is $\Omega(n/k) = \Omega(\sqrt{n})$. □

Claim 6. *For any parameter* r *of the* `Threshold round probability` *schedule, there exists a probability vector* p, *and a set* R_r, *such that* $p(R_r) \geq r$, $cost(R_r) \geq n/2$, *and* $cost(S_{opt}) < 3$.

PROOF. For $r \leq 1 - 1/n$, let $p_1 = 1/2$, $p_2 = 1/2 - 1/n$ and $p_i = 1/(n(n-2))$. Then $cost(S_{opt}) = 3 - 1/n$. For $r \in [1/2, 1 - 1/n]$ we have $S = \{1\}$ and otherwise $S = \emptyset$, therefore $cost(S) = (n+1)/2$ or $cost(S) = n$.

For $r \geq 1 - 1/n$ let $p_1 = 1 - 2/(n-1)$, and $p_i = 2/(n-1)^2$. Then $cost(S_{opt}) = 3 - 2/(n-1)$ and $cost(S) = n$, since $S = \emptyset$. □

3.2 n paging rounds

Let $D = n$ which is the case where there are no restrictions on the number of rounds. Recall that $m_i(T)$ is the number of times the user was in location i during time interval $[1, T]$. Clearly, $\sum_{i=1}^{n} m_i(T) = T$. Fix the time T, set $m_i = m_i(T)$, and without loss of generality assume that $m_1 \geq m_2 \geq \cdots \geq m_n$. By Observation 1, the optimal schedule pages the cells in the order of their frequencies which is equivalent to the order of their probabilities.

Theorem 7. *For* $D = n$ *paging rounds, the optimal off-line schedule is to page cell* i *at round* i *until the mobile is found, and its cost is* $OPT(n) = \sum_{i=1}^{n} i \cdot m_i$.

3.3 Arbitrary number of paging rounds

We first describe an off-line schedule for $D = \lceil \log_2 n \rceil$ rounds. For a clearer presentation, we assume that $n = 2^D - 1$ and that $D = \log_2 n$ although $D = \log_2(n+1) = \lceil \log_2 n \rceil$. We call this schedule L_2. Assume again that $m_1 \geq m_2 \geq \cdots \geq m_n$. The main idea is that L_2 doubles the size of the set of cells it pages in a round if it fails to find the mobile in the previous round.

Schedule L_2: In round i, $1 \leq i \leq D$, L_2 pages the cells $[2^{i-1}, 2^i - 1]$ following the non-decreasing order of m_i. In particular, L_2 pages cell 1 in the first round, cells 2 and 3 in the second rounds, and the last 2^{D-1} cells in the last round.

We compare L_2 with $OPT(n)$ and not $OPT(D)$ since we do not know the structure of $OPT(\log_2 n)$. Surprisingly the approximation factor is only 2.

Lemma 8. *The cost of* L_2 *is less than* $2 \cdot OPT(n)$.

PROOF. Fix a round i for $1 \leq i \leq D$. Note that the size of the set of cells paged in this round is 2^{i-1}. When L_2 pages the cells $[2^{i-1}, 2^i - 1]$ in this round, it is because the mobile was not found in cells $[1, 2^{i-1} - 1]$. This paging is successful exactly $\sum_{j=2^{i-1}}^{2^i-1} m_j$ times for a total cost of $(2^i - 1)\sum_{j=2^{i-1}}^{2^i-1} m_j$. This is because the number of cells paged in the first i rounds is $2^i - 1$. It follows that the overall cost of all rounds is

$$Cost(L_2) = \sum_{i=1}^{D} (2^i - 1) \sum_{j=2^{i-1}}^{2^i-1} m_j \ .$$

Note that the coefficient of m_j is at most $2j$, and therefore

$$Cost(L_2) \leq \sum_{j=1}^{n} 2j \cdot m_j \ .$$

The result is now implied by Theorem 7. □

The next theorem follows since we compare the schedule L_2 with $OPT(n)$ and since by adding "dummy" paging rounds any off-line schedule for x rounds can be transformed to a schedule for $y > x$ rounds with at least the same performance.

Theorem 9. *For any* D *paging rounds,* $\log_2 n \leq D \leq n$, *there exists an off-line schedule whose cost is at most twice the optimal off-line schedule cost.*

We now generalize L_2 to a schedule L_h for an integer $2 \leq h \leq \log_2 n$. Again, we omit all ceilings and assume that $n = \frac{h^D - 1}{h-1}$. The main idea is that instead of doubling the size of the new set of cells to be paged, L_h multiplies this size by h.

Schedule L_h: In round i, $1 \leq i \leq D$, L_h pages the cells $[\frac{h^{i-1}-1}{h-1} + 1, \frac{h^i-1}{h-1}]$ following the non-decreasing order of m_i.

Lemma 10. *The cost of* L_h *is less than* $h \cdot OPT(n)$.

PROOF. Fix a round i for $1 \leq i \leq D$ and let $a_i = \frac{h^{i-1}-1}{h-1} + 1$ and $b_i = \frac{h^i-1}{h-1}$. By definition, L_h pages the cells $[a_i, b_i]$ in round i. Note that L_h pages h^i cells in round i since $b_i - a_i + 1 = h^i$. When L_h pages these cells, it is because the mobile was not found in cells $[1, a_i - 1]$. Thus, a successful round i pages b_i cells. Since this happens $\sum_{j=a_i}^{b_i} m_j$ times, it follows that the total cost of a successful round i is $b_i \sum_{j=a_i}^{b_i} m_j$. Therefore, the overall cost of all rounds is

$$Cost(L_h) = \sum_{i=1}^{D} b_i \sum_{j=a_i}^{b_i} m_j \ .$$

Note that if $j \in [a_i, b_i]$ then the coefficient of m_j is b_i. Since $b_i/a_i < h$, we get that this coefficient is at most $h \cdot j$, and therefore

$$Cost(L_h) \leq \sum_{j=1}^{n} h \cdot j \cdot m_j \ .$$

The result is now implied by Theorem 7. □

With proper definitions of sets to be paged in each round, we can define a schedule L_x for any real number $2 \leq x \leq \log_2 n$ and can prove that the cost of L_x is at most $x \cdot OPT(n)$. Fixing D and setting x accordingly imply the following

Theorem 11. *For any* $3 \leq D \leq \log_2 n$ *paging rounds, there exists a schedule whose cost is at most* $n^{1/D}$ *times* $OPT(n)$.

PROOF. Fix D in the range $[3, \log_2 n]$. We are looking for x such that

$$n = \frac{x^D - 1}{x - 1} .$$

This happens when $x \approx n^{1/D}$. The lemma follows by Lemma 10. □

4. COMPETITIVE DETERMINISTIC ON-LINE ALGORITHMS

In this section, we describe on-line deterministic algorithms for various values of D. We compare them with the near-optimal off-line schedules from the previous section by computing their relative competitive ratio. Particularly, We present a 6-competitive greedy algorithm for$D = 2$. We show that a natural generalization of this algorithm to $D = 3$ fails. For $D = n$, we show a $2 - \epsilon$ competitive algorithm and prove that no deterministic on-line algorithm could perform better. For $D = \log_2 n$, we show a 3-competitive algorithm that generalizes to a $h + 1$-competitive algorithm for $D \approx \log_h n$. The former implies a 3-competitive algorithm for $\log_2 n \leq D \leq n$ and the latter implies an $O(n^{1/D})$-competitive algorithm for $D \leq \log_2 n$.

4.1 Two paging rounds

Assume that $D = 2$. In this case, an on-line algorithm, at any time t, partitions the n cells into two sets. Therefore, a full specification of the algorithm is to define a set of cells to be paged in the first round of any paging event. We first design an on-line algorithm assuming that the number of paging events T is given in advance, and we are interested only in the cost after the last paging event.

`Algorithm Cell Threshold I (CL-I):` Let Q_t be the set of all cells i whose corresponding $m_i(t)$ is greater than $2T/n$ times, i.e., $Q_t = \{i : m_i(t) \geq 2T/n\}$. At time $t+1$, page the cells of Q_t in the first round.

Theorem 12. *For $D = 2$ paging rounds, algorithm* `Cell Threshold I` *is 6 competitive.*

PROOF. We compare the performance of algorithm `CL-I` to the schedule that pages the set of cells $S_{2/n}$ composed of all the cells of final frequency $2/n$ after T paging events. Note that Q_t is monotone and that $Q_T = S_{2/n}$. Therefore, the cost of both `CL-I` and $S_{2/n}$ on the cells not in $S_{2/n}$ is n. For a cell ℓ in $S_{2/n}$, after the first $2T/n$ times that the mobile was in cell ℓ, the cost of `CL-I` is at most that of $S_{2/n}$. This implies that for the above two cases the cost of `CL-I` is either the same or less than $S_{2/n}$. During the first $2T/n$ times that the mobile is in cell $\ell \in S_{2/n}$, we have that the cost of `CL-I` is n while that of $S_{2/n}$ is $|S_{2/n}|$. This implies that

$$\begin{aligned} cost(\texttt{CL-I}) &\leq cost(S_{2/n}) + |S_{2/n}|(n - |S_{2/n}|)2T/n \\ &\leq cost(S_{2/n}) + 2T|S_{2/n}| \\ &\leq 3cost(S_{2/n}) . \end{aligned}$$

The last inequality follows since each paging round in $S_{2/n}$ has cost of at least $|S_{2/n}|$. By Claim 4, the cost of $S_{2/n}$ is at most twice the cost of S_{opt}. Hence, `CL-I` is 6 competitive. □

We now omit the assumption that the value of T is known in advance and define the following on-line algorithm that relies on the local threshold $2t/n$ instead of the global threshold $2T/n$.

`Algorithm Cell Threshold II (CL-II):` Let Q_t be the set of all cells i whose corresponding $m_i(t)$ is greater than $2t/n$ times, i.e., $Q_t = \{i : m_i(t)/t \geq 2/n\}$. At time $t + 1$ page in the first round the cells of Q_t.

Theorem 13. *For $D = 2$ paging rounds, algorithm* `Cell Threshold II` *is 6 competitive.*

PROOF. As for algorithm **CL-I**, we will compare the performance to that of the off-line schedule that pages the set of cells $S_{2/n}$ composed of all the cells of final frequency $2/n$ after T paging events whose cost is at most twice the optimal cost. However, we do the accounting somewhat differently. For cells not in $S_{2/n}$, we sum the cost when they where paged (which is at most n) and the number of times they are in Q_t. For cells in $S_{2/n}$, each time they are paged, they will be charged as follows: (1) if they are in Q_t they are charged $|Q_t \cap S_{2/n}|$, which is bounded by $|S_{2/n}|$ (2) if they are not in Q_t they are charged n.

We claim that the sum of the charges bounds the online cost. We show this according to the type of the cell. If a cell is not in $S_{2/n}$ or a cell is in $S_{2/n}$ and not in Q_t, we charge n, which clearly upper bounds the cost of the paging round. For a cell in $S_{2/n} \cap Q_t$, we charge $|S_{2/n} \cap Q_t|$ the cell we paged, however the online incurs a cost of $|Q_t|$. For each cell in $Q_t \setminus S_{2/n}$, we charge one for each time it is in Q_t, and therefore, we have that this additional charge is $|Q_t \setminus S_{2/n}|$. Summing the two types of charges gives a total of $|Q_t|$, which is the online cost.

By definition, cells that are not in $S_{2/n}$ have frequency $k < 2T/n$. Such a cell can be in at most $kn/2$ different Q_t. Its cost is at most kn, when it is paged, and at most $kn/2$ when other cells are paged. Summing the two, we bound the cost by $(3/2)kn$. In the $S_{2/n}$ schedule, this cell cost is kn.

Cells in $S_{2/n}$ are paged at least $\ell \geq 2T/n$ times, out of which at most $k < 2T/n$ times they are not in Q_t. The online cost of such a cell is bounded by $\ell|S_{2/n}| + kn$, which is bounded by $\ell|S_{2/n}| + 2T$. The cost of that cell in $S_{2/n}$ is $\ell|S_{2/n}|$. Summing over all cells in $S_{2/n}$, we get a cost of $|S_{2/n}|(2T + \sum_{i \in S_{2/n}} \ell_i)$, which is bounded by $3T|S_{2/n}|$. Clearly the cost of $S_{2/n}$ is at least $T|S_{2/n}|$.

Consequently, algorithm **Cell Threshold II** is at most 3 times the cost of $S_{2/n}$ and is 6-competitive by Claim 4. □

One might hope that a similar scheme can be used for $D = 3$. Namely, have two thresholds, α and β, and define an algorithm $A(\alpha, \beta)$ as follows. All the cells with frequencies more than α are paged in the first round. All the cells with frequencies between α and β are paged in the second round. In the third round, all the cells with frequencies less than β are paged. The following theorem states that this would yield a "bad" approximation even in the off-line setting.

Theorem 14. *There exists a vector of frequencies p, such that $cost(A(\alpha, \beta)) = \Omega(n^{1/5} cost(OPT(3)))$ for any α and β.*

PROOF. Let ρ be the competitive ratio of algorithm $A(\alpha, \beta)$. Consider the following three scenarios and denote by ρ_1, ρ_2, ρ_3 the ratio between the cost of the algorithm and the optimal cost in the three scenarios respectively.

Scenario I: There is one cell with probability $1 - \sqrt{\alpha}$, $1/\sqrt{\alpha}$ cells with probabilities slightly more than α, and the other

cells have negligible probabilities. The optimal schedule pages the first cell in the first round, pages the $1/\sqrt{\alpha}$ cells in the second round, and pages the rest of the cells in the third round. The cost of the optimal schedule is at most 2. Algorithm $A(\alpha,\beta)$ pages all the $1+1/\sqrt{\alpha}$ "heavy" cells in the first round, with a cost of $1+1/\sqrt{\alpha}$. Hence,

$$\rho_1 \geq \frac{1+\frac{1}{\sqrt{\alpha}}}{2} \geq \frac{1}{2\sqrt{\alpha}} .$$

Scenario II: There are $1/\beta$ cells with probabilities slightly less than β, while the other cells have negligible probability. The optimal schedule pages half of the $1/\beta$ cells in the first round, the other half in the second round, and the rest of the cells in the third round. The optimal cost is $3/(4\beta)$. Algorithm $A(\alpha,\beta)$ pages all the cells in one round and has cost n. Hence,

$$\rho_2 \geq \frac{n}{\frac{3}{4\beta}} \geq \beta n .$$

Scenario III: There are $(1-\epsilon)/\alpha$ cells with probabilities slightly less than α, ϵ/β cells with probabilities slightly more than β, and the other cells have negligible probabilities. The optimal schedule pages the $(1-\epsilon)/\alpha$ "heavy" cells in the first round, the next ϵ/β cells in the second round, and the rest of the cells in the third round. The optimal cost is $(1-\epsilon)/\alpha+\epsilon^2/\beta$. The algorithm $A(\alpha,\beta)$ pages all the "heavy" cells in one round and has cost $(1-\epsilon)/\alpha+\epsilon/\beta$. Set $\epsilon=\sqrt{\beta/\alpha}$. The cost of the optimal schedule is at most $2/\alpha$. The cost of algorithm $A(\alpha,\beta)$ is at least $1/\sqrt{\alpha\beta}$. Hence,

$$\rho_3 \geq \frac{\frac{1}{\sqrt{\alpha}\sqrt{\beta}}}{\frac{2}{\alpha}} \geq \frac{\sqrt{\alpha}}{2\sqrt{\beta}} .$$

We now prove that the competitive ratio of algorithm $A(\alpha,\beta)$ is $\Omega\left(n^{1/5}\right)$ for the following three cases for the values of α and β.

Case $1/n > \alpha > \beta$: Consider only the first scenario. Thus $\rho \geq \rho_1$, and therefore

$$\rho \geq \frac{1}{2\sqrt{\alpha}} \geq \frac{\sqrt{n}}{2} = \Omega\left(n^{1/2}\right) .$$

Case $\alpha \geq 1/n > \beta$: Consider scenarios I and III. Thus $\rho \geq \max\{\rho_1,\rho_3\}$, and therefore

$$\rho \geq \max\left\{\frac{1}{2\sqrt{\alpha}}, \frac{\sqrt{\alpha}}{2\sqrt{\beta}}\right\} = \Omega\left(n^{1/4}\right).$$

Case $\alpha > \beta \geq 1/n$: Consider all three scenarios. Thus $\rho \geq \max\{\rho_1,\rho_2,\rho_3\}$, and therefore

$$\rho \geq \max\left\{\frac{1}{2\sqrt{\alpha}}, \beta n, \frac{\sqrt{\alpha}}{2\sqrt{\beta}}\right\} = \Omega(n^{1/5}).$$

□

4.2 n paging rounds

Assume that $D = n$. We introduce an on-line algorithm that "imitates" the optimal schedule using the frequencies known before the t-th paging event. We show, unlike the setting in [17, 10], that there is no need to add randomization to achieve a good competitive ratio.

`Follow the optimal (FTO):` After the paging event t, compute the optimal off-line schedule up to time t, and apply it in time $t+1$.

Theorem 15. *The competitive ratio of algorithm* `Follow the optimal` *is* $2-2/(n+1)$*, and no deterministic on-line algorithm has a better competitive ratio.*

Proof. We first prove the upper bound. The cost associated with cell i is the cost of finding the mobile when it is located in cell i. Denote this cost by c_i. First consider cell 1, the cell in which the mobile is located the most times. In the last m_1-m_2 times that the mobile was in cell 1, this cell was the most frequently visited, and therefore `FTO` paged it first. This contributes a cost of m_1-m_2 to c_1. Similarly, there are m_2-m_3 times where cell 1 was paged in at most the second round. In general, there are m_k-m_{k+1} times `FTO` finds the mobile in cell 1 after at most k rounds. This implies that c_1 is bounded by

$$c_1 \leq \sum_{i=1}^{n-1} i\cdot(m_i-m_{i+1}) + nm_n = m_1 + \sum_{i=2}^{n} m_i$$

In general for c_k, we claim that at least m_i-m_{i+1} times cell k was paged after at most i rounds, for $i \geq k$. This implies that c_k is bounded by

$$c_k \leq \sum_{i=k}^{n-1} i\cdot(m_i-m_{i+1}) + nm_n = km_k + \sum_{i=k+1}^{n} m_i$$

Note that the first term of c_i contributes i to the coefficient of m_i and that the second terms of all c_k, for $k<i$, contribute $i-1$ to this coefficient. Thus, summing the cost associated with all cells yields a bound of

$$\sum_{i=1}^{n} c_i \leq \sum_{i=1}^{n}(2i-1)m_i = 2\left(\sum_{i=1}^{m} i\cdot m_i\right) - T = 2\cdot OPT(n) - T .$$

It follows that $OPT(n)$ is maximized when $m_i = T/n$ for each cell i. This implies that $OPT(n) \leq (n+1)T/2$, which establishes an upper bound of $2-2/(n+1)$ on the competitive ratio.

For the lower bound, consider the following scenario. At each time t, the adversary forces the on-line to conduct n rounds. This is always possible since the adversary knows the on-line paging order. Therefore, the cost of any on-line algorithm could be nT. However, the cost of the optimal schedule is bounded by $OPT(n) \leq (n+1)T/2$ even when all the frequencies are the same. This establishes the lower bound claim. □

4.3 Arbitrary number of paging rounds

We first introduce an on-line algorithm `Follow` L_2 (FL_2) that operates as the L_2 schedule using the sequence $m_1(t) \geq m_2(t) \geq \cdots \geq m_n(t)$ instead of the final sequence $m_1 \geq m_2 \geq \cdots \geq m_n$. Again, for clarity of presentation we assume that $n = 2^D-1$ and that $D = \log_2 n$ although $D = \log_2(n+1) = \lceil \log_2 n \rceil$. Recall, that the main idea of L_2 and therefore of FL_2 is to double the size of the set of cells it pages in a round if it fails to find the mobile in the previous round.

`Algorithm Follow-`L_2`:` In round $1 \leq i \leq D$ of time t , this algorithm pages the cells $[2^{i-1}, 2^i-1]$ following the non-decreasing order of $m_i(t)$. In particular, FL_2 pages cell 1 in the first round, cells 2 and 3 in the second rounds, and the last 2^{D-1} cells in the last round.

We show that FL_2 is 3-competitive for $D = \log_2 n$. We first compare the cost of FL_2 to L_2 and show that it is 3/2-competitive. Then we obtain the desired result since

L_2 is a 2-approximation to the optimal off-line schedule by Lemma 8.

Theorem 16. *The competitive ratio of* `Follow-`L_2 *for* $D = \log_2 n$ *paging rounds is at most* 3.

The next theorem follows since we compare the schedule FL_2 with $OPT(n)$ and since, by adding "dummy" paging rounds, any on-line algorithm for x rounds can be transformed to an on-line algorithm for $y > x$ rounds with at least the same performance.

Theorem 17. *For* $\log_2 n \leq D \leq n$ *paging rounds, there exists a* 3*-competitive on-line algorithm.*

We now generalize FL_2 to an on-line algorithm FL_h for an integer $2 \leq h \leq \log_2 n$. Again, we omit all ceilings and assume that $n = \frac{h^D-1}{h-1}$. The main idea is that instead of doubling the size of the new set of cells to be paged, FL_h multiplies this size by h.

`Algorithm Follow-`L_h`:` In round $1 \leq i \leq D$ of time t , this algorithm pages the cells $[\frac{h^{i-1}-1}{h-1} + 1, \frac{h^i-1}{h-1}]$ following the non-decreasing order of $m_i(t)$.

We show that FL_h is h+1-competitive for $D \approx \log_h n$. We first compare the cost of FL_h to L_h and show that it is $(h+1)/h$-competitive. Then we obtain the desired result since FL_h is a h-approximation to the optimal off-line schedule by Lemma 10.

Theorem 18. *The competitive ratio of* `Follow-`L_h *for* $D \approx \log_h n$ *paging rounds is at most* $h + 1$.

With proper definitions of sets to be paged in each round, we can define an on-line algorithm FL_x for any real number $2 \leq x \leq \log_2 n$. Moreover, we can prove that the competitive ratio of FL_x compared to F_x is at most $(x+1)/x$. Theorem 11 implies that the competitive ratio compared to $OPT(n)$ is $x + 1$. Fixing D and setting x accordingly imply the following theorem.

Theorem 19. *For any* $3 \leq D \leq \log_2 n$ *paging rounds, there exists an on-line algorithm whose competitive ratio is at most* $1 + n^{1/D}$.

5. ALMOST OPTIMAL RANDOMIZED ON-LINE ALGORITHMS

In this section we apply the methodology of the best experts [20, 11, 4, 17] to derive an almost optimal on-line algorithm. The objective is that for "long" periods of time, we will have an almost optimal cost. The basic idea is to use the on-line best expert algorithms, and to associate an expert with each possible off-line schedule. The best expert algorithm is able to track the lowest cost expert, i.e., the best off-line schedule. When we come to implement this idea there is a clear computational difficulty, since the number of experts, even for $D = 2$, is exponential in n. (The works [10, 17] deal with a similar phenomena.)

We propose two ways to overcome this computational difficulty. The first is using a recent result of [17] and showing how it applies to our case for any value of D. The second on-line algorithm uses an interesting simulation technique of the best experts algorithms that we developed for the special case of $D = 2$. This technique can be generalized, for $D > 2$, to an on-line algorithm whose running time is exponential in D. Thus the algorithm runs in polynomial time only for constant D. The benefit of the second algorithm is that it attains the near-optimality for a much smaller value of T (the number of paging events).

5.1 Arbitrary number of paging rounds

Our result in this section is a direct application of [17] for general experts. They assume that each expert is associated with a vector $\pi \in \mathcal{R}^n$ and that the cost at time t is associated with a vector $c_t \in [0,1]^n$. In [17] it is shown how to track the best expert in polynomial time when the following two conditions are satisfied:

1. The loss of expert π on cost vector c_t is $< \pi, c_t >$.
2. There exists a polynomial time algorithm, that given any cumulative cost vector $C_T = \sum_{t=1}^{T} c_t$, finds the expert π that minimizes the cost, i.e., minimizes $< \pi, C_T >$.

Let us discuss an encoding of the cost and the experts. If at time t, the mobile was located in cell i, we represent this cost by a length n vector c_t that has 1 in entry i and 0 elsewhere. Note that $C_T = \sum_{t=1}^{T} c_t$ is simply the vector $m_i(T)$. For each possible schedule $\mathcal{S} =< S_1, \ldots, S_D >$, let $n_i = |S_i|$ be the number of cells paged in round i and let r_j be the round in which cell j is paged. We associate with the schedule $\mathcal{S}$ a vector π where $\pi(j)$ is the number of cells paged when the mobile is in cell j, i.e., $\pi(j) = \sum_{i=1}^{r_j} n_i$. We denote the schedule by π. Note that π is a unique representation of the schedule and that its length is n. The cost of schedule π on the input sequence is simply $< C_T, \pi >= \sum_i m_i(T)\pi(i)$. This answers the requirement 1 of [17].

For the second requirement, we need to provide a polynomial time algorithm that given C_T finds the best off-line schedule π for C_T. This can be done using the dynamic programming algorithm [16, 21, 24]. This answers requirement 2 of [17].

Based on the above two properties, we can derive the proof of Theorem 20 directly from the general result in [17].

Theorem 20. *For any* D *paging rounds, there exists a randomized on-line algorithm whose expected cost is bounded by* $OPT(n)+O(Dn\sqrt{T})$ *for any sequence of* T *paging events.*

The next corollary follows since each paging round has a cost of at least one (even in the optimal schedule).

Corollary 21. *For any* D *paging rounds, there exists a randomized on-line whose expected cost is bounded by* $(1+\epsilon)OPT(n)$ *for* $T = \Omega(D^2n^2/\epsilon^2)$.

5.2 Two paging rounds

In this section, we derive an alternative almost optimal on-line algorithm, R-Hedge, for the special case of $D = 2$. The main benefit of the algorithm is a smaller additive bound (which implies faster convergence time). This alternative algorithm is done by directly simulating the known best expert algorithms [20, 11, 4]. The following is the performance bound of the algorithm.

Theorem 22. *For* $D = 2$*, there exists a randomized on-line algorithm* R-Hedge*, such that for any input sequence* c_t*, the expected cost of* A *is bounded by* $OPT + O(\sqrt{nT} + n)$.

Comparing the bounds of Theorems 20 and 22, there is a difference of a $O(\sqrt{n})$ factor in the additive term. This implies that Theorem 22 has a guarantee of $(1+\epsilon)OPT$ for smaller values of T, i.e., $T = \Omega(n/\epsilon^2)$. A slight weakness of the second approach, aside from its running time for large values of D, is that it requires T, the number of paging events, to be fixed in advanced.

The rest of this section describes *R-Hedge* that would run the Hedge algorithm ([4]). The R-Hedge algorithm would do a perfect simulation of Hedge and therefore its performance would be identical to that of Hedge. The main contribution would be in the running time of the algorithm (since a naive implementation would require exponential time). Recall that $m_i(t)$ is the number of times that the mobile was in cell i during $[1,t]$. Let $m_S(t)$ be $\sum_{i\in S} m_i(t)$.

In the case of $D = 2$ an expert π is define by a set Q_π which is paged in the first time step. The cost of expert π up to time t is $L^t(\pi) = L^t(|Q_\pi|, m_{Q_\pi}(t))$, where $L^t(s,k) = sk+(t-k)n$. The weight of an expert π at time t is $w^t(\pi) = b^{L^t(\pi)}w^0(\pi)$, where $b \in (0,1)$ is a constant which is predetermined by the algorithm (and may depend on the final number of steps taken, i.e., T, but not on the input sequence). The Hedge algorithm needs to choose randomly a set Q_π based on the distribution w^t.

We start with the following observation on the structure of the weights. (The observation follows directly from the definition of the weights.)

Observation 23. *Let π_1 and π_2 be two different experts. If $|Q_{\pi_1}| = |Q_{\pi_2}|$ and $m_{Q_{\pi_1}}(t) = m_{Q_{\pi_2}}(t)$ then $w^t(\pi_1) = w^t(\pi_2)$.*

The above observation suggests that we can divide the possible sets to equivalence groups, based on their size and their cost. Now we will have two tasks. The first is to select an equivalence class (with the right probabilities) and the second is to select a set from a given equivalence class.

A helpful tool for R-Hedge would be the ability to count the size of an equivalence class. Let $f(n,s,k)$ be the number of sets $S \subset \{1,\ldots,n\}$ of size s whose cost (up to time t) is k. Note that,

$$f(n,s,k) = f(n-1,s,k) + f(n-1,s-1,k-m_n(t))$$

where the boundary conditions are: $f(n,s,k) = 0$ for $n < s$, $f(n,s,k) = 0$ for $k < 0$, and $f(n,s,k) = 1$ for $s = 0$ and $k = 0$. One can easily see that in time $O(n^2T)$ we can compute all values of $f(n,s,k)$, since s has at most $n+1$ values and k has at most T values.

Given $f(n,s,k)$, define $W^t(s,k) = f(n,s,k)b^{L^t(s,k)}/2^n$. The value of $W^t(s,k)$ is the total weight assigned to an equivalence class. That is, the sum of all the weights of sets of size s that had cost k up to time t. Algorithm R-Hedge, at the start of time step t, computes $W^t(s,k)$ for every s and k.

Recall that the Hedge algorithm chooses a set randomly based on the distribution w^t. The R-Hedge will perform the random selection in two steps. In the first step R-Hedge selects randomly a pair (s,k) based on the distribution induced by $W^t(s,k)$. In the second step, given the pair (s,k), it randomly selects the set S of size s and cost k. This is done by selecting the elements of S one at a time. More precisely,

$$Pr[n \in S] = \frac{f(n-1,s-1,k-m_n(t))}{f(n,s,k)}$$

In general, assume we selected randomly S_j for elements from $\{j,\ldots,n\}$, then

$$Pr[j-1 \in S_{j-1} | S_j \subset \{j,\ldots,n\}] = \frac{f(j-1,s-1-|S_j|,k-m_{S_j}(t)-m_j(t))}{f(j,s-|S_j|,k-m_{S_j}(t))}$$

The following properties of R-Hedge follows from is definition, and complete the proof of Theorem 22.

Claim 24. *The algorithm R-Hedge runs in time $O(n^2T)$ and for any input sequence the probability that R-Hedge selects π is $w^t(\pi)$.*

PROOF. For the running time, note that $f(i,s,k)$ has at most $n+1$ values for i and s, and at most T values for k. The claim on the probability follows from the fact that $w^t(\pi) = W(s,k)/f(n,s,k)$, where $s = |Q_\pi|$ and $k = L^t(\pi)$. □

6. OPEN PROBLEMS

- We do not have deterministic algorithms for $D = 3$ and a constant D with a good competitive ratio. However, we know that `follow the optimal` is competitive for the $D = 2$ case and we conjecture that it is competitive for all values of D. For most of our on-line algorithms, we compared them with $OPT(n)$. This is the main reason why the performance deteriorated when D approaches 2. We do not know how to compare our algorithm with $OPT(D)$. For randomized algorithms, it will be interesting to find faster algorithms that achieve an almost optimal performance for a small value of T.

- An interesting research topic is to design solutions for paging several mobiles together. The off-line case was presented and studied in [6, 7].

- We ignored the reporting operation. In the next research stage, one could look for competitive algorithms for a system with some reporting schemes. See the survey [2] for the many papers that addressed the off-line case.

- The case of n paging rounds resembles the list update problem ([26]). The cost function is different since in this problem the algorithm "pays" for changing the order of the list where in our model the cost of changing the schedule in each round is free. Nevertheless, it seems that the Move-to-Front algorithm ([26]) which is a 2-competitive algorithm for the list update problem will have the same competitive ratio in our model. In a way, our follow the optimal algorithm is simpler and more natural. Also, it seems like the $1+\epsilon$ ratio randomized algorithm of [10] will work in our model with some modifications. We will explore further the relationship between the two problems in the full version of the paper.

Acknowledgement

We thank the PODC reviewers for their helpful comments.

7. REFERENCES

[1] A. Abutaleb and V. O. K. Li. *Paging Strategy Optimization in Personal Communication Systems.* ACM/Baltzer Wireless Networks Journal (WINET), **3:** 205–216 (1997).

[2] I. F. Akyildiz, J. Mcnair, J. S. M. Ho, H. Uzunalioğlu, and W. Wang. *Mobility Management in Next-Generation Wireless Systems.* IEEE Proceedings Journal, **87:** 1347–1385 (1999).

[3] N. Alon, K. Kalai, M. Ricklin, and L. .J. Stockmeyer. *Lower Bounds on the Competitive Ratio for Mobile User Tracking and Distributed Job Scheduling.* Theoretical Computer Science (TCS), **130:** 175–201 (1994)

[4] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. *Gambling in a rigged casino: The adversarial multi-armed bandit problem.* the 36th Symposium on Foundations of Computer Science (FOCS), 322–331 (1995).

[5] B. Awerbuch and D. Peleg. *Online Tracking of Mobile Users.* Journal of the ACM (JACM), **42:** 1021–1058 (1995).

[6] A. Bar-Noy and G. Malewicz, *Establishing Wireless Conference Calls Under Delay Constraints.* Journal of Algorithm, **51:** 145–169 (2004).

[7] A. Bar-Noy and Z. Naor. *Establishing a Mobile Conference Call Under Delay and Bandwidth Constraints.* The 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), 2004.

[8] Y. Bartal, A. Fiat, and Y. Rabani. *Competitive Algorithms for Distributed Data Management.* Journal of Computer and System Science (JCSS), **51:** 341–358 (1995).

[9] A. Bhattacharya and S. K. Das. *LeZi-Update: An Information-theoretic framework for personal mobility tracking in PCS networks.* ACM/Baltzer Wireless Networks Journal (WINET), **8:** 121–135 (2002).

[10] A. Blum, S. Chawla, and A. Kalai. *Static Optimality and Dynamic Search-Optimality in Lists and Trees.* Algorithmica, **36:** 249 – 260 (2003).

[11] N. Cesa-Bianchi, Y. Freund, D. P. Helmbold, D. Haussler, R. E. Schapire, and M. K. Warmuth. *How to use expert advice.* The 25th ACM Symposium on Theory of Computing (STOC), 382–391 (1993).

[12] EIA/TIA. *Cellular Radio-Telecommunications Intersystem Operations.* EIA/TIA Technical Report IS-41 Revision C (1995).

[13] ETSI/TC. *Mobile Application Part (MAP) Specification, version 4.8.0.* Technical Report, Recommendation GSM 09.02 (1994).

[14] R .H. Gau and Z. J. Haas. *Concurrent Search for Mobile Users in Cellular Networks.* IEEE Communications Letters, **7:** 287–289 (2003).

[15] D. J. Goodman. *Cellular Packet Communications.* IEEE Transactions on Communications, **38:** 1272–1280 (1990).

[16] D. J. Goodman, P. Krishnan, and B. Sugla. *Minimizing Queuing Delays and Number of Messages in Mobile Phone Location.* Mobile Networks and applications (MONET), **1:** 39–48 (1996).

[17] A. Kalai and S. Vempala. *Efficient Algorithms for On-line Optimization.* The 16th Annual Conference on Learning Theory (COLT), 26–40 (2003).

[18] B. Krishnamachari, R. H. Gau, S. B. Wicker, and Z. J. Haas. *Optimal Sequential Paging in Cellular Networks.* ACM/Baltzer Wireless Networks Journal (WINET), **10:** 121–131 (2004).

[19] W. Lee. *Mobile Cellular Telecommunications Systems.* New York: Mcgraw-Hill (1989).

[20] N. Littlestone and M. K. Warmuth. *The weighted majority algorithm.* Information and Computation, **108:** 212–261 (1994).

[21] S. Madhavapeddy, K. Basu, and A. Roberts. *Adaptive Paging Algorithms for Cellular Systems.* Wireless Information Networks: Architecture, Resource Management and Mobile Data 83–101 (1996)

[22] S. J. Mullender and P. B. M. Vitányi. *Distributed Match-Making.* Algorithmica, **3:** 367–391 (1988).

[23] C. Rose and R. Yates. *Location Uncertainty in Mobile Networks: a Theoretical Framework.* IEEE Communications Magazine, **35:** (1997).

[24] C. Rose and R. Yates. *Minimizing the Average Cost for Paging Under Delay Constraints.* ACM/Baltzer Wireless Networks Journal (WINET), **1:** 211–219 (1995).

[25] C. Rose and R. Yates. *Ensemble Polling Strategies for Increased Paging Capacity in Mobile Communication Networks.* ACM/Baltzer Wireless Networks Journal (WINET), **3:** 159–167 (1997).

[26] D. Sleator and R. E. Tarjan. *Amortized Efficiency of List Update and Paging Rules.* Communications of the ACM, **28:** 202–208 (1985).

[27] W. Wang and I. F. Akyildiz. *An Optimal Paging Scheme for Minimizing Signaling Costs Under Delay Bounds.* IEEE Communication Letters, **5:** 43–45 (2001).

[28] W. Wang, I. F. Akyildiz, and G. L. Stüber. 6*Effective Paging Schemes with Delay Bounds as QoS Constraints in Wireless Systems.* ACM/Baltzer Wireless Networks Journal (WINET), **7:** 455–466 (2001).