

# Finding Mobile Data: Efficiency vs. Location Inaccuracy <sup>\*</sup>

Amotz Bar-Noy<sup>1,2</sup> and Joanna Klukowska<sup>2</sup>

<sup>1</sup> Computer Science Department, Brooklyn College, CUNY,

<sup>2</sup> Computer Science Department, The Graduate Center, CUNY

amotz@sci.brooklyn.cuny.edu

jdklukowska@gc.cuny.edu

**Abstract.** A token is hidden in one out of  $n$  boxes following some known probability distribution and then all the boxes are locked. The goal of a searcher is to find the token in at most  $D \leq n$  rounds by opening as few boxes as possible, where in each round any set of boxes may be opened. We design and analyze strategies for a searcher who does not know the exact values of the probabilities associated with the locked boxes. Instead, the searcher might know only the complete order or a partial order of the probabilities, or ranges in which these probabilities fall. We show that with limited information the searcher can find the token without opening significantly more boxes compared to a searcher who has full knowledge. This problem is equivalent to finding mobile users (tokens) in cellular networks (boxes) and finding data (tokens) in sensor networks (boxes).

**Key words:** wireless networks, sensor networks, location management, mobile computing

## 1 Introduction

Consider the following combinatorial game. A *token* is placed and hidden in one out of  $n$  *boxes* following some probability distribution. The boxes are then locked and the only information about the location of the token is the probability distribution. A *searcher* needs to find the token as fast as possible while opening a minimum number of boxes. The searcher is given  $D$ ,  $1 \leq D \leq n$ , rounds to find the token where, in each round the searcher may open any set of locked boxes.

More formally, let  $C_1, C_2, \dots, C_n$  be the  $n$  boxes and let  $\vec{p} = \langle p_1, p_2, \dots, p_n \rangle$ ,  $\sum_{i=1}^n p_i = 1$ , be the vector of probabilities associated with the  $n$  boxes. We call this vector the *prediction vector* or the *probability vector*. Without loss of generality, assume that  $p_1 \geq p_2 \geq \dots \geq p_n$ . Let  $D$  be the number of *rounds* by which the token must be found.  $D$  is the *delay constraint* for the searcher. In particular, if the token is not found after  $D - 1$  rounds, then the searcher must open all the remaining boxes in the last round. Clearly, in order to minimize the

---

<sup>\*</sup> This research was made possible by NSF Grant number 6531300

number of opened boxes, the searcher should open the boxes following a non-decreasing order of their probabilities. That is, for  $i < j$ , any box opened at round  $i$  should be associated with at least as large a probability as that associated with a box opened at round  $j$ . Therefore, a search strategy is equivalent to partitioning the boxes into  $D$  disjoint sets (rounds). This partitioning is defined by the vector  $\bar{d} = \langle d_0, d_1, d_2, \dots, d_D \rangle$  where by definition  $d_0 = 0$  and  $d_D = n$ . The meaning of this vector is that if the token was not found during the first  $i - 1$  rounds, then in round  $i$  the searcher opens the boxes  $C_{d_{i-1}+1}, \dots, C_{d_i}$ .

Let **ALG** denote a search strategy for constructing the vector  $\bar{d}$  based on the vector  $\bar{p}$ . The cost of searching for the token in  $D$  rounds using **ALG** is defined as the expected number of boxes opened until the token is found. If the token is in box  $C_i$  and that box is opened in round  $j + 1$  (i.e.,  $d_j < i \leq d_{j+1}$ ), then a searcher that follows the strategy **ALG** would open in total  $d_{j+1}$  boxes. Thus:

$$Cost_{ALG} = \sum_{j=1}^D \left( d_j \times \sum_{i=d_{j-1}+1}^{d_j} p_i \right). \quad (1)$$

The cost for any searcher is largest when  $D = 1$  because all the boxes have to be opened in the first and only round and therefore the cost is  $n$ . On the other hand, the cost for an optimal searcher is minimized if it has  $D = n$  rounds. In this case, the boxes should be opened one per round following the non-increasing order of their probabilities yielding a  $\sum_{i=1}^n p_i i$  cost. For other values of  $D$ , the cost for  $D$  rounds is smaller than the cost for  $D'$  rounds for  $n \geq D > D' \geq 1$ . A simple formula for the cost does not exist for  $1 < D < n$  rounds. Nevertheless, dynamic programming solutions exist that find the optimal partitioning in polynomial time for any  $1 \leq D \leq n$ , [1], [2], [3], [4].

In order to achieve the optimal partitioning, one needs to have an accurate knowledge of the prediction vector. The scope of this paper is to explore how the cost is affected when only partial information about the values of the probabilities is given. One motivation for exploring the tradeoff between such *location inaccuracies* and *search efficiency* is the fact that there exist situations in which there is no benefit in having the complete knowledge of the prediction vector. For example, if the number of allowed rounds is equal to the number of boxes ( $D = n$ ), then a searcher who knows only the relative order of the probabilities is as efficient as a searcher who has complete knowledge. We show that there are other situations in which a searcher with a partial knowledge obtains results very close to a searcher who has a complete knowledge of probabilities.

**Applications:** This general search game was introduced for the problem of searching for a mobile user in a cellular network. In this setting, the token is a cellphone holder (user) and the boxes are the cells in which a particular user can be located. Opening a box is interpreted as paging the user in a cell using the wireless communication links. Finding the token fast is equivalent to finding the user while paging as few as possible cells.

A cellular phone system is composed of many cells and mobile users. The system has to be able to locate a mobile when phone calls arrive. The cost of delivering the call to a user is lower if the exact location of that user is known.

To that end, users can report their position whenever they cross boundaries of cells. However, reporting consumes wireless bandwidth (*uplink*) and battery power, and it is desirable for a user to report infrequently in order to save these resources. On the other hand, if the user does not report often enough, then when calls arrive, the system needs to perform a search that pages the user in each cell in which the user may be located. This method consumes the wireless bandwidth (*downlink*) resource. A compromise solution is a division of the cells into *location areas* (*zones*), each composed of several cells [5], [6]. Users report only when crossing boundaries of location areas. When a call to the user occurs, the system needs to search only within a given zone.

The cost of searching for a user depends upon the particular search strategy used. When a call for a user arrives, the system can either perform a *blanket search* in which all cells are paged at the same time, or a *sequential search* in which the cells are paged one after another. The first strategy guarantees that the user is found after one *round*, but uses the most downlink bandwidth. The second increases the number of rounds to possibly as many as the number of cells in the zone, but uses on average the minimum downlink bandwidth. Assuming that, at the time of the call arrival, the system knows the probabilities associated with every cell in the zone indicating the likelihood of the user being in that cell, the system can use a dynamic programming solution to find the user optimally for any given number of rounds.

The scope of this paper is to design efficient search strategies for locating a user even when the prediction vector is not accurate. The inaccuracy can be caused by, for example, limited tracking methods used by the system, large irregularities in users' mobility patterns, or even intentional decreases in accuracy of location information in an attempt to give the user more *privacy*. Sometimes an accurate prediction vector reveals a lot of information about the user and may even open up potential security risks. Some users may prefer to maintain a higher level of privacy even at a higher price. For an individual user, the delay in delivery of a call that occurs when the system needs to page all possible locations is not noticeable and not important as long as the call is received. On the other hand, keeping one's whereabouts private may be valued a lot. We show that the system can offer more privacy to its users by storing only an approximation to the location probability vector without paying significantly more for delivering each call.

Another application is in wireless sensor networks. The sensors are not mobile, however the system does not always know where to find specific data that is gathered by the sensors. Hence, one can view this data as mobile data. Nevertheless, the system might know the probability (or an estimate probability) of finding the answer at each sensor. One can view the sensors as the boxes and the answer to the query as the token. Opening few boxes is translated into communicating with few sensors which is a paramount objective in sensor networks due to battery limitations for the sensors. Using only a few rounds is equivalent to finding an answer to a query quickly.

**Our Contributions:** We analyze and evaluate the performance of strategies for a searcher that has limited information about the probability vector. We compute the *worst case ratio*,  $\rho$ , between algorithms that are based on some limited information and the optimal solution that relies on complete knowledge of the prediction vector. We also compare the solutions via a simulation assuming the Zipf distribution for the prediction vector. Let  $Opt(D)$  be the cost of the algorithm that has complete knowledge of the probability vector and has  $D$  rounds to find the token.

Our first contribution is the analysis of the case where the searcher knows only the relative order of the values in the prediction vector. Following [7], we propose algorithm *DRoot* whose performance in the worst case is  $\Theta(\sqrt[D]{n})Opt(n)$  and we prove that no other algorithm can perform better. According to the performance analysis results, the cost computed by this algorithm is much closer to the optimal solution. The performance of the algorithm depends on the allowed delay in finding the token. When that delay is equal to the number of boxes, *DRoot* finds a solution with optimal cost.

Our second contribution is the analysis of various scenarios with only partial knowledge of the prediction vector. In general the probabilities are assigned to  $B$  bins, where the relative order of the probabilities between the bins is known but within a bin the relative order of the probabilities is unknown. We analyze the performance of only sequential search given that limited information. We discuss several rules for bin assignment. *Assignments based on size* considerations guarantee a cost that is  $\frac{n}{B}Opt(n)$  in the case of bins with the same size, and  $\Theta(n^{1/B})Opt(n)$  when the sizes of bins vary exponentially. Using *assignments based on threshold* considerations, the computed worst case costs are similar:  $\Theta(\frac{n}{B})Opt(n)$  for *linear threshold bins*, and  $\Theta(n^{1/B})Opt(n)$  for *exponential threshold bins*. For *assignments based on optimal solution for 2 rounds*, the worst case cost for a searcher is  $O(\sqrt{n})Opt(n)$ . Our simulation results demonstrate that the theoretical bounds are truly limited to the worst case and on average the cost computed with the limited knowledge offered by different types of bins is close to optimal.

**Related Work:** Modeling uncertainty of location of mobiles as a probability distribution vector was studied in [8], which also presented a framework for measuring uncertainty. Location management schemes were discussed in [5], [6]. The optimal solution for the partitioning of  $n$  cells into  $1 \leq D \leq n$  rounds was proposed by [1], [2], [3]. Recently the running time of the dynamic programming method has been improved by the authors of [4]. In [9] the authors presented a suboptimal solution which is computationally more efficient than dynamic programming. Other variations of the problem have been proposed. One of them is paging multiple users for a conference call, [10]. Another is the added effect of queuing requests to search for single users and performing a search for several mobiles at once, [2], [11], [12].

Privacy issues have been raised in recent years regarding the tradeoff between privacy and the benefits of location based applications [13], [14], [15]. Most of the research is based on assuming that the wireless carrier is a trusted party

and developing algorithms to limit access of third parties to the location data of mobiles, [14], [15]. In [13] the author provides a survey of the technology, issues, and regulations related to location-aware mobile devices.

## 2 Relative Order

One of the motivations for this research was the fact that knowing only the relative order of probabilities associated with the boxes is enough to enable the sequential search strategy (i.e.,  $D = n$ ) to achieve optimal cost. In this case, the searcher can find a token with the same cost as if he/she knew the exact value of the prediction vector. In this section, we decrease the allowed delay and examine how well the searcher can perform. Note that knowing only relative order makes  $\langle 1, 0, \dots, 0 \rangle$  and  $\langle 1/n, \dots, 1/n \rangle$  indistinguishable. A searcher cannot achieve the optimal cost since the strategies for opening these boxes associated with the two vectors may be very different. The authors of [7] show that for  $D \geq \log_2(n)$  the searcher can find the token with cost at most two times larger than an optimal solution that uses complete knowledge, and that for  $D < \log_2(n)$  the cost is  $\Theta(n^{1/D})Opt(n)$ .

Let  $DRoot$  be the algorithm that creates  $D$  partitions by assigning  $d_i = \lfloor n^{i/D} \rfloor$  for  $1 \leq i \leq D$ .

**Theorem 1 (from [7]).** *The cost of searching for a token, given only the relative ordering of the boxes, computed by  $DRoot$  using at most  $D$  rounds is  $O(n^{1/D})Opt(n)$  in the worst case.*

In the next theorem, we show that  $DRoot$  is optimal.

**Theorem 2.** *The lower bound on the performance in the worst case for any algorithm that is given only the relative order of the probabilities associated with the boxes is  $\Omega(n^{1/D}Opt(D))$ , for  $D < \log_2(n)$ .*

*Proof.* Let  $\langle d_1, d_2, \dots, d_D \rangle$  be the partition vector used by an arbitrary algorithm,  $ALG$ . By the pigeon hole principal, there is at least one  $d_j$  such that  $d_j/d_{j-1} \geq n^{1/D}$  (for  $j = 1$ , assume that  $d_0 = 1$ ; this allows us to treat  $d_1$  the same as other  $d_j$ 's). The assumption that  $D < \log_2 n$  implies that  $n^{1/D} > 2$  and therefore  $d_j > 2d_{j-1}$ . Consider a distribution that has  $2d_{j-1}$  boxes each with probability  $1/(2d_{j-1})$  and the remaining boxes with probability equal to zero. For this distribution the algorithm with the above partitions assigns  $d_{j-1}$  boxes with nonzero probability to the  $j^{th}$  round. Ignoring the cost that this algorithm has to pay for the previous  $j - 1$  rounds, its total cost is:

$$Cost_{ALG} \geq \left( \frac{1}{2d_{j-1}} d_{j-1} \right) d_j = \frac{1}{2} d_j = \frac{1}{2} d_{j-1} n^{1/D} .$$

The optimal cost has to be no greater than the solution for 2 rounds in which the first  $d_{j-1}$  boxes are opened in the first round, and the next  $d_{j-1}$  boxes are opened in the second round. Therefore,

$$Cost_{Opt} \leq \left( \frac{1}{2d_{j-1}} d_{j-1} \right) d_{j-1} + \left( \frac{1}{2d_{j-1}} d_{j-1} \right) 2d_{j-1} = \frac{3}{2} d_{j-1} .$$

These two costs give a lower bound on the ratio of any algorithm and the optimal solution:  $\rho \geq n^{1/D}/3$ .

*Remark:* For  $D = 2$  it can be shown that the lower bound is larger,  $\rho \geq n^{1/D}/\sqrt{3}$

### 3 Bins

A natural next step in limiting the knowledge available to a searcher is a *prediction bin* strategy. We represent the knowledge of a searcher by bins that are groupings of boxes based on some characteristic. The motivation for this limitation in the applications described in the introduction is that it might be easier to estimate the values of the probabilities this way, than trying to arrange them in the complete relative order. The  $B$  bins,  $b_1, b_2, \dots, b_B$ , are a partition of the set of boxes,  $C_1, C_2, \dots, C_n$ , with probabilities  $p_1, p_2, \dots, p_n$ , such that  $C_i \in b_k$  and  $C_j \in b_{k+1}$  implies that  $p_i \geq p_j$ , where  $1 \leq i, j \leq n$  and  $1 \leq k < B$ . We do not know anything about the relationship of boxes that are in the same bin; i.e., if  $C_i, C_j \in b_k$ , then  $p_i \geq p_j$  or  $p_i < p_j$ .

We consider several criteria for putting boxes into bins. In *size bin*, each bin contains some predefined number of boxes, for example, uniform size with  $\frac{n}{B}$  boxes per bin. When  $B = n$  this uniform size bin assignment is equivalent to the relative order case discussed in the previous section. In *threshold bin*, the boundaries of the bins are assigned based on some threshold values. For example, with  $B = 2$  and threshold  $\frac{1}{n}$ , all the boxes with probability greater than or equal to  $\frac{1}{n}$  are placed into  $b_1$  and all the boxes with probability less than  $\frac{1}{n}$  are placed into  $b_2$ . Note that some bins can be empty. In *solution for  $D$  bin*, the  $B = D$  bins are based on the optimal partition of  $n$  boxes into  $D$  rounds.

In this section we consider only sequential search using various bin assignments. Given the set of *prediction bins* there is essentially one way to search, since the boxes within a bin are indistinguishable from each other; the only rule that any smart solution should follow is to search the bins according to their order. In the analysis of the various types of bin strategies that follow, we call the procedure for opening boxes using *prediction bins*, *ALG*, and the cost obtained by it  $Cost_{ALG}$ .

From the application point of view, consider the wireless cellular network that provides some privacy to its users. The bins reveal much less information about a mobile's location than a full relative order among the cells does. With bins, the system knows the relative order only among the bins, but within each bin, it has no way of deciding what the order of cells is.

#### 3.1 Size Bins

The idea of size bins is that the algorithm is given  $B$  bins with specific sizes. When searching within a single bin, the worst case is when the searcher inadvertently opens the boxes within that bin in reverse order starting with the one with the smallest probability, going to the largest. Performance in the worst case

depends on how the sizes of particular bins are chosen. If this happens in the first bin (the one that contains the boxes with highest probabilities) and that bin has large size, then the boxes with high values contribute to the cost a lot more than they do if the opening process is done in the correct order. We provide matching bounds for the performance of a searcher in the worst case.

*Linear size bins.* In this strategy, the sizes of all bins differ by at most one. The first  $n - (n \bmod B)$  bins have size  $\lfloor n/B \rfloor$  and the remaining bins have size  $\lceil n/B \rceil$ . For simplicity of computation assume that  $n$  is a multiple of  $B$ ; then  $|b_1| = |b_2| = \dots = |b_B| = \frac{n}{B}$ . The bins with lower index numbers contain the boxes with higher probability, i.e.,  $C_i \in b_k$  and  $C_j \in b_{k+1}$  implies that  $p_i \geq p_j$  for any  $1 \leq i, j \leq n$  and  $1 \leq k < B$ . For an arbitrary bin  $b_i$ , the boxes in it are opened in  $n/B$  rounds one after another without knowing which are opened first.

**Theorem 3.** *Any algorithm that opens boxes in  $n$  rounds given linear size prediction bins achieves, in the worst case, a cost of  $\frac{n}{B} \text{Opt}(n)$ .*

*Proof Sketch.* The worst case occurs when *ALG* opens the boxes placed in a single bin from the one with smallest probability to largest. Then  $\text{Cost}_{\text{ALG}} \leq \frac{n}{B}\pi_1 + \frac{2n}{B}\pi_2 + \dots + \frac{Bn}{B}\pi_B$ , where  $\pi_i$  is the sum of all the probabilities in a bin  $b_i$ . The optimal algorithm that has complete knowledge of the prediction vector opens the boxes from each bin in decreasing order, so its cost is  $\text{Cost}_{\text{Opt}} \geq 1\pi_1 + \frac{n}{B}\pi_2 + \dots + \frac{(B-1)n}{B}\pi_B$ . This guarantees a ratio  $\rho \leq n/B$ .

The worst case is met when the prediction vector is  $\langle 1, 0, \dots, 0 \rangle$ . Optimally the box with probability 1 should be open first, but a searcher may open it as the last box in the first bin causing the worst case ratio  $\rho = n/B$ .

*Exponential size bins.* In the exponential bin strategy, the size of a bin  $b_i$  is  $n^{i/B} - n^{(i-1)/B}$ , where  $1 \leq i \leq B$ . For simplicity assume that  $n = k^B$  for some positive integer  $k$ . This type of bin assignment puts the boxes with higher probability into smaller bins, thereby reducing the potential cost of the worst case in which the boxes in the first few bins are opened in reverse order.

The proof of the next theorem is in the same style as the one for Theorem 3.

**Theorem 4.** *Any algorithm that opens the boxes in  $n$  rounds given exponential size prediction bins achieves, in the worst case, a cost of  $\Theta(n^{1/B}) \text{Opt}(n)$ .*

### 3.2 Threshold Bins

In the threshold bin strategy, bins are assigned probability ranges, and boxes with probabilities within a bin's range are assigned to that bin. Consequently, it is possible to have empty bins. Like the size bin strategy, within any bin one does not know the relative values of the probabilities. Ideally one would assign bin boundaries in such a way that the boxes are nearly evenly distributed among the bins, but there is no way to do this. A bad choice might place all the boxes into a single bin. In practice, this method leaves a lot of bins empty, which leads to bad performance. As for size bins, we provide matching bounds for the performance of a searcher in the worst case.

*Linear threshold bins.* The simplest threshold bin strategy is the one in which the width of the range of every bin is  $\frac{1}{B}$ , i.e., the bins would be:  $(1 - \frac{1}{B}, 1]$ ,  $(1 - \frac{2}{B}, 1 - \frac{1}{B}]$ ,  $\dots$ ,  $(\frac{1}{B}, \frac{2}{B}]$ ,  $[0, \frac{1}{B}]$ . Thus, bin  $b_i$  has boxes with probabilities  $p_i$  such that  $\frac{B-i}{B} < p_i \leq \frac{B-i+1}{B}$ . The worst case occurs when all of the boxes are placed into the last bin (the one with range  $[0, \frac{1}{B}]$ ), because no information about relative values of probabilities is revealed.

**Theorem 5.** *Any algorithm that opens the boxes in  $n$  rounds given linear threshold prediction bins achieves, in the worst case, a cost of  $\Theta(\frac{n}{B}) \text{Opt}(n)$ .*

*Proof Sketch.* Denote by  $z_i$  the number of boxes in the bins with indices smaller than  $i$  and by  $x_i$  the number of boxes in the bin  $b_i$ . No matter what the values of  $z_i$  and  $x_i$  are, the costs for bin  $i$  are as follows:

$$\text{Cost}_{ALG} \leq \frac{B-i+1}{B} \sum_{j=z_i+1}^{z_i+x_i} j \quad \text{and} \quad \text{Cost}_{Opt} \geq \frac{B-i}{B} \sum_{j=z_i+1}^{z_i+x_i} j.$$

The worst case for the game with only linear threshold prediction bins is  $\rho < \frac{B-i+1}{B-i} < 2$  for  $1 \leq i \leq B-1$ . The last bin requires special consideration. Let  $x_{B+}$  be the number of boxes in the last bin with nonzero probability and  $x_{B0}$  be the number of boxes in  $b_B$  that have a probability of 0. The above costs become:

$$\text{Cost}_{ALG} \leq \frac{1}{B} \sum_{j=z_B+x_{B+}+x_{B0}+1}^{z_B+x_{B+}+x_{B0}} j \quad \text{and} \quad \text{Cost}_{Opt} \geq p_{min} \sum_{j=z_B+1}^{z_B+x_{B+}} j,$$

in which  $p_{min}$  is the smallest nonzero probability associated with a box in  $b_B$ . Using the fact that  $z_B + x_{B+} + x_{B0} = n$  the ratio can be simplified to  $\rho \leq (2n - x_{B+} + 1) / (Bp_{min}(2z_B + x_{B+} + 1))$ . This ratio has maximum value when  $z_B = 0$ , for which  $\rho = O(n/B)$ . The ratio of the costs of two algorithms is less then or equal to the maximum ratio of the costs of the two algorithms for any given bin, so it is less than or equal to the worst ratio for the last bin,  $\rho = O(n/B)$ .

This worst case is met by the prediction vector  $\langle \frac{1}{B} - \epsilon', \dots, \frac{1}{B} - \epsilon', \epsilon, \dots, \epsilon \rangle$ . Without affecting much the value of  $\rho$ , both  $\epsilon'$  and  $\epsilon$  may be replaced by zero. If a searcher opens boxes in order of nondecreasing probabilities, then the ratio of the cost of this searcher to that of an optimal algorithm is  $\rho \geq \frac{2n-B+1}{B+1} = \Omega(\frac{n}{B})$ .

*Exponential threshold bins.* The exponential threshold bin strategy takes advantage of the fact that there can be only a few boxes with large probabilities but possibly many with very small probabilities. The exponential threshold bin strategy provides finer granularity of bins for the boxes with smaller probabilities. This makes it less likely that all the boxes are placed into one or just a few bins. Bin  $b_i$  contains all the boxes with probabilities in the semi-closed range from  $n^{-(i-1)/B}$  to  $n^{-i/B}$ , to be precise, the bins are:  $[n^{-1/B}, 1]$ ,  $[n^{-2/B}, n^{-1/B})$ ,  $\dots$ ,  $[0, n^{-(B-1)/B})$ .

The proof of the next theorem is in the same style as the one for Theorem 5.

**Theorem 6.** *Any algorithm that opens the boxes in  $n$  rounds given exponential threshold prediction bins achieves, in the worst case, a cost of  $\Theta(n^{1/B}) \text{Opt}(n)$ .*

### 3.3 Optimal Solution Bins

This bin assignment strategy assumes that somehow the searcher knows the values of the optimal partitions for  $D$  rounds and needs to find the token in  $D' \neq D$  rounds. Specifically we analyze the case of going from a solution for 2 rounds to a sequential search, i.e.,  $n$  rounds, and provide an upper bound on the worst case performance of a searcher. Other cases remain open problems and we present only simulation results for them in Section 4.

*Solution for 2 paging rounds.* Given the solution for 2 rounds, the only information present is about which boxes are opened in which round. It is known that the threshold between two partitions is between  $1/2n$  and  $2/n$ , [7]; hence, the searcher can approximate the location of the threshold that separates two bins.

**Theorem 7.** *Any algorithm that opens the boxes in  $n$  rounds given prediction bins that follow the optimal partition for two rounds achieves, in the worst case, a cost of  $O(\sqrt{n}) \text{Opt}(n)$ .*

We use the above-mentioned characteristics of the optimal partitioning into two paging rounds. This allows us to analyze two cases depending on the value of the optimal partition. In the first, the worst case for an arbitrary algorithm is caused by opening boxes in the second bin in the wrong order, in the second case an algorithm has to pay more for opening in the wrong order the boxes from the first bin. In both cases we show that the ratio of that algorithm to the optimal cannot be more than  $O(\sqrt{n})$ . The detailed proof is beyond the space limitations of this publication.

## 4 Performance Analysis

We implemented all of the proposed algorithms and strategies and compared all results. We ran the algorithms on different combinations of the input parameters  $n$ ,  $D$  and  $B$  for various distributions. We present here just a selection of the results. We mainly used the *Zipf* distribution, [16], which is observed in many commonly occurring phenomena. The values of the probabilities associated with the boxes according to this distribution are proportional to  $(1/i)^\alpha$  for  $1 \leq i \leq n$  and a parameter  $\alpha \geq 0$  and normalized by  $\sum (1/j)^\alpha$  so that  $\sum_{j=1}^n p_j = 1$ . When the parameter  $\alpha$  is equal to zero the distribution is uniform (all the values are the same), and as  $\alpha$  increases the values become skewed towards the  $p_i$ 's with lower indices. When  $\alpha = 3$ , the value of  $p_1$  is already very close to 1 and the remaining  $p_i$ 's are negligible.

The first set of tests examined the performance of algorithm *DRoot* that knows only the relative order of probabilities of the token being located in each box. The numerical results show that the performance is very close to the optimal algorithm that knows the entire probability vector. The chart in Figure 1 shows that the cost computed by *DRoot* algorithm remains very close to the cost of the optimal algorithm as the number of boxes increases. We ran the algorithm

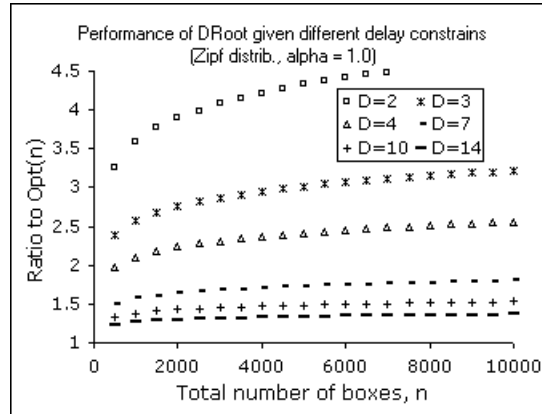


Fig. 1. The performance of *DRoot* algorithm improves as  $D$  increases.

on input sizes of up to  $n = 10,000$  to see if the performance is affected by the large number of boxes. We found out that *DRoot* performs well independent of the input size. Table 1 shows more specific results for  $n = 10,000$ . Independent of the parameter  $\alpha$  and the number of rounds allowed,  $D$ , algorithm *DRoot* performs much better than suggested by the theoretical analysis. Similar results were observed with other distributions (e.g. Gaussian).

The second set of tests compared the results of different bin strategies. We compared five different types of inaccurate prediction vectors. The tests showed that even with a very small number of bins the cost obtained by the searcher that is given only knowledge about the bins is very close to the optimal cost. The chart in Figure 2(a) shows these results for the Zipf distribution with parameter  $\alpha = 1.0$  and  $n = 10,000$ . The results for smaller values of  $n$  were even closer to optimal. The tests on other distributions showed also that the searcher only needs a small numbers of bins to obtain good results. The chart in Figure 2(b) demonstrates the results obtained by the searcher with only 5 bins for different types of bins as the number of boxes increases. For all the Zipf distributions the *optimal solution for  $D$  bin* gave the cost closest to optimal.

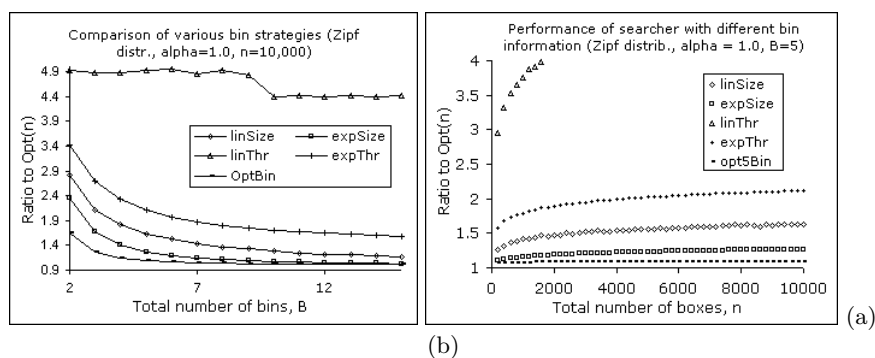
## 5 Open Problems

We have presented several ways of finding a token in one out of  $n$  boxes, when the knowledge about the prediction vector associated with these boxes is limited. The limited information increases the cost of searching for the token, but the increase is not significant even in the worst case. The performance analysis shows that on average the increase in cost is much smaller than the one suggested by the worst case analysis. There are several questions that still remain open (some of which are work in progress).

In section 2, we showed a lower bound of the worst case behavior of any algorithm that knows only the relative order of probabilities. Narrowing the gap

**Table 1.** The ratio of cost of the *DRoot* algorithm to the cost of an optimal solution for different values of the delay ( $D$ ) and different values of parameter  $\alpha$  to Zipf distribution in comparison to the theoretical worst case of that ratio.

n = 10,000	Parameter to Zipf distribution, $\alpha$											Theor. worst case
	0.4	0.6	0.8	1	1.2	1.4	1.6	1.8	2	2.2	2.4	
D=2	1.438	1.500	1.541	1.531	1.449	1.298	1.123	1.010	1.033	1.221	1.569	100.000
D=4	1.524	1.543	1.537	1.496	1.411	1.282	1.133	1.026	1.037	1.219	1.577	10.000
D=6	1.435	1.434	1.417	1.378	1.312	1.218	1.109	1.023	1.019	1.144	1.405	4.642
D=8	1.357	1.350	1.331	1.298	1.247	1.174	1.090	1.025	1.033	1.171	1.409	3.162
D=10	1.299	1.290	1.273	1.245	1.203	1.144	1.075	1.020	1.023	1.111	1.255	2.512
D=12	1.256	1.247	1.232	1.208	1.172	1.123	1.067	1.025	1.043	1.134	1.277	2.154
D=14	1.223	1.215	1.201	1.180	1.149	1.107	1.057	1.013	1.001	1.029	1.071	1.931



**Fig. 2.** The performance of the 5 bin strategies: linear (linSize) and exponential (expSize) size bins, linear (linThr) and exponential (expThr) threshold bins, and optimal solution bin (OptBin) for (a) increasing number of bins, (b) increasing number of boxes.

between the upper bound of algorithm *DRoot* and the general lower bound for  $D \geq 3$  as well as matching the two bounds for  $D = 2$  remain open problems.

In Section 3, we discussed the performance of sequential search given various types of bins. An interesting research topic is to provide similar analysis of cases in which the delay constraint is  $D < n$ . In Subsection 3.3, we analyzed the sequential search performance of a searcher who knows the optimal solution for  $D = 2$  rounds. We encountered similar problems as the authors of [7] with the analysis for the case of  $D \geq 3$  rounds. A matching lower bound for the case of  $D = 2$  is also work in progress.

Other bin assignments can be introduced and analyzed, for example a combination of size type bins and threshold type bins, strategies with a variable number of bins, or bins whose content adds up to some predetermined constant.

In practice, our results may suggest a method of measuring the privacy level offered by the camouflaging strategies used by cellular networks. The question is whether the optimality of a strategy could be an indication of privacy that

it provides to the user. That is, if the system can find the user with cost close to optimal, does it follow that the strategy does not guarantee much privacy? This type of analysis requires a specific definition of privacy that includes some metrics by which it can be measured

**Acknowledgments:** We thank the reviewers of the previous version of the paper for the ideas and comments that were incorporated into this version.

## References

1. Madhavapeddy, S., Basu, K., Roberts, A.: Adaptive paging algorithms for cellular systems. Kluwer Academic Publishers, Norwell, MA, USA (1996)
2. Goodman, D.J., Krishnan, P., Sugla, B.: Minimizing queuing delays and number of messages in mobile phone location. *Mobile Networks and Applications* **1**(1) (1996) 39–48
3. Krishnamachari, B., Gau, R.H., Wicker, S.B., Haas, Z.J.: Optimal sequential paging in cellular wireless networks. *Wirel. Netw.* **10**(2) (2004) 121–131
4. Bar-Noy, A., Feng, Y.: Efficiently paging mobile users under delay constraints. In: Proc. of 26th IEEE Conf. on Computer Communications. (2007) 1910–1918
5. Jain, R., Lin, Y.B., Mohan, S. In: Location Strategies for Personal Communications Services. *Mobile Communications Handbook* (chapter 18), CRC Press (1996)
6. Akyildiz, I., McNair, J., Ho, J., Uzunalioglu, H., Wang, W.: Mobility management in next-generation wireless systems. In: Proceedings of the IEEE. Volume 87. (1999) 1347–1384
7. Bar-Noy, A., Mansour, Y.: Competitive on-line paging strategies for mobile users under delay constraints. In: Proceedings of the 23rd ACM Symposium on Principles of Distributed Computing. (2004) 256–265
8. Rose, C., Yates, R.: Location uncertainty in mobile networks: a theoretical framework. *Communications Magazine, IEEE* **35**(2) (1997) 94–101
9. Wang, W., Akyildiz, I., Stuber, G.: An optimal partition algorithm for minimization of paging costs. *IEEE Comm. Letters* **5**(2) (2001) 42–45
10. Bar-Noy, A., Malewicz, G.: Establishing wireless conference calls under delay constraints. *J. Algorithms* **51**(2) (2004) 145–169
11. Rose, C., Yates, R.: Ensemble polling strategies for increased paging capacity in mobile communication networks. *Wirel. Netw.* **3**(2) (1997) 159–167
12. Gau, R.H., Haas, Z.J.: Concurrent search of mobile users in cellular networks. *IEEE/ACM Trans. Netw.* **12**(1) (2004) 117–130
13. Minch, R.P.: Privacy issues in location-aware mobile devices. In: Proceedings of the 37th Hawaii International Conference on System Sciences. (2004)
14. Gruteser, M., Liu, X.: Protecting privacy in continuous location-tracking applications. *IEEE Security and Privacy Magazine* **02**(2) (2004) 28–34
15. Hoh, B., Gruteser, M.: Protecting location privacy through path confusion. In: Proceedings of the 1st International Conference on Security and Privacy for Emerging Areas in Communications Networks. (2005) 194–205
16. Zipf, G.K.: *Human Behaviour and the Principle of Least Effort*. Addison-Wesley (1949)