Finding Mobile Data under Delay Constraints with Searching Costs

Amotz Bar-Noy
Computer Science
The Graduate Center and
Brooklyn College
City University of New York
365 Fifth Avenue
New York, NY 10016, U.S.A.
amotz@sci.brooklyn.
cuny.edu

Panagiotis Cheilaris
Center for Advanced Studies
in Mathematics
Ben-Gurion University
P.O.B. 653, Be'er Sheva
84105, Israel
panagiot@math.bgu.ac.il

Yi Feng
Computer Science
The Graduate Center
City University of New York
365 Fifth Avenue
New York, NY 10016, U.S.A.
yfeng@gc.cuny.edu

Asaf Levin
Industrial Engineering and
Management
Technion
Haifa 32000, Israel
levinas@ie.technion.ac.il

ABSTRACT

A token is hidden in one of several boxes and then the boxes are locked. The probability of placing the token in each of the boxes is known. A searcher is looking for the token by unlocking boxes where each box is associated with an unlocking cost. The searcher conducts its search in rounds and must find the token in a predetermined number of rounds. In each round, the searcher may unlock any set of locked boxes concurrently. The optimization goal is to minimize the expected cost of unlocking boxes until the token is found. The motivation and main application of this game is the task of paging a mobile user (token) who is roaming in a zone of cells (boxes) in a cellular network system. Here, the unlocking costs reflect cell congestions and the placing probabilities represent the likelihood of the user residing in particular cells. Another application is the task of finding some data (token) that may be known to one of the sensors (boxes) of a sensor network. Here, the unlocking costs reflect the energy consumption of querying sensors and the placing probabilities represent the likelihood of the data being found in particular sensors. In general, we call mobile data any entity that has to be searched for.

The special case, in which all the boxes have equal unlocking costs has been well studied in recent years and several optimal polynomial time solutions exist. To the best of our knowledge, this paper is the first to study the general problem in which each box may be associated with a different unlocking cost. We first present three special interesting and important cases for which optimal polynomial time algorithms exist: (i) There is no a priori knowledge about the location of the token and therefore all the placing probabilities are the same. (ii) There are no delay constraints so in each round only one box is unlocked. (iii) The token is atypical in the sense that it is more likely to be placed in boxes whose unlocking cost is low. Next, we consider the case of a typical token for which the unlocking cost of any box is proportional to the probability of placing the token in this box. We show that computing the optimal strategy is strongly NP-Hard for any number of unlocking rounds, we provide a PTAS algorithm, and analyze a greedy solution. We propose a natural dynamic programming heuristic that unlocks the boxes in a non-increasing order of the ratio probability over cost. For two rounds, we prove that this strategy is a 1.143-approximation solution for an arbitrary token and a 1.108-approximation for a typical token and that both bounds are tight. For an arbitrary token, we provide a more complicated PTAS.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity

General Terms

Algorithms

Keywords

Partitioning and scheduling, design and analysis of algorithms, approximation algorithms

1. INTRODUCTION

Consider the following combinatorial game. A token is hidden in one out of N boxes following some probability distribution. The boxes are then locked and the only known information about the location of the token is the probability distribution. Each box is associated with an unlocking cost. A searcher needs to find the token as fast as possible by unlocking boxes while minimizing the expected unlocking cost. The searcher is given D rounds, where $1 \leq D \leq N$, to find the token where in each round the searcher may unlock any set of locked boxes.

Let the boxes be $C_1, \ldots, C_N$, let $w_1, \ldots, w_N$ be the unlocking costs, and let $p_1, \ldots, p_N$ be the placing probabilities: with probability p_i the token is placed in box C_i. The fastest but the most expensive search strategy would unlock all the N boxes in one round, (*a blanket search*). The other extreme is to unlock one box per round terminating once the token is found (*a sequential search*). In general, a search strategy for D rounds is an ordered D-partition $\mathcal{A} = \langle A_1, \ldots, A_D \rangle$ of the boxes, such that in the ith round, all the boxes in the set A_i are unlocked if the token was not found during the previous $(i-1)$ rounds. The search process terminates in round d if the token is found in one of the boxes of the set A_d. Then the cost for the searcher is the total cost of unlocking all the boxes from the sets $A_1, \ldots, A_d$.

The ultimate goal is to minimize both the number of rounds and the expected unlocking cost until the token is found. These are the two main criteria in evaluating the efficiency of a specific search strategy. The problem studied in this paper is a common way to attack bi-criteria optimization problems by constraining one criterion and optimizing the other: *Given the delay constraint of finding the token in at most D search rounds, design a search strategy with minimum expected unlocking cost.*

Example: Let $N = 3$, the placing probabilities be 0.5, 0.2, 0.3, and the unlocking costs be 0.1, 0.2, 0.7 for boxes C_1, C_2, C_3, respectively. The cost of unlocking all the boxes in one round is 1. Suppose now that the token must be found in $D = 2$ rounds. One possible search strategy is $\langle\{C_1\}, \{C_2, C_3\}\rangle$. For this strategy, with probability 0.5 the token is found in C_1 for a cost of 0.1. Otherwise, with probability $(0.2 + 0.3)$ all the boxes are unlocked for a cost of 1. Thus, the total expected cost is $0.5 \cdot 0.1 + 0.5 \cdot 1 = 0.55$. Another possible search strategy is $\langle\{C_1, C_2\}, \{C_3\}\rangle$. For this strategy, with probability $(0.5 + 0.2)$ the token is found in the first round for a cost of $(0.1 + 0.2)$. Otherwise, with probability 0.3, all the boxes are unlocked for a cost of 1. Thus, the total expected cost is $0.7 \cdot 0.3 + 0.3 \cdot 1 = 0.51$. The above two search strategies follow the non-increasing order p_i/w_i. The expected cost of the search strategy $\langle\{C_2\}, \{C_1, C_3\}\rangle$ that "violates" this order is $0.2 \cdot 0.2 + (0.5 + 0.3) \cdot 1 = 0.84$. Finally, it is not hard to see that with three rounds the best strategy is to unlock the boxes following the order C_1, C_2, C_3. With probability 0.5, only C_1 is unlocked for a cost of 0.1, with probability 0.2, both C_1 and C_2 are unlocked for the cost of $(0.1 + 0.2)$, and with probability 0.3, all boxes are unlocked for the cost of 1. The total expected cost is therefore $0.5 \cdot 0.1 + 0.2 \cdot (0.1 + 0.2) + 0.3 \cdot 1 = 0.41$ which is the best possible for this example.

Motivation: The main application to the above game is the task of paging a mobile user (token) that is roaming among the cells (boxes) of a cellular network (e.g., [14]). When a call to a user arrives, the system must locate the exact cell in which the user resides to establish a connection. If the user reports its new location whenever it crosses boundaries of cells, then the system "knows" its exact location at any time and the task of finding this user becomes trivial. Since cellular networks are expected to have many cells (mini-cells or micro-cells) and mobile users are expected to move very fast, the user might cross boundaries of cells very frequently. This would make it infeasible for the user to report its new location each time it enters a different cell because the reporting process consumes the "expensive" resources: time, energy, and uplink bandwidth. Indeed, many existing location management schemes instruct mobile users to report less often. A common location management framework partitions the cells into location areas (zones), each with possibly many cells. A user reports its new location to the system only when it crosses zone boundaries (e.g., [23]). When a call to a user arrives, the system may page some or all the N cells (boxes) in some zone to find the user. Although the choice of a location management scheme to minimize the overall use of system resources depends on many parameters, such a paging step is common to most of the schemes. Frequently, the system is looking for a mobile user without knowing the exact location of this user. However, in many cases, some a priori knowledge about the whereabouts of the user is known. This knowledge can be modeled with N probability values, one value associated with each of the N cells: with probability p_i the mobile user resides in cell C_i. This a priori knowledge could either be supplied by the user itself, be extracted from history logs maintained by the system, be based on recent reports and calls involving this user, or be based on some mobility patterns. A paramount task in any location management scheme is to design, analyze, implement, and evaluate efficient paging (search) strategies for a mobile user while taking advantage of any partial knowledge of its whereabouts. The optimization goals are to find the user as fast as possible while "paying" as little as possible for paging the cells.

A special case of this problem in which $w_i = 1$ for all $i \in \{1, \ldots, N\}$ has been studied thoroughly. This special case corresponds to searching for a mobile user in D rounds while minimizing the expected number of cells paged. Efficient polynomial time dynamic programming solutions are known for this case. The scope of this paper is the general case for which the paging costs are not the same for all cells. This mainly reflects the fact that cellular networks are highly correlated with user density and cell congestion. As a result, paging a cell with more users could be more expensive than paging a cell with less users. In addition, there are a lot of other factors affecting the cost of paging a cell, which include the maintenance cost of the base stations, the different regulations on radiation emission, etc.

Mobile Data: The scope of this paper is very general. Let *mobile data* be an abstraction of any entity in a network whose exact location is not known to the system at the time when a specific query is looking for this data. Instead, the system knows that the mobile data may be found in one out of N locations. The system has a *profile* for the data which is represented as a vector of probabilities: with probability p_i the data is in location C_i. Whenever the system queries location C_i to see if it has the data it pays a cost of w_i. Paging mobile users in cellular networks is one application to this general setting but there are more applications. Consider a wireless sensor network that accumulates some information (e.g., weather or traffic). Mobile data may be any information that can be found in this sensor network. In order to save battery energy, the sensors do not push the information but only reply to queries. As a result, the system needs to pull the data by probing the sensors. The above framework models the pull task where the objectives are to minimize the time it takes to get the data and to minimize the expected cost incurred by the sensors that are probed. The above two applications are for wireless networks, but one could think of similar applications in any kind of network, for example, the task of looking for some data in the Internet or in a peer-to-peer network.

Prior art and related work: The general framework of mobile user location management has been studied a lot in the past fifteen years; see the survey [1]. Modeling uncertainty of locations of mobiles as a probability distribution vector was first studied in [19, 21]. The paper [20] introduced the user profile based paging scheme, under which the problem solved in this paper is discussed. The papers [14, 17, 20, 15, 3] described optimal solutions based on dynamic programming when all cells are of equal cost. The papers [15, 20] studied how to minimize the expected number of paged cells given an *average* (as opposed to worst-case) delay constraint using relaxation to a continuous model [20] or with a weakly polynomial dynamic programming solution [15]. The problem of paging more than one user for a conference call was studied in [5, 7, 13, 11, 12]. The problem of online paging a mobile user (in contrast to predetermined offline paging) was studied in [6]. The paper [9] explored a similar problem in which the order of cells is dictated in the context of TTL flood searching in sensor networks. The problem of paging mobile users with inaccurate information of the user location probabilities was studied in [4].

Contributions: To the best of our knowledge, this generalized version of the problem, i.e., w_i being an arbitrary number for each C_i, was not studied prior to our work. Indeed, the algorithms that generate optimal search strategies when $w_i = 1$ may have very bad performance because they ignore the different costs. Thus, our goal is to explore different solutions and approaches for the general case of arbitrary cost values.

We start with three interesting and important special cases for which polynomial time optimal strategy algorithms exist. In the first, the searcher has no a priori knowledge about the location of the token, so it assumes that all the probabilities are the same. We show that this case is "dual" to the traditional case in which all the costs are the same. Therefore, the known optimal dynamic programming solutions can be applied to this case as well. In the second, there are no delay constraints and the searcher may search the token in N rounds. We show that opening the boxes in a non-increasing order of the ratio p_i/w_i is an optimal strategy. In the third, the token is *atypical* in the sense that it is more likely to be placed in boxes with low unlocking costs. We show that applying known dynamic programming solutions on the boxes ordered by the non-increasing order of the ratio p_i/w_i yields an optimal search strategy. We refer to this algorithm as FRO (Follow Ratio Order).

Next, we consider the case of a token that has a higher probability of being placed in "expensive" boxes (in cellular networks, this corresponds to a mobile user who follows the massive behavior of other users). Let a *typical* token be a token for which p_i is proportional to w_i. We show that for a typical token the problem has a relatively simple Polynomial Time Approximation Scheme (PTAS). This is best possible because we also show that the problem is strongly NP-hard already for $D = 2$ and therefore a Fully PTAS (FPTAS) is impossible. We also show that the problem for a typical token is similar to a known version of the load balancing problem. As a result, we analyze the natural greedy solution borrowed from the load balancing problem and show similar results to those that were known for the load balancing problem.

Next we address the case of $D = 2$ rounds. For an arbitrary token that may be placed in any box with any probability, we show that the FRO strategy has a tight guaranteed $8/7 \approx 1.143$ approximation ratio. For a typical token, we design another solution that implies a slightly better tight guaranteed $\frac{7-2\sqrt{7}}{28-10\sqrt{7}} \approx 1.108$ approximation ratio. We also provide a PTAS for the general case, which is however more complicated than the PTAS given for typical tokens. The PTAS can be modified to work for an arbitrary constant number of rounds.

Paper organization: In section 2 we provide formal definitions and some preliminaries. In section 3 we present three cases for which optimal polynomial time algorithms exist. In section 4 we study the case of typical tokens. In Section 5 we analyze the performance of FRO for two rounds and describe the PTAS for the general case. In section 6 we conclude with some open problems.

2. PRELIMINARIES

Denote the N boxes by $C_1, C_2, \ldots, C_N$. Let $\mathbf{p} = \langle p_1, p_2, \ldots, p_N \rangle$ be the vector of probabilities of the token being placed in these boxes respectively. Let $\mathbf{w} = \langle w_1, w_2, \ldots, w_N \rangle$ be the vector of costs of unlocking these boxes respectively. Denote the delay constraint for finding the token by D, $1 \le D \le N$. An *instance* to the problem is the quadruple $I = (N, D, \mathbf{w}, \mathbf{p})$. An instance I is a *uniform cost instance* if $w_1 = \cdots = w_N = 1/N$ and a *uniform probability instance* if $p_1 = \cdots = p_N = 1/N$.

A *search strategy* $\mathcal{A} = \langle A_1, \ldots, A_D \rangle$ is an ordered D-partition of the boxes, such that in the ith round, all the boxes in the set A_i are unlocked. The search process terminates in round d if the token is found in one of the boxes of the set A_d. For a given search strategy $\mathcal{A}$ and a round $1 \le d \le D$, the *round probability* is $P_d = \sum_{C_i \in A_d} p_i$ and the *round cost* is $W_d = \sum_{C_i \in A_d} w_i$. The cost of search strategy $\mathcal{A}$ on an instance $I = (N, D, \mathbf{w}, \mathbf{p})$ is denoted by cost$(\mathcal{A}, \mathcal{I})$ and when the definition of I is clear it is denoted by cost$(\mathcal{A})$.

Proposition 1. *The following are two different but equivalent ways to compute the* search cost *of strategy* $\mathcal{A} = \langle A_1, \ldots, A_D \rangle$ *on instance* $I = (N, D, \mathbf{w}, \mathbf{p})$*:*

$$\begin{aligned} cost(\mathcal{A}, \mathcal{I}) &= \textstyle\sum_{d=1}^{D} \left(P_d \sum_{i=1}^{d} W_i \right) \\ cost(\mathcal{A}, \mathcal{I}) &= \textstyle\sum_{d=1}^{D} \left(W_d \sum_{i=d}^{D} P_i \right) \end{aligned} \tag{1}$$

Proof. The first equation follows since with probability P_d the token is found during the dth round and the strategy pays the cost of the first d sets $A_1, \ldots, A_d$ of the partition. The second equation follows since the strategy pays the cost of the dth round only if the token is in a box belonging to the last $(D - d + 1)$ sets $A_d, \ldots, A_D$ of the partition. □

An *optimal search strategy* is a search strategy whose cost is the minimum among all possible search strategies. An *optimal algorithm* is an algorithm that generates optimal search strategies for all possible instances. Let OPT be an optimal algorithm and ALG be any search algorithm. ALG is a $(1 + \varepsilon)$-approximation, if cost(ALG(x))/cost(OPT(x)) $\le (1 + \varepsilon)$ for any instance x.

Normalizing the cost and the probability vectors: We observe the following basic fact that allows us to assume without loss of generality that $\sum_{i=1}^{N} w_i = 1$ and that $\sum_{i=1}^{N} p_i = 1$.

Proposition 2. *Let O be an optimal search strategy on boxes with probabilities p_i and costs w_i, $1 \le i \le N$. O is also an optimal search strategy on boxes with probabilities p'_i and costs w'_i, if $w'_i = c_w \cdot w_i$ and $p'_i = c_p \cdot p_i$, where $1 \le i \le N$ and c_w and c_p are positive constants. The approximation ratio of any non-optimal solution is retained, too.*

Types of tokens: We classify tokens by their *typicality*. In one extreme, the probability vector of a *typical token* is proportional to its cost vector. By Proposition 2, without loss of generality, for a typical token, $p_i = w_i$ for any box C_i. In the other extreme, an *atypical token* is more likely to be located in lower cost boxes. Formally, the costs and the probabilities are in opposite order. A token with no typicality association is called an *arbitrary* token. Such a token may be located in any box C_i with arbitrary cost w_i and arbitrary probability p_i.

Optimal polynomial time algorithms: We say that an ordered partition $\mathcal{A} = \langle A_1, \ldots, A_D \rangle$ *respects the order of boxes* $C_1, C_2, \ldots, C_N$ if there are no A_i, A_j with $i < j$, such that $C_{i'} \in A_i$ and $C_{j'} \in A_j$ with $i' > j'$. Given an order of the boxes, one can find a minimum cost partition that respects that order in polynomial time by slightly modifying the dynamic programming methods described in [14, 20, 17, 15, 3] to include costs of boxes. A naive implementation implies an $O(N^2 D)$ algorithm. We mention below a more efficient implementation. The proof is omitted for space considerations.

Theorem 1. *The dynamic programming scheme from [3] can be implemented in $\Theta(ND)$ time to find a minimum cost partition that respects a given order on the boxes.*

Unfortunately, for the general problem, as we will prove later, it is impossible to find in polynomial time the order of boxes in an optimal search strategy, unless P = NP.

Algorithm FRO: Consider the N boxes in a non-increasing order of the p_i/w_i ratio. That is, $p_1/w_1 \geq p_2/w_2 \geq \cdots \geq p_N/w_N$. We call the algorithm that computes the minimum cost partition that respects the above order Algorithm FRO (follow ratio order). Algorithm FRO can also be implemented in time $\Theta(ND)$ by adapting the dynamic programing scheme from [3].

3. OPTIMAL SEARCH STRATEGIES

The traditional version of the problem with instances of uniform cost can be solved by polynomial time algorithms [14, 20, 15, 3]. In this section, we first present two lemmas that reveal some properties an optimal searching strategy must retain; and then, based on the lemmas, we show three special cases in which optimal search strategies can be found with polynomial time algorithms.

Lemma 1. *If there is an order of the boxes such that $p_1 \geq \ldots \geq p_N$ while $w_1 \leq \ldots \leq w_N$, then algorithm FRO follows this order and generates an optimal search strategy.*

Proof. Assume for the sake of contradiction that there exists an optimal solution $\mathcal{A} = \langle A_1, \ldots, A_D \rangle$ that does not respect the non-decreasing order of the ratios p_i/w_i. This implies the existence of indices $j < i$ (and therefore $p_i < p_j$ and $w_i > w_j$) such that $C_i \in A_d$ and $C_j \in A_{d+1}$ for some $1 \leq d < D$. Define a new partition $\mathcal{A}'$ which is almost identical to $\mathcal{A}$ except that C_i and C_j are swapped. To get a contradiction, we show that the cost of $\mathcal{A}'$ is smaller than the cost of $\mathcal{A}$. The probability and cost for rounds d and $d+1$ are related as follows: $P_d = P'_d + p_i$, $W_d = W'_d + w_i$, $P_{d+1} = P'_{d+1} + p_j$, and $W_{d+1} = W'_{d+1} + w_j$. A careful examination of the terms in Eq. (1) from Proposition 1 reveals that both cost($\mathcal{A}$) and cost($\mathcal{A}'$) share identical terms except that cost($\mathcal{A}$) has unique terms $P'_d w_i + p_j w_i + p_j W'_{d+1}$ and cost($\mathcal{A}'$) has unique terms $P'_d w_j + p_i w_j + p_i W'_{d+1}$ Now, since $p_i < p_j$ and $w_i > w_j$, it follows that cost($\mathcal{A}'$) < cost($\mathcal{A}$). □

Lemma 2. *Let $\mathcal{A} = \langle A_1, \ldots, A_D \rangle$ be an optimal solution for an arbitrary instance I. Then $P_d/W_d \geq P_{d+1}/W_{d+1}$, for $1 \leq d < D$.*

Proof. Fix d, $1 \leq d < D$. Define another search strategy $\mathcal{A}' = \langle A_1, \ldots, A_{d-1}, A_{d+1}, A_d, \ldots, A_D \rangle$ that is obtained from $\mathcal{A}$ by swapping A_d and A_{d+1}. By Proposition 1, it follows that cost($\mathcal{A}'$) − cost($\mathcal{A}$) = $P_d W_{d+1} - P_{d+1} W_d$. This is because in both strategies all the costs that incurred by W_i for $i < d$ and $i > d+1$ are the same and in both strategies the terms $P_d W_d$ and $P_{d+1} W_{d+1}$ are part of the cost. Now, since $\mathcal{A}$ is optimal, it follows that $P_d W_{d+1} - P_{d+1} W_d \geq 0$, which is equivalent to $P_d/W_d \geq P_{d+1}/W_{d+1}$. □

Uniform Probabilities: The uniform probability case is when all the probabilities are $1/N$ (in contrast to uniform cost case in [14, 20, 15, 3]). This is the case when the searcher has no clue for the whereabouts of the token but still knows the unlocking costs associated with the boxes. In Section 2, we show that the algorithm FRO yields optimal solution if the order of optimal searching strategy is known for any number of rounds $2 \leq D \leq N$. In the uniform probabilities case, Lemma 1 provides this optimal order by sorting the boxes in *non-decreasing* order of unlocking cost w_i, by which FRO generates optimal polynomial time solutions.

Atypical token: Recall that an atypical token is one that appears in boxes of lower unlocking costs with higher probabilities. For an atypical token, the non-increasing order of the probability vector corresponds to the non-decreasing order of the cost vector. Lemma 1 directly implies that an optimal search strategy can be found in polynomial time for such tokens.

D = N rounds: The following corollary is a direct implication of Lemma 2, which implies that the polynomial time algorithm FRO generates an optimal search strategy for the case $D = N$.

Corollary 1. *For $D = N$, the optimal search strategy unlocks the boxes in a non-increasing order of p_i/w_i, one box per round.*

4. TYPICAL TOKENS

In this section, we discuss and analyze search strategies for the special case of typical tokens. Recall that a typical token is more likely to be found in high cost boxes and therefore, after normalizing the cost and probability values, we assume that $p_i = w_i$ for all $1 \leq i \leq N$. We prove that the problem is strongly NP-Hard even in this special case. In particular, we show that the problem is essentially a variation of a known load balancing problem. This enables us to apply a known *polynomial time approximation scheme* (PTAS) solution for the load balancing problem to our problem of searching for typical tokens. In addition, we analyze the performance of a natural load balancing greedy algorithm applied to the search problem. We first show how to compute the cost for a typical token by simplifying the equations from Proposition 1.

Proposition 3. *Let $I = (N, D, \mathbf{p}, \mathbf{p})$ be an instance of the typical token problem and $\mathcal{A} = \langle A_1, \ldots, A_D \rangle$ be a search strategy. Then, cost($\mathcal{A}, I$) = $\frac{1}{2} + \frac{1}{2}\sum_{d=1}^{D}(P_d)^2$.*

Proof. For all $1 \leq d \leq D$, denote by $P_d = W_d$ the probability as well as the cost of the set A_d. By Proposition 1,

$$\text{cost}(\mathcal{A}, I) = \sum_{d=1}^{D}\left(P_d \sum_{i=1}^{d} W_i\right) = \sum_{1 \leq i \leq j \leq D} P_i P_j = \frac{1}{2}\left(\left(\sum_{d=1}^{D} P_d\right)^2 + \sum_{d=1}^{D}(P_d)^2\right)$$

The proposition follows since $\sum_{d=1}^{D} P_d = 1$. □

The above proposition implies that for a typical token and a given search strategy $\mathcal{A}$, the cost is the same regardless of the order of the D sets in $\mathcal{A}$. That is, the task of finding an efficient D-round search strategy becomes the task of partitioning the N boxes into D sets while minimizing a particular cost function. One can view the sets as machines and the probabilities as the processing times of tasks. Then the problem is almost identical to the known *load balancing* problem from [8, 10] whose goal is to minimize the L_2 norm of the tasks' completion time on all the machines. Formally,

Definition 1. *Let $\{T_1, \ldots, T_N\}$ be N tasks and $\{M_1, \ldots, M_D\}$ be D machines. The processing time for T_i on any machine is p_i. The* load balancing problem *is to find an allocation of tasks to machines that minimizes the L_2 norm of the completion time on all machines $\sum_{d=1}^{D}(\sum_{T_i \in M_d} p_i)^2$.*

If X is the optimization goal of the load balancing problem and Y is the optimization goal of the search problem, then $X = (1 + Y)/2$ by Proposition 3 and Definition 1. As a result, we now show that any approximation algorithm to the load balancing problem is also an approximation algorithm to the searching for typical tokens problem with an even smaller approximation factor.

Lemma 3. *Let ALG be a $(1+\varepsilon)$-approximation algorithm to the load balancing problem where $(1+\varepsilon)$ is a tight bound. Then ALG is a $(1+\varepsilon')$-approximation algorithm to the searching for typical tokens problem, where $\varepsilon' < \frac{\varepsilon}{2}$ for all instances and $\varepsilon' \ge \frac{\varepsilon}{D+1}$ for some instance.*

Proof. Let OPT be an optimal solution to the load balancing problem. Let O and $\mathcal{A}$ be the L_2 norm of solutions OPT and ALG, respectively. Let OPT′ be an optimal solution to the search typical tokens problem. Let ALG′ = ALG be a $(1+\varepsilon')$-approximation solution to the search problem. Let $O' = \text{cost}(\text{OPT}')$ and $\mathcal{A}' = \text{cost}(\text{ALG}')$. By definition, we have $\mathcal{A} \le (1+\varepsilon)O$ for all instances in the load balancing problem. Therefore,

$$1+\varepsilon' = \frac{\mathcal{A}'}{O'} = \frac{0.5+0.5\mathcal{A}}{0.5+0.5O} \le \frac{1+(1+\varepsilon)O}{1+O} = 1+\frac{O}{1+O}\varepsilon$$

for all instances. Since $(1+\varepsilon)$ is a tight bound to the load balancing problem, we have $\mathcal{A} = (1+\varepsilon)O$ for some instance. Therefore,

$$1+\varepsilon' = \frac{\mathcal{A}'}{O'} = \frac{0.5+0.5\mathcal{A}}{0.5+0.5O} = \frac{1+(1+\varepsilon)O}{1+O} = 1+\frac{O}{1+O}\varepsilon$$

for some instance.

Since all $P_i \ge 0$ and $\sum_{i=1}^{D} P_i = 1$, it follows that the cost $\sum_{i=1}^{D} P_i^2$ (Definition 1) of any solution to the load balancing problem is greater than $1/D$ and smaller than 1. In particular $1/D \le O \le 1$ and therefore $O/(1+O) \le 1/2$ for all instances and $O/(1+O) \ge 1/(D+1)$ for some instance. ☐

The paper [2] pointed out that the load balancing problem is strongly NP-Hard. The next theorem follows since Lemma 3 gives a lower bound to the approximation ratio to the searching for typical tokens problem without N as a parameter.

Theorem 2. *For any $N > D \ge 2$, the search problem is* strongly NP-Hard*, even for a typical token.*

Constant approximation ratio algorithms to the load balancing problem have been introduced in [8, 10] and a PTAS to this problem has been presented in [2]. The following theorem is another corollary of Lemma 3.

Theorem 3. *For any $N > D \ge 2$, there exists a constant approximation ratio algorithm and a PTAS to the searching for typical tokens problem.*

Algorithm $\mathcal{S}$ in [8] is a natural greedy algorithm. Essentially, it allocates boxes one by one in a non-increasing order of p_i to the set A_d currently with the smallest P_d. In some cases, like when $D = 2$, one can compute exactly the approximation ratio of $\mathcal{S}$ for both the load balancing and the searching for typical tokens problems. We omit the technical details. The specific approximation ratios for different values of N and D of Algorithm $\mathcal{S}$ for the load balancing problem and for the searching for typical tokens problem (by lemma 3) are shown in Table 1. In the table, the meaning of $[x, y]$ is that there exists a y approximation factor and there cannot be an approximation factor smaller than x.

Case	Load Balancing	Typical Tokens
$N \le 4$, any D	1	1
$D = 2, N \ge 5$	tight ≈ 1.0285	tight ≈ 1.0143
$D = 3, N \ge 5$	$[83/81, 25/24]$	$[326/324, 49/48]$
even $D \ge 4, N \ge 5$	$[37/36, 25/24]$	$[1+\frac{1}{36(D+1)}, 49/48]$
odd $D \ge 5, N > 5$	$[1+\frac{(D-1)}{36D}, 25/24]$	$[1+\frac{(D-1)}{36D(D+1)}, 49/48]$

Table 1: Approximation ratio of $\mathcal{S}$-algorithm

5. ALGORITHM FRO AND A PTAS

In this section, we analyze the performance of Algorithm FRO for two rounds on arbitrary tokens whose cost vector has no correlation to the probability vector. Note that the problem for an arbitrary token is strongly NP hard when $2 \le D < N$ because it is strongly NP-hard already for a typical token. Therefore, our goal is to prove a guaranteed approximation factor. We show that the approximation ratio of FRO is $8/7 \approx 1.143$ for $D = 2$ rounds. For typical tokens, we slightly improve the ratio to $\frac{7-2\sqrt{7}}{28-10\sqrt{7}} \approx 1.108$. For both cases, we provide instances that show that these bounds are tight. We also provide a PTAS for the general case, for two rounds, that can be extended to a PTAS for any constant number of rounds D.

5.1 FRO for arbitrary tokens

Theorem 4. *Algorithm FRO has an approximation ratio $8/7$ when $D = 2$ and $N > 2$, and the ratio $8/7$ is attainable.*

Proof. First, we show that an upper bound on the approximation ratio of FRO for $N = 3, 4$ is also an upper found for all $N > 2$ (Lemma 4). Next, we prove that the upper bound on the approximation ratio of FRO for $N = 3, 4$ is $8/7$ (Lemmas 5, 6). Finally, for every N, we construct an instance with approximation ratio $8/7$ (Lemma 7). ☐

Lemma 4. *For each instance with $N \ge 4$ for which the approximation ratio of FRO is ρ, there exists an instance, with either $N = 3$ or 4, for which the approximation ratio of FRO is at least ρ.*

Proof. Let an instance I_N consist of boxes $\{C_1, \ldots, C_N\}$, where $N \ge 4$, with optimal partition $\text{OPT}_N = \langle X, Y\rangle$ and FRO partition $\text{FRO}_N = \langle X', Y'\rangle$. Define the following four subsets of the N boxes: $A = X \cap X'$, $B = X' \setminus X$, $C = Y' \setminus Y$, $D = Y \cap Y'$. Then, $\text{OPT}_N = \langle A \cup C, B \cup D\rangle$ and $\text{FRO}_N = \langle A \cup B, C \cup D\rangle$.

If all four sets A, B, C, D are not empty, we construct an instance I_4 with $N = 4$ boxes as follows. The boxes are $\{C_A, C_B, C_C, C_D\}$, such that $p_S = \sum_{c_i \in S} p_i$ and $w_S = \sum_{c_i \in S} w_i$, where $i = 1 \ldots N$ and $S \in \{A, B, C, D\}$. Let OPT_4 be the optimal solution for I_4. It follows that $\text{cost}(\text{OPT}_4) = \text{cost}(\text{OPT}_N)$, because: (i) $\text{cost}(\text{OPT}_4)$ cannot be less than $\text{cost}(\text{OPT}_N)$, otherwise OPT_N would not be optimal (taking the corresponding OPT_4 partition); (ii) $\text{cost}(\text{OPT}_4)$ can reach $\text{cost}(\text{OPT}_N)$ by taking $\text{OPT}_4 = \langle\{C_A C_C\}, \{C_B C_D\}\rangle$ (by definition of partition $\{A, B, C, D\}$). It also follows that the FRO partition on I_4 is $\text{FRO}_4 = \langle\{C_A C_B\}, \{C_C C_D\}\rangle$ because $\text{FRO}_N = \langle A \cup B, C \cup D\rangle$. Thus, $\text{cost}(\text{FRO}_4) \ge \text{cost}(\text{FRO}_N)$. Therefore,

$$\text{cost}(\text{FRO}_N)/\text{cost}(\text{OPT}_N) \le \text{cost}(\text{FRO}_4)/\text{cost}(\text{OPT}_4)\,.$$

If either of the four sets A, B, C, D is empty, we can similarly construct an instance of $N = 3$ with at least the same approximation ratio. Finally, if there are at least two empty sets among A, B, C, D, then OPT *is* FRO, and thus the approximation ratio is 1. ☐

Corollary 2. *An upper bound ρ on the approximation ratio of FRO for all instances with $N = 3$ or $N = 4$ is also an upper bound on the approximation ratio for all instances with $N > 4$.*

It remains to prove the following two lemmas to complete the proof of Theorem 4. The full proofs of Lemma 5 and Lemma 6 are technically non-trivial, involving tedious case analysis; thus, we do not include them because of space considerations.

Lemma 5. *For all instances with $N = 3$, the approximation ratio $\rho = cost(FRO)/cost(OPT) \le 8/7$.*

Lemma 6. *For all instances with $N = 4$, the approximation ratio $\rho = cost(FRO)/cost(OPT) \leq 8/7$.*

In the next lemma we show that the 8/7 upper bound on the approximation ratio is tight.

Lemma 7. *The approximation ratio* $8/7$ *is attainable for any* $N > 2$.

Proof. Consider the instance with $p_1 = \frac{1}{4}$, $p_2 = \frac{3}{4}$, $p_3 = \cdots = p_N = 0$ and $w_1 = \frac{1}{5}$, $w_2 = \frac{3}{5}$, $w_3 = \cdots = w_N = (5(N-2))^{-1}$. Algorithm FRO outputs the partition $(1|23\ldots N)$ the cost of which is $4/5$, while the optimal partition is $(2|13\ldots N)$ the cost of which is $7/10$. The approximation ratio is thus $8/7$. □

5.2 FRO for typical tokens

For a typical token, FRO does not distinguish among the boxes since all the ratios are 1. Therefore, we assume that an adversary picks the worst permutation on the boxes that is respected by FRO. Still, we can prove a smaller guaranteed approximation factor for this case.

Theorem 5. *Algorithm FRO has approximation ratio* $\frac{7-2\sqrt{7}}{28-10\sqrt{7}} \approx 1.108$ *when* $D = 2$ *and* $N > D$, *and this ratio is attainable.*

Proof. (outline) The proof is very similar to the proof of Theorem 4. First, the reduction lemma, Lemma 4, is correct for any type of token. Then the proofs of the equivalent Lemmas (but with a different ratio) to Lemma 5 and Lemma 6 are easier since there are fewer variables. Finally, the next lemma demonstrates the instance for which the ratio is attainable. □

Lemma 8. *The approximation ratio* $\frac{7-2\sqrt{7}}{28-10\sqrt{7}}$ *is attainable for any* $N > 2$.

Proof. For simplicity, assume that $0/0 = 1$ since one can replace each zero value with a small ε. Consider the instance with $p_1 = w_1 = x$, $p_2 = w_2 = 1-2x$, $p_3 = w_3 = x$, and $p_4 = w_4 = \cdots = p_N = w_N = 0$. FRO, that respects this order, outputs the partition $(1|23\ldots N)$ the cost of which is $1-x+x^2$ while the optimal partition is $(2|13\ldots N)$ the cost of which is $1-2x+4x^2$. The maximum of the ratio is achieved for $x = (3-\sqrt{7})/2$ and is $\frac{7-2\sqrt{7}}{28-10\sqrt{7}}$. □

5.3 A general PTAS for D=2

We present a PTAS for the problem of finding a strategy of optimal expected cost, when $D = 2$.

Fix an optimal solution OPT. Denote by $F_1 = 1$ and F_2 the probability that OPT will get to first and second round respectively (i.e., F_2 is the probability that the token is not found by OPT in the first round). Similarly, let W_1 and W_2 be the total weight of the cells that OPT probes in the first and second round, respectively. Therefore, the cost of OPT is $C = F_1W_1 + F_2W_2$.

Consider a positive constant $\varepsilon < 1$. The first step of the PTAS is to guess an approximation within a factor of $(1+\varepsilon)$ of each one of the values of F_2, W_1, and W_2 in the optimal solution. We denote these approximations with F_2', W_1', W_2', respectively.

For F_2', we do it as follows: We first guess the maximum probability of a cell which is probed in the second round of the optimal solution. We call this probability M. There are N possible values for M: $p_1, \ldots, p_N$. Since $M \leq F_2 \leq NM$, F_2 can be $(1+\varepsilon)$-approximated by a value in $\{M, (1+\varepsilon)M, \ldots, (1+\varepsilon)^K M\}$, where $K = \lceil \log_{1+\varepsilon} N \rceil$.

Similarly, for each one of W_1', W_2' we first guess the maximum weight of a cell probed in the first and second round, respectively, of the optimal solution. We call these weights M_1 and M_2, respectively, and each one can take any of the values $w_1, \ldots, w_N$. Then, $M_1 \leq W_1 \leq NM_1$ and $M_2 \leq W_2 \leq NM_2$ and thus W_1 can be approximated by a value in $\{M_1, (1+\varepsilon)M_1, \ldots, (1+\varepsilon)^K M_1\}$ and W_2 can be approximated by a value in $\{M_2, (1+\varepsilon)M_2, \ldots, (1+\varepsilon)^K M_2\}$, where $K = \lceil \log_{1+\varepsilon} N \rceil$.

The number of possible triples (F_2', W_1', W_2') is thus $(N(K+1))^3$, which is bounded by a polynomial in N and $1/\varepsilon$. For every triple, we consider the cost $C' = F_1W_1' + F_2'W_2'$. For one of the triples, each of F_2', W_1', W_2', is ε-close to the corresponding value in the optimal solution, and thus,

$$\begin{aligned} C' &\leq F_1(1+\varepsilon)W_1 + (1+\varepsilon)F_2(1+\varepsilon)W_2 \\ &\leq (1+\varepsilon)^2(F_1W_1 + F_2W_2) \\ &\leq (1+3\varepsilon)C. \qquad (2) \end{aligned}$$

We apply the following algorithm for each possible triple: With the help of a linear program, we compute a feasible solution (if it exists) with corresponding values close enough to the values of the triple and record the solution's cost. Then, we return as an output the feasible solution with minimum cost. In the analysis of the scheme it suffices to consider the iteration of the algorithm in which we tried the values of F_2', W_1', W_2' which are ε-close to the corresponding values in OPT. In particular, for each triple of values (F_2', W_1', W_2'), we consider the following linear program, over variables $x_{1,1}, \ldots, x_{1,N}, x_{2,1}, \ldots, x_{2,N}$ (An integral feasible solution to this linear program has the following meaning: $x_{i,n} = 1$ if cell n is probed in round i.):

minimize $f = W_1' + \sum_{n=1}^N W_2' p_n x_{2,n}$ such that:
(a) $\sum_{n=1}^N p_n x_{2,n} \leq F_2'$
(b) $\sum_{n=1}^N w_n x_{1,n} \leq W_1'$ and $\sum_{n=1}^N w_n x_{2,n} \leq W_2'$
(c) $x_{1,n} + x_{2,n} = 1$ for $n = 1, \ldots, n$
(d) $x_{d,n} \geq 0$ for $d = 1,2$ and $n = 1, \ldots, n$

We say that a cell n is a *large cell* if $p_nW_2' > \varepsilon C'$; otherwise we say it is a *small cell*. We denote by Y the set of small cells, and by Z the set of large cells. Intuitively, the large cells have a major influence on the value of the goal function f, so we will treat them separately. Assume B large cells are assigned to round 2 in the optimal solution. Then,

$$C \geq F_2W_2 \geq F_2W_2'(1+\varepsilon)^{-1} > B\varepsilon C'(1+\varepsilon)^{-1} \geq B\varepsilon(1+\varepsilon)^{-1}C.$$

(The last inequality holds, because $C' \geq C$.) Therefore, it must be the case that $B\varepsilon(1+\varepsilon)^{-1} < 1$, or $B < 1+\varepsilon^{-1}$. i.e., we have a constant upper bound on the cardinality B of the set X of large cells that are assigned to round 2 in the optimal solution. Thus, the number of such sets is $O(N^B)$, i.e., polynomial in N.

Our second guessing step would be to guess the set X. That is, for every subset $X \subseteq Z$, such that $|X| < 1+\varepsilon^{-1}$, we set the values of $x_{i,n}$ for $i = 1,2$ and $n \in Z$ as follows. For all $n \in X$, we set $x_{1,n} = 0$ and $x_{2,n} = 1$, and for all $n \in Z \setminus X$ we set $x_{1,n} = 1$ and $x_{2,n} = 0$. The value of the other decision variables are determined by the solution of the following linear program, on the set of variables $\{x_{1,n} \mid n \in Y\} \cup \{x_{2,n} \mid n \in Y\}$:

minimize $f = W_1' + \sum_{n=1}^N W_2' p_n x_{2,n}$ such that:
(a) $\sum_{n=1}^N p_n x_{2,n} \leq F_2'$
(b) $\sum_{n=1}^N w_n x_{1,n} \leq W_1'$ and $\sum_{n=1}^N w_n x_{2,n} \leq W_2'$
(c) $x_{1,n} + x_{2,n} = 1$ for $n \in Y$
(d) $x_{d,n} \geq 0$ for $d = 1,2$ and $n \in Y$

The last linear program has $2|Y|$ variables and four types of constraints:

(a) One constraint with p_n coefficients (constraint of type a),

(b) Two constraints with w_n coefficients (constraints of type b),

(c) $|Y|$ equality constraints where coefficients equal to 1 (constraints of type c),

(d) $2|Y|$ non-negativity constraints.

We compute an optimal basic solution to the linear program [18]. Such a solution exists for the correct value of F'_1, W'_1, W'_2 and X because for such values OPT corresponds to a feasible solution for the above linear program whose goal function value is

$$W'_1 + W'_2 F_2 \leq W'_1 + W'_2 F'_2 = C'.$$

Therefore, the cost of the basic optimal solution which we find, is at most C'.

Moreover, a basic solution of the above linear program has the property that $2|Y|$ linearly independent constraints of the linear program are set to equality (i.e., as many constraints as the number of variables). This means that at least $|Y| - 3$ of the (d)-constraints are set to equality, i.e., at least $|Y| - 3$ variables are set to 0. However, if $x_{d,n} = 0$ for some $d = 1, 2$ and some n, then (because of the corresponding (c)-constraint) $x_{(3-d),n} = 1$. This further implies that the variables that correspond to at most 3 cells (i.e., 6 variables) are set to fractional (non-integral) values in a basic optimal solution. A similar method of bounding the number of fractional values of a basic solution was first employed in [16], in the context of scheduling unrelated parallel machines.

Let x^* be a basic optimal solution to the linear program. We will associate with it a *cost* $C^* = W^*_1 + W^*_2 P^*_2$, where $P^*_2 = F^*_2 = \sum_{n=1}^N p_n x^*_{2,n}$, $W^*_1 = \sum_{n=1}^N w_n x^*_{1,n}$, $W^*_2 = \sum_{n=1}^N w_n x^*_{2,n}$. Since $W^*_1 \leq W'_1$, $W^*_2 \leq W'_2$, and $P^*_2 = F^*_2 = F'_2$, we have

$$C^* \leq C' \tag{3}$$

Assume without loss of generality that the non-integral values are at a subset of the cells 1, 2, and 3 (that is, we assume that cells $4, 5, \ldots, N$ have integral solution). We round the non-integral variables of the linear program so that $x^{\mathrm{r}}_{11} = x^{\mathrm{r}}_{12} = x^{\mathrm{r}}_{13} = 0$ and $x^{\mathrm{r}}_{21} = x^{\mathrm{r}}_{22} = x^{\mathrm{r}}_{23} = 1$, i.e., we assign the non-integral cells to the second round, and $x^{\mathrm{r}}_{d,n} = x^*_{d,n}$, for every other variable. Consider the cost C^{r} of the rounded solution; we intend to compare C^{r} and C^*. We define:

$$\Delta p = x^*_{11} p_1 + x^*_{12} p_2 + x^*_{13} p_3$$

and

$$\Delta w = x^*_{11} w_1 + x^*_{12} w_2 + x^*_{13} w_3.$$

Then,

$$\begin{aligned}\Delta C &= C^{\mathrm{r}} - C^* \\ &= (W^*_1 - \Delta w) + (W^*_2 + \Delta w)(P^*_2 + \Delta p) - (W^*_1 + W^*_2 P^*_2) \\ &= \Delta w (P^*_2 + \Delta p - 1) + \Delta p \cdot W^*_2. \end{aligned} \tag{4}$$

Define $P^-_2 = P^*_2 - (x^*_{21} p_1 + x^*_{22} p_2 + x^*_{23} p_3)$, i.e., P^-_2 is the sum of the probabilities of the integral cells of round 2, i.e., it does not contain any probability for cells 1, 2, and 3. But then, $P^*_2 + \Delta p = P^-_2 + p_1 + p_2 + p_3 \leq 1$, which implies

$$\Delta w (P^*_2 + \Delta p - 1) \leq 0. \tag{5}$$

On the other hand,

$$\begin{aligned}\Delta p \cdot W^*_2 &\leq \Delta p \cdot W'_2 = x^*_{11} p_1 W'_2 + x^*_{12} p_2 W'_2 + x^*_{13} p_3 W'_2 \\ &\leq x^*_{11} \varepsilon C' + x^*_{12} \varepsilon C' + x^*_{13} \varepsilon C' \leq 3 \varepsilon C'. \end{aligned} \tag{6}$$

Then, using inequalities (5) and (6), equation (4) implies

$$C^{\mathrm{r}} \leq C^* + 3\varepsilon C' \leq C' + 3\varepsilon C' = (1+3\varepsilon)C' \leq (1+3\varepsilon)^2 C \leq (1+15\varepsilon)C,$$

where the second inequality is true because of (3) and the fourth inequality because of (2). Therefore the rounded solution is an $(1 + \varepsilon')$-approximation, if we choose $\varepsilon = \varepsilon'/15$ above.

Although the PTAS we described above is of theoretical interest, it might not be very attractive for use in a real system, because it is quite complicated to implement and it relies on computational tools like linear program solvers. Compare, for example, with the simplicity of algorithm FRO. We supply theoretical bounds on the worst-case performance for both the PTAS and the FRO algorithm, so that a designer of a real system can choose what suits best the application.

Since we have to solve linear programs in our PTAS, the complexity of our PTAS as described above is dependent on the representation size of the probability and weight values. However, we can make the complexity independent of this representation size as follows: In the first linear program given above, for every constraint of type (a) or (b), which is of the form $\sum_{n=1}^N a_n x_n \leq A$, we substitute it with a constraint of the form $\sum_{n=1}^N a'_n x_n \leq A(1 + \varepsilon)$, where $a'_n = \lceil a_n \cdot N/(A\varepsilon) \rceil A\varepsilon/N$. It is not difficult to see that every feasible solution of the original linear program is also feasible for the transformed linear program and that we lose another factor of $1 + O(1)\varepsilon$ in the approximation ratio that can be absorbed in ε' above. Moreover, if we multiply both sides of the above constraint with $N/(A\varepsilon)$, we get integer coefficients in the constraint matrix whose values are bounded by a polynomial of N and $1/\varepsilon$. Then, we can use Tardos' method from [22] to solve the new linear program in strongly polynomial time and thus get a PTAS which is only polynomial in N and $1/\varepsilon$.

Finally, we can adapt the methods described in this section to give a PTAS for any constant number of rounds D. In this case, we have to guess approximations to the values of $F_1, F_2, \ldots, F_D$ and $W_1, W_2, \ldots, W_D$; then, we solve a linear program with a constraint for each one of the above values. We omit the details because of space considerations.

6. OPEN PROBLEMS

We conjecture that the 8/7 bound for the FRO algorithm, from section 5.1, holds also for $D > 2$. Similarly to the $D = 2$ case, we can reduce the problem instances with any N to instances with $D \leq N \leq D^2$. However, we do not know how to handle all these cases, even for $D = 3$. We only know how to resolve an instance with $N = 4$ and $D = 3$ showing a 8/7 lower bound for algorithm FRO.

In section 5.3, we presented a PTAS for $D = 2$, running in polynomial time with respect to the number of boxes, that can be generalized to a PTAS for a constant number of searching rounds D. It would be interesting to find a PTAS for an arbitrary number of rounds $D \leq N$ (i.e., not necessarily constant D).

The uniform cost case was investigated also in many settings for paging multiple users in cellular networks. These settings of finding more than one hidden token could be addressed in the non-uniform case as well.

Acknowledgement

The work of Amotz Bar-Noy and Yi Feng was partially sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

Yi Feng would also like to thank the Research Computing group at the Graduate Center of the City University of New York for providing computational resources.

7. REFERENCES

[1] I. F. Akyildiz, J. McNair, J. Ho, H. Uzunalioglu, and W. Wang. Mobility management in next-generation wireless systems. In *Proc. IEEE*, pages 1347–1384, 1999.

[2] N. Alon, Y. Azar, G. J. Woeginger, and T. Yadid. Approximation schemes for scheduling on parallel machines. *J. Scheduling*, 1(1):55–66, 1998.

[3] A. Bar-Noy, Y. Feng, and M. J. Golin. Paging mobile users efficiently and optimally. In *Proc. IEEE Conference on Computer Communications*, pages 1910–1918, 2007.

[4] A. Bar-Noy and J. Klukowska. Finding mobile data: efficiency vs. location inaccuracy. In *Proc. Annual European Symposium on Algorithms (ESA)*, pages 111–122, 2007.

[5] A. Bar-Noy and G. Malewicz. Establishing wireless conference calls under delay constraints. *J. Algorithms*, 51(2):145–169, 2004.

[6] A. Bar-Noy and Y. Mansour. Competitive on-line paging strategies for mobile users under delay constraints. In *Proc. ACM Symposium on Principles of Distributed Computing (PODC)*, pages 256–265, 2004.

[7] A. Bar-Noy and Z. Naor. Efficient multicast search under delay and bandwidth constraints. *Wireless Networks*, 12(6):747–757, 2006.

[8] A. K. Chandra and C. K. Wong. Worst-case analysis of a placement algorithm related to storage allocation. *SIAM J. Comput.*, 4(3):249–263, 1975.

[9] N. B. Chang and M. Liu. Revisiting the TTL-based controlled flooding search: optimality and randomization. In *Proc. 10th Annual International Conference on Mobile Computing and Networking (MOBICOM)*, pages 85–99, 2004.

[10] R. A. Cody and E. G. Coffman. Record allocation for minimizing expected retrieval costs on drum-like storage devices. *J. ACM*, 23(1):103–115, 1976.

[11] L. Epstein and A. Levin. The conference call search problem in wireless networks. *Theor. Comput. Sci.*, 359(1-3):418–429, 2006.

[12] L. Epstein and A. Levin. A PTAS for delay minimization in establishing wireless conference calls. *Discrete Optimization*, 5(1):88–96, 2008.

[13] R.-H. Gau and Z. J. Haas. Concurrent search of mobile users in cellular networks. *IEEE/ACM Trans. Netw.*, 12(1):117–130, 2004.

[14] D. J. Goodman, P. Krishnan, and B. Sugla. Minimizing queuing delays and number of messages in mobile phone location. *Mobile Netw. and Appl.*, 1(1):39–48, 1996.

[15] B. Krishnamachari, R.-H. Gau, S. B. Wicker, and Z. J. Haas. Optimal sequential paging in cellular wireless networks. *Wireless Netw.*, 10(2):121–131, 2004.

[16] J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46:259–271, 1990.

[17] S. Madhavapeddy, K. Basu, and A. Roberts. *Adaptive paging algorithms for cellular systems*, volume 1, pages 83–101. Kluwer Academic Publishers, Norwell, MA, USA, 1996.

[18] J. Matoušek and B. Gärtner. *Understanding and Using Linear Programming*. Springer, 2006.

[19] C. Rose. State-based paging/registration: a greedy technique. *IEEE Trans. Veh. Tech.*, 48(1):166–173, January 1999.

[20] C. Rose and R. D. Yates. Minimizing the average cost of paging under delay constraints. *Wireless Netw.*, 1(2):211–219, 1995.

[21] C. Rose and R. D. Yates. Ensemble polling strategies for increased paging capacity in mobile communication networks. *Wireless Netw.*, 3(2):159–167, 1997.

[22] É. Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research*, 34:250–256, 1986.

[23] TSG/WG. *Section 7.6.5.18, 3GPP TS 09.02 Mobile Application Part (MAP) Specification, ver. 8.8.1*, 2008.