The Conference Call Search Problem in Wireless Networks*

Leah Epstein[1] and Asaf Levin[2]

[1] Department of Mathematics, University of Haifa, 31905 Haifa, Israel. lea@math.haifa.ac.il
[2] Department of Statistics, The Hebrew University, Jerusalem, Israel. levinas@mscc.huji.ac.il.

Abstract. Cellular telephony systems, where locations of mobile users are unknown at some times, are becoming more common. In such systems, mobile users are roaming in a zone and a user reports its location only if it leaves the zone entirely. The Conference Call Search problem (CCS) deals with tracking a set of mobile users in order to establish a call. To find a single roaming user, the system may need to search each cell where the user may be located. The goal is to identify the location of all users, within bounded time, satisfying some additional constraints on the search scheme.

We consider cellular systems with n cells and m mobile users (cellular phones). The uncertain location of users is given by m probability distribution vectors. Whenever the system needs to find the users, it conducts a search operation lasting at most d rounds. A *request* for a single *search step* specifies a user and a cell. In this search step, the cell is asked whether the given user is located there. In each round the system may perform an arbitrary number of such requests. An integer number $B \geq 1$ bounds the number of distinct requests per cell in every round. The bound d results from quality of service considerations whereas the bound B results from the wireless bandwidth allocated for signaling being scarce.

Every search step consumes expensive wireless links, which motivates search techniques minimizing the expected number of requests thus reducing the total search costs.

We distinguish between oblivious, semi-adaptive and adaptive search protocols. An oblivious search protocol decides on all requests in advance, and stops only when all users are found. A semi-adaptive search protocol decides on all the requests in advance, but it stops searching for a user once it is found. An adaptive search protocol stops searching for a user once it has been found (and its search strategy may depend on the subsets of users that were found in each previous round). We establish the difference between those three search models. We show that for oblivious "single query per cell" systems ($B = 1$), and a tight environment ($d = m$), it is NP-hard to compute an optimal solution (the case $d = m = 2$ was proven to be NP-hard already by Bar-Noy and Naor) and we develop a PTAS for these cases (for fixed values of $d = m$). However, we show that semi-adaptive systems allow polynomial time algorithms. This last result also shows that the case $B = 1$ and $d = m = 2$ is polynomially solvable also for adaptive search systems, answering an open question of Bar-Noy and Naor.

1 Introduction

Cellular phone systems allow us to contact and talk to people that are not residing in predetermined locations. In systems where a user reports its new location each time it moves to a new cell, the task of finding the user is simple. Many existing systems allow the users

* A preliminary version of this paper appeared in the *Proceedings of the 3rd Workshop on Approximation and Online Algorithms (WAOA 2005)*, LNCS volume 3879, Springer, pages 133-146.

to report their locations more rarely. Furthermore, future systems are planned to have more and smaller cells, which makes it infeasible for a user to report each time it crosses a border between a pair of cells.

The Conference Call Search problem (CCS) deals with establishing wireless conference calls under delay and bandwidth constraints. The goal is to establish a conference call between m roaming users in a cellular network consisting of n cells. The search for the users places another step in the process of establishment of the conference call. I.e., the system needs to find out to which cell each user is connected at the moment. Using historical data the system has an a-priori assumption of the likelihood of each user to reside in each cell. This is represented by a probability vector for each user describing the probabilities for the system to find the user in each cell. We denote by $p_{i,j}$ the probability to find user i in cell j. Following previous work [2], we assume that $p_{i,j} > 0$ for all values of i, j. We assume that each user is connected to exactly one cell in the system and that the locations of the different users are independent random variables. The tool for finding the users are *search requests*. Given a request for a user i and a cell j, the system pages cell j and asks whether user i is located there. Delay constraints limit the whole search process into d synchronous search rounds (such that $1 \leq d \leq mn$). Bandwidth constraints limit the number of requests per cell in each round to at most a given integer number B such that $1 \leq B \leq m$. Both delay and bandwidth constraints are motivated by quality of service considerations. We are interested in designing search protocols which efficiently utilize the wireless links, i.e. given the constraints, minimize the expected cost of the search, where each search request incurs a uniform cost of 1.

Note that even if at some time it is already clear that a given user must be located in a specific cell, (i.e., this user was paged in all other cells and was not located there), we still need to page this user in the cell where it is located in order to be able to initiate a communication link.

We consider three types of search protocols. An *oblivious search protocol* makes a full plan of search requests for d rounds, and does not change it. It stops completely if all users are found. We can view this protocol as one where we are not notified when a single user was located, but only at the time that all of them were found. A *semi-adaptive search protocol* makes a full plan of search requests for d rounds, and does not change it, however once a user is found we stop search for it. An *adaptive search protocol* decides on the search requests per round after it is notified which users were found in the previous round. It never continues a search for a user that has already been found. As one can imagine an optimal adaptive search protocol is much more complex than the optimal oblivious search protocol or the optimal semi-adaptive search protocol, as it has to define the search strategy according to the subsets of users that were found in each of the previous rounds.

We define a *tight instance* of the conference call search problem to be an instance where $B = 1$ and $d = m$. To motivate our study of tight instances we note that the case of $B = 1$ is the elementary case where each cell can be asked about a single user in each round. Clearly this means that the process of finding all users may take up to m rounds. In order to minimize the worst case delay, we require that all users are found within exactly m rounds (i.e. $d = m$). Note that when $B = 1$ then $d = m$ is the minimum number of rounds that enables a feasible solution to the problem. So one may consider the restriction to tight instances to have a primary goal of minimizing the worst case delay and minimizing the maximum load on a cell within a particular round, and a secondary goal that is to minimize the consumption of wireless bandwidth defined as the expected number of requests.

Previous work. There has been a fair amount of work on problems related to the conference call search problem in the past, see e.g. [1, 7]. The paper [3] introduced the model where search requests for different users for the same cell are made separately (i.e., we can not ask a cell what is the subset of the users that are currently connected to it). They showed that the case $B = 1$, $d = m = 2$, is NP-hard for oblivious search protocols. It was left open to find out whether the same case is NP-hard for the adaptive search protocol as well. A similar model was introduced by Bar-Noy and Malewicz [2]. In that model once a cell is requested in some round, it does not search for a single user (or a limited number of users), but outputs a list of all users in that cell. The paper focuses on oblivious search techniques. It is shown in that paper that for any constant number of users ($m > 1$,$d > 1$), and any constant number of rounds $1 \leq d \leq n$, the problem is NP-hard. Note that the problem for a single user, which is equivalent to the problem studied in this paper in this case, is polynomially solvable using a simple dynamic programming [6, 8]. Bar-Noy and Malewicz [2] suggested a simple algorithm which combines users and reduces to the algorithm for the case $m = 1$. This is a $\frac{4}{3}$-approximation for $m = d = 2$ and $\frac{e}{e-1} \sim 1.581977$-approximation for other values of d, m. In a previous paper [4] we designed a PTAS (Polynomial Time Approximation Scheme) for that last problem. The PTAS is defined for the oblivious search model, but can be modified easily to work for the adaptive search model as well.

Paper outline. In Section 2 we prove that finding an optimal oblivious search protocol of a tight instance is NP-hard for all fixed values of $d \geq 2$. This last result extends an earlier result of Bar-Noy and Naor [3] for $d = 2$. We also show that if d is a part of the input, then finding an optimal oblivious search protocol of a tight instance becomes NP-hard in the strong sense. In Section 3 we present our PTAS for oblivious search problems that are tight. We first present a relatively simple PTAS for the case $d = m = 2$ and afterwards we present a more complicated PTAS for an arbitrary constant $d = m$. Finally, in Section 4 we show that computing an optimal semi-adaptive search protocol for tight instances where the number of users is a constant, can be done in a polynomial time. This last result shows the barrier in the tractability of the conference call search problem between the oblivious and semi-adaptive search protocols; the first is NP-hard whereas the second is polynomially solvable. The case of semi-adaptive search protocol with $d = m = 2$ also implies a polynomial time algorithm for computing an optimal adaptive search protocol.

2 NP-Hardness for the Oblivious Problem

We recall that Bar-Noy and Naor [3] proved that finding the optimal oblivious search protocol is NP-hard for $B = 1$ and $d = m = 2$. In this section we extend this result to the general tight case.

Theorem 1. *Finding an optimal oblivious search protocol is NP-hard even when restricted to tight instances with $d = m$ rounds and $B = 1$ for all fixed values of $d \geq 2$.*

Proof. The claim for $d = m = 2$ is proved in [3]. We prove the claim for $d \geq 3$ using a reduction from the PARTITION problem (see problem SP12 on page 223 in [5]). In this problem we are given N integer numbers $a(1), \ldots, a(N)$, such that $\sum_{i=1}^{N} a(i) = 2S$ for some integer $S \geq 2$, and the question is whether there exists a subset $J \in \{1, \ldots, N\}$ such that $\sum_{i \in J} a(i) = S$. We create an instance of the oblivious search problem as follows. Let $\delta > 0$ be a small positive value such that $\delta < \frac{1}{8S^2d^2}$. There are $N + m - 2$ cells, $c_1, \ldots, c_{N+m-2}$. The (identical) probabilities of the first two users are as follows. The probability for cell c_j, $j \leq N$ is $p_{1,j} = p_{2,j} = (1 - \delta)\frac{a(j)}{2S}$. The probability of every other cell $j > N$ is $p_{1,j} = p_{2,j} = \frac{\delta}{m-2}$. As for the other $m - 2$ users, user i $(3 \leq i \leq m)$ has probability of $1 - \delta$ in cell $i + N - 2$ $(p_{i,i+N-2} = 1 - \delta)$ and probability $p_{i,j} = \frac{\delta}{N+m-3}$ for all $j \neq i + N - 2$. This completes the description of the reduction.

We upper-bound the cost of an optimal oblivious search protocol in case there exists an exact partition (i.e., the PARTITION instance is a YES instance). Let J be the subset of $\{1, \ldots, N\}$ such that $\sum_{i \in J} a(i) = S$. In the first round, the requests are as follows. The cells in J are asked about the first user and the cells in $\{1, \ldots, N\} - J$ are asked about the second user. Note that $\sum_{i \notin J} a(i) = S$ as well. Each other cell $N + k$ is asked for user $k + 2$. Recall that the probability of this user and cell is $1 - \delta$. In the second round, the cells in J are asked about the second user. The cells in $\{1, \ldots, N\} - J$ are asked about the first user. All other search requests are made in some arbitrary order. The probability to find each one of the first two users in the first round is exactly $\frac{1-\delta}{2}$. The probability to find any other user in the first round is exactly $1 - \delta$. Therefore, the probability to find all the users in the first round is $\frac{(1-\delta)^m}{4}$, and so the probability to have a second round is $1 - \frac{(1-\delta)^m}{4}$. For every user, the probability to find it within the first two rounds is at least $1 - \delta$. Therefore, the probability to find all the users within the first two rounds is at least $(1 - \delta)^m$, and thus the probability that the search will last at least three rounds (and at most m rounds) is at most $1 - (1 - \delta)^m$. We conclude that the total cost is at most

$$n + n\left(1 - \frac{(1-\delta)^m}{4}\right) + n(m-2)(1 - (1-\delta)^m) \leq$$

$$n + n\left(\frac{3}{4} + \frac{m\delta}{4}\right) + nm(m-2)\delta \leq \frac{7n}{4} + nm^2\delta < \frac{7n}{4} + \frac{n}{8S^2}$$

where the first inequality holds since $(1 - \delta)^m \geq 1 - m\delta$, the second inequality follows by simple algebra and the last inequality holds as $\delta < \frac{1}{8Sd^2} = \frac{1}{8Sm^2}$.

Consider now the situation where there is no exact partition (i.e., the PARTITION instance is a NO instance). Therefore, for every subset $J' \subseteq \{1, \ldots, N\}$ either $\sum\limits_{i \in J'} a(i) \leq S - 1$ or $\sum\limits_{i \in J'} a(i) \geq S + 1$. First note that if one of the cells $N + 1, \ldots, N + m - 2$ is not paged in the first round for the user who has probability $1 - \delta$ to be in this cell, then the probability for a second round is at least $1 - \delta$, and the cost is at least $2n - n\delta$. Otherwise, consider the cells $1, \ldots, N$. A subset $A_1 \subseteq \{1, \ldots, N\}$ of these cells is paged for the first user in the first round, and a disjoint subset $A_2 \subseteq \{1, \ldots, N\}$ is paged for the second user in the first round. Let $p(1)$ ($p(2)$) be the sum of probabilities of cells paged for the first (second) user in the first round. I.e., $p(1) = \sum_{j \in A_1} p_{1,j}$ and $p(2) = \sum_{j \in A_2} p_{2,j}$. Since $A_1 \cap A_2 = \emptyset$ and $p_{1,j} = p_{2,j}$ for all j, we conclude that $p(1) + p(2) \leq 1 - \delta$. Due to the definitions of probabilities for the first two users in the first N cells, we know that $p(i) = (1-\delta)\frac{X(i)}{2S}$, where $X(i)$ for $i = 1, 2$ are integers. Since there is no exact partition, we know that $X(i) \neq S$. If $X(i) \leq S - 1$ for $i = 1, 2$, then the probability to reach the second round is at least $1 - (1-\delta)^2 \cdot (\frac{S-1}{2S})^2 > 1 - (1-\delta)^2 \frac{S^2-1}{4S^2}$ where the last inequality holds since $S \geq 1$. Otherwise, since $X(1) + X(2) \leq 2S$, and none of the values can be S, we have that if for one of the users i, $X(i) = S + u \geq S + 1$, then for the other user $3 - i$ we have $X(3-i) \leq S - u \leq S - 1$. In this case the probability for a second round is at least $(1 - (\frac{S+u}{2S})(\frac{S-u}{2S})(1-\delta)^2) \geq 1 - (1-\delta)^2 \frac{S^2-1}{4S^2}$ (since $u \geq 1$). The cost in the last two cases is therefore at least $n + n(1 - \frac{S^2-1}{4S^2}(1-\delta)^2) \geq n + n(1 - \frac{S^2-1}{4S^2}) = \frac{7n}{4} + \frac{n}{4S^2}$. Note that the cost we got in the first case ($2n - n\delta$) is not smaller since $2n - n\delta \geq \frac{7n}{4} + \frac{n}{4S^2}$ is equivalent to $\delta + \frac{1}{4S^2} \leq \frac{1}{4}$ which holds since $\delta < \frac{1}{8Sd^2}$ and $S, d \geq 2$.

We got that if there is an exact partition, then the optimal cost is at most $\frac{7n}{4} + \frac{n}{8S^2}$, whereas if there is no exact partition, the optimal cost is at least $\frac{7n}{4} + \frac{n}{4S^2}$. Therefore we got that the question, whether the cost is at most $\frac{7n}{4} + \frac{3n}{16S^2}$, is NP-hard. □

Next, we prove the following theorem. We show that if d is not fixed, but a part of the input, the problem becomes strongly NP-hard.

Theorem 2. *Finding an optimal oblivious search protocol is strongly NP-hard even when restricted to tight instances with $d = m$ rounds and $B = 1$.*

Proof. We prove the claim using a reduction from the 3-PARTITION problem (see problem SP15 on page 224 in [5]). In this problem we are given $N = 3M$ integer numbers $a(1), \ldots, a(N)$, such that $\sum\limits_{i=1}^{N} a(i) = MS$ for some integer $S \geq 3$. Each number $a(i)$ satisfies $\frac{S}{4} < a(i) < \frac{S}{2}$. The question is whether there exists a partition of the indices into M disjoint sets $A_1, \ldots, A_M$ such that $\sum\limits_{j \in A_i} a(j) = S$. If such a partition exists, clearly each set has exactly three elements.

We create an instance of the oblivious search problem as follows. We introduce M users and $N = 3M$ cells. The probability of user i in cell j $p_{i,j} = \frac{a(j)}{MS}$.

The cost of an oblivious search protocol is

$$N\sum_{r=1}^{M}\left(1-\prod_{i=1}^{M}u_{i,r}\right)=N\left(M-\sum_{r=1}^{M}\left(\prod_{i=1}^{M}u_{i,r}\right)\right) ,$$

where $u_{i,r}$ is the probability to find user i within the first r rounds. Note that since $p_{i,j}=p_{i',j}$ for all $i\neq i'$ and for all j, no matter which requests are made in some round, and the sum of probabilities of the requested cells is 1 in each round. Therefore, $\sum\limits_{i=1}^{M}u_{i,r}=r$. We claim that there exists a solution of cost at most $N\left(M-\sum\limits_{r=1}^{M}\left(\frac{r}{M}\right)^{M}\right)$ iff the 3-PARTITION instance is a YES instance.

Consider first a YES instance of the 3-PARTITION problem. Let $A_1,\ldots,A_M$ be the disjoint partition of indices such that $\sum\limits_{j\in A_i}a(j)=S$. We define the following protocol. The cells with indices in A_i are requested for user $(i+k-1) \mod M$ in round k. For this protocol, the probability to find user i within r rounds is exactly $\frac{r}{M}$ and therefore the cost is exactly $N\left(M-\sum\limits_{r=1}^{M}\left(\frac{r}{M}\right)^{M}\right)$.

Next, consider a solution which minimizes $N\left(M-\sum_{r=1}^{M}\left(\prod_{i=1}^{M}u_{i,r}\right)\right)$. This is equivalent to maximizing $\sum_{r=1}^{M}\prod_{i=1}^{M}u_{i,r}$ subject to the constraints $\sum\limits_{i=1}^{M}u_{i,r}=r$, $1\leq r\leq M$. Consider the minimization problem of $\prod_{i=1}^{M}u_{i,r}$ subject to $\sum\limits_{i=1}^{M}u_{i,r}=r$. Due to the means inequality, the unique minimum of this function is achieved at $u_{i,r}=\frac{r}{M}$ for $1\leq i\leq M$. This is also an optimal solution for the minimization problem of the sum (since it is feasible for that problem as well). Moreover, this is the unique minimum point. The existence of another minimum point would imply another minimum point for at least one of the parts of the sum, which does not exist. We get that if a solution costs at most $N\left(M-\sum\limits_{r=1}^{M}\left(\frac{r}{M}\right)^{M}\right)$, then this is its exact cost. Moreover, it means that $u_{i,1}=\frac{1}{M}$ $\forall i$. Let B_i be the set of indices whose request in the first round is for user i. We get that $\sum\limits_{j\in B_i}\frac{a(j)}{S}=1$, i.e. $\sum\limits_{j\in B_i}a(j)=S$. The sets B_i are a proper disjoint partition of the indices of $1,\ldots,N$ and we get that the 3-PARTITION instance is a YES instance. □

3 A PTAS for the Oblivious Problem

Properties. Recall that we assume non-zero probabilities for each pair of user and cell. In this case, each cell must be asked regarding exactly one user per round. Therefore, each cell needs to be assigned a permutation of the users. Recall that an oblivious search is defined in advance, and lasts as long as some user is still not located. Since we solve tight instances, already the first round costs n, and therefore $OPT\geq n$.

Let ε be a value such that $0 < \varepsilon < \frac{1}{(20m)^{m+1} \cdot m!}$. We show polynomial time approximation schemes where the running time is polynomial in n, but the values ε, and also m are seen as constants. The approximation ratios of the algorithms are $1 + \Theta(\varepsilon)$.

Our schemes are composed of several guessing steps. In these guessing steps we guess certain information about the structure of OPT. Each guessing step can be emulated via an exhaustive enumeration of all the possibilities for this piece of information. Our algorithm runs all the possibilities, and among them chooses the best solution achieved. In the analysis it is sufficient to consider the solution obtained when we check the correct guess.

3.1 Two Users

We start with a relatively simple PTAS for this case. Here the search takes one or two rounds. For a given algorithm, its cost is simply $2n - n(1-p)(1-q)$, where p and q are the probabilities of finding the first user and the second user (respectively) in the second round. In this section, let p and q denote these probabilities in an optimal solution.

Let $p_j = p_{1,j}$ be the probability for the first user to be located in cell j, and let $q_j = p_{2,j}$ be the probability of the second user to be located in that cell.

Denote the probability intervals $I_0 = (0, \frac{\varepsilon}{n}]$, and for $1 \le i \le \left\lceil \log_{1+\varepsilon}\left(\frac{n}{\varepsilon}\right) \right\rceil$,

$$I_i = \left(\frac{\varepsilon}{n}(1+\varepsilon)^{i-1}, \frac{\varepsilon}{n}(1+\varepsilon)^{i}\right].$$

First guessing step: we guess k, which is the number of cells that are paged in the second round for the first user. Moreover, we guess the probability p of finding the first user in the second round. That is, we guess the index i such that $p \in I_i$.

Lemma 1. *The number of possibilities for the first guessing step is*

$$O\left(n \left[\log_{1+\varepsilon}\left(\frac{n}{\varepsilon}\right) + 2\right]\right) .$$

Proof. Clearly $0 < k < n$, since paging all cells for the same user in the first round always results in a second round, which gives cost $2n$, and this is sub-optimal. To conclude the proof, note that the number of intervals is at most $\log_{1+\varepsilon}\left(\frac{n}{\varepsilon}\right) + 2$. □

By Lemma 1, performing an exhaustive enumeration for the first guessing step can be done in polynomial time. We continue to analyze the iteration of this step in which we guess the "correct" values that correspond to OPT. We denote *the guess of* p by p' to be the upper bound of I_i; i.e., $p' = \frac{\varepsilon}{n}(1+\varepsilon)^i$.

The next step is to scale the probabilities of only the first user as follows. For all j define $r_j = p_j/p'$ to be the *scaled probability of cell* j *and the first user.* We consider the vector $R = (r_j)$ of the scaled probabilities that the first user is in cell j. We remove all cells with scaled probability larger than 1. Such cells cannot be paged for the first user in the second round, and therefore must be paged for the first user in the first round.

We further assign a *type* to each cell according to the following way. We define a set of intervals $\mathcal{J}$ as follows: $J_0 = (0, \varepsilon]$, and for all $\ell \geq 1$, $J_\ell = (\varepsilon \cdot (1+\varepsilon)^{\ell-1}, \varepsilon \cdot (1+\varepsilon)^{\ell}]$, and $\mathcal{J} = \{J_0, J_1, \ldots\}$. For each cell $1 \leq j \leq n$, we find the interval from $\mathcal{J}$ that contains r_j. That is, we compute a value t_j such that $r_j \in J_{t_j}$. The index t_j is the type of cell j. For values of t_j such that $t_j > 0$, we replace r_j with r'_j which is the upper bound of the interval J_{t_j}, i.e., $r'_j = \varepsilon(1+\varepsilon)^{t_j}$. Otherwise the value remains unchanged, i.e., $r'_j = r_j$. Note that the number of types is at most $\log_{1+\varepsilon}\left(\frac{1}{\varepsilon}\right) + 2 = O\left(\log_{1+\varepsilon}\left(\frac{1}{\varepsilon}\right)\right)$. We let S be the sum of scaled probabilities for type 0 cells (paged in round 2 for the first user). Let S' be the upper bound of this interval that contains S.

Second guessing step: We guess the amount of cells of each type that are paged for the first user in the second round. Moreover, we guess the value of S'.

Lemma 2. *The number of possibilities in the second guessing step is*

$$O\left(n^{\log_{1+\varepsilon}\left(\frac{1}{\varepsilon}\right)+2} \log_{1+\varepsilon}\left(\frac{1}{\varepsilon}\right)\right) .$$

Proof. The number of cells from each type is an integer between 0 and $k \leq n-1$ (clearly, bounded from above by the number of cells that exist for this type). The number of options for guessing S' is equal the number of intervals in $\mathcal{J}$ that is $O\left(\log_{1+\varepsilon}\left(\frac{1}{\varepsilon}\right)\right)$. □

Note that the number of possibilities for this guessing step is polynomial (for a fixed value of ε).

Next, for a given type of cell, $i > 0$, consider the cells which belong to this type. After the rounding, the difference between these cells is the probability of the second user to reside in this cell. Clearly, given that s such cells need to be paged for the first user in round 2, it means that the same set of cells should be paged for the second user in the first round. We prefer to page the cells with highest probability for the second user among the cells with a common type. A simple exchange argument shows that considering this option only (for rounded instances) may never increase the cost of the solution. For cells of type 0, define the density of a cell j to be q_j/p_j, this is the ratio between probabilities of the two users. Sort all cells of type 0 by non-increasing densities. Afterwards, take a minimal prefix of the sorted cells, such that the sum of scaled probabilities of the first user is at least $S' = \varepsilon \cdot (1+\varepsilon)^\ell$. If the sum of all the scaled probabilities of type 0 cells does not exceed S', then all these cells will be paged for the first user in round 2. If $S' > \varepsilon$, then the second user would prefer to page the most profitable such cells in round 1. We allow a slightly higher probability in the second round, and pick the most profitable cells greedily. Therefore, we may only increase the probability of finding the second user in round 1. If we could not exceed S', but instead page all type 0 cells in round 2 for the first user, then this may slightly harm the first user (see details below), but again may only increase the probability of finding the second user in round 1.

Consider the guess which guesses correctly the value k, the amounts of cells from each type, and the value of S'. The first step in the analysis would be to bound the relation between the probabilities of finding the users in the first step in the optimal solution and the solution we find. Let $\hat{p}$ and $\hat{q}$ be the corresponding probabilities to p and q in the resulting solution. Since the probability of the second user to be found in the first round may only increase, we get $1-\hat{q} \geq 1-q$, i.e. $\hat{q} \leq q$.

To bound $\hat{p}$ in terms of p, note that $p' \leq (1+\varepsilon)p + \frac{\varepsilon}{n}$. The rounding for cells whose rounded probabilities are not of type 0, results in a possible increase of probabilities by a multiplicative factor of $1+\varepsilon$. For cells of type 0, assume that $S' \in J_\ell$. If $\ell > 0$, we allow the sum (of scaled probabilities) for the chosen cells to exceed the value $\varepsilon \cdot (1+\varepsilon)^\ell$. However, since all values are at most ε, we get an additive error of at most that amount, in addition to a multiplicative rounding error of $1+\varepsilon$. For type 0, the worst case would be that the sum of scaled probabilities should have been zero, but it reaches ε and exceeds it by the same amount. Therefore, $\hat{p} \leq p'(1+\varepsilon) + 2\varepsilon p' = (1+3\varepsilon)p' \leq (1+3\varepsilon)(1+\varepsilon)p + (1+3\varepsilon)\frac{\varepsilon}{n} \leq (1+7\varepsilon)p + \frac{4\varepsilon}{n}$. The last inequality holds since $\varepsilon < 1$.

The cost is therefore

$$\begin{aligned} APX = n(1+\hat{p}+\hat{q}-\hat{p}\hat{q}) = n(1+\hat{p}+\hat{q}(1-\hat{p})) \leq n(1+\hat{p}+q(1-\hat{p})) \\ = n(1+\hat{p}(1-q)+q) \leq n(1+(1+7\varepsilon)p(1-q)+\frac{4\varepsilon}{n}+q) \\ \leq (1+7\varepsilon)n(1+p+q-pq)+4\varepsilon \leq (1+11\varepsilon)OPT = (1+\Theta(\varepsilon))OPT \end{aligned}$$

The last inequality follows from $OPT \geq 1$ which holds for any instance of the problem. Therefore, we have established the following theorem:

Theorem 3. *Problem CCS with two users, two rounds and $B = 1$ has a polynomial time approximation scheme.*

Remark 1. We can easily extend the scheme of this section to the case where there are also zero probabilities. To do so, we first guess the number of cells n_1 (n_2) to page the first (second) user in the first round such that the second (first) user has zero probability to be placed in this cell. Over the set of cells where both users have positive probability we apply the scheme of this section. Among the cells where the first (second) user has zero probability we will page for the second (first) user in the first round in the set of the n_2 (n_1) cells with the highest probability.

3.2 m Users

We continue with a PTAS for a general (constant) number of users. We prove the following theorem.

Theorem 4. *Problem CCS with a constant $d = m$ and $B = 1$ has a polynomial time approximation scheme.*

In this scheme, instead of using exact probabilities, we use rounded values. This is done both for input probabilities and probabilities uses by the algorithm. As a result, the sum of probabilities for an event whose probability must be 1, can change to a value which is not 1 (though it should still be close to 1). This does not affect the correctness of the algorithm since we do not treat the rounded values as probabilities of events throughout the execution. Instead of that, we fix all the (rounded) probabilities which the algorithm uses, and based on this, we solve a generalized knapsack problem.

The number of rounds that the search takes is at least 1 and at most m. Since locations of users are again independent, we can compute the expectation of the number of requests by calculating for each r, the probability of finding all users in at most r rounds. Given an algorithm (search scheme) let $q_{i,r}$ be the probability of finding user i in round r by a given solution, Then, the cost of this solution is

$$n\sum_{r=1}^{m}\left(1-\prod_{i=1}^{m}\left(\sum_{s=1}^{r-1}q_{i,s}\right)\right)=n\sum_{r=1}^{m}\left(1-\prod_{i=1}^{m}\left(1-\sum_{s=r}^{m}q_{i,s}\right)\right).$$

In this section we use these $(q_{i,r})$ notations to denote the values in a fixed optimal solution.

We start with a uniform rounding of the values $p_{i,j}$. In this section we use the following set of intervals for all rounding procedures. We define $\mathcal{J}$ as follows: $J_0=(0,\varepsilon^{2m+5}]$, and for all $k\geq 1$, $J_k=(\varepsilon^{2m+5}\cdot(1+\varepsilon)^{k-1},\varepsilon^{2m+5}\cdot(1+\varepsilon)^{k}]$, and $\mathcal{J}=\{J_0,J_1,\ldots\}$. Let s be such that $1\in J_s$. We replace the interval J_s by $(\varepsilon^{2m+5}\cdot(1+\varepsilon)^{s-1},1]$, and use only the $s+1$ first intervals. For each pair i,j where $1\leq i\leq m$, $1\leq j\leq n$, we find the interval from $\mathcal{J}$ that contains $p_{i,j}$. That is, we compute a value $t_{i,j}$ such that $p_{i,j}\in J_{t_{i,j}}$, and we define the *type* of the cell j to be the vector $(t_{1,j},\ldots,t_{m,j})$. For values of $p_{i,j}$ such that $t_{i,j}>0$, we replace $p_{i,j}$ with $p'_{i,j}$ which is the upper bound of the interval $J_{t_{i,j}}$, i.e., $p'_{i,j}=\varepsilon^{2m+5}\cdot(1+\varepsilon)^{t_{i,j}}$. Otherwise, the value remains unchanged, i.e., $p'_{i,j}=p_{i,j}$.

Corollary 1. *If $t_{i,j}>0$ then $p_{i,j}\leq p'_{i,j}\leq(1+\varepsilon)p_{i,j}$.*

We assign *sub types* to cells, based on the (unchanged) values of probabilities of type 0. If for all users $1\leq i\leq m$, $t_{i,j}>0$, there is no further partition to sub types. Otherwise, let the *weight* of cell j denoted as w_j be defined as $w_j=\max_{\{i|t_{i,j}=0\}}p_{i,j}$. For a type vector of a given cell j, create the following vector a^j of length m. The i-th entry a^j_i is -1 if $t_{i,j}>0$, and otherwise $a^j_i=\frac{p_{i,j}}{w_j}$.

We use the same partition into intervals in order to round and classify the vectors a^j. For an entry a^j_i, find the interval from $\mathcal{J}$ that contains a^j_i. Compute a value $t'_{i,j}$ such that $a^j_i\in J_{t'_{i,j}}$, then the sub type of the cell j is the vector $(t'_{1,j},\ldots,t'_{m,j})$ (where $t'_{i,j}=-1$ if $t_{i,j}>0$). We use the vector a'^j, where a'^j_i is the upper bound of the interval $J_{t'_{i,j}}$. If $a^j_i=-1$ then also $a'^j_i=-1$. We scale the probabilities again in the following way: if $t_{i,j}>0$ then $p''_{i,j}=p'_{i,j}$ and otherwise $p''_{i,j}=w_j a'^j_i$.

Corollary 2. *If $t_{i,j}=0$, $p''_{i,j}\leq w_j\left((1+\varepsilon)a^j_i+\varepsilon^{2m+5}\right)=(1+\varepsilon)p_{i,j}+w_j\varepsilon^{2m+5}$.*

Note that at least one entry in a'^j is 1, that is an entry ℓ_j for which $w_j = p_{\ell_j,j}$. We call the user ℓ_j *the leader* of the cell. Note that there may be other such unit entries, in the case that the maximum is not unique (in that case, ℓ_j is picked to be such a user with a minimum index), or if some user has a slightly smaller probability, but still in the last interval.

A cell is specified by its type, sub type, leader and weight (excluding cells with no sub type). Two cells j_1, j_2 have the same *general type* if they both have no sub type, or if they have the same type, same sub type and same leader. Their weights w_{j_1} and w_{j_2} may both take arbitrary values in $(0, \varepsilon^{2m+5}]$. Therefore, the number of general types is at most

$$m\left(2\log_{1+\varepsilon}\left(\frac{1}{\varepsilon}\right)^{2m+5}+3\right)^m \le \frac{m}{\varepsilon^{2m}} \ .$$

This follows from the choice of $\varepsilon < \frac{1}{(20m)^{m+1}\cdot m!}$, and from $\ln\frac{1}{\varepsilon} \le \frac{1}{\varepsilon}$, $\ln(1+\varepsilon) \ge \frac{\varepsilon}{2}$, and $m \ge 2$.

Given a cell, in order to specify a solution when restricted to this cell, we need to give a permutation of the users. That would be the order in which the cell is paged for the users.

Guessing step: For every general type and every permutation π (out of the $m!$ possible ones), we guess the number of cells of this general type that are paged in the order of the permutation π. Note that the sum of these numbers should be exactly the total number of cells of this general type. For every general type t, excluding the general type with no sub types, we also guess an interval for the total probability $P(t,\pi)$ that the cells of the general type t, paged using the permutation π, induce in the round where the leader of this general type is paged (i.e., the sum of their weights belongs to the guessed interval).

Lemma 3. *The number of possibilities for the first guessing step is polynomial.*

Proof. The number of combinations of general types and permutations is at most $\frac{m!m}{\varepsilon^{2m}}$. The number of possible guesses for a given permutation and cell is at most $n+1$ (this is an integer between 0 and n). The number of possibilities for a probability guess is

$$\left((2m+5)\log_{1+\varepsilon}\left(\frac{1}{\varepsilon}\right)+2\right) \le \frac{1}{\varepsilon^2} \ .$$

Therefore, the number of possibilities for the guessing step is at most $\left(\frac{n+1}{\varepsilon^2}\right)^{\frac{m!m}{\varepsilon^{2m}}}$. □

Given a guess, we distribute the cells to the permutations as follows. For a general type with no sub types, allocate the guessed number of cells of this type to each permutation, if possible. The exact distribution is not important. For other general types (i.e., with subtypes), given a specific general type, let ℓ be its leader. Denote the permutations by $\pi_1, \ldots, \pi_{m!}$. Given a permutation π_i, let t_i be the index for the probability interval guessed for this class, and let a_i be the number of cells guessed for it. Let n' be the number of cells that need to be distributed. Re-number the cells from 1 to n' and denote by w_j the weight associated with cell j. We need to distribute the n' cells to the $m!$ permutations, where for every permutation, an upper bound is given on the sum of probabilities of the cells allocated to it as well as an upper bound on the number of these cells. This corresponds to the following integer program. Let $X_{i,j}$ be an indicator variable whose value is 1 if cell j is allocated to permutation i. We apply the upper bounds of numbers and probabilities as follows. For each $1 \le i \le m!$,

$$\sum_{i=1}^{n'} X_{i,j} \le a_i \quad \text{and} \quad \sum_{i=1}^{n'} w_j \cdot X_{i,j} \le \varepsilon^{2m+5}(1+\varepsilon)^t \ .$$

We clearly have $\sum\limits_{i=1}^{m!} X_{i,j} \ge 1$ for all $1 \le j \le n'$, since each cell is assigned to at least one permutation. If it is assigned to more than one, one of its occurrences can be removed without violating the other constraints. The goal is to find a feasible integer point.

We relax the integrality constraint, and replace it with $X_{i,j} \ge 0$. We are left with a linear program which clearly has a solution if the original integer program does. Solving the linear program we can find a basic solution. This basic solution has at most $2m! + n'$ non zero variables (as the number of constraints). Clearly, each cell j has at least one non zero variable $X_{i,j}$ and thus we get that the number of cells that are not assigned completely to a permutation (i.e., that have more than one non zero variable associated with them) is at most $2m!$. These cells are removed and re-distributed to the permutations in order to satisfy the amounts of cells. In the worst case, all additional cells are assigned to one permutation, increasing its total probability in the round of the leader (i.e., its total weight) by an additive factor of $2m!w_j\varepsilon^{2m+5}$, and values which are no larger than $2m!w_j\varepsilon^{2m+5}$ in other rounds.

From now on, we consider the correct set of guesses. We would like to compute the differences between the values $q_{i,r}$ used by an optimal algorithm and the ones used by our scheme. Let $q'_{i,r}$ be the values used by the algorithm. I.e., $q'_{i,r}$ is the total probability of finding user i during round r by the scheme.

Lemma 4. $q'_{i,r} \le (1+7\varepsilon)q_{i,r} + \varepsilon^2$.

Proof. There are two types of changes in the value $q_{i,r}$, multiplicative changes and additive changes. The first two rounding steps are taken for pairs of cells and users. By Corollary 1 and 2, we conclude that $p''_{i,j} \le (1+\varepsilon)p_{i,j} + w_j\varepsilon^{2m+5}$. Therefore, the sum of additive changes in all pairs of cells and leaders is bounded by ε^{2m+5} times the sum of all probabilities, which is m. Hence, $m\varepsilon^{2m+5}$ bounds the resulting additive change in each value $q_{i,r}$.

The next rounding is of $P(t, \pi)$. Another multiplicative factor of $1+\varepsilon$ is introduced at this time. Moreover, the probability of a given permutation π may increase by an additive factor of $(2m!+1)\varepsilon^{2m+5}$. In the worst case, this additive growth may happen for every pair of general type and permutation. The term $2m!\varepsilon^{2m+5}$ is due to the last phase where the fractional solution to the linear program is rounded. An additional ε^{2m+5} is due to the rounding of $P(t, \pi)$ to right end points of probability intervals.

Recall that the number of combinations of general types and permutations is at most $\binom{m!m}{\varepsilon^{2m}}$, thus the additive factor is at most $\binom{m!m}{\varepsilon^{2m}}(2m!+1)\varepsilon^{2m+5} \le \varepsilon^4$. Summarizing we get,

$$q'_{i,r} \le ((1+\varepsilon)q_{i,r} + \varepsilon^{2m+4})(1+\varepsilon) + \varepsilon^4 \le (1+3\varepsilon)q_{i,r} + \varepsilon^2 \ .$$

□

We compute an upper bound for the change in the goal function value.

$$\begin{aligned}
& n\sum_{r=1}^{m}\left(1-\prod_{i=1}^{m}\left(1-\sum_{s=r}^{m}q'_{i,s}\right)\right) && (1)\\
\le\ & n\sum_{r=1}^{m}\left(1-\prod_{i=1}^{m}\left(1-\sum_{s=r}^{m}\left((1+3\varepsilon)q_{i,s}+\varepsilon^2\right)\right)\right) && (2)\\
\le\ & n\sum_{r=1}^{m}\left(1-\prod_{i=1}^{m}\left(1-m\varepsilon^2-(1+3\varepsilon)\sum_{s=r}^{m}q_{i,s}\right)\right) && (3)\\
\le\ & n\sum_{r=1}^{m}\left(m^2\varepsilon^2+(1+3\varepsilon)\left(1-\prod_{i=1}^{m}\left(1-\sum_{s=r}^{m}q_{i,s}\right)\right)\right) && (4)\\
\le\ & n\left(m^3\varepsilon^2+(1+3\varepsilon)\sum_{r=1}^{m}\left(1-\prod_{i=1}^{m}\left(1-\sum_{s=r}^{m}q_{i,s}\right)\right)\right) && (5)\\
\le\ & \varepsilon OPT+(1+3\varepsilon)OPT=(1+4\varepsilon)OPT\ , && (6)
\end{aligned}$$

where (2) follows by Lemma 4, and (3),(5) follow by simple algebra. Next, (4) follows since given a set of m independent random events, the probability of their union is multiplied by at most $(1+3\varepsilon)$ if we multiply the probability of each event in this set by that amount, and if we increase the probability of each event by an additive factor of $\rho = m\varepsilon^2$, then the probability of the union increases by at most $m\rho = m^2\varepsilon^2$. Finally (6) follows since $OPT \ge n$ and $\varepsilon < \frac{1}{m^3}$. This completes the proof of Theorem 4.

4 Polynomial Time Algorithms for Finding Optimal Semi-Adaptive Search Protocols

In this section we consider the problem of computing an optimal semi-adaptive search protocol for tight instances of CCS. We show polynomial time algorithms for solving this problem. We describe a fast algorithm to solve the two-users two-rounds case (this solution holds for adaptive systems as well). Further, we present a dynamic programming based algorithm to solve the CCS problem with a constant number of users.

4.1 Two Users

We assume that there are two users and two rounds and $B = 1$. Bar-Noy and Naor [3] showed that computing an optimal oblivious protocol for this case is an NP-hard problem. They left as an open question to decide if computing an optimal adaptive search protocol is polynomially solvable.

We note that for this case, given a search plan for one round, since we may need to search each of the two users in every cell, the plan for the second round is fixed. Changes in this plan can follow only from users being found already in the first round. However, a user that was not found, must be searched in every cell, and there is no room for changes in the plan. This means that in this case the semi-adaptive search protocol is equivalent to the adaptive search protocol.

Therefore, by computing an optimal semi-adaptive search protocol in polynomial time, we provide a positive answer for this question.

Our algorithm, denoted by Alg, guesses k that is defined as the number of cells that an optimal solution pages for the first user in the first round. This guess is implemented by an exhaustive enumeration using the fact that k is an integer in the interval $[0, n]$, and then returning the best solution obtained during the exhaustive enumeration. We next analyze the iteration in which the guess is correct.

Denote by $I_i^k = p_{1,i} \cdot (n-k) - p_{2,i} \cdot k$ the *index* of cell i in the k-th iteration. Our algorithm sorts the indices of the cells in non-decreasing order, and then it picks the first k cells (in the sorted list). These picked cells are paged for the first user in the first round, whereas the other cells are paged for the second user in the first round.

Theorem 5. *Alg returns an optimal semi-adaptive search protocol.*

Proof. To prove the theorem, it is sufficient to prove the following claim: Assume that there exists a pair of cells i, j with $I_i^k \geq I_j^k$ such that the optimal solution pages j for the first user in the first round, and it pages i for the second user in the first round. Then, replacing the role of i and j (i.e., the new solution pages i for the first user in the first round, and it pages j for the second user in the first round), results in another optimal solution.

To prove the claim we first argue that the decrease in the solution cost resulting by this replacement is $(n-k) \cdot (p_{1,i} - p_{1,j}) + k \cdot (p_{2,j} - p_{2,i})$. To see this, note that the probability of finding the first user in the first round increases by $p_{1,i} - p_{1,j}$, thus gaining an expected decrease of the cost by $(n-k) \cdot (p_{1,i} - p_{1,j})$. Similarly for the second user the expected decrease in the cost is $k \cdot (p_{2,j} - p_{2,i})$.

However, $(n-k) \cdot (p_{1,i} - p_{1,j}) + k \cdot (p_{2,j} - p_{2,i}) = p_{1,i} \cdot (n-k) - p_{2,i} \cdot k - [p_{1,j} \cdot (n-k) - p_{2,j} \cdot k] = I_i^k - I_j^k \geq 0$, where the last inequality follows by the assumption. Therefore, the replacement of the roles of i and j results in another optimal solution, as we claimed. □

The next corollary answers the open question implied by [3].

Corollary 3. *Alg returns an optimal adaptive search protocol.*

Proof. As stated above, the definitions of the adaptive and semi-adaptive search protocols are equivalent for the case of two users and two rounds. □

4.2 m Users

We assume that there are $m = d$ users and d rounds where d is a constant, and $B = 1$. We show a dynamic programming procedure that computes an optimal semi-adaptive search protocol for this case.

Our first step is to guess for each permutation π of the users the number of cells that are paged according to the permutation π (i.e., the i-th user is paged for the cell in round $\pi(i)$). Note that the number of possibilities for this guessing step is bounded by $(n+1)^{d!}$ (since the number of occurrences of each permutation is an integer between 0 and n). Therefore we can implement an exhaustive enumeration of this guessing step in polynomial time. We assume that the guessing step outputs a vector $T = (t_1, \ldots, t_{d!})$ where t_j is the number of cells with permutation π^j.

Using T, we can compute for each user i, and each round r, the number of cells that the optimal semi-adaptive search protocol pages for user i in round r. We denote this number by $n_{i,r}$.

We next compute for all i and all $r' \leq d-1$ the *gain* that we will get if we find user i in the r'-th round, i.e., $N_{i,r'} = \sum_{r=r'+1}^{d} n_{i,r}$. To motivate this definition note that if we find user i in the r'-th round, then we stop looking for user i and thus we gain $\sum_{r=r'+1}^{d} n_{i,r}$ requests with respect to the solution that pages user i in all the cells.

For a cell c, the *expected gain of using permutation* π^j *for cell* c is defined as $E_{j,c} = \sum_{i=1}^{d} p_{i,c} \cdot N_{i,\pi^j(i)}$. Note that our goal is equivalent to allocating permutations to cells so as to maximize the total expected gain while satisfying the bounds on the number of cells that can be allocated to each permutation (the bounds that are given by the guess).

This last problem can be solved by using a dynamic programming procedure as follows: Define $F_c(t_1, t_2, \ldots, t_{d!}))$ to denote the optimal total expected gain by allocating permutations to cells $c, c+1, \ldots, n$ such that permutation π^j is allocated at most t_j cells. Then

$$F_n(t_1, t_2, \ldots, t_{d!}) = \max_{j: t_j \geq 1} E_{j,n},$$

and for $c < n$ the following holds:

$$F_c(t_1, t_2, \ldots, t_{d!}) = \max_{j: t_j \geq 1} \{E_{j,c} + F_{c+1}(t_1, t_2, \ldots, t_{j-1}, t_j - 1, t_{j+1}, \ldots, t_{d!})\}.$$

This completes our algorithm. Therefore, we established the following theorem:

Theorem 6. *There is a polynomial time algorithm that computes an optimal semi-adaptive search protocol for tight instances of CCS where d is a constant.*

5 Open Questions

We list several open questions that are left for future research:

- Determine the complexity status of computing an optimal adaptive search protocol for tight instances with $d = m > 2$.
- Find an FPTAS or prove its non-existence (by showing that the problem is NP-hard in the strong sense for fixed constant values of $d = m$) for computing an optimal oblivious search protocol for tight instances with a fixed constant number of users.
- Find a PTAS or prove its non-existence (by showing that the problem is APX-hard) for computing an optimal oblivious search protocol for an arbitrary tight instance. The running time of the PTAS should be polynomial in n and in $d = m$.

References

1. A. Bar-Noy and I. Kessler. Tracking mobile users in wireless networks. *IEEE Transaction on Information Theory*, 39:1877–1886, 1993.
2. A. Bar-Noy and G. Malewicz. Establishing wireless conference calls under delay constraints. *Journal of Algorithms*, 51(2):145–169, 2004.
3. A. Bar-Noy and Z. Naor. Establishing a mobile conference call under delay and bandwidth constraints. In *The 23rd Conference of the IEEE Communications Society (INFOCOM2004)*, volume 1, pages 310–318, 2004.
4. L. Epstein and A. Levin. A PTAS for delay minimization in establishing wireless conference calls. In *Proc. of the 2nd Workshop on Approximation and Online Algorithms (WAOA2004)*, pages 36–47, 2004.
5. M. R. Garey and D. S. Johnson. *Computer and Intractability.* W. H. Freeman and Company, New York, 1979.
6. D. Goodman, P. Krishnan, and B. Sugla. Minimizing queuing delays and number of messages in mobile phone location. *Mobile Networks and Applications*, 1(1):39–48, 1996.
7. S. Madhavapeddy, K. Basu and A. Roberts. Adaptive paging algorithms for cellular systems. In *Wireless Information Networks: Architecture, Resource Management and Mobile Data*, pages 83–101, 1996.
8. C. Rose and R. Yates. Minimizing the average cost of paging under delay constraints. *Wireless Networks*, 1(2):211–219, 1995.