Paging Multiple Users in Cellular Network: Yellow Page and Conference Call Problems

Amotz Bar-Noy[1], Panagiotis Cheilaris[2], and Yi Feng[1]

[1] Department of Computer Science, City University of New York
[2] Center for Advanced Studies in Mathematics, Ben-Gurion Univ. of the Negev, Israel

Abstract. Mobile users are roaming in a zone of cells in a cellular network system. The probabilities of each user residing in each cell are known, and all probabilities are independent. The task is to find any one, or all, of the users, by paging the cells in a predetermined number of rounds. In each round, any subset of the cells can be paged. When a cell is paged, the list of users in it is returned. The paging process terminates when the required user(s) are found. The objective is to minimize the expected number of paged cells. Finding any one user is known as the yellow page problem, and finding all users is known as the conference call problem. The conference call problem has been proved NP-hard, and a polynomial time approximation scheme exists. We study both problems in a unified framework. We introduce three methods for computing the paging cost. We give a hierarchical classification of users. For certain classes of users, we either provide polynomial time optimal solutions, or provide relatively efficient exponential time solutions. We design a family of twelve fast greedy heuristics that generate competitive paging strategies. We implement optimal algorithms and non-optimal heuristics. We test the performance of our greedy heuristics on many patterns of input data with different parameters. We select the best heuristics for both problems based on our simulation. We evaluate their performances on randomly generated Zipf and uniform data and on real user data.

1 Introduction

In cellular network systems, when a call arrives for a user, the system must know in which cell the user is located in order to establish a connection. Such a locating process is usually conducted by paging.

Let a user roam in a set of N cells: $\{C_1, \ldots, C_N\}$. With probability p_n the user is located in cell C_n. All p_is are independent. The system pages the cells in rounds. Once the cell that contains the user is paged, it reports to the system and the paging process terminates. To ensure the quality of service, the paging process must be conducted in at most D rounds. In each round, any subset of the cells could be paged. Thus, a paging strategy is an ordered D-partition of the cells, such that, in the dth round, the cells in dth part are paged. The *cost* of a paging strategy is the expected number of cells paged until the user is located. Our objective is to design algorithms that compute paging strategies that minimizes paging costs.

In a variety of applications, e.g., to establish a conference call, we need to mutually page multiple users in a cellular network system. Suppose M users roam in a set of N cells, $\{C_1, \ldots, C_N\}$. With probability $p_{m,n}$, user m is located in cell C_n. All probabilities $p_{m,n}$s are independent. Our goal is find any one, some, or all of the users in at most D rounds. When a cell is paged, we become aware of the list of user(s) that reside in it. The paging process terminates as soon as the desired user(s) are found. The same objective remains: to minimize the expected number of paged cells.

In the multiple user paging problem, on one extreme, we want to find all the users so that a conference call can be established. We call this the *conference call* problem. On the other extreme, we only need to find any one of the users, no matter who. This is similar to when we look for information in a yellow page book: We stop after finding the first useful information in a category. We call this the *yellow page* problem. In this paper, we study both the yellow page and conference call problems, showing a kind of duality between the two problems.

Example: Suppose 2 users roam in 3 cells, C_1, C_2, C_3, with probabilities 0.5, 0.3, 0.2 (user 1) and 0.4, 0.1, 0.5 (user 2), respectively. For the paging strategy that pages C_1 and C_2 in the first round and C_3 in the second round, if we only search for user 1, by probability (0.5+0.3) we page 2 cells; by probability 0.2 that we fail in the first round, we page 3 cells. The paging cost is $(0.5+0.3)\cdot 2+0.2\cdot 3 = 2.2$. For the same paging strategy, in the conference call problem, the probability that both users are in cells C_1, C_2 (i.e., paging stops after the first round and only 2 cells are paged) is $(0.5 + 0.3) \cdot (0.4 + 0.1) = 0.4$. Otherwise, all 3 cells are paged. The expected cost is $0.4 \cdot 2 + (1 - 0.4) \cdot 3 = 2.6$. Again for the same strategy, in the yellow page problem, both users are in C_3 with probability $0.2 \cdot 0.5 = 0.1$, in which case no user is found in the first round and we page all 3 cells. Otherwise, we only page the first 2 cells. The expected cost is $0.1 \cdot 3 + (1 - 0.1) \cdot 2 = 2.1$.

Motivation: The cost of updating locations by users in Cellular Networks could be very expensive if users update their location each time they move from cell to cell. As a result, many systems partition the cells into zones where users report their new locations only when they enter a different zone. To locate a user, the system needs to page it in the zone of cells where it last reports its location. This scheme is part of one of the *location management* solutions (see survey [AMHUW]) where the paging step described above is a common task. It follows that any a priori knowledge of user locations, that can either be provided by the users or can be extracted from log files, will help the system to reduce the expected paging cost. There are several other applications to the paging problem. In wireless sensor networks, the system collects information from sensors by probing them. Such probes are costly (usually batteries of wireless sensors) and therefore the system needs to arrange efficient information collection strategy such that the number of probes is minimized. Another application is the task of searching information on the Internet where search engines consume computational resource by accessing Internet cache databases. Since search request comes massively and frequently, the search engine system needs to better schedule the resource accessing by processing multiple search requests together, and arrange them in a good schedule.

Previous work: The problem of searching a single user in cellular networks under delay constraint D has been studied in [GKS,RY1,KGWH,BFG]. An efficient, $O(ND)$ time, algorithm computing the optimal cost solution exists based on dynamic programming. The papers [RY2,GH] considered the task of searching multiple users but as a collection of many single user search tasks that occur concurrently and therefore finding each one of the users is a success.

The conference call problem was introduced in [BM2]. The authors proved NP-hardness of the problem, and showed a natural greedy algorithm is an $\frac{e}{e-1}$-approximation. In [EL2], the authors introduced a PTAS to the conference call problem for any constant number of paging rounds. In [BN], the authors studied the conference call problem with an additional bandwidth constraint, such that in each round, a limited number of users in each cell can be paged. The NP-hardness of the problem was shown, and a fast heuristic that minimized both delay constraint and paging cost was presented.

In [EL1], the authors explored another version of the conference call problem: instead of paging a cell and collecting all users in it, the system queries a cell by asking if a specific user is in it, and gets a boolean answer. The authors showed hardness and provided approximation algorithms in this setting.

The yellow page problem has not been studied to the best of our knowledge in the context of partitioning and scheduling. In [KKM], the authors explored a more general dynamic version of the problem. They showed it is NP-hard and provided a 4-approximation algorithm. In [CFK], the authors studied a similar problem. The problem differs from the yellow page problem in the parameter of number of users M. They proved its NP-hardness and provided an efficient approximation algorithm. In [W], the author discussed the yellow page problem in the context of efficiently finding alternative investment and provided heuristical solutions in a continuous model (in contrast to our discrete model).

Our contribution: In the problem of paging multiple users in a cellular network, computing the paging cost itself becomes an important task and is essential to understand the problem. We present three methods for computing the paging cost: two of them will be used in proving lemmas and constructing optimal algorithms, the other (the most efficient) one will be used in heuristics. We conjecture that, in addition to the conference call problem, the yellow page problem is also NP-hard, therefore we give a hierarchical classification of users and provide efficient optimal algorithms for certain interesting classes of users. When the delay constraint equals the number of cells, i.e., $D = N$, we present polynomial time algorithms for monotonic users and disjointed users, which are very representative cases for some applications. The optimal solution in the $D = N$ case is a permutation of the cells and thus a naive optimal algorithm has time complexity on the order of $N!$. However, by exploiting properties of the optimal solution, we design instead a dynamic programming algorithm of time complexity on the order of 2^N. This algorithm will be later used as a benchmark to evaluate our heuristics. In addition to optimal algorithms, we design a family of twelve greedy heuristics that belong to four groups. To evaluate their performance, we test them on several types of data with many settings and parameters. We also test them on a real data with 171929 appearances of 996 users in 5625 cells in 31 consecutive days, acquired from a cell phone provider. We find the best heuristic for each problem that outperform the other heuristics on almost all instances. We also measure the running time of our algorithms on a real machine in addition to the theoretical analysis.

2 Preliminaries

Let M users roam N cells and $p_{m,n}$ be the probability of user m being in cell C_n. Given a bound D on the number of rounds (with $1 \leq D \leq N$), a paging strategy $\mathcal{A} = \langle A_1, \ldots, A_D \rangle$ is an ordered partition of the set of cells $\{C_1 \ldots C_N\}$, such that, in the dth round, cells in part A_d are paged. Given a paging strategy $\mathcal{A} = \langle A_1, \ldots, A_D \rangle$, let $P_{m,d}$ be the the probability of user m being in any cell in part A_d, i.e., $P_{m,d} = \sum_{C_n \in A_d} p_{m,n}$. Denote the suffix probability by $R_{m,d} = \sum_{i=d+1}^{N} P_{m,i}$ and the prefix probability by $Q_{m,d} = \sum_{i=1}^{d} P_{m,i}$. Let S_d be the number of cells in the first d parts, A_1, …, A_d. By convention, $S_0 = 0$.

We describe three methods that compute the paging cost of the yellow page problem and the conference call problem. The first two are used in our proofs while the third is used by our simulations since it is computationally the most efficient. Let $\mathrm{YP}(\mathcal{A})$ be the cost of the yellow page problem and $\mathrm{CC}(\mathcal{A})$ be the cost of the conference call problem, on paging strategy $\mathcal{A}$. Consider the vector $\mathbf{d} = (d_1, \ldots, d_M) \in \{1, \ldots, D\}^M$, which encodes in which part each user is (i.e., user m is in part d_m, for $1 \leq m \leq M$).

Combinatorial Computation: For a part location vector $(d_1, \ldots, d_M)$, which occurs with probability $\prod_{m=1}^{M} P_{m,d_m}$, the strategy pays a cost of $S_{\min\{d_1,\ldots,d_M\}}$ for the yellow page problem (i.e., it pages in parts until it finds the first part that contains some user) and therefore:

$$\mathrm{YP}(\mathcal{A}) = \sum_{\mathbf{d} \in \{1,\ldots,D\}^M} \left(S_{\min\{d_1,\ldots,d_M\}} \cdot \prod_{m=1}^{M} P_{m,d_m} \right) . \qquad (1)$$

Similarly, the cost of the conference call problem is:

$$\mathrm{CC}(\mathcal{A}) = \sum_{\mathbf{d} \in \{1,\ldots,D\}^M} \left(S_{\max\{d_1,\ldots,d_M\}} \cdot \prod_{m=1}^{M} P_{m,d_m} \right) , \qquad (2)$$

where the only difference is that for each part location vector, the strategy pays a cost of $S_{\max\{d_1,\ldots,d_M\}}$, because it has to page also all cells in the last part that contains a user.
Time Complexity: $\Theta(MN + (M + D)D^M)$.

Recursive computation: In the yellow page problem, we extend the definition of cost, so that $\mathrm{YP}(\langle A_d, \ldots, A_D \rangle)$ is the expected cost of paging parts $\langle A_d, \ldots, A_D \rangle$ given the condition that no user is found in parts $A_1, \ldots, A_{d-1}$. If no user has been found in the first $(D-1)$ rounds, we must page all the cells in A_D, i.e., $\mathrm{YP}(\langle A_D \rangle) = |A_D|$. The recursion step is

$$\mathrm{YP}(\langle A_d, \ldots, A_D \rangle) = |A_d| + \frac{\prod_{m=1}^{M} R_{m,d}}{\prod_{m=1}^{M} R_{m,d-1}} \mathrm{YP}(\langle A_{d+1}, \ldots, A_D \rangle) \ , \tag{3}$$

because to page users in $A_d, \ldots, A_D$ given that no users are in $A_1, \ldots, A_{d-1}$, we must page cells in A_d by paying a cost of $|A_d|$ and if no user is found there (an event with probability $\prod_{m=1}^{M} R_{m,d} / \prod_{m=1}^{M} R_{m,d-1}$) we pay an extra cost of $\mathrm{YP}(\langle A_{d+1}, \ldots, A_D \rangle)$.

Let $\mathrm{CC}(\langle A_1, \ldots, A_d \rangle)$ be the conference call cost of paging all users in parts $A_1, \ldots, A_d$. The recursion base is $\mathrm{CC}(\langle A_1 \rangle) = \prod_{m=1}^{M} Q_{m,1} |A_1|$, since if all users are in cells of part A_1, we page $|A_1|$ cells. The recursion step is

$$\mathrm{CC}(\langle A_1, \ldots, A_d \rangle) = \mathrm{CC}(\langle A_1, \ldots, A_{d-1} \rangle) + \left(\prod_{m=1}^{M} Q_{m,d} - \prod_{m=1}^{M} Q_{m,d-1} \right) |A_d|, \tag{4}$$

because to page all users in parts $A_1, \ldots, A_d$, we must page parts $A_1, \ldots, A_{d-1}$ first and pay a cost of $\mathrm{CC}(\langle A_1, \ldots, A_{d-1} \rangle)$, and with probability that at least one user is in A_d, we pay an extra cost of $|A_d|$.
Time Complexity: $\Theta(MN + MD)$.

Exclusive Computation: With probability $\prod_{m=1}^{M} R_{m,d-1}$, all users are in parts $\{A_d \ldots A_D\}$; with probability $\prod_{m=1}^{M} R_{m,d}$, all users are in parts $\{A_{d+1} \ldots A_D\}$. Thus, with the difference of the above probabilities, at least one user is in part A_d but no user is in parts $A_1 \ldots A_{d-1}$, in which case we need to page exactly S_d cells. Summing through $d = 1, \ldots, D$, we have the cost for the yellow page problem.

$$\mathrm{YP}(\mathcal{A}) = \sum_{d=1}^{D} \left(S_d \cdot \left(\prod_{m=1}^{M} R_{m,d-1} - \prod_{m=1}^{M} R_{m,d} \right) \right) \tag{5}$$

Similarly, in the conference call problem, with probability $\prod_{m=1}^{M} Q_{m,d}$, all users are in parts $\{A_1 \ldots A_d\}$ and with probability $\prod_{m=1}^{M} Q_{m,d-1}$, all users are in parts $\{A_1 \ldots A_{d-1}\}$. Thus, with the difference of the above probabilities, at least one user is in part A_d and all users are in parts $A_1 \ldots A_d$, in which case we need to page exactly S_d cells.

$$\mathrm{CC}(\mathcal{A}) = \sum_{d=1}^{D} \left(S_d \cdot \left(\prod_{m=1}^{M} Q_{m,d} - \prod_{m=1}^{M} Q_{m,d-1} \right) \right) \tag{6}$$

Time Complexity: $\Theta(MN + MD)$

3 Types of Users

In [BM2], the authors proved that the conference call problem is NP-hard. We conjecture that the yellow page problem is also NP-hard. In Sec.4, we observe some "duality" between the two problems. Since the general setting is hard to tackle, we study some interesting restricted classes of instances, for which we provide more efficient optimal solutions – some have polynomial running time and some have improved exponential running time. Toward that goal, we present a hierarchical classification of types of users.

We define a few properties for a set of M users according to their probabilities in the set of cells. A set of users is *identical* if for any cell $C_n \in \{C_1, \ldots, C_N\}$, $p_{1,n} = \cdots = p_{M,n}$. A set of users is *uniform* if for each user $m = 1 \ldots M$, $p_{m,n}$ is either 0 or $1/k$, where k is the number of non-zero entries in $\{p_{m,1}, \ldots, p_{m,N}\}$. A set of users is *similar*, if for all users $m = 2, \ldots, M$, $\{p_{m,1}, \ldots, p_{m,N}\}$ is some permutation of user 1's probabilities $\{p_{1,1}, \ldots, p_{1,N}\}$. A set of users is *disjoint* if for each cell $C_n \in \{C_1, \ldots, C_N\}$, there is exactly one non-zero entry $p_{m,n}$, for some $m = 1, \ldots, M$. We present the combinations of these properties in a hierarchy in Fig. 2 in Appendix A.

4 Optimal Solutions

In this section, we first present efficient algorithms for both problems to compute the paging cost for a predetermined order of the cells. Based on these algorithms, we describe relatively efficient optimal solutions to some types of users (and their ancestor types in the hierarchy described in Fig. 2 in Appendix A). We defer some of the proofs of our statements to Appendix B.

Given an order of the cells, say $\langle C_1, \ldots, C_N \rangle$ (without loss of generality), a paging strategy $\mathcal{A} = \langle A_1, \ldots, A_D \rangle$ is said to *respect the above order* if for any cells C_i, C_j with $i < j$, we have $C_i \in A_{d_i}$ and $C_j \in A_{d_j}$ with $d_i \leq d_j$. Given an order of the cells, Algorithms 1 (for the YP problem) and 2 (for the CC problem) compute the optimal paging cost and corresponding strategy that respects this order, in polynomial time.

In Algorithm 1, let $h^{\text{yp}}_{n,d}$ denote the optimal cost of paging cells $\{C_n, \ldots, C_N\}$ in d rounds given the condition that no user locates in cells $\{C_1, \ldots, C_{n-1}\}$. Our objective is to find $h^{\text{yp}}_{1,D}$. It is not difficult to see that $h^{\text{yp}}_{n,1} = N - n + 1$. To compute $h^{\text{yp}}_{n,d}$, we need to search through all possible js with $n + 1 \leq j \leq n - d + 1$ for the strategy of minimum cost that pages cells $\{C_j, \ldots, C_N\}$ in the last $(d - 1)$ rounds and pages cells $\{C_n, \ldots, C_{j-1}\}$ in the previous round; the inner loop $(*)$ follows equation (3)).

Algorithm 1 Dynamic programming algorithm, respect order $\langle C_1, \ldots, C_N \rangle$, yellow page, cost = DPYP$(p[M][N], D)$

```
for $n = 1 \ldots N$ do
    $h^{\text{yp}}_{n,1} \leftarrow N - n + 1$
for $d = 2 \ldots D$ do
    for $n = 1 \ldots (N - d)$ do
        $h^{\text{yp}}_{n,d} \leftarrow \min_{j=n+1}^{n-d+1} |j - n| + (\prod_{m=1}^{M} R_{m,n} - \prod_{m=1}^{M} R_{m,j}) / \prod_{m=1}^{M} R_{m,n} \cdot h^{\text{yp}}_{j,d-1}$ $(*)$
return $h^{\text{yp}}_{1,D}$
```

In Algorithm 2, let $h^{\text{cc}}_{n,d}$ denote the optimal cost of paging cells $\{C_1, \ldots, C_n\}$ in d rounds. Our objective is to find $h^{\text{cc}}_{N,D}$. It is not difficult to see that $h^{\text{cc}}_{n,1} = n$. To compute $h^{\text{cc}}_{n,d}$, we need to search through all possible js with $(d-1) \leq j < n$ for the strategy of minimum cost that pages cells $\{C_1, \ldots, C_j\}$ in the first $(d-1)$ rounds and pages cells $\{C_{j+1}, \ldots, C_n\}$ in the dth round; the inner loop $(*)$ follows equation (4).

Algorithm 2 Dynamic programming algorithm, respect order $\langle C_1, \ldots, C_N \rangle$, conference call, cost = DPCC($p[M][N], D$)

```
for $n = 1 \ldots N$ do
    $h^{\text{cc}}_{n,1} \leftarrow n$
for $d = 2 \ldots D$ do
    for $n = d \ldots N$ do
        $h^{\text{cc}}_{n,d} \leftarrow \min_{j=d-1}^{n-1} h^{\text{cc}}_{j,d-1} + \left(\prod_{m=1}^{M} Q_{m,n} - \prod_{m=1}^{M} Q_{m,j}\right) |n-j|$ $(*)$
return $h^{\text{cc}}_{N,D}$
```

The correctness of Algorithms 1 and 2 follows from the fact that, under the particular order constraint, any sub-partition of an optimal paging strategy must be sub-optimal within itself; otherwise, replacing the sub-partition with the alternative sub-optimal paging strategy would gain a better paging strategy than optimal. We omit details of a proof here, but a formal proof can be adapted from [KGWH].

Lemma 1. *The running time of Algorithms 1 and 2 is $\Theta(MDN^2)$*

Monotonic Users: A set of users is called *monotonic* if there is a permutation of cells, w.l.o.g., say $\langle C_1, \ldots, C_N \rangle$, such that $p_{m,1} \geq \cdots \geq p_{m,N}$ for every $m \in \{1, \ldots, M\}$. Let this permutation be the monotonic order of the cells for the monotonic users.

Lemma 2. *For monotonic users, the optimal paging strategies for both the yellow page and conference call problems follow the monotonic order of the cells.*

Applying Algorithms 1 and 2 on the monotonic order yields:

Corollary 1. *The optimal paging strategies for both the yellow page problem and the conference call problem for monotonic users can be computed in polynomial time for any D, M, and N.*

D=N, Duality: An interesting case is when $D = N$, i.e., *sequential* searching, in which a paging strategy is a permutation of the cells. The conference call problem has been proved NP-Hard in [BM2]. Although we have not yet proved the NP-hardness for yellow page problem, we show that there is some kind of *duality* between the two problems.

Lemma 3. *Let $\mathcal{A}$ be a paging strategy that pages M users in N cells in $D = N$ rounds. Let an instance of yellow page with probabilities $p_{m,n}$. Let another instance for the conference problem with probabilities $q_{m,n} = p_{m,N+1-n}$. Then, YP$(\mathcal{A}, p_{m,n})$ + CC$(\mathcal{A}, q_{m,n}) = N + 1$.*

Corollary 2. *When $D = N$, the maximization problem of yellow page is equivalent to the minimization problem of conference call, and vice versa.*

D=N, Arbitrary User: When $D = N$, a brute-force approach to find the optimal strategy is to test all permutations of the cells. This method requires $\Theta(N!)$ time. We present lemmas, which allow us to give instead a $\Theta(2^N)$ algorithm that generates the optimal permutation.

Lemma 4. *In the yellow page problem, if $\langle C_{i_1}, \ldots C_{i_n}\rangle$ is an optimal paging strategy of paging cells $\{C_{i_1}, \ldots C_{i_n}\}$ given the condition that no user is located in the rest of cells, then any suffix of it, $\langle C_{i_k}, \ldots C_{i_n}\rangle$, for $1 \leq k \leq n$, is an optimal paging strategy that pages cells $\langle C_{i_k}, \ldots C_{i_n}\rangle$ given no user is located in any other cells (except in $\{C_{i_k}, \ldots C_{i_n}\}$).*

Lemma 5. *In the conference call problem, if $\langle C_{i_1}, \ldots C_{i_n}\rangle$ is an optimal paging strategy of paging cells $\{C_{i_1}, \ldots C_{i_n}\}$ in the first i_n rounds, then any prefix of it, $\langle C_{i_1}, \ldots C_{i_k}\rangle$, for $1 \leq k \leq n$, is an optimal paging strategy that pages cells in $\{C_{i_1}, \ldots C_{i_k}\}$ in the first i_k rounds.*

In light of Lemma 4, Algorithm 3 computes an optimal paging strategy for the yellow page problem. A dedicated array Best$[2^N]$ is used in the algorithm. Best$[k]$ records the optimal sub-strategy of paging cells $\{C_{i_1}, \ldots, C_{i_l}\}$ where $i_1, \ldots, i_l$ are the bits of 1 after converting k into binary. In the first for loop, we initialize the optimal paging strategy of a single cell given no user is found in other cells which is to page the cell itself in the only around. For paging l cells in l rounds, we search through all possible cases that page one of the l cells in the first round, and page the other $(l-1)$ cells optimally in the remaining $(l-1)$ rounds. The data structure Best is set up for random access any optimal sub-strategy that has been already computed.

Similarly, we construct Algorithm 4 for the conference call problem. The only difference with the yellow page algorithm is the recursive computation of the cost according to (4).

Algorithm 3 $D = N$, arbitrary users: compute the optimal cost and strategy for yellow page using dynamic programming; opt = YPDP(A)

```
for $\forall A$, that $|A| = 1$ do
    $\text{Best}[A]_{\text{cost}} \leftarrow 1$
    $\text{Best}[A]_{\text{strategy}} \leftarrow \langle A \rangle$
for $\forall A$ that $|A| = 2 \ldots N$ do
    $\text{Best}[A]_{\text{cost}} \leftarrow \min_{C_i \in A}\{1 + \frac{\prod_{m=1}^{M} \sum_{C_n \in A \setminus C_i} p_{m,i}}{\prod_{m=1}^{M} \sum_{C_n \in A} p_{m,i}} \cdot \text{Best}[A \setminus C_i]_{\text{cost}}\}$
    $\text{Best}[A]_{\text{strategy}} \leftarrow \langle \arg\min\{C_i | \text{Best}[A \setminus C_i]_{\text{cost}}\}, \text{Best}[A \setminus C_i]_{\text{strategy}} \rangle$
return $\{\text{Best}[A] |$ that $|A| = N\}$
```

Theorem 1. *The time complexity of Algorithms 3 and 4 is $\Theta(MN \cdot 2^N)$. The space complexity is $\Theta(N \cdot 2^N)$.*

D=N, Disjoint users: For disjoint users we can reduce the running time of optimal algorithm to $O(N^M)$ based on the following lemma. The algorithm involved is outlined in Appendix B. We implement the algorithm and use it in our simulation. It runs very fast for small M.

Algorithm 4 $D = N$, arbitrary users: compute the optimal cost and strategy for conference call using dynamic programming; opt = CCDP(A)

```
for $\forall A$, that $|A| = 1$ do
    $\text{Best}[A]_{\text{cost}} \leftarrow \prod_{m=1}^{M} P_{m,A}$
    $\text{Best}[A]_{\text{strategy}} \leftarrow \langle A \rangle$
for $\forall A$ that $|A| = 2 \dots N$ do
    $\text{Best}[A]_{\text{cost}} \leftarrow \min_{C_i \in A}\{\text{Best}[A \setminus C_i]_{\text{cost}} + \left(\prod_{m=1}^{M} P_A - \prod_{m=1}^{M} P_{A \setminus C_i}\right) \cdot |A|\}$
    $\text{Best}[A]_{\text{strategy}} \leftarrow \langle \arg\min\{\text{Best}[A \setminus C_i]_{\text{cost}}, \text{Best}[A \setminus C_i]_{\text{strategy}}, C_i\}\rangle$
return $\{\text{Best}[A] | \text{ that } |A| = N\}$
```

Lemma 6. *For disjoint users, in an optimal strategy, for every user, the cells where the user is located must be paged by order of non-increasing probability.*

5 Experiments

We design a family of 12 heuristics that compute good paging strategies in practice. All our heuristics are of the following form: First, we compute an order of the cells (according to some greedy method) and then, we apply Algorithms 1 or 2 to find the best strategy that follows this order of the cells. We have four criteria to order the cells. Define $X_n = \prod_{m=1}^{M} p_{m,n}$ (the probability all users are in cell C_n). Define $Y_n = \prod_{m=1}^{M}(1 - p_{m,n})$ (the probability no user is in cell C_n). Define $S_n = \sum_{m=1}^{M} p_{m,n}$ (the sum of user probabilities being in cell C_n). Define $Z_n = \max_{m=1}^{M} p_{m,n}$ (the maximum user probability in cell C_n). Heuristics X, Y, S and Z page the cells in the orders $X_1 \geq \dots \geq X_N$, $Y_1 \leq \dots \leq Y_N$, $S_1 \geq \dots \geq S_N$[3] and $Z_1 \geq \dots \geq Z_N$, respectively.

We use the above four basic heuristics to compute paging strategies for both problems. For each basic greedy heuristic $\mathcal{G} \in \{X, Y, Z, S\}$, we design two adaptive versions, namely $BF\mathcal{G}$ and $WL\mathcal{G}$. In the *Best First* (BF) version, each time we select a cell to form an order, we select the next available cell that has the best value (maximum for X, S, Z and minimum for Y), and then normalize the probabilities among unselected cells. In the *Worst Last* (WL) version, we select the available cell that has the worst value (minimum for X, S, Z and maximum for Y) as the last cell in the order, and then normalize probabilities among unselected cells.

Select the Best Heuristics: We test our heuristics for both the yellow page and the conference call problem. For each problem, we check regular users and

[3] [BM2] showed Y is an $\frac{e}{e-1}$-approximation for any $D \leq N$.

disjoint users. We also conduct our simulation on small instances and large instances with respect to number of cells. For small instances, we try all possible inputs (exhaustive search) up to some granularity (values of probabilities are integer multiples of some small value), then we compute strategies for Zipf and Uniform distributed user location data. We check the cases of $D = 2$ and $D = N$ (for efficient optimal algorithms). We compare the paging costs between different greedy heuristics and between greedy heuristics and OPT. For large instances, we conduct simulation on Zipf and Uniform distributed data, for several values of D, and then we compare the paging costs between the heuristics. We measure the average and worst case ratio of costs between each pair of heuristics and between each heuristic and OPT for the instances in each group of test with a combination of parameters and for real data. We evaluate the performances of heuristics by comparing among heuristics and comparing heuristics and optimal algorithms. We use a kind of"voting system" to generate a ranking of heuristics. We describe details of the setup and results of our experiments in Appendix C.

Our results show that the ranking of heuristics is the same for large and small instances, when doing exhaustive search, for Zipf and uniform data, for average case and worst case measurement, and when comparing heuristics among themselves or when comparing heuristics with OPT. Table 1 shows the ranking of heuristics and of the three versions of the best heuristic (Y, BFY, WLY) for the two problems.

Table 1. Performance ranking of heuristics. $\mathcal{G} > \mathcal{H}$: heuristic $\mathcal{G}$ is consistently better than $\mathcal{H}$; $\mathcal{G} \geq \mathcal{H}$: $\mathcal{G}$ is no worse than $\mathcal{H}$ (sometimes better, sometimes comparable); $\mathcal{G} \sim \mathcal{H}$: $\mathcal{G}$ is comparable to $\mathcal{H}$.

Yellow Page	Conference Call
$Y \geq S > Z \geq X$	$Y \geq S > Z \geq X$
$BFY > Y \geq WLY$	$Y > BFY \sim WLY$

Next we evaluate the performance of our best greedy heuristics. Our simulation found some "bad" instances. Based on these instances, we are able to craft some examples that show lower bounds on the competitiveness of our heuristics for both problems. We elaborate in two cases in Appendix D.

Performances of Best Greedy Heuristics: We evaluate the performances of the best heuristics (BFY for yellow page and Y for conference call) on small instances, large instances. We measure their average case and worst case cost ratios over OPT for small instances and real data. We measure the average and worst case cost ratios of other heuristics over BFY or Y for large instances.

Tables 2(a), 2(c) and 2(d) show the results. We observe that our selected heuristics perform well in the worst case and average performance for all types of input data. We also present the results in bar chart in Fig. 4 in Appendix C.

Simulation on Real Data: We obtain from a cellular phone company 996 users' 171929 appearances in 5625 cells in 31 consecutive days. For each user, we extract (from the above real data) the number of appearances in every cell. We randomly pick two ($M = 2$), three ($M = 3$), and four ($M = 4$) users and compute the probabilities $p_{m,n}$s. For each $M = 2, 3, 4$, we pick 10,000 instances as above. We conduct the same simulations as in the non-real data. We observe that each real user is close to Zipf distributed and users are almost disjoint. The results coincide with the results from non-real data as shown in Tables. 1 and 2(b).

Table 2. Cost Ratios of BFY for Yellow Page (YP) and Y for Conference Call (CC)

(a) Cost Ratios Over OPT: Small Instances

Problem	Average	Worst
YP (BFY)	1.00638	1.19415
CC (Y)	1.00173	1.03609

(b) Cost Ratios Over OPT: Real Data

Problem	Average	Worst
YP (BFY)	1.03352	1.54384
CC (Y)	1.00643	1.05930

(c) Cost Ratios Over BFY: Large Instances, YP

Heuristic	Average	Worst
BFX	1.19102	2.23126
BFZ	1.28360	2.51777
BFS	1.00003	1.00961

(d) Cost Ratios Over Y: Large Instances, CC

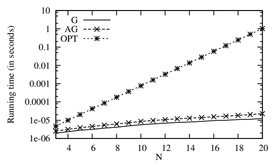
Heuristic	Average	Worst
X	1.15884	1.62346
Z	1.00626	1.03000
S	1.00000	1.00316

Running Time: We compare the running time of our greedy heuristics and efficient optimal algorithms. For accurate time measurement, for each N, we run our algorithms for many iterations and compute the average running times.

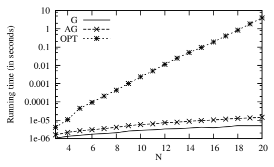
For $D = 2$, we test the running time of the static version of the greedy heuristics G (i.e., Y) of complexity $\Theta(MDN \log N)$, the adaptive version of the greedy heuristics AG (i.e., BFY and WLY) of complexity $\Theta(MDN^2 \log N)$, and the D^N optimal algorithm of complexity $\Theta(MND^N)$. The result is in Fig. 1(a) For $D = N$, we test the running time of the static greedy heuristic G, the adaptive greedy heuristic AG, and the fast optimal algorithm of complexity $\Theta(MN2^N)$. The result is in Fig. 1(b) In the two experiments, we observe a complexity hierarchy of the algorithms and a tradeoff between complexity and optimality. For $D = N$, we also compare the running time of the straight forward optimal $N!$ algorithm and fast 2^N algorithm. The result is in Fig. 1(c). As expected, the $N!$ algorithm is super exponential and the 2^N algorithm is comparably much faster, which allows us do experiments on larger problem instances.

6 Open Problems

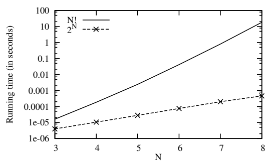
A natural generalization is to efficiently find k out of M users. In the yellow page problem $k = 1$ and in the conference call problem $k = M$. The motivation for this general case could be the task of finding a team of k doctors out of a pool of M doctors. In this paper, we assume that the costs of paging cells are all the same. However, they may vary from cell to cell due to different level of congestions. Another generalization, is to compute good paging strategies when cells have arbitrary paging costs. We have results for paging a single users while paging multiple users with paging costs is work in progress. Finally, our paging strategies are static since they are predetermined before the paging process starts. In a dynamic setting, we may select the cells to be paged in the next round according to the users found in previous rounds. The dynamic yellow page problem remains the same while a solution to the dynamic conference call problem can outperform the optimal solution for the static call conference problem. How to compute a good dynamic paging strategy remains open.



(a) $D = 2$, heuristics and optimal algorithm



(b) $D = N$, heuristics and optimal algorithm



(c) $D = N$, two optimal algorithms

Fig. 1. Running time

References

[AMHUW] Akyildiz, I. F., Mcnair, J., Ho, J., Uzunalioglu, H., Wang, W.: Mobility management in next-generation wireless systems. Proc. IEEE. (1999) 1347–1384

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

[BM2] Bar-Noy, A., Malewicz, G.: Establishing wireless conference calls under delay constraints. J. Algorithms **51**(2) (2004) 145–169

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

[B] Buchanan, M.: Ecological modelling: the mathematical mirror to animal nature. Nature **453** (2008) 714–716

[CFK] Cohen, E., Fiat, A., Kaplan, H.: Efficient sequences of trials. Proceedings of the ACM-SIAM symposium on Discrete algorithms (SODA) (2003) 737–746

[EL1] Epstein, L., Levin, A.: The conference call search problem in wireless networks. Theor. Comput. Sci. **359**(1-3) (2006) 418–429

[EL2] Epstein, L., Levin, A.: A PTAS for delay minimization in establishing wireless conference calls. Discrete Optimization **5**(1) (2008) 88–96

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

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

[KKM] Kaplan, H., Kushilevitz, E., Mansour, Y.: Learning with attribute costs. Proceedings of ACM Symposium on Theory of Computing (STOC) (2005) 356–365

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

[RY1] Rose, C., Yates, R.D.: Minimizing the average cost of paging under delay constraints. Wireless Netw. **1**(2) (1995) 211–219

[RY2] Rose, C., Yates, R.D.: Ensemble polling strategies for increased paging capacity in mobile communication networks. Wireless Netw. **3**(2) (1997) 159–167

[W] Weitzman, M. L.: Optimal search for the best alternative. Econometrica **47**(3) (1979) 641–654

A Hierarchy of User Types

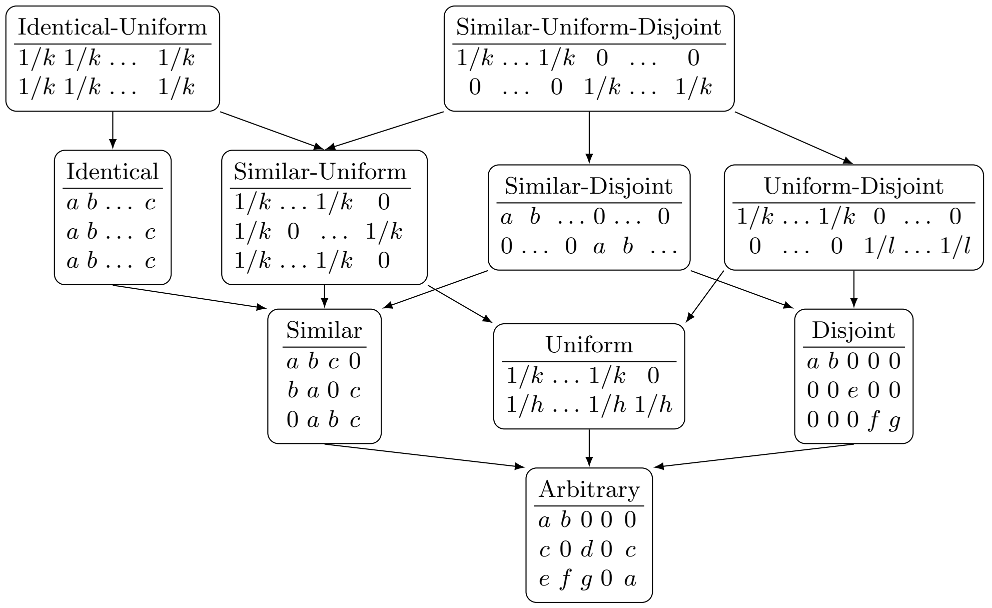


Fig. 2. Hierarchy among Different User Types; $a, b, c, d, e, g, h \leq 1$ are positive real numbers, $h, k, l \leq N$ are positive integers.

B Proof of Lemmas in Sec. 4

Proof (of Lemma 2). In the yellow page problem, denote the per-cell probability suffix with $r_{m,n} = \sum_{i=n+1}^{N} p_{m,i}$. Assume for the sake of contradiction, that in the optimal paging strategy $\mathcal{O}$, there are two cells, C_i, C_j, for which $C_i \in O_d$ and $C_j \in O_{d+1}$, but $p_{m,i} < p_{m,j}$. Using equation (5), the paging cost of $\mathcal{O}$ is:

$$\begin{aligned}\mathrm{YP}(\mathcal{O}) = \text{cost before round } d + S_d \cdot \left(\prod_{m=1}^{M} r_{m,S_{d-1}} - \prod_{m=1}^{M} r_{m,S_d} \right) + \\ S_{d+1} \cdot \left(\prod_{m=1}^{M} r_{m,S_d} - \prod_{m=1}^{M} r_{m,S_{d+1}} \right) + \text{cost after round } (d+1). \quad (7)\end{aligned}$$

Consider the strategy $\mathcal{O}'$ where we switch the round assignments of cells C_i and C_j. Then,

$$\begin{aligned}\mathrm{YP}(\mathcal{O}') = \text{cost before round } d + S_d \cdot \left(\prod_{m=1}^{M} r_{m,S_{d-1}} - \prod_{m=1}^{M} r'_{m,S_d}\right) + \\ S_{d+1} \cdot \left(\prod_{m=1}^{M} r'_{m,S_d} - \prod_{m=1}^{M} r_{m,S_{d+1}}\right) + \text{cost after round } (d+1). \quad (8)\end{aligned}$$

Besides identical terms, for every m, r'_{m,S_d} contains $p_{m,i}$ whereas r_{m,S_d} contains $p_{m,j}$, i.e., $r'_{m,S_d} - r_{m,S_d} = p_{m,i} - p_{m,j}$, which implies $r'_{m,S_d} < r_{m,S_d}$, because $p_{m,i} < p_{m,j}$. Subtracting (8) from (7), we get

$$\mathrm{YP}(\mathcal{O}) - \mathrm{YP}(\mathcal{O}') = (S_d - S_{d+1}) \cdot \left(\prod_{m=1}^{M} r'_{m,S_d} - \prod_{m=1}^{M} r_{m,S_d}\right) > 0$$

because both factors are negative. This is a contradiction to the optimality of $\mathcal{O}$.

Similarly, we can prove the lemma for the conference call problem (proof omitted).

Proof (of Lemma 3). From (2), we have (the summation is over all possible location vectors $\mathbf{d} = (d_1, \ldots, d_M) \in \{1, \ldots, N\}^M$):

$$\begin{aligned}\mathrm{CC}(\mathcal{A}, q_{m,n}) &= \sum_{\mathbf{d}} \left(\max_{m=1}^{M} d_m \cdot \prod_{m=1}^{M} q_{m,d_m}\right) \\ &= \sum_{\mathbf{d}} \left((N+1 - \min_{m=1}^{M}(N+1-d_m)) \cdot \prod_{m=1}^{M} p_{m,N+1-d_m}\right) \\ &= \sum_{\mathbf{d}} \left((N+1) \cdot \prod_{m=1}^{M} p_{m,N+1-d_m}\right) - \sum_{\mathbf{d}} \left(\min_{m=1}^{M}(N+1-d_m) \cdot \prod_{m=1}^{M} p_{m,N+1-d_m}\right) \\ &= N+1 - \sum_{\mathbf{d}'} \left(\min_{m=1}^{M} d'_m \cdot \prod_{m=1}^{M} p_{m,d'_m}\right) \\ &= N+1 - \mathrm{YP}(\mathcal{A}, p_{m,n})\end{aligned}$$

where $d'_m = N+1-d_m$ and $\mathbf{d}' = (d'_1, \ldots, d'_m)$, because the mapping $d_m \mapsto d'_m$ is a bijection in $\{1, \ldots, N\}$, after using equation (1).

Proof (of Lemma. 4). Assume for contradiction that, $\langle C_{j_k}, \ldots C_{j_n}\rangle$ $(\neq \langle C_{i_k}, \ldots C_{i_n}\rangle)$ is the optimal paging strategy for the suffix, then

$$\mathrm{YP}(\langle C_{j_k}, \ldots C_{j_n}\rangle) < \mathrm{YP}\,\langle C_{i_k}, \ldots C_{i_n}\rangle \ .$$

Using equation (3),

$$\mathrm{YP}(\left\langle C_{i_1}, \ldots, C_{i_{k-1}}, C_{j_k}, \ldots, C_{j_N}\right\rangle) < YP(\left\langle C_{i_1}, \ldots, C_{i_{k-1}}, C_{i_k}, \ldots, C_{i_N}\right\rangle) \ ,$$

which is a contradiction to the optimality of $\left\langle C_{i_1}, \ldots, C_{i_{k-1}}, C_{i_k}, \ldots, C_{i_N}\right\rangle$.

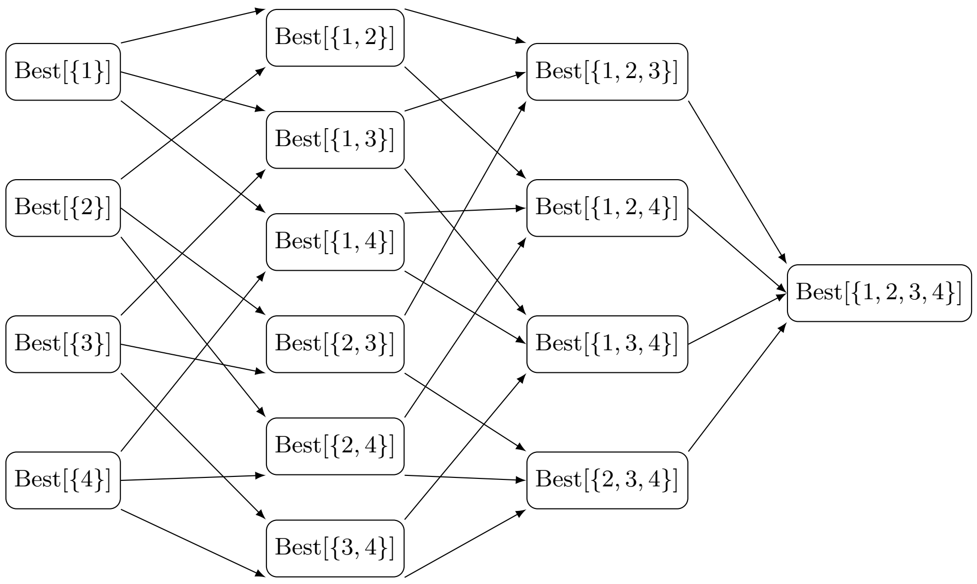


Fig. 3. Illustration of Algorithm 3 for $D = N = 4$.

Proof (of Theorem 1). : Fig. 3 demonstrates an example of running process of Algorithm 3. We note that there are $\Theta(2^N)$ nodes, for general N, in the above figure. To compute each node, on average, one need to randomly access $\lceil N/2 \rceil + 1$ nodes from the previous column. Each random access can be executed in $\Theta(M)$ time after some linear time preprocessing. The overall time complexity of the algorithm is $\Theta(MN \cdot 2^N)$.

To randomly access from one column to the previous column, we need to maintain at least $\binom{N}{\lfloor N/2 \rfloor} + \binom{N}{\lfloor N/2 \rfloor + 1}$ nodes in the memory, and the average size of a node is $\lceil N/2 \rceil + 1$. Thus, the overall space complexity of the algorithm is $\Theta(N \cdot 2^N)$.

Proof (of Lemma 6). (Sketch) By contradiction. Without loss of generality, suppose in the optimal paging strategy $\mathcal{O}$, for user 1, cell C_i is paged in d_ith round and C_j is paged in d_jth round, with $d_i < d_j$, but $p_{1,i} < p_{1,j}$. By swapping C_i and C_j, we obtain another paging strategy, $\mathcal{O}'$. We have

$$\text{YP}(\mathcal{O}) = \text{terms group } 1 + p_{1,i} \cdot d_i \cdot (\text{terms group } 2) + p_{1,j} \cdot d_j \cdot (\text{terms group } 3)$$
$$\text{YP}(\mathcal{O}') = \text{terms group } 1 + p_{1,j} \cdot d_i \cdot (\text{terms group } 2) + p_{1,i} \cdot d_j \cdot (\text{terms group } 3)$$

We observe that $p_{1,i}d_i + p_{1,j}d_j > p_{1,j}d_i + p_{1,i}d_j$. Thus $\text{YP}(\mathcal{O}) > \text{YP}(\mathcal{O}')$, which is a contradiction to $\mathcal{O}$'s optimality. (A similar proof can be applied to the conference call problem.)

Optimal algorithm [for disjoint users]: We show the algorithm for $M = 2$ users in the yellow page problem. It can be extended to any number of users. The algorithm also works with slight modifications for the conference call problem.

Let two users roam in $(k+l)$ cells. User 1 only appears in the first k cells with probabilities $p_1, \dots, p_k$ and user 2 only appears in the other l cells with probabilities $q_1, \dots, q_l$. Without loss of generality, we assume $p_1 \geq \cdots \geq p_k$ and $q_1 \geq \cdots \geq q_l$. Define matrix $\text{Best}_{k\times l}$ such that $\text{Best}[i, j]$ is the best paging strategy within cells with probabilities $p_i, \dots, p_k, q_j, \dots, q_l$ given the condition that no user is located in the other $(i+j-2)$ cells. By defining $p_{k+1} = q_{l+1} = 0$, we observe that $\text{Best}[k+1, l+1]_{\text{cost}} = 0$, $\text{Best}[k+1, j]_{\text{cost}} = 1 + \frac{\sum_{t=j+1}^{l} q_t}{\sum_{t=j}^{l} q_t}\text{Best}[k+1, j+1]_{\text{cost}}$ for $1 \leq j \leq l$, and $\text{Best}[i, l+1]_{\text{cost}} = 1 + \frac{\sum_{t=i+1}^{k} p_t}{\sum_{t=i}^{k} p_t}\text{Best}[i+1, l+1]_{\text{cost}}$ for $1 \leq i \leq k$. We recursively compute other entries in Best, according to Lemma 6:

$$\text{Best}[i, j]_{\text{cost}} = \min \begin{cases} 1 + \frac{p_{i+1}+\cdots+p_k}{p_i+\cdots+p_k}\text{Best}[i+1, j]_{\text{cost}} \\ 1 + \frac{q_{j+1}+\cdots+q_l}{q_j+\cdots+q_l}\text{Best}[i, j+1]_{\text{cost}} \end{cases} \tag{9}$$

$\text{Best}[1, 1]$ is the optimal paging strategy for all cells.

For $M > 2$ users, a similar algorithm can be designed by using an M-dimensional Best array. The size of data structure Best is $O(N^M)$. Computing an entry of Best, takes $\text{Poly}(M, N)$ time. The overall complexity of the algorithm is $\Theta(N^M \text{Poly}(M, N))$, which is polynomial with respect to the number of cells.

C Experiment Details

Algorithms structure: Input: Probability matrix $P_{M\times N}$, delay constraint D. Output: Paging cost and paging strategy array $part[N]$, where $part[n]$ is the part index of cell C_n.

Algorithms implemented: 1) OPT: A simple optimal algorithm for arbitrary users on any D: It tests all assignments of cells $\langle C_1, \dots, C_N\rangle$ to parts $\langle A_1, \dots, A_D\rangle$ and selects the best. The complexity is $\Theta(MN \cdot D^N)$. For $D = 2$, its complexity is $\Theta(MN \cdot 2^N)$. 2) FAST OPT: Algorithms 3 and 4. Optimal algorithms for arbitrary users when $D = N$. Their complexity is $\Theta(MN \cdot 2^N)$. 3) DISJOINT OPT: Optimal algorithm for disjoint users when $D = N$. 4) All 12 greedy heuristics for $1 \leq D \leq N$. Their general complexity is $O(MD \cdot N^4)$. When $D = N$ their complexity becomes $\Theta(MN^2)$. The static version of heuristics has complexity $\Theta(MDN^2)$ when $D < N$.

User Types: 1) Regular users: directly generate $p_{m,n}$s according to data type. 2) Disjoint users: select non-zero entries $p_{m,n}$s for each user such that all users have (almost) same number of non-zero $p_{m,n}$s, then generate $p_{m,n}$ values within non-zero entries of each user.

Instance scale: 1) Small instances: For Zipf and random user location data, we select $M = 2, 3, 4, 5$ and $N = 10, 12, 14, 16$, $D = 2$ and $D = N$. 2) Large instances: For Zipf and random user location data, we select $M = 2, 3, 4, 5$ and $N = 10, 20, \ldots, 100$, $D = 2, 4, 8, \ldots, 2^{\lfloor \log_2 N \rfloor}$.

Data type: 1) EXT: Exhaustive search on small instances: for each set of parameters $\{M, N\}$ and granularity K, we enumerate all possible $p_{m,n}$s in increment of $1/k$, where $2 \le k \le K$. The number of instances is $\Theta((K^2)^{MN})$. We choose $N = 3, 4, 5, 6$, $M = 2$ and $K = 13$ so that the experiments can be completed in a reasonable time frame (hours to days). 2) RANDOM: Random data: for each set of parameters $\{M, N\}$, we generate 1000 instances of random data, such that $p_{m,n} \sim \mathrm{U}(0, 1)$. We normalize $p_{m,n}$s such that $\sum_{n=1}^{N} p_{m,n} = 1$ for $1 \le m \le M$. 3) ZIPF: Zipf data: Empirical study [B] on user location distribution shows that $p_{m,n}$ follows *Zipf* distribution, i.e., $p_{m,i} = i^{-\alpha} / \sum_{n=1}^{N} n^{-\alpha}$. Zipf distribution is a power-law distribution and $\alpha \ge 0$ is its parameter. When $\alpha = 0$, Zipf distribution is uniform. As α grows, it becomes more uneven. Using the method of least squares on real data, we acquire the following estimate of the parameter $\alpha = 0.4429$. We generate Zipf data $p_{m,n}$s for each user m, and randomly shuffle for each user (otherwise we would just create monotonic users). For each set of parameters $\{N, M, \alpha\}$, we generate 1000 instances. We select $\alpha = 0.25, 0.4429, 0.5, 0.75, 1.0$.

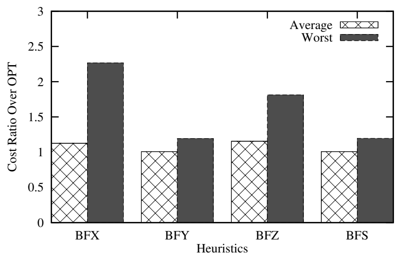
Variation: We set up a group of instances for each combination of the following seven criteria:

– Problem: yellow page versus conference call
– User type: regular users versus disjoint users
– Instance scale: large versus small
– Data: EXT versus RANDOM versus ZIPF
– Delay constraint: $D = 2$ versus $D = N$ for small instances only
– Measurement: worst case (lower bound) versus average ratios
– Evaluation criteria: between heuristics versus between heuristic and OPT (to be explained next).

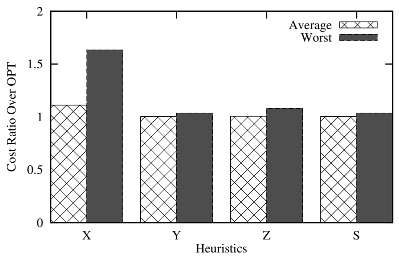
Example of a group of tests: For the *yellow page* problem, we create 1,000 *small* random instances ($N = 10, 12, 14, 16$, $M = 2, 3, 4, 5$) of *regular users* with probabilities that follow *Zipf* distribution. We measure the *average* cost ratios between each *pairs of heuristics*.

Measurements: 1) For each instance in each group of data, we measure the cost ratio of one heuristic over another heuristic, as well as each heuristic over OPT. We have 12×12 such ratios for each instance (12×11 between pairs of heuristics and 12 between all heuristics and OPT). 2) For each group of instances, we measure the worst case and average case of ratios.

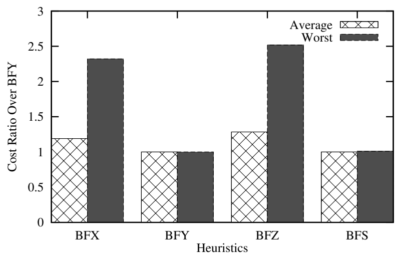
Evaluation Criteria: 1) Between heuristics: Let $\mathcal{G}$ and $\mathcal{H}$ be two greedy heuristics. Let $\rho_{\mathcal{G}/\mathcal{H}}$ be the cost ratio of $\mathcal{G}$ over $\mathcal{H}$ on a group of instances; let $\rho_{\mathcal{H}/\mathcal{G}}$ be the ratio of $\mathcal{H}$ over $\mathcal{G}$ on the same group of instances. Define the competitive factors $\varepsilon_{\mathcal{G}/\mathcal{H}} = \rho_{\mathcal{G}/\mathcal{H}} - 1$ and $\varepsilon_{\mathcal{H}/\mathcal{G}} = \rho_{\mathcal{H}/\mathcal{G}} - 1$. Let $\mu \in [2, 4]$ be a constant that



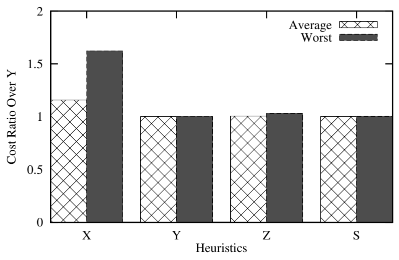
(a) Cost Ratios Over OPT: Small Instances, YP



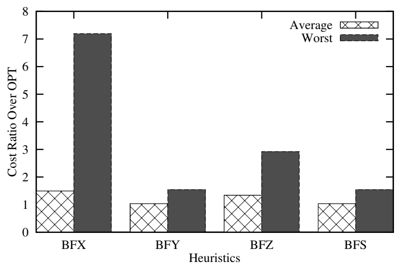
(b) Cost Ratios Over OPT: Small Instances, CC



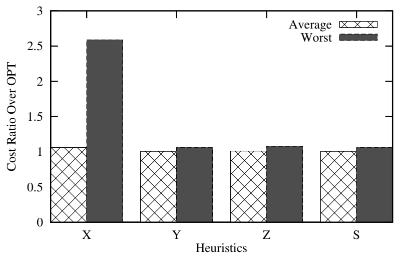
(c) Cost Ratios Over BFY: Large Instances, YP



(d) Cost Ratios Over Y: Large Instances, CC



(e) Cost Ratios Over OPT: Real Data, YP



(f) Cost Ratios Over OPT: Real Data, CC

Fig. 4. Cost Ratios of Greedy Heuristics

we will choose and that will be used to distinguish performances of different algorithms. If $\varepsilon_{\mathcal{G}/\mathcal{H}}$ and $\varepsilon_{\mathcal{H}/\mathcal{G}}$ are of same sign and $1/\mu \leq \varepsilon_{\mathcal{G}/\mathcal{H}}/\varepsilon_{\mathcal{H}/\mathcal{G}} \leq \mu$, we say heuristics $\mathcal{G}$ and $\mathcal{H}$ have comparable performance. Otherwise, if $\varepsilon_{\mathcal{G}/\mathcal{H}} < \varepsilon_{\mathcal{H}/\mathcal{G}}$, we say $\mathcal{G}$ outperforms $\mathcal{H}$. 2) Between a heuristic and optimal: Let $\mathcal{G}$ be a heuristic. Let $\rho_{\mathcal{G}} \geq 1$ be the cost ratio of $\mathcal{G}$ over OPT. Define competitive factor $\varepsilon_{\mathcal{G}} = \rho_{\mathcal{G}} - 1$. For any heuristics $\mathcal{G}$ and $\mathcal{H}$, we say their performances are comparable if $1/\mu \leq \varepsilon_{\mathcal{G}}/\varepsilon_{\mathcal{H}} \leq \mu$; otherwise, we say $\mathcal{G}$ outperforms $\mathcal{H}$ if $\varepsilon_{\mathcal{G}}/\varepsilon_{\mathcal{H}} < 1/\mu$. 3) $\mu = 2, 3, 4$.

We observe that the two evaluation criteria above coincide very well in our experiments. The disagreement rate varies from 2.78% to 26.7%, with a mean of 9.37%. Varying μ from 2 to 4 does not remarkably affect results.

Statistics Collection: For each group of instances, we create a 12×12 table that records the superiority of one heuristic over another and OPT for each variation of tests. Each table entry can be of three values, -1, 0 and 1 denoting the row heuristic is worse, comparable or better than the column heuristic, respectively. The value is determined by both evaluation criteria.

We measure the superiority among four groups of heuristics: X, Y, Z and S in their corresponding versions (e.g., WLX, WLX, WLZ and WLS), and among the three versions within each group (e.g., Y, BFY and WLY).

We conduct a voting on the corresponding table entries of one heuristic over another. If heuristic $\mathcal{G}$ has a 2/3 majority of superiority over $\mathcal{H}$, we denote this by $\mathcal{G} \geq \mathcal{H}$. If heuristic $\mathcal{G}$ has a 100% majority of superiority over $\mathcal{H}$, we denote this by $\mathcal{G} > \mathcal{H}$. Otherwise, we say $\mathcal{G} \sim \mathcal{H}$

Results: We first observe that a few factors do not affect the performance of heuristics; these are: delay constraint: $D = 2$ or $D = N$; Data type: EXT, RANDOM OR ZIPF; Measurement: average or worst case; Evaluation criteria: between heuristics or between a heuristic and OPT.

A summary of the results is shown in Table 1 in the main paper.

Besides Table 1, we observe a few facts: 1) All heuristics perform quite well, on average with ratios of less than 1.15 competitive to OPT. In the worst case ratios, only heuristics X, BFX and WLX perform badly with a lower bound ratio of $N-1$ for the yellow page problem and $(N+1)/2$ for the conference call problem. All other worst case ratios are less than 2. For the conference call problem we record better cost ratios than for the yellow page problem because the latter has smaller value in paging cost for the same instance. 2) The average case and worst case performance of heuristics coincides with each other, which suggests that their performance is consistent and stable.

D Some Examples: Lower Bounds of Greedy Heuristics

Case 1: $D = N$ and $M = 1$. The probability matrix:

$$\begin{pmatrix} 1-(N-2)\varepsilon & 0 & \varepsilon \ldots \varepsilon \\ 0 & 1-(N-2)\varepsilon & \varepsilon \ldots \varepsilon \end{pmatrix}$$

Lemma 7. *In Case 1, $YP(X)/YP(OPT) = N-1$, $CC(X)/CC(OPT) = N/2$, when $\varepsilon \to 0^+$.*

Proof. (sketch) Let $\varepsilon \to 0^+$. The paging strategy generated by X is $\langle C_3, \ldots, C_N, C_1, C_2 \rangle$ while the optimal paging strategy is $\langle C_1, C_2, C_3, \ldots, C_N \rangle$.

$$\mathrm{YP}(X) = N-1 \qquad \mathrm{CC}(X) = N$$
$$\mathrm{YP}(\mathrm{OPT}) = 1 \qquad \mathrm{CC}(\mathrm{OPT}) = 2$$

By applying the above equations we have Lemma 7.

Case 2: $D = N$ and $M = m+1$. The probability matrix:

$$\begin{array}{c} \\ m\left\{\begin{array}{c} \\ \\ \\ \end{array}\right. \\ \end{array}
\begin{array}{ccc}
\underbrace{\begin{array}{ccc} 0 & \ldots & 0 \\ 1/k & \ldots & 1/k \\ & \ldots & \\ 1/k & \ldots & 1/k \end{array}}_{k \text{ terms}} &
\underbrace{\begin{array}{ccc} \frac{k^m-(k-1)^m}{k^m} & \ldots & \frac{k^m-(k-1)^m}{k^m} \\ 0 & \ldots & 0 \\ & \ldots & \\ 0 & \ldots & 0 \end{array}}_{x \text{ terms}} &
\begin{array}{c} y \\ 0 \\ \vdots \\ 0 \end{array}
\end{array}$$

Lemma 8. *In Case 2, $YP(BFY)/YP(OPT) \geq 2 - \varepsilon$.*

Proof. (Sketch) Since

$$1 - \prod_{i=1}^{m}(1 - 1/k) = \frac{k^m - (k-1)^m}{k^m} ,$$

an adversary can force BFY select cell C_1 in the first round by adding small turbulence. Consequently, BFY will sequentially select cells $C_2, \ldots, C_k$ in later rounds. OPT will page the last $(x+1)$ cells in sequentially. We compute the following steps by induction (details omitted).

$$\mathrm{YP}(BFY) = \frac{\sum_{i=1}^{k} i^m}{k^m}$$

$$\mathrm{YP}(BFY) \geq \frac{k}{m+1}$$

$$\mathrm{YP}(\mathrm{OPT}) \leq \frac{k}{2m} + O(1)$$

Fixing m and let $k \to \infty$, we have

$$\frac{\mathrm{YP}(BFY)}{\mathrm{YP}(\mathrm{OPT})} \geq \frac{\frac{k}{m+1}}{\frac{k}{2m} + O(1)} \approx \frac{k}{k + O(m)} \cdot \frac{2m}{m+1} .$$

Make $m \to \infty$ and force $k \gg m$,

$$\frac{\mathrm{YP}(BFY)}{\mathrm{YP}(\mathrm{OPT})} \geq \frac{2m}{m+1} = 2 - \varepsilon .$$

E 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.