Pergamon

0966-8349(95)00009-7

Location Science, Vol. 3, No. 2, pp. 107–124, 1995
Elsevier Science Ltd. Printed in Great Britain
0966-8349/95 $9.50 + 0.00

AN ANALYSIS OF SERVICE SCHEDULES FOR THE MOBILE *k*-SERVER PROBLEM

WEIZHEN MAO*
Department of Computer Science, The College of William and Mary, Williamsburg, VA 23187-8795, U.S.A.

and

REX K. KINCAID†
Department of Mathematics, The College of William and Mary, Williamsburg, VA 23187-8795, U.S.A.

(*Received March* 1994; *in revised form June* 1995)

Abstract—In the mobile *k*-server problem, servers with fixed home locations travel to locations of requests to provide service and then return to their home locations. Given the home locations of the *k*-servers, the service times, release times and locations of the *n* requests, we seek to determine a service schedule so that the total waiting time of all requests is minimized. In this paper, we exploit the strong connection between the mobile *k*-server problem and job scheduling. We present optimal algorithms for a few special cases of the problem and provide worst-case performance analysis of some heuristics for the general case. For the special cases it is possible to rank order competing sets of home locations with respect to total (or average) waiting time.

Keywords: Mobile server, scheduling, complexity, algorithms.

1. INTRODUCTION

Mobile servers arise in both the public and private sectors. Examples are widespread and include automated guided vehicles, elevators, heads on a disk drive, firetrucks, repair services for utilities, and taxicabs. The most often cited example is a public service system such as an emergency medical system, in which ambulances are sent to accident sites, perform on-site care, and return with the patients to hospitals. In each of these examples some metric must be imposed to quantify the level of service provided by the mobile servers. Although effectiveness and equity can be important measures of service, we focus on efficiency as measured by the total waiting time of the requests. Savas (1978) provides a discussion of appropriate performance measures for public service systems.

In the standard multi-server model of the traditional queueing theory, mobile requests (jobs or customers) travel to the stationary servers (machines or processors) seeking service. An interesting variant of the traditional multi-server model is the *mobile k-server* model. Here the travel component is reversed with respect to the requests and the servers. The servers are mobile and travel on a network to service stationary requests at the network nodes. A wealth of locational research has focused on the problem of determining the home locations of the servers and, surprisingly, almost no analytical results have been obtained

* Supported in part by NSF grant CCR-9210372.
† Supported in part by a faculty research award from the College of William and Mary.

for the case where the home locations of the servers are known *a priori*. One of the potential uses of our analysis is in the evaluation of sets of competing home locations. In some cases the analytical results we present determine the optimal queue discipline given a set of home server locations and, consequently, the minimum total waiting time is known. In these cases competing sets of home locations can be compared with respect to the performance measure of total (or average) waiting time. We begin by reviewing some of the work on the home location problem, with particular emphasis on the efforts to include stochastic elements.

Finding good home locations for the k servers is difficult, even in a deterministic environment. For example, the goal in the NP-complete p-median problem is to determine the set of p server home locations that minimizes the sum of the distances from each request to its closest server home location. In a stochastic environment, if the congestion is low then it is intuitive that the k server home locations should be the p-median solution. For more details on the p-median problem and location analysis, the interested reader is referred to Hakimi (1964), the seminal reference, or Brandeau and Chiu (1989), who provide an excellent survey of fifty representative problems in location analysis. The desire to include stochastic components in the mobile k-server home location problem has led to two broad analytic approaches. Although many interdependencies exist, one approach can be categorized as queueing theory based and the other approach as integer programming based.

The queueing theory based approach can be traced to the seminal paper of Larson (1974), which introduces the hypercube model. In hypercube models the steady state behavior of the queue is exploited to develop efficient algorithms. Berman *et al.* (1990) provide a recent summary of the research efforts associated with this approach. There are two main assumptions placed on the queue of requests for service. The queue is *conservative* (if a request for service arrives and a server is available then a server responds immediately to the request), and the queue discipline is *first-come-first-serve* (*FCFS*). Much of the work in this area assumes that only one request is serviced per trip and that the server must return to its home location before the servicing of another request can begin.

The integer programming based approach has its roots in the seminal papers of Toregas *et al.* (1971) and Church and ReVelle (1974). Both employ integer programming models of covering problems as a mechanism for generating good solutions to the k-server home location problem. Extensions of these covering models have included constraints for backup coverage, expected coverage, and probabilistic coverage. These models typically assume that the probability of "busy-ness" of one server is independent of any other server. Schilling *et al.* (1993) provide a comprehensive summary of the literature related to covering models and the siting of facilities. With respect to probabilistic covering and backup models the reader is referred to Daskin *et al.* (1988) and Marianov and ReVelle (1992).

In addition to these analytical approaches there are several simulation-based mobile server models. Uyeno and Seeberg (1984) develop a detailed simulation model to find ambulance home locations in British Columbia while Lubicz and Mielczarek (1987) determine the number of ambulances needed to service a rural emergency medical system in Poland. In addition to the siting problem, Fujiwara *et al.* (1987) examine the effects of a demand increase on current ambulance deployment in Bangkok, Thailand. Park *et al.* (1992) use a simulation model to study the effect of the transient behavior of the queue and to examine the performance of both traditional (first available and districting) server disciplines as well as non-traditional (look ahead) server disciplines that include knowledge of when the next server will become available.

One final category of literature related to the mobile k-server problem is the on-line k-server problem studied by Manasse *et al.* (1990) and Chrobak *et al.* (1990). In the on-line models, requests have no service times and release times (the time a request for service is made) and are generated randomly in some underlying metric space; home locations for the servers are not prescribed and congestion is not considered as part of the performance measure of the system. The objective is to determine a strategy which minimizes the sum of the distances traveled by each of the k servers to service all requests. A typical algorithm moves all servers towards a request until one server covers it.

Our statement of the mobile k-server problem most closely follows Berman *et al.* (1990) with the following exceptions. First, we assume that the home locations of the servers are given *a priori* and that it is not necessary to have a conservative FCFS queue discipline. Second, the problem we study makes no assumptions about the underlying distribution associated with the requests for service. The servers $S_1, S_2, \ldots, S_k$ are mobile and travel on a network, usually a two-dimensional metric space, to serve the stationary requests $R_1, R_2, \ldots, R_n$. Each server S_i has a fixed home location $(x(S_i), y(S_i))$, and after traveling to a request at speed v_i and providing service to the request, a server always returns home to wait for another service request. Each request R_j requires service time p_j and is released at time r_j at location $(x(R_j), y(R_j))$. When a request arrives at its designated location in the network, the scheduler either assignes an idle server to it or lets the request wait. The waiting time or response time w_j of request R_j is the difference between the time when the request is given service, i.e. the starting time s_j, and the time when the request is available, i.e. the release time r_j. So we have

$$w_j = s_j - r_j.$$

Let d_{ij} be the Euclidean distance (or the shortest path distance if an underlying graph is specified) between the home location of S_i and the location of R_j, i.e.

$$d_{ij} = \sqrt{(x(S_i) - x(R_j))^2 + (y(S_i) - y(R_j))^2}.$$

Note that even if a server S_i is assigned to a request R_j immediately after R_j is released, R_j still has to wait d_{ij}/v_i time units before it is serviced. So $w_j \geqslant d_{ij}/v_i$ if R_j is served by S_i. We wish to design a service schedule such that the total waiting time of the requests, $\Sigma_j w_j$, is minimized. Note that an equivalent measure that is often used is the average waiting time $\frac{1}{n}\Sigma_j w_j$. It is clear that minimizing $\Sigma_j w_j$ also minimizes $\frac{1}{n}\Sigma_j w_j$ and vice versa.

In this paper we view the mobile k-server problem as a job scheduling problem and use this approach to design optimal algorithms for a few special cases and analyze the worst-case performance of some heuristics for the general case. We organize the paper as follows. In Section 2, we consider the case when all requests have equal release times and present efficient optimal algorithms for the mobile k-server problem when $k = 1$ and when $k \geqslant 2$. In Section 3, we consider the case when requests have arbitrary release times. First, we show that when release times are arbitrary the mobile k-server problem is NP-complete, even when there is only one server. We then analyze the worst-case performance of two heuristics for the queue disciplines—*first-come-first-serve* (*FCFS*) and *shortest-available-request-first* (*SARF*)—when there is only one server. For the multi-server version, we define *earliest-request-closest-server* (*ERCS*) and *shortest-request-closest-server* (*SRCS*), which are the generalizations of FCFS and SARF, respectively, and establish some worst-case performance bounds. In Section 4, we conclude with a summary of our contributions.

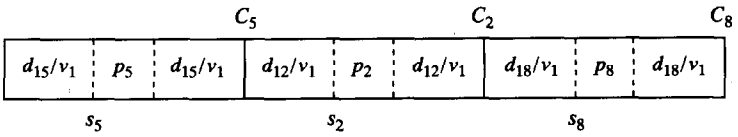

Fig. 1. A schedule.

2. EQUAL RELEASE TIMES

In this section, we consider the case when $r_j = r$ for $j = 1, 2, \ldots, n$ and some fixed constant r. Without loss of generality, let $r_j = 0$ for $j = 1, 2 \ldots, n$, i.e. all requests are available at time 0.

2.1. *Single mobile server*

When there is only one server, i.e. $k = 1$, the mobile 1-server problem is in fact equivalent to a job scheduling problem, where n jobs are to be scheduled on a single machine in such a way that the total completion time of all jobs, $\Sigma_j C_j$, is minimized. Using the $\alpha|\beta|\gamma$ notation introduced by Lawler *et al.* (1990), this scheduling problem is denoted as $1 \| \Sigma_j C_j$. To be more specific, let the single machine M_1 correspond to the single server S_1 with speed v_1 and the n jobs $J_1, J_2, \ldots, J_n$ to the n requests $R_1, R_2, \ldots, R_n$. Let the execution time of J_j be $p_j + 2d_{1j}/v_1$ and the release time of J_j be 0.

Note that $w_j = s_j - r_j = s_j - 0 = s_j$. From Fig. 1, we then have that the completion time C_j of job J_j is

$$C_j = s_j + p_j + d_{1j}/v_1 = w_j + p_j + d_{1j}/v_1$$

and the total completion time $\Sigma_j C_j$ is

$$\prod_j C_j = \prod_j w_j + \prod_j p_j + \prod_j d_{1j}/v_1.$$

Since $\Sigma_j p_j$ and $\Sigma_j d_{1j}/v_1$ have nothing to do with the schedule and are solely determined by the input parameters, they can be treated as constants. Therefore, minimizing $\Sigma_j C_j$ in $1 \| \Sigma_j C_j$ is equivalent to minimizing $\Sigma_j w_j$ in the mobile 1-server problem with $r_j = 0$ for $j = 1, 2, \ldots, n$. According to Smith (1956), the *shortest-job-first* (*SJF*) queue discipline gives the optimal solution for $1 \| \Sigma_j C_j$. The algorithm first sorts the jobs by nondecreasing execution time $p_j + 2d_{1j}/v_1$, and then schedules the jobs on the machine in that order. We then have the following theorem.

Theorem 1. The mobile k-server problem is solvable in $O(n \log n)$ when $k = 1$ and $r_j = 0$ for $j = 1, 2, \ldots, n$.

2.2. *Multiple mobile servers*

When there are two or more servers, i.e. $k \geqslant 2$, the mobile k-server problem is related (but not equivalent) to the scheduling problem $R \| \Sigma_j C_j$, where n jobs are to be scheduled on k unrelated machines in such a way that the total completion time of all jobs, $\Sigma_j C_j$, is minimized. To be more specific, let the k machines $M_1, M_2, \ldots, M_k$ correspond to the k servers $S_1, S_2, \ldots, S_k$ and the n jobs $J_1, J_2, \ldots, J_n$ to the n requests $R_1, R_2, \ldots, R_n$. Let

the execution time of J_j be $p_j + 2d_{ij}/v_i$ if J_j is scheduled on M_i and the release time of J_j be 0. See Table 1 for the definition of the jobs' execution times.

Table 1. Execution times

	J_1	J_2	$\cdots$	J_n
M_1	$p_1 + 2d_{11}/v_1$	$p_2 + 2d_{12}/v_1$	$\cdots$	$p_n + 2d_{1n}/v_1$
M_2	$p_1 + 2d_{21}/v_2$	$p_2 + 2d_{22}/v_2$	$\cdots$	$p_n + 2d_{2n}/v_2$
$\vdots$	$\vdots$	$\vdots$		$\vdots$
M_k	$p_1 + 2d_{k1}/v_k$	$p_2 + 2d_{k2}/v_k$	$\cdots$	$p_n + 2d_{kn}/v_k$

Note that $w_j = s_j - r_j = s_j - 0 = s_j$. Assume that J_j is scheduled on $M_{i(j)}$, where $i(j)$ is a function determined by the job-machine assignment, we then have that the completion time C_j of job J_j is

$$C_j = s_j + p_j + d_{i(j)j}/v_{i(j)} = w_j + p_j + d_{i(j)j}/v_{i(j)}$$

and that the total completion time $\Sigma_j C_j$ is

$$\prod_j C_j = \prod_j w_j + \prod_j p_j + \prod_j d_{i(j)j}/v_{i(j)}.$$

Since $\Sigma_j p_j$ has nothing to do with the schedule and is solely determined by the input parameters, it can be treated as a constant. However, $\Sigma_j d_{i(j)j}/v_{i(j)}$ is related to the schedule, in the sense that $i(j)$ is the index of the machine that executes J_j in the schedule. Therefore, minimizing $\Sigma_j C_j$ in $R \| \Sigma_j C_j$ is not equivalent to minimizing $\Sigma_j w_j$ in the mobile k-server problem with $k \geqslant 2$ and $r_j = 0$ for $j = 1, 2, \ldots, n$.

We observe that despite their differences, $R \| \Sigma_j C_j$ and the mobile k-server problem with $k \geqslant 2$ and $r_j = 0$ for $j = 1, 2, \ldots, n$ are similar in many aspects. These similarities allow us to use the same idea as in the algorithm that solves $R \| \Sigma_j C_j$, which is given by Bruno *et al.* (1974), to solve the mobile k-server problem. In their algorithm, the scheduling problem $R \| \Sigma_j C_j$ is formulated as an integer program. The structure of this integer program is such that it can be converted to an equivalent problem of finding a minimum weight matching of a weighted bipartite graph, which can then be solved efficiently. Here, to solve the mobile k-server problem with $k \geqslant 2$ and $r_j = 0$ for $j = 1, 2, \ldots, n$, we use a similar idea.

We first observe that since all requests are available at time 0, a server never has to wait before serving the next request. This means that for any server, there is no idle period between the services of two requests, otherwise, we can always obtain a shorter total waiting time by removing the idle period. Given any instance of the mobile k-server problem with $k \geqslant 2$ and $r_j = 0$ for $j = 1, 2, \ldots, n$, a feasible assignment of the servers to the requests can be described by a set of 0–1 variables $x_{i,j,l}$, where $x_{i,j,l} = 1$ if R_j is the lth last request served by S_i and $x_{i,j,l} = 0$ otherwise. Since $r_j = 0$, then $w_j = s_j - r_j = s_j - 0 = s_j$. Now consider the total waiting time $\Sigma_j w_j$, which is also the $\Sigma_i W_i$, where W_i is the sum of the waiting times (starting times) of all requests served by S_i. To compute W_i, let us consider the example in Fig. 1. For this example let $k \geqslant 2$ and let server S_1 first provide service to R_5, then to R_2, and finally to R_8. It is obvious that $W_1 = w_5 + w_2 + w_8 = s_5 + s_2 + s_8 = (d_{15}/v_1) + (p_5 + 2d_{15}/v_1 + d_{12}/v_1) + (p_5 + 2d_{15}/v_1 + p_2 + 2d_{12}/v_1 + d_{18}/v_1) = (2p_5 + 5d_{15}/v_1)x_{1,5,3} + (p_2 + 3d_{12}/v_1)x_{1,2,2} + (d_{18}/v_1)x_{1,8,1}$. To generalize the formula,

we have

$$W_i = \prod_{j,l} ((l-1)p_j + (2l-1)d_{ij}/v_i)x_{i,j,l}$$

and

$$\prod_j w_j = \prod_i W_i = \prod_{i,j,l} ((l-1)p_j + (2l-1)d_{ij}/v_i)x_{i,j,l}.$$

The mobile k-server problem then becomes the following 0–1 integer program:

$$\min \prod_{i,j,l} ((l-1)p_j + (2l-1)d_{ij}/v_i)x_{i,j,l}$$

subject to

$$\prod_{i,l} x_{i,j,l} = 1 \quad \text{for } j = 1, 2, \ldots, n \tag{1}$$

$$\prod_j x_{i,j,l} \leqslant 1 \quad \text{for } i = 1, 2, \ldots, k,\ l = 1, 2, \ldots, n \tag{2}$$

$$x_{i,j,l} = 0, 1 \quad \text{for } i = 1, 2, \ldots, k,\ j = 1, 2, \ldots, n,\ l = 1, 2, \ldots, n. \tag{3}$$

Constraint (1) ensures that each request is served exactly once and constraint (2) ensures that each position on each server is occupied by at most one request.

To solve this integer program, we define a weighted complete bipartite graph $G = (V_1 \cup V_2, E)$, where $V_1 = \{R_1, R_2, \ldots, R_n\}$, $V_2 = \{S_{11}, S_{12}, \ldots, S_{1n}, S_{21}, S_{22}, \ldots, S_{2n}, \ldots, S_{k1}, S_{k2}, \ldots, S_{kn}\}$, and the weight $w(R_j, S_{il})$ on edge (R_j, S_{il}) is $(l-1)p_j + (2l-1)d_{ij}/v_i$ for $i = 1, 2, \ldots, k$, $j = 1, 2, \ldots, n$ and $l = 1, 2, \ldots, n$. Figure 2(a) shows a complete bipartite graph with $k = 2$ and $n = 3$. We see that any definition of $x_{i,j,l}$ for $i = 1, 2, \ldots, k$, $j = 1, 2, \ldots, n$ and $l = 1, 2, \ldots, n$ which satisfies constraints (1), (2) and (3) defines a matching in G, i.e. $x_{i,j,l} = 1$ if and only if edge (R_j, S_{il}) is in the matching. Constraint (1) implies that for each $R_j \in V_1$ there is exactly one edge with R_j as one endpoint in the matching and constraint (2) implies that for each $S_{il} \in V_2$ there is at most one edge with S_{il} as one endpoint

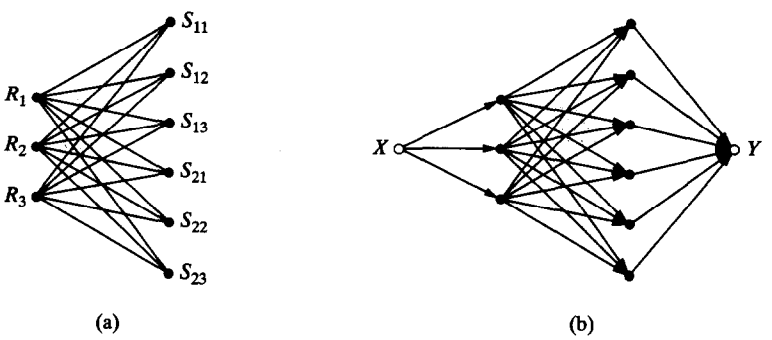

Fig. 2. (a) A complete bipartite graph G; (b) A network G' with source X and sink Y.

in the matching. The integer program is then equivalent to finding a matching in G with the minimum total weight.

As pointed out by Tarjan (1983), the above minimum weight matching problem can further be reduced to a minimum cost maximum flow problem. Let $G = (V_1 \cup V_2, E)$ be the complete bipartite graph defined earlier. Let X and Y be two new vertices. Construct a directed graph G' with vertex set $V_1 \cup V_2 \cup \{X, Y\}$, source X, sink Y, arcs $\langle X, R_j\rangle$ of cost 0 for every $R_j \in V_1$, arcs $\langle R_j, S_{il}\rangle$ of cost $w(R_j, S_{il})$ for every $(R_j, S_{il}) \in E$, and arcs $\langle S_{il}, Y\rangle$ of cost 0 for every $S_{il} \in V_2$. Finally, we assume that the flow capacity of each arc in G' is 1. See Fig. 2(b). Therefore, finding the minimum weight matching in G is equivalent to finding the minimum cost maximum flow from source X to sink Y in network G'. By Tarjan (1983), if a unit-capacity network has v vertices and a arcs, then the minimum cost maximum flow can be found in $O(v \cdot a \cdot \log_{2+a/v} v)$. Since $v = n + kn + 2$ and $a = n + n \cdot kn + kn$, the mobile k-server problem can therefore be solved in $O((n + kn + 2)(n + kn^2 + kn)\log_{2+(n+kn^2+kn)/(n+kn+2)}(n + kn + 2)) = O(k^2n^3 \log kn)$. We have the following theorem.

Theorem 2. The mobile k-server problem is solvable in $O(k^2n^3 \log kn)$ when $k \geqslant 2$ and $r_j = 0$ for $j = 1, 2, \ldots, n$.

3. ARBITRARY RELEASE TIMES

In this section, we consider the case when r_j is arbitrary. Since requests arrive at arbitrary times, it becomes difficult to determine a service schedule that minimizes the total waiting time. We shall see that whether $k = 1$ or $k \geqslant 2$, the mobile k-server problem with arbitrary release times is intractable.

3.1. *Single mobile server*

The mobile k-server problem with $k = 1$ is equivalent to the scheduling problem $1 \mid r_j \mid \Sigma_j C_j$, where n jobs with arbitrary release times are to be scheduled on a single machine in such a way that the total completion time of all jobs, $\Sigma_j C_j$, is minimized. The equivalence is achieved by defining the execution time of job J_j to be $p_j + 2d_{1j}/v_1$ and the release time of J_j to be r_j.

Since $w_j = s_j - r_j$, we have that the completion time C_j of job J_j is

$$C_j = s_j + p_j + d_{1j}/v_j = w_j + r_j + p_j + d_{1j}/v_1$$

and that the total completion time $\Sigma_j C_j$ is

$$\prod_j C_j = \prod_j w_j + \prod_j r_j + \prod_j p_j + \prod_j d_{1j}/v_1.$$

Since $\Sigma_j r_j$, $\Sigma_j p_j$ and $\Sigma_j d_{1j}/v_1$ have nothing to do with the schedule and are solely determined by the input parameters, they can be treated as constants. Therefore, minimizing $\Sigma_j C_j$ in $1 \mid r_j \mid \Sigma_j C_j$ is equivalent to minimizing $\Sigma_j w_j$ in the mobile 1-server problem. Since $1 \mid r_j \mid \Sigma_j C_j$ is proved to be NP-complete by Lenstra *et al.* (1977), the mobile 1-server problem is also NP-complete. This is stated in the following theorem.

Theorem 3. The mobile k-server problem with $k = 1$ and arbitrary r_j is NP-complete.

Because of the equivalence between $1|r_j|\Sigma_j C_j$ and the mobile 1-server problem, the heuristics designed for $1|r_j|\Sigma_j C_j$ are also suitable to the mobile 1-server problem. The two most common heuristics are first-come-first-serve (FCFS) and shortest-available-request-first (SARF). Let S_1 be the only server, and Q_R be a queue containing all requests that were released but have not been served. In FCFS, the requests in Q_R are ordered by nondecreasing r_j, while in SARF, the requests in Q_R are ordered by nondecreasing $p_j + 2d_{1j}/v_1$. Both FCFS and SARF can be described by actions at the requests' end, i.e. the queue (request) discipline, and by actions at the server's end, i.e. the server discipline.

FCFS and SARF:

- Queue (request) discipline: When a new request R_j arrives, it is inserted in the correct position in Q_R using r_j in FCFS and $p_j + 2d_{1j}/v_1$ in SARF, as the key for insertion.
- Server discipline: When S_1 is idle at home and Q_R is not empty, S_1 leaves home immediately to serve the first request in Q_R.

Mao *et al.* (1995) give a rigorous analysis of the performance of FCFS and SARF when they are applied to $1|r_j|\Sigma_j C_j$. They show that in the worst case the total completion times of these two heuristics are both within n times that of the optimal solution, where n is the number of jobs in $1|r_j|\Sigma_j C_j$. This result cannot be applied directly to the mobile 1-server problem, because in the mobile 1-server problem the cost function is no longer the total completion time, but instead the total waiting time. Since

$$\prod_j w_j = \prod_j C_j - \prod_j r_j - \prod_j p_j - \prod_j d_{1j}/v_1$$

and $\Sigma_j r_j$, $\Sigma_j p_j$ and $\Sigma_j d_{1j}/v_1$ are all treated as constants, our first guess is that in the worst case the total waiting times of FCFS and SARF are within $f(n)$ times that of the optimal solution, where $f(n)$ is perhaps in the form of $an + b$. To our surprise, the result has nothing to do with n. We present the following theorem.

Theorem 4. For any instance I of the mobile 1-server problem, define $\Sigma_j w_j[\text{FCFS}(I)]$, $\Sigma_j w_j[\text{SARF}(I)]$ and $\Sigma_j w_j[\text{OPT}(I)]$ to be the total waiting times of all requests in the service schedules obtained by the FCFS algorithm, the SARF algorithm and the optimal (OPT) algorithm, respectively. Then we have

$$\prod_j w_j[\text{FCFS}(I)] \leqslant \frac{M}{m} \cdot \prod_j w_j[\text{OPT}(I)] + \frac{M-m}{m} \prod_j r_j,$$

and

$$\prod_j w_j[\text{SARF}(I)] \leqslant \frac{M}{m} \cdot \prod_j w_j[\text{OPT}(I)] + \frac{M-m}{m} \prod_j r_j,$$

where $M = \max_j\{p_j + 2d_{1j}/v_1\}$ and $m = \min_j\{p_j + 2d_{1j}/v_1\}$. Furthermore, there exist instances for which the equalities hold asymptotically.

Proof. We will only prove the result for FCFS. The proof for SARF is almost identical and is therefore omitted. Consider any instance I of the mobile 1-server problem with

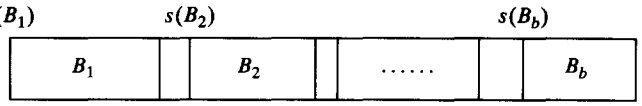

Fig. 3. A FCFS service schedule.

server S_1 and requests $R_1, R_2, \ldots, R_n$. Assume R_j requires service time p_j and is available at release time r_j. Let the distance between S_1 and R_j be d_{1j} and the traveling speed of S_1 be v_1.

Consider the service schedule for I obtained by FCFS, which is denoted as FCFS(I). It must have a block structure as shown in Fig. 3: $B_1, B_2, \ldots, B_b$, where in each block there is no idle time, and between two consecutive blocks there is an idle period. Let $s(B_i)$ be the starting time of block B_i. Obviously, $r_j \geqslant s(B_i)$ for any $R_j \in B_i$.

We define a new instance I' as follows. Let I' contain server S_1' and requests $R_1', R_2', \ldots, R_n'$, where the service time p_j' of R_j' is p_j, the distance d_{1j}' between S_1' and R_j' is d_{1j}, the traveling speed v_1' of S_1' is v_1, and finally, the release time r_j' of R_j' is $s(B_i)$ if $R_j \in B_i$ in FCFS(I). The optimal schedule for I', denoted as OPT(I'), obviously has the same block structure as FCFS(I). Each block B_i' in OPT(I') contains the same requests as the corresponding block B_i in FCFS(I). Furthermore, the requests in each B_i' are ordered by nondecreasing $p_j + 2d_{1j}/v_1$. For notational simplicity, let $q_j = p_j + 2d_{1j}/v_1$ throughout the rest of the proof.

Let $\Sigma_j w_j[\text{OPT}(I)]$ and $\Sigma_j w_j[\text{OPT}(I')]$ be the total waiting times of requests in OPT(I) and OPT(I'), respectively. Because I and I' are the same in all parameters except that requests in I' arrive at least as early as those in I, the total starting time of all requests in OPT(I) is at least as large as that in OPT(I'). We then have

$$\prod_j s_j[\text{OPT}(I)] = \prod_j w_j[\text{OPT}(I)] + \prod_j r_j \geqslant \prod_j w_j[\text{OPT}(I')] + \prod_j r_j' = \prod_j s_j[\text{OPT}(I')].$$

Now, let $\Sigma_j w_j[\text{FCFS}(I), B_i]$ and $\Sigma_j w_j[\text{OPT}(I'), B_i']$ be the total waiting times of requests in block B_i of FCFS(I) and block B_i' of OPT(I'), respectively. Assume $M_i = \max_{R_j \in B_i}\{q_j\}$ and $m_i = \min_{R_j \in B_i}\{q_j\}$. Next we prove that $\Sigma_j w_j[\text{FCFS}(I), B_i] \leqslant \frac{M_i}{m_i} \cdot \Sigma_j w_j[\text{OPT}(I'), B_i'] + \Sigma_{R_j' \in B_i'} r_j' - \Sigma_{R_j \in B_i} r_j$.

Without loss of generality, we prove the above inequality for blocks B_1 and B_1'. We assume that B_1 contains $R_1, R_2, \ldots, R_l$ and that B_1' contains $R_1', R_2', \ldots, R_l'$. Moreover, let $q_1 \leqslant q_2 \leqslant \cdots \leqslant q_l$. Obviously, $M_1 = q_l$ and $m_1 = q_1$. We first have

$$\begin{aligned}
&\prod_j w_j[\text{OPT}(I'), B_1'] \\
&= [s(B_1) + d_{11}/v_1 - s(B_1)] + [s(B_1) + q_1 + d_{12}/v_1 - s(B_1)] + \cdots \\
&\quad + [s(B_1) + q_1 + q_2 + \cdots + q_{l-1} + d_{1l}/v_1 - s(B_1)] \\
&= (d_{11}/v_1 + \cdots + d_{1l}/v_1) + (l-1)\cdot q_1 + (l-2)\cdot q_2 + \cdots + 2\cdot q_{l-2} + 1\cdot q_{l-1}.
\end{aligned}$$

We then have the following upper bound on the start times for FCFS by servicing the requests from the longest (q_l) to the shortest (q_1):

$$\prod_j s_j[\mathrm{FCFS}(I), B_1]$$

$$\leqslant [s(B_1) + d_{1l}/v_1] + [s(B_1) + q_l + d_{1(l-1)}/v_1] + \cdots$$
$$+ [s(B_1) + q_l + q_{l-1} + \cdots + q_2 + d_{11}/v_1]$$

$$= \prod_{R_j' \in B_1'} r_j' + (d_{11}/v_1 + \cdots + d_{1l}/v_1)$$
$$+ (l-1)\cdot q_l + (l-2)\cdot q_{l-1} + \cdots + 2\cdot q_3 + 1\cdot q_2.$$

The following claim simplifies the proof of the theorem.

Claim 1. $\frac{q_l}{q_1}\cdot 1\cdot q_{l-1} + \frac{q_l}{q_1}\cdot 2\cdot q_{l-2} + \cdots + \frac{q_l}{q_1}\cdot (l-3)\cdot q_3 + \frac{q_l}{q_1}\cdot (l-2)\cdot q_2 \geqslant (l-2)\cdot q_{l-1} + (l-3)\cdot q_{l-2} + \cdots + 2\cdot q_3 + 1\cdot q_2$.

Proof. See Appendix.

Therefore, after some algebra and involving Claim 1, we have

$$\prod_j w_j[\mathrm{FCFS}(I), B_1]$$

$$= \prod_j s_j[\mathrm{FCFS}(I), B_1] - \prod_{R_j \in B_1} r_j$$

$$\leqslant \prod_{R_j' \in B_1'} r_j' + (d_{11}/v_1 + \cdots + d_{1l}/v_1) + (l-1)\cdot q_l$$
$$+ [(l-2)\cdot q_{l-1} + (l-3)\cdot q_{l-2} + \cdots + 2\cdot q_3 + 1\cdot q_2] - \prod_{R_j \in B_1} r_j$$

$$\leqslant (d_{11}/v_1 + \cdots + d_{1l}/v_1) + (l-1)\cdot q_l$$
$$+ \left[\frac{q_l}{q_1}\cdot 1\cdot q_{l-1} + \frac{q_l}{q_1}\cdot 2\cdot q_{l-2} + \cdots + \frac{q_l}{q_1}\cdot (l-3)\cdot q_3 + \frac{q_l}{q_1}\cdot (l-2)\cdot q_2\right]$$
$$+ \prod_{R_j' \in B_1'} r_j' - \prod_{R_j \in B_1} r_j$$

$$\leqslant \frac{q_l}{q_1}\cdot [(d_{11}/v_1 + \cdots + d_{1l}/v_1) + (l-1)\cdot q_1 + (l-2)\cdot q_2 + \cdots + 2\cdot q_{l-2} + 1\cdot q_{l-1}]$$
$$+ \prod_{R_j' \in B_1'} r_j' - \prod_{R_j \in B_1} r_j$$

$$= \frac{M_1}{m_1}\cdot \prod_j w_j[\mathrm{OPT}(I'), B_1'] + \prod_{R_j' \in B_1'} r_j' - \prod_{R_j \in B_1} r_j.$$

Since the above holds for any block B_i, we then have

$$\prod_j w_j[\text{FCFS}(I)]$$

$$= \prod_j w_j[\text{FCFS}(I), B_1] + \cdots + \prod_j w_j[\text{FCFS}(I), B_b]$$

$$\leqslant \frac{M_1}{m_1} \cdot \prod_j w_j[\text{OPT}(I'), B'_1] + \cdots + \frac{M_b}{m_b} \cdot \prod_j w_j[\text{OPT}(I'), B'_b] + \prod_j r'_j - \prod_j r_j$$

$$\leqslant \frac{M}{m} \cdot \left[\prod_j w_j[\text{OPT}(I'), B'_1] + \cdots + \prod_j w_j[\text{OPT}(I'), B'_b]\right] + \prod_j r'_j - \prod_j r_j$$

$$= \frac{M}{m} \cdot \prod_j w_j[\text{OPT}(I')] + \prod_j r'_j - \prod_j r_j$$

$$\leqslant \frac{M}{m} \cdot \left(\prod_j w_j[\text{OPT}(I)] + \prod_j r_j - \prod_j r'_j\right) + \prod_j r'_j - \prod_j r_j$$

$$= \frac{M}{m} \cdot \prod_j w_j[\text{OPT}(I)] + \frac{M-m}{m} \cdot \left(\prod_j r_j - \prod_j r'_j\right)$$

$$\leqslant \frac{M}{m} \cdot \prod_j w_j[\text{OPT}(I)] + \frac{M-m}{m} \cdot \prod_j r_j.$$

The instance I^* for which $\Sigma_j w_j[\text{FCFS}(I^*)]$ is asymptotically equal to $\frac{M}{m} \cdot \Sigma_j w_j[\text{OPT}(I^*)] + \frac{M-m}{m} \Sigma_j r_j$ can be constructed as follows. Let $n = 2$ and $d_{11} = d_{12} = 0$. Let $r_1 = 0$, $r_2 = \varepsilon$ and $p_1 - p_2 > 2\varepsilon$. Obviously, $M = p_1$ and $m = p_2$. In FCFS(I^*), S_1 goes to serve R_1 and then R_2. So

$$\prod_j w_j[\text{FCFS}(I^*)] = 0 + (p_1 - \varepsilon) = p_1 - \varepsilon$$

In OPT(I^*), S_1 waits ε time units, goes to serve R_2 and then R_1. So

$$\prod_j w_j[\text{OPT}(I^*)] = (p_2 + \varepsilon) + 0 = p_2 + \varepsilon.$$

Therefore,

$$\frac{M}{m} \cdot \prod_j w_j[\text{OPT}(I^*)] + \frac{M-m}{m} \prod_j r_j - \prod_j w_j[\text{FCFS}(I^*)]$$

$$= \frac{p_1}{p_2} \cdot (p_2 + \varepsilon) + \frac{p_1 - p_2}{p_2} \varepsilon - (p_1 - \varepsilon)$$

$$= \frac{2p_1}{p_2} \cdot \varepsilon$$

$$\to 0$$

as $\varepsilon \to 0$. □

We see that in Theorem 4, when $M = m$, i.e. $p_j + 2d_{1j}/v_1 = q$ for some constant q, FCFS and SARF give the optimal service schedule. That is, when $p_j + 2d_{1j}/v_1 = q$ for all j, it makes no difference which available request is served first. In fact, any conservative queue discipline gives a service schedule with the minimum total completion time and, therefore, the total waiting time $\Sigma_j w_j$ is minimized as well. We then have the following corollary.

Corollary 1. For any instance of the mobile 1-server problem, where $p_j + 2d_{1j}/v_1 = q$ for $j = 1, 2, \ldots, n$ and some constant q, both FCFS and SARF give the optimal service schedule that minimizes the total waiting time.

Another heuristic of interest is called *look-ahead* (*LA*), in which a look-ahead queue Q'_R is used to keep the next arriving request. After server S_1 returns home, it checks both Q_R and Q'_R and then decides whether to go ahead to serve the first request in Q_R or wait for a while until the request in Q'_R is released. This strategy is useful especially when the first request in Q_R asks for long service at a location far away and the request in Q'_R that will be released soon asks for short service at a location close to S_1. We note that LA is *not* a conservative queue discipline. Mao and Kincaid (1994) introduce this heuristic and apply it to $1|r_j|\Sigma_j C_j$. They also prove that in the worst case the total completion time of jobs in a LA schedule is about 67% better than that of FCFS and SARF. However, we do not know the worst-case behavior of LA when applied to the mobile 1-server problem with minimizing total waiting time as its performance measure.

3.2. *Multiple mobile servers*

Now let us consider the mobile k-server problem with $k \geqslant 2$ and arbitrary r_j. We claim that this problem is also NP-complete since the NP-complete $1|r_j|\Sigma_j C_j$ is a special case of it. To see this, for any instance of $1|r_j|\Sigma_j C_j$, in which job J_j has execution time p_j and release time r_j, we construct an instance of the mobile k-server problem with $k = 2$ as follows. Let request R_j correspond to job J_j, with service time p_j and release time r_j. Assume that servers S_1 and S_2 are very far apart and that all requests appear at the home location of S_1, i.e. $d_{1j} = 0$ and $d_{2j} = +\infty$ for $j = 1, 2, \ldots, n$. Since S_2 is too far away, all requests will have to be served by S_1. It is clear that $\Sigma_j C_j$ is minimized in $1|r_j|\Sigma_j C_j$ if and only if $\Sigma_j w_j$ is minimized in the mobile k-server problem constructed. We then have the following theorem.

Theorem 5. The mobile k-server problem with $k \geqslant 2$ and arbitrary r_j is NP-complete.

We can generalize the heuristics for the mobile 1-server problem, FCFS and SARF, to heuristics for the mobile k-server problem with $k \geqslant 2$. We still use Q_R for the queue of requests that were released but have not been served. In addition, we use Q_S for the queue of servers idle at their home locations. We begin with a generalization of FCFS, in which the requests in Q_R are ordered by nondecreasing r_j. When two or more idle servers in Q_S are available to provide service for a request R_j, the server S_i with the smallest d_{ij}/v_i is assigned to R_j. We call this algorithm *earliest-request-closest-server* (*ERCS*). It can be described by the following queue and server disciplines.

ERCS

- Queue (request) discipline: When a new request R_j arrives, it is inserted in the correct position in Q_R using r_j as the key for insertion.

- Server discipline: When both Q_S and Q_R are not empty, server S_i with the smallest d_{ij}/v_i in Q_S is assigned to the first request R_j in Q_R.

The generalization of SARF is not as obvious as that of FCFS. This is because the key $p_j + 2d_{ij}/v_i$ that is used to sort Q_R is determined not only by the request, but also by the server–request assignment. Thus, Q_R can no longer be sorted since the server assignment cannot, in general, be made when a request is released. When both Q_S and Q_R are not empty, the algorithm chooses $S_i \in Q_S$ and $R_j \in Q_R$ with the smallest $p_j + 2d_{ij}/v_i$ and assigns S_i to R_j. Note that R_j may not be the first request in Q_R. We call this algorithm *shortest-request-closest-server* (*SRCS*). It can be described by the following queue and server disciplines.

SRCS

- Queue (request) discipline: When a new request R_j arrives, it is added to Q_R.
- Server discipline: When both Q_S and Q_R are not empty, server $S_i \in Q_S$ is assigned to request $R_j \in Q_R$ with the smallest $p_j + 2d_{ij}/v_i$.

Studying the worst-case performance of ERCS and SRCS in comparison with the optimal schedule OPT, we have the following two theorems. Each theorem establishes a lower bound on the worst-case performances of ERCS and SRCS.

Theorem 6. For any instance I of the mobile k-server problem, define $\Sigma_j w_j[\mathrm{ERCS}(I)]$, $\Sigma_j w_j[\mathrm{SRCS}(I)]$ and $\Sigma_j w_j[\mathrm{OPT}(I)]$ to be the total waiting times of all requests in the service schedules obtained by ERCS, SRCS and OPT, respectively. Then there exists an instance I_1 such that

$$\frac{\Sigma_j w_j[\mathrm{ERCS}(I_1)]}{\Sigma_j w_j[\mathrm{OPT}(I_1)]} \to \frac{M}{m}$$

and an instance I_2 such that

$$\frac{\Sigma_j w_j[\mathrm{SRCS}(I_2)]}{\Sigma_j w_j[\mathrm{OPT}(I_2)]} \to \frac{M}{m},$$

where $M = \max_{i,j}\{p_j + 2d_{ij}/v_i\}$ and $m = \min_{i,j}\{p_j + 2d_{ij}/v_i\}$.

Proof. The instance I we construct works for the proofs of both ERCS and SRCS. Let $n = 2k$, $d_{ij} = 0$ for $i = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, n$. Hence, the home locations of the servers and the location of all requests for service are coincident. The service times and release times of the requests are defined in Table 2, where $a \gg b > 0$ and $\varepsilon > 0$. Obviously, $M = a$ and $m = b$.

Table 2. An instance I with $n = 2k$ and $d_{ij} = 0$ for all i and j

	R_1	$\cdots$	R_k	R_{k+1}	$\cdots$	R_n
p_j	a	$\cdots$	a	b	$\cdots$	b
r_j	0	$\cdots$	0	ε	$\cdots$	ε

It is easy to see that in both ERCS and SRCS schedules, all k servers provide service to the k long requests $R_1, \ldots, R_k$ first, and then provide service to the k short requests $R_{k+1}, \ldots, R_n$. So

$$\prod_j w_j[\text{ERCS}(I)] = \prod_j w_j[\text{SRCS}(I)] = k \cdot 0 + k \cdot (a - \varepsilon) = k \cdot (a - \varepsilon).$$

In the OPT schedule, all k servers wait ε time units, provide service to the k short requests first, and then provide service to the k long requests. So

$$\prod_j w_j[\text{OPT}(I)] = k \cdot (b + \varepsilon) + k \cdot 0 = k \cdot (b + \varepsilon).$$

Therefore,

$$\frac{\Sigma_j w_j[\text{ERCS}(I)]}{\Sigma_j w_j[\text{OPT}(I)]} = \frac{\Sigma_j w_j[\text{SRCS}(I)]}{\Sigma_j w_j[\text{OPT}(I)]} = \frac{k \cdot (a - \varepsilon)}{k \cdot (b + \varepsilon)} = \frac{M - \varepsilon}{m + \varepsilon} \to \frac{M}{m}$$

as $\varepsilon \to 0$. □

Theorem 7. Let $\Sigma_j w_j[\text{ERCS}(I)]$, $\Sigma_j w_j[\text{SRCS}(I)]$ and $\Sigma_j w_j[\text{OPT}(I)]$ be as defined before. Then there exists an instance I_1 such that

$$\frac{\Sigma_j w_j[\text{ERCS}(I_1)]}{\Sigma_j w_j[\text{OPT}(I_1)]} \to 2n - k$$

and there exists an instance I_2 such that

$$\frac{\Sigma_j w_j[\text{SRCS}(I_2)]}{\Sigma_j w_j[\text{OPT}(I_2)]} \to 2n - k.$$

Proof. The instance I we construct works for the proofs of both ERCS and SRCS. Let $n = hk + 1$ for some h, $v_i = 1$ for $i = 1, 2, \ldots, k$ and $p_j = 1$ for $j = 1, 2, \ldots, n$. The home locations of the servers and the release times and locations of the requests are depicted in Fig. 4.

The network topology is a line of $k + 1$ nodes, 0, 1, 2, ..., k, where the distance between any two adjacent nodes is L. The home location of S_i is node i for $i = 1, 2, \ldots, k$. Request

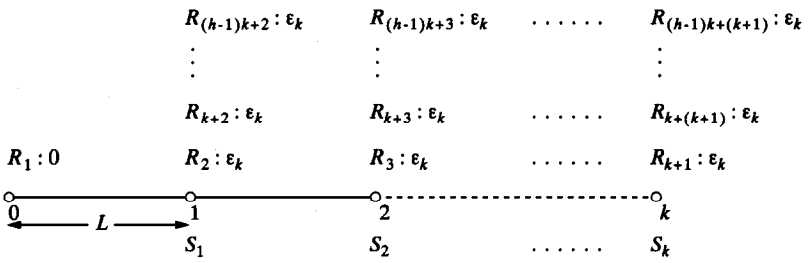

Fig. 4. An instance I with $n = hk + 1$, $v_i = 1$ and $p_j = 1$ for all i and j.

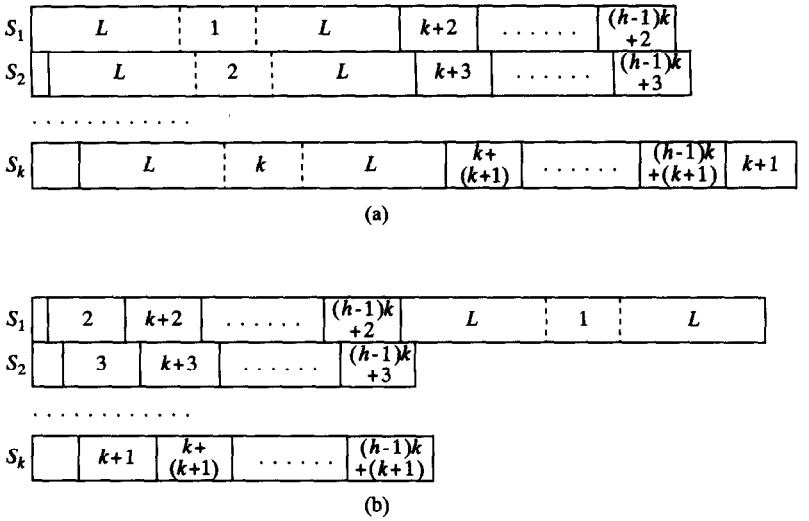

Fig. 5. (a) The ERCS and SRCS schedules for I; (b) The OPT schedule for I.

R_1 arrives at the node 0 at time 0, R_j arrives at node $j-1$ at time ε_{j-1} for $j = 2, 3, \ldots, k+1$. Furthermore, $R_{j_1+j_2}$ arrives at node $j_2 - 1$ at time ε_k for $j_1 = k, 2k, \ldots, (h-1)k$ and $j_2 = 2, 3, \ldots, k+1$. Assume that $0 < \varepsilon_1 < \varepsilon_2 < \cdots < \varepsilon_k < 1$.

In both ERCS and SRCS schedules, after R_1 arrives at node 0 at time 0, S_1 leaves its home immediately to serve R_1. At time ε_1, R_2 arrives at node 1. Then, S_2 is assigned to R_2. We can see that in ERCS(I) and SRCS(I), S_j travels the distance L to serve R_j for $j = 1, 2, \ldots, k$ and, after returning home, each server then provides service to the remaining requests that are at its home location. This service schedule is illustrated in Fig. 5(a). For simplicity, the expressions in Fig. 5 other than L are indices of requests. We then have

$$\begin{aligned}\prod_j w_j[\text{ERCS}(I)] &= \prod_j w_j[\text{SRCS}(I)] \\ &= kL + (k-1)(h-1)\cdot 2L + h\cdot 2L + f_1 \\ &= (2hk - k + 2)L + f_1,\end{aligned}$$

where f_1 is an expression of all parameters of I except L.

In the OPT schedule, however, each server first serves the requests that are at its home location and then S_1 travels to node 0 to serve R_1. The service schedule is illustrated in Fig. 5(b). We then have

$$\prod_j w_j[\text{OPT}(I)] = L + f_2,$$

where f_2 is an expression of all parameters of I except L.

Therefore,

$$\frac{\Sigma_j w_j[\text{ERCS}(I)]}{\Sigma_j w_j[\text{OPT}(I)]} = \frac{\Sigma_j w_j[\text{SRCS}(I)]}{\Sigma_j w_j[\text{OPT}(I)]}$$
$$= \frac{(2hk - k + 2)L + f_1}{L + f_2}$$
$$= \frac{(2n - k)L + f_1}{L + f_2}$$
$$\to 2n - k$$

as $L \to \infty$. □

From Theorem 6, we know that there exists at least one instance of the mobile k-server problem for which the total waiting times in both ERCS and SRCS schedules are at least $\frac{M}{m}$ times that in the OPT schedule. From Theorem 7, we know that there exists at least one instance of the mobile k-server problem for which the total waiting times in both ERCS and SRCS schedules are at least $2n - k$ times that in the OPT schedule. We see that $\frac{M}{m}$ and $2n - k$ are both lower bounds. At present, we are unable to determine any nontrivial upper bound for ERCS and SRCS as we have done for FCFS and SARF.

4. CONCLUSION

In this paper we presented an analysis of service schedules for the mobile k-server problem with total waiting time as the performance measure to be optimized. In particular, we gave efficient optimal algorithms for the case when all requests have equal release times. We then considered the case when requests have arbitrary release times. We showed that for the single server problem, both FCFS and SARF construct service schedules with total waiting times within approximately $\frac{M}{m}$ times that of the optimal schedule. For the multiple server problem, the total waiting times of the service schedules constructed by ERCS and SRCS are bounded below by $\max\{\frac{M}{m}, 2n - k\}$ times that of the optimal schedules. Together these results provide the first analytical results of service schedules for the mobile k-server problem. The limiting assumptions are that the mobile servers must return to their home locations before servicing another request, and that these home locations are known *a priori*. Otherwise, the model is completely general. There are no restrictions placed upon where the requests for service must originate, or what distribution describes these requests for service.

An additional feature of our approach is that it provides a means by which competing home location schemes (p-median problem, probabilistic covering model, or the hypercube model) can be evaluated and compared for the case when all the release times are identical (Section 2). For this case we presented polynomial time algorithms which find the optimal service schedules with respect to minimizing total waiting time. Hence, competing sets of home locations can be ranked according to the total waiting times resulting from the corresponding optimal service schedules. For the case when release times are not identical we showed (Section 3) that the resulting mobile k-server problems were NP-complete. Hence, it is not possible (in general) to rank order competing sets of home locations since the optimal service schedule is typically not known.

REFERENCES

Berman, O., Chiu, S. S., Larson, R. C., Odoni, A. R. & Batta, R. (1990) Location of mobile units in a stochastic environment. In P. B. Mirchandani and R. L. Francis (Eds), *Discrete Location Theory*. New York: John Wiley and Sons.

Brandeau, M. L. & Chiu, S. S. (1989) An overview of representative problems in location research. *Management Science*, **35**, 645–673.

Bruno, J. L., Coffman, E. G. Jr. & Sethi, R. (1974) Scheduling independent tasks to reduce mean finishing time. *Communications of the ACM*, **17**, 382–387.

Chrobak, M., Karloff, H., Payne, T. & Vishwanathan, S. (1990) New results on server problems. Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms, 291–300.

Church, R. & ReVelle, C. (1974) The maximal covering location problem. *Papers of the Regional Science Association*, **32**, 101–118.

Daskin, M. S., Hogan, K. & Revelle, C. (1988) Integration of multiple, excess, backup, and expected covering models. *Environment and Planning B: Planning and Design*, **15**, 15–35.

Fujiwara, O., Makjamroen, T. & Gupta, K. K. (1987) Ambulance deplolyment analysis: a case study of Bangkok. *European Journal of Operational Research*, **31**, 9–18.

Hakimi, S. L. (1964) Optimal location of switching centers and the absolute centers and medians of a graph. *Operations Research*, **12**, 450–459.

Larson, R. C. (1974) A hypercube queueing model for facility location and redistricting in urban emergency services. *Computers and Operations Research*, **1**, 67–95.

Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G. & Shmoys, D. B. (1990) Sequencing and scheduling: Algorithms and complexity. In S. C. Graves, A. H. G. Rinnooy Kan and P. Zipkin (Eds), *Handbooks in Operations Research and Management Science, Volume 4: Logistics of Production and Inventory*. Amsterdam: North-Holland.

Lenstra, J. K., Rinnooy Kan, A. H. G. & Brucker, P. (1977) Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, **1**, 343–362.

Lubicz, M. & Mielczarek, B. (1987) Simulation modeling of emergency medical services. *European Journal of Operational Research*, **29**, 178–185.

Manasse, M. S., McGeoch, L. A. & Sleator, D. D. (1990) Competitive algorithms for server problems. *Journal of Algorithms*, **11**, 208–230.

Mao, W. & Kincaid, R. K. (1994) A look-ahead heuristic for scheduling jobs with release dates on a single machine. *Computers and Operations Research*, **21**, 1041–1050.

Mao, W., Kincaid, R. K. & Rifkin, A. (1995) On-line algorithms for a single machine scheduling problem. In S. G. Nash and A. Sofer (Eds), *The Impact of Emerging Technologies on Computing Science and Operations Research* (pp. 157–173). Dordrecht: Kluwer Academic Publishers.

Marianov, V. & ReVelle, C. (1992) The queueing probabilistic location set covering location problem and some extensions. Technical Report, Operations Research Group, Johns Hopkins University, Baltimore, MD 21218-2686.

Park, S. K., Harvey, S., Kincaid, R. K. & Miller, K. (1992) Alternative server disciplines for mobile-servers on a congested network. In O. Balci, R. Sharda and S. A. Zenios (Eds), *Computer Science and Operations Research: New Developments in Their Interfaces* (pp. 105–116). Oxford: Pergamon Press.

Savas, E. S. (1978) On equity in providing public service. *Management Science*, **24**, 800–808.

Schilling, D. A., Jayaraman, V. & Barkhi, R. (1993) A review of covering problems in facility location. *Location Science*, **1**, 25–56.

Smith, W. E. (1956) Various optimizers for single-state production. *Naval Research Logistics Quarterly*, **3**, 56–66.

Tarjan, R. E. (1983) *Data structures and network algorithms*. Philadelphia: Society for Industrial and Applied Mathematics.

Toregas, C., Swain, R., ReVelle, C. & Bergmann, L. (1971) The location of emergency service facilities. *Operations Research*, **19**, 1363–1373.

Uyeno, D. H. & Seeberg, C. (1984) A practical methodology for ambulance location. *Simulation*, **43**, August 1984, 79–87.

APPENDIX

Proof of Claim 1. Consider two cases: (1) the even case, when $l = 2h$ for some h, and (2) the odd case, when $l = 2h + 1$ for some h. When $l = 2h$, then

$$\frac{q_l}{q_1} \cdot 1 \cdot q_{l-1} + \frac{q_l}{q_1} \cdot 2 \cdot q_{l-2} + \cdots + \frac{q_l}{q_1} \cdot (l-3) \cdot q_3 + \frac{q_l}{q_1} \cdot (l-2) \cdot q_2$$

$$= \left[\frac{q_l}{q_1} \cdot 1 \cdot q_{l-1} + \frac{q_l}{q_1} \cdot (l-2) \cdot q_2\right] + \cdots + \left[\frac{q_l}{q_1} \cdot (h-1) \cdot q_{h+1} + \frac{q_l}{q_1} \cdot h \cdot q_h\right]$$

$$\geqslant [q_l + (l-2) \cdot q_l] + \cdots + [(h-1) \cdot q_l + h \cdot q_l]$$

$$\geqslant [1 \cdot q_2 + (l-2) q_{l-1}] + \cdots + [(h-1) \cdot q_h + h \cdot q_{h+1}]$$

$$= (l-2) \cdot q_{l-1} + (l-3) \cdot q_{l-2} + \cdots + 2 \cdot q_3 + 1 \cdot q_2.$$

The first inequality follows since $\frac{q_i}{q_1} \geqslant 1$ for $i = 2, 3, \ldots, l-1$. The second inequality follows since $q_l \geqslant q_i$ for $i = 2, 3, \ldots, l-1$.

When $l = 2h + 1$, the proof follows much as before, and we have

$$\frac{q_l}{q_1} \cdot 1 \cdot q_{l-1} + \frac{q_l}{q_1} \cdot 2 \cdot q_{l-2} + \cdots + \frac{q_l}{q_1} \cdot (l-3) \cdot q_3 + \frac{q_l}{q_1} \cdot (l-2) \cdot q_2$$

$$= \left[\frac{q_l}{q_1} \cdot 1 \cdot q_{l-1} + \frac{q_l}{q_1} \cdot (l-2) \cdot q_2\right] + \cdots + \left[\frac{q_l}{q_1} \cdot (h-1) \cdot q_{h+2} + \frac{q_l}{q_1} \cdot (h+1) \cdot q_h\right] + \frac{q_l}{q_1} \cdot h \cdot q_{h+1}$$

$$\geqslant [q_l + (l-2) \cdot q_l] + \cdots + [(h-1) \cdot q_l + (h+1) \cdot q_l] + h \cdot q_l$$

$$\geqslant [(l-2) q_{l-1} + 1 \cdot q_2] + \cdots + [(h+1) \cdot q_{h+2} + (h-1) \cdot q_h] + h \cdot q_{h+1}$$

$$= (l-2) \cdot q_{l-1} + (l-3) \cdot q_{l-2} + \cdots + 2 \cdot q_3 + 1 \cdot q_2. \qquad \square$$