Iranian Journal of Operations Research
Vol. 2, No. 2, 2011, pp. 1-16

Maximal covering location-allocation problem with M/M/k queuing system and side constraints

F. Moeen Moghadas[1*], H. Taghizadeh Kakhki[2]

We consider the maximal covering location-allocation problem with multiple servers. The objective is to maximize the population covered, subject to constraints on the number of service centers, total number of servers in all centers, and the average waiting time at each center. Each center operates as an M/M/k queuing system with variable number of servers. The total costs of establishing centers and locating servers should not exceed a predetermined amount. We present a mathematical model for the problem, and propose a heuristic solution procedure with two local search algorithms for improving the solutions. Finally, some computational results are presented.

Keywords: *Maximal covering location-allocation problem, M/M/k queuing system, Heuristic algorithm.*

Manuscript received on 27/07/2009 and Accepted for publication on 28/08/2010.

1. Introduction

Stochastic location problems can be divided into three main streams: median, center and covering-type models. The first median-type location model with mobile servers is the stochastic queue model (SQM), introduced by Berman et al. [7]. This model was extended to locate a facility with *k* servers by Batta and Berman [5]. A review of other extensions of the SQM is given by Berman and Krass [8]. Wang et al. [25] considered the problem of locating fixed facilities in which the objective was to minimize customers' total travel and waiting costs. They imposed a restriction on the allowable expected waiting time at a facility. Berman and Drezner [6] generalized the model proposed by Wang et al. [25] allowing the facilities to act as an *M/M/k* systems with each customer selecting the closest center.

Boffey et al. [10] considered locating a single service center operating as an $M/E_r/1/N$ queuing system. Marianov et al. [15] considered an extension of this problem for $M/E_r/m/N$ systems with a constraint on the probability of a costumer being lost.

A second type of stochastic location model is the center location model. A model of this type is the multiple-sever center location problem introduced by Aboolian et al. [1], in which the objective is to minimize the maximum time spent by any customer, including travel time and waiting time, at a service site.

Aboolian et al. [2] considered an application of this problem to a Web Service Provider (WSP). The problem was to find the location of facilities, the number of servers at each facility, and customer allocation to a new WSP competing with a competitor that provides the same service.

The third type of stochastic location models is the covering location model. The queuing maximal availability location problem is a model of this type which has been proposed by Marianov and ReVelle [16] in which each facility operates as an *M/G/s/s* queuing system.

* Corresponding author.
[1] Department of Mathematics, Ferdowsi University of Mashhad, Vakil Abad Blvd., Mashhad 91775-1159, Iran.
Email: fo_mo59@stu-mail.um.ac.ir
[2] Department of Mathematics, Ferdowsi University of Mashhad, Vakil Abad Blvd., Mashhad 91775-1159, Iran.
Email: taghizad@math.um.ac.ir

Marianov and Serra [18] considered a probabilistic maximal covering model with constraints on waiting time and queue length. They modeled the problem for both a single server facility, and for multi-server facilities, *m* servers for each facility. An extension with each facility acting as an *M/M/1* system with finite capacity was considered by Marianov and Rios [17]. This model was used for locating switches in ATM's (Automatic Teller Machines). Marianov and Serra [19] considered the location set covering problem with constraints on queue length and on waiting time. They studied models with fixed and variable number of servers at each service center. Marianov [14] considered location of *p* centers, each with *m* servers, in order to maximize the total expected demand over a region. Correa et al. [11] proposed a clustering search for the probabilistic maximal covering location-allocation problem with an *M/M/1* system and with constraints on waiting time or queue length. Silva and Serra [24] presented a maximal covering location problem for priority queues. A GRASP type heuristic procedure was proposed to solve this problem. Shavandi and Mahlooji [23] considered a fuzzy queuing location model for congested systems. They utilized fuzzy set theory to develop a queuing maximal covering location-allocation model.

Here, we consider a covering-type model having fixed facilities. The objective is to maximize the population covered. We allow one or more servers in each facility. Each demand point should be serviced only by one service center and there are some constraints on the number of centers and servers in each center and on the total number of servers that can be placed in each center. Demand points should be assigned to centers so that the average waiting time in each center does not exceed a predetermined amount. In addition there is a constraint, on the total cost of establishing centers and locating servers.

The remainder of the paper is organized as follows. In Section 2, we define the notations used throughout the paper and present a mathematical formulation for the problem. In section 3, we first develop a heuristic algorithm, and then present two local search algorithms to improve the solution. Computational results and some concluding remarks are presented in Sections 4 and 5.

2. Model Formulation

In order to formulate the problem, we use the following notations:

I : the set of all existing demand points $\left(|I|=n\right)$

J : the set of all candidate centers $\left(|J|=m\right)$

p: the maximum number of new centers

a_i : population at point i

N_i : the set of candidate centers that are within a covering radius from i; *i.e.,* $N_i=\{j\in J: d(i,j)\le R\}$, where R is the covering radius, and $d(i,j)$ is the Euclidean distance between node i and candidate center j

$\bar{k}$: maximum number of servers in all centers

$k_{\max}$: maximum number of servers in each center

τ_j : maximum allowable waiting time at center j

Fs_j : cost of placing a server at center j

Fc_j : cost of placing a service center at candidate location j

C : maximum allowable cost for establishing centers and locating servers

λ_j : arrival rate at service center j

μ_j : service rate for service center j

$W(\lambda,\mu,k)$: average waiting time at a service center with arrival rate λ, service rate μ and with k

servers.

We also define variables as follows:
k_j : the number of servers at center j
y_j : 1 if a new service center is located at site $j \in J$; and 0, otherwise
x_{ij} : 1 if a call from point i is answered by center j; and 0, otherwise.

Now, the problem can be stated as follows:

$$P:\ Max \sum_{i\in I}\sum_{j\in N_i} a_i x_{ij}$$

$$s.t. \quad \sum_{j\in N_i} x_{ij} \le 1 \qquad \forall i \in I \tag{1}$$

$$x_{ij} \le y_j \qquad \forall i \in I, j \in N_i \tag{2}$$

$$\sum_{j\in J} y_j \le p \tag{3}$$

$$\sum_{j\in J} k_j \le \bar{k} \tag{4}$$

$$k_j \le k_{\max} y_j \qquad \forall j \in J \tag{5}$$

$$W(\lambda_j, \mu_j, k_j) \le \tau_j \qquad \forall j \in J \tag{6}$$

$$\sum_{j\in J} Fs_j k_j + \sum_{j\in J} Fc_j y_j \le C \tag{7}$$

$$x_{ij}, y_j \in \{0,1\} \qquad \forall\ i \in I\ ,\ j \in J \tag{8}$$

$$k_j = 0,1,...,k_{\max} \qquad \forall j \in J. \tag{9}$$

In this model, the objective is to maximize the population covered. Constraints (1) ensure that each demand point i is allocated to at most one service center j. Constraints (2) ensure that point i is only serviced by an established center j. Constraints (3) establish at most p centers, and constraints (4) limit the number of servers at all selected facilities. Constraints (5) ensure that at most $k_{\max}$ servers are placed at each center constraints (6) guarantee that the average waiting time at each center does not exceed a predetermined amount, τ_j. Constraint (7) ensures that the total costs of locating service centers and placing servers at centers does not exceed a given amount C.
If we assume that the arriving calls from demand points are *Poisson* processes with intensity f_i, then λ_j can be shown to be given by $\lambda_j = \sum_{i\in I\,:\, j\in N_i} f_i x_{ij}$ (Marianov and Serra [18]).

It is also known that for an *M/M/k* queuing system with given λ, μ and k, the average waiting time can be calculated as follows (Kleinrock [12]):

$$W(\lambda,\mu,k) = \begin{cases} \dfrac{\mu\left(\lambda/\mu\right)^k}{(k\mu-\lambda)^2(k-1)!}\left(\dfrac{1}{k!}\left(\lambda/\mu\right)^k\left(\dfrac{k\mu}{k\mu-\lambda}\right)+\sum_{r=0}^{k-1}\dfrac{1}{r!}\left(\lambda/\mu\right)^r\right)^{-1}, & \text{if } \dfrac{\lambda}{k\mu}<1 \\ +\infty, & \text{otherwise.} \end{cases}$$

Note that for the stability condition to hold, we must have $\frac{\lambda}{k\mu} < 1$.

Pasternack and Drezner [22] showed that if λ/μ is large, then the above formula can be written as:

$$W(\lambda,\mu,k) = \frac{\lambda}{(k\mu-\lambda)^2\left(a_k + \frac{\lambda}{k\mu-\lambda}\right)} \tag{10}$$

where,

$$a_1 = 1 \quad , \quad a_i = 1 + \frac{\mu}{\lambda}(i-1)\, a_{i-1} \qquad i \geq 2.$$

This relation is also used by Berman and Drezner [6] and Aboolian et al. [1] to approximate the waiting time. In Section 4, we will show that by selecting a suitable μ, λ/μ can be made large for a large $\bar{k}$, which enables us to use (10), as well. Substituting (10) for (6) makes the problem highly nonlinear, and for large k we are only able to solve small instances with commercial software packages such as CPLEX.

Note that for fixed values of μ and k, $W(\lambda,\mu,k)$ is a strictly increasing function of λ. If $\lambda_{\max}$ is the value of λ which makes the constraint $W(\lambda,\mu,k) \leq \tau$ binding, then $\lambda \leq \lambda_{\max}$ is a linear equivalent of $W(\lambda,\mu,k) \leq \tau$, as shown by Marianov and Serra [18] (see also Marianov and Serra [19], and Aboolian et al. [2]). However, here we propose another approach to handle the waiting time constraint without a need to know the exact value of $\lambda_{\max}$. We then propose an algorithm based on decomposing the problem into smaller sub-problems, similar to the solution methods suggested in the literature for the capacitated *p*-median problem (see e.g., Baldacci et al. [4], Lorena and Senne [13]). Each sub-problem has only one constraint. The procedure starts with an approximate value for $\lambda_{\max}$ and a starting solution for *P*. It then tries to improve this approximate solution until a stopping criterion is met.

3. Heuristic Procedure for Solving Problem *P*

Before presenting the algorithm, we state and prove a lemma.

Lemma 3.1. If $W(\lambda,\mu,k)$ is defined by (10) and if $(k.k!\,\mu^k - \lambda^k) > 0$, for λ, μ and k, then

$$W(\lambda,\mu,k) \leq \frac{\lambda^k}{\mu(k.k!\,\mu^k - \lambda^k)}. \tag{11}$$

Proof. See the appendix.

Now, let $G(\lambda,\mu,k) = \frac{\lambda^k}{\mu(k.k!\,\mu^k - \lambda^k)}$. It is not difficult to see that G is an increasing function of λ as $\frac{\partial}{\partial\lambda} G(\lambda,\mu,k) > 0$. Also, let λ_G be that value of λ for which $G(\lambda_G,\mu,k) = \tau$. Then, $\lambda \leq \lambda_G$ and $G(\lambda,\mu,k) \leq \tau$ are equivalent and we have $\lambda_G = \left(\frac{k\,.\,k!\;\mu^{k+1}\tau}{1+\mu\tau}\right)^{\frac{1}{k}} = \mu\,(k\,.\,k!)^{\frac{1}{k}}\left(\frac{\mu\tau}{1+\mu\tau}\right)^{\frac{1}{k}}$.

On the other hand, for fixed values of μ and k, if λ satisfies $\lambda \le \lambda_G$ and $\left(k.k!\,\mu^k - \lambda^k\right) > 0$, then $W(\lambda,\mu,k) \le G(\lambda,\mu,k) \le \tau$. This means that λ is feasible for $W(\lambda,\mu,k) \le \tau$, or equivalently λ satisfies the constraint $\lambda \le \lambda_{\max}$. Now, if $W(\lambda_G,\mu,k) \le \tau$, then λ_G can be used as a starting solution for the algorithm. Otherwise, we need to construct an initial solution.

3.1. Heuristic Algorithm

For any candidate center *j* with *t* servers, we define sub-problem $v_j(t)$ as follows:

$$
\begin{aligned}
v_j(t): \quad z\ (v_j(t)) = Max \quad & \sum_{i \in I: j \in N_i} a_i x_{ij} \\
s.t. \quad & W(\lambda_j, \mu_j, t) \le \tau_j \\
& x_{ij} \in \{0,1\} \qquad\qquad \forall\ i \in I, j \in N_i,
\end{aligned}
$$

where $\lambda_j = \sum_{i \in I: j \in N_i} f_i x_{ij}$, and $W(\lambda_j,\mu_j,t)$ is given by (10).

Suppose that we have determined the location of $q-1$ $(1 \le q \le p)$ centers and want to determine the location of the $_{qth}$ center from among the candidate locations. If we remove the demand points covered by the previous $q-1$ centers, constraints (1) would be satisfied for any new facility. In addition, if we ignore the constraints (2)-(5) and (7), and consider them only implicitly, then the problem can be decomposed into smaller sub-problems $v_j(k_j)$. Details of the algorithm are shown in Fig. 1.

If $j \in J$ is selected as a service center with k_j servers, then it is added to J^*, and any demand point assigned to *j* is removed from *I*. Define $\overline{I_j}$ as the set of demand points *i* in the neighborhood of *j*, i.e., $j \in N_i$, which are not yet assigned to a center. If a demand point *i* $(i \in \overline{I_j})$ is assigned to *j*, then it is removed from $\overline{I_j}$ and added to I_j^*.

The q^{th} center is chosen from among candidate centers which have the maximum value of $z\ (v_j(t))\ (\forall\ j \notin J^*)$. This process is continued until either $q > p$, or the number of servers, $\sum_{j \in J^*} k_j$, is larger than $\overline{k}$, or all demand points are covered ($I = \phi$), or the total cost exceeds *C*.

Here, $k_{\max}^q$ is the maximum number of servers that can be located at q^{th} center. The number of servers at center *j*, t_j, the value of objective function, $z(v_j(t_j))$, and the set of demand points that are assigned to center *j*, I_j^*, are determined in lines (4)-(17). In line (7), if the first condition is not satisfied, then there is no demand point that can be covered by *j*. If the second condition does not hold, then increasing *t* will not improve the objective value. If the third condition is not satisfied, then *j* has the maximum number of servers, and we cannot increase *t*. If the fourth condition does not hold, then increasing *t* causes the total cost to exceed *C*.

(1) *Set* $q=1$, $J^*=\phi$, $\vec{K}=(k_1,k_2,...,k_m)=\vec{0}$, $I=\{1,2,...,n\}$, *cost=0.*

(2) *While* $(q \le p$ *and* $\sum_{j\in J^*} k_j < \bar{k}$ *and* $I \ne \phi$ *and* $cost \le C)$ *do*

(3) $k_{\max}^q = \min\left\{k_{\max}, \bar{k} - \sum_{r\in J^*} k_r\right\}$

(4) *For (any* $j \notin J^*$ *s.t.* $cost + (Fs_j + Fc_j) \le C)$ *do*

(5) *Set* $M_0^j = 0$, $\overline{I_j} = \{\, i \in I : j \in N_i$, $i \notin I_r^*$ $\forall$ $r \in J^* \}$, $I_j^* = \phi$

(6) *t=1, solve* $v_j(t)$ *and let* $M_t^j = z(v_j(t))$

(7) *While* $(\overline{I_j} \ne \phi$ *and* $M_t^j > M_{t-1}^j$ *and* $t < k_{\max}^q$ *and* $cost + (t.Fs_j + Fc_j) \le C)$ *do*

(8) *t=t+1. compute* $M_t^j = z(v_j(t))$

(9) *If* $M_t^j > M_{t-1}^j$ them

(10) *Remove any node assigned to j from* $\overline{I_j}$

(11) *End* if

(12) *End while*

(13) *If* $(M_t^j \le M_{t-1}^j$ *or* $cost + (t.Fs_j + Fc_j) > C)$ *then*

(14) *t=t-1*

(15) *End if*

(16) *Let* $M_j^* = M_t^j$, $t_j = t$ *and update* I_j^*

(17) *End for*

(18) $j^* = \arg\{\max\{M_j^* : j \notin J^*\}\}$, $k_{j^*} = t_{j^*}$

(19) $J^* = J^* \cup \{j^*\}$, $I = I \setminus I_{j^*}^*$, $cost = cost + (k_{j^*} Fs_{j^*} + Fc_{j^*})$

(20) $q = q + 1$

(21) *End while*

Figure1. A heuristic algorithm for solving problem *P*

3.1.1 Solving Sub-Problem $v_j(t)$

For *t*=1 or *t*=2, $W(\lambda,\mu,t) \le \tau$ is linear, as mentioned earlier, and the problem can be easily solved. For $t \ge 3$, let $\lambda_{\max}^j$ and $\lambda_G^{t,j}$ be the values of λ that satisfy $W(\lambda_{\max}^j, \mu_j, t) = \tau_j$ and $G(\lambda_G^{t,j}, \mu_j, t) = \tau_j$, respectively. If $W(\lambda_G^{t,j}, \mu_j, t) \le \tau_j$, then we have $\lambda_G^{t,j} \le \lambda_{\max}^j$; Let us define $\overline{\lambda_j} := \lambda_G^{t,j}$. Else, we have $\lambda_G^{t,j} > \lambda_{\max}^j$. Let s_1 be a given step size, and decrease $\lambda_G^{t,j}$ by s_1. If the feasibility condition is not satisfied at $\lambda_G^{t,j} - s_1$, then again decrease $\lambda_G^{t,j}$ by setting $s_1 \leftarrow 2s_1$, and continue this process until $\overline{\lambda_j} := \lambda_G^{t,j} - s_1$ satisfies $W(\overline{\lambda_j}, \mu_j, t) \le \tau_j$, so that we can replace

$W(\lambda_j, \mu_j, t) \le \tau_j$ by $\lambda \le \overline{\lambda_j}$ and solve $v_j(t)$.

We define the sub-problem $u_j(t,s)$, with a given step size s, similar to $v_j(t)$ but with $W(\lambda_j, \mu_j, t) \le \tau_j$ replaced by $\lambda \le \overline{\lambda_j} + s$:

$$u_j(t,s): \quad z(u_j(t,s)) = Max \quad \sum_{\substack{i \in I \\ j \in N_i}} a_i x_{ij}$$

$$s.t. \quad \sum_{\substack{i \in I \\ j \in N_i}} f_i x_{ij} \le \overline{\lambda_j} + s$$

$$x_{ij} \in \{0,1\} \qquad \forall\, i \in I,\ j \in N_i .$$

If $s = 0$, then $u_j(t,s)$ gives a feasible solution for $v_j(t)$. Suppose $s > 0$. If a solution for $u_j(t,s)$ is feasible for $v_j(t)$, then we check the next solution with $s \leftarrow s + \frac{s}{2}$. Otherwise, we set $s \leftarrow s - \frac{s}{2}$. This process is continued until the number of iterations is larger than *MaxItr*, or the solution does not improve after *Maxcount* iterations. Fig. 2 shows the steps of the algorithm for solving $v_j(t)$ $(t \ge 3)$.

Let $\lambda_G^{t,j}$ *be the value of* λ *that satisfies* $G(\lambda_G^{t,j}, \mu_j, t) = \tau_j$.

(1) If $W(\lambda_G^{t,j}, \mu_j, t) \le \tau_j$ *then*
(2) Let $\overline{\lambda_j} = \lambda_G^{t,j}$
(3) Else
(4) Define s_1 *as a step size and let key=0*
(5) While key ==0 do
(6) Let $\overline{\lambda_j} = \lambda_G^{t,j} - s_1$
(7) If $W(\overline{\lambda_j}, \mu_j, t) \le \tau_j$ then
(8) key=1
(9) Else
(10) $s_1 \leftarrow 2s_1$
(11) End if
(12) End while
(13) End if
(14) Let (X_1, f_1) *= solution of* $u_j(t,0)$
(15) Define s as a step size and set Itr=1 and count=0
(16) While (Itr $\le$ *MaxItr and count<Maxcount) do*
(17) Let (X_2, f_2) *= solution of* $u_j(t,s)$
(18) If X_2 *is feasible for* $v_j(t)$ *then*
(19) $= ss + \frac{s}{2}$
(20) Else
(21) $= ss - \frac{s}{2}$, $f_2 = 0$

(22) End if
(23) If $f_1 < f_2$ *then*
(24) $f_1 \leftarrow f_2$, $X_1 \leftarrow X_2$
(25) Else
(26) count=count+1
(27) End if
(28) Itr=Itr+1
(29) End while
(30) Return X_1, f_1.

Figure 2. A heuristic algorithm for solving sub-problem $v_j(t)$ $(t \geq 3)$

To solve the knapsack sub-problem $u_j(t,s)$, we use a branch-and-bound algorithm (Balas [3]) with Dantzig's upper bounds (see Martello and Toth [21]) .

3.2. Local Search

Here, we propose two local search procedures to improve the solutions obtained using the heuristic. In the first local search procedure (local search 1), we try to transfer *t* servers $(1 \leq t \leq k_j)$ from a selected center *j* to a center *r* such that the value of the objective function is improved. Center *r* can be chosen from previous selected facilities, or from candidate facilities not yet selected. Local search should be performed so that center r gives the maximum benefit among all candidate centers that can accept *t* additional servers. Fig. 3 shows the details of local search 1.

(1) For all $j \in J^*$ *do*
(2) Set $r_{\max} = j$, $f_{\max} = 0$, $t_{\max} = 0$
(3) For $t = 1 : k_j$ *do*
(4) For any facility $r \in J$ *(with* $k_r = 0$ *or* $k_r > 0$ *) do*
(5) If $(k_r + t \leq k_{\max})$ *and* $(z(v_r(k_r + t)) - z(v_r(k_r))) > f_{\max}$
and $(z(v_r(k_r + t)) - z(v_r(k_r))) > (z(v_j(k_j)) - z(v_j(k_j - t)))$ *then*
(6) $f_{\max} = z(v_r(k_r + t)) - z(v_r(k_r))$
(7) $r_{\max} = r$, $t_{\max} = t$
(8) End if
(9) End for
(10) End for
(11) If $r_{\max} \neq j$ *then*
(12) $k_{r_{\max}} = k_{r_{\max}} + t_{\max}$
(13) If $r_{\max} \notin J^*$ *then*
(14) $J^* = J^* \cup \{r_{\max}\}$
(15) End if
(16) End if
(17) End for

Figure 3. A procedure for local search 1

Note that $z(v_j(k_j))$ (the value of the objective function $v_j(k_j)$) is zero if $k_j = 0$. If r is among centers not yet selected and if $t = k_j$, then j is replaced by r and if $t < k_j$, and local search should be carried out so that the number of selected facilities does not exceed p.

In local search 2, we try to distribute t servers $\left(2 \le t \le k_j\right)$ j among two service centers r_1 and r_2 so that the objective function is improved. As before, r_1 and r_2 may be chosen from previous selected centers or from the centers not yet selected; or one may be from selected centers and the other from centers not yet selected.

4. Computational Results

We first compare the solutions obtained using the heuristic algorithm with that of enumerative method for two small size problems with 5 and 10 points, taken from the 30-node problem of Marianov and Serra [18]. We assumed that f_i and τ_j were the same for each demand point and for each candidate center, and were defined as 0.005 times the population and 10 minutes, respectively. Covering radius (R) was taken to be 2.5 and average service times were 10 and 7.5 minutes for the 5 and 10 point problems, respectively $(\mu = 6, \mu = 8)$, Fs_j and Fc_j were set to 100 and 200, for all candidate centers, and C was set to 600 . The algorithms were coded in MATLAB 6.5 and all computations were performed on a Pentium: IV processor with 2.80 GHz and 2.50 GB of RAM.

These problems were solved for three cases, (1) selecting one center with one server, (2) selecting at most two centers each with one server, and (3) selecting one center with at most two severs. The results are shown in Table 1.

Table 1. Comparison of the results for enumerative method and the heuristic

n	$\bar{k}=1$, $k_{\max}=1$, $p=1$		$\bar{k}=2$, $k_{\max}=1$, $p=2$		$\bar{k}=2$, $k_{\max}=2$, $p=1$	
	Enumerative method	Heuristic algorithm	Enumerative method	Heuristic algorithm	Enumerative method	Heuristic algorithm
5	50.57	50.57	100	100	100	100
10	61.34	57.98	100	100	100	100

In order to test the efficiency of the algorithm we have tried to solved some randomly generated problems. Data for other problems were taken from OR-library (Beasley [9]) for the p-median and covering problems.

In all examples, the distances were considered to be Euclidean and covering radii were defined to be 2.5 for 100 and 200 points, 250 for 324 points, and 1000 for other problems. The number of candidate centers, m, was taken to be equal to the number of demand points, n; i.e., each demand point was also assumed to be a candidate facility location.

f_i and τ_j were the same for all demand points and all candidate centers. The daily call rate, f_i, was taken to be 0.05 times the population, and maximum waiting time in each candidate center, τ_j,

was set to be 10 minutes.

We also assumed that the average costs of establishing a service center and locating servers were the same for all candidate centers, and were 200 and 100, respectively. The parameters were set as follows: $C = 2500$, $MaxItr = 50$, $Maxcount = 20$, $s = 10$ and $s_1 = 5$.

Note that defining $\bar{\lambda}$ as $\bar{\lambda} = \sum_{i \in I} f_i$ (total arrival rate), and μ as $\mu = \frac{\bar{\lambda}}{\bar{k} - 1}$, would yield: $\bar{\lambda}/\mu = \bar{k} - 1$ which is large for a large value of $\bar{k}$; hence, the stability condition $\frac{\bar{\lambda}}{\bar{k}\mu} < 1$, is satisfied.

Table 2 shows the results for values of $k_{\max}$ equal to 1, 3 and 5, and for different values of $\bar{k}$. In each case, first the percentage of population covered without a local search, and then solutions with two local searches 1 and 2 are shown. The hyphen, "-", means that the local search cannot improve the solution. For $k_{\max} = 1$, since each service center can only have one server, we have $p = \bar{k}$. For $k_{\max} = 3$ and $k_{\max} = 5$, p was set to be 3. For cases marked with a*, p was set to be 5.

In Table 3, solutions with Fs_j set equal to 150 are shown. The CPU column shows the maximum CPU time for a given $k_{\max}$. The first row presents maximum CPU consumed time without using a local search, the second row with local search1, and the third row with local search 2.

The followings are the main results:

(1) Comparison of the results for enumerative method and the proposed heuristic (table 1) shows that 100% of population is covered in cases (2) and (3). The total cost of establishing centers and locating servers in three cases are 300, 600 and 400, respectively. For C less than 600, we obtain the same solutions for cases (1) and (2). Therefore, selecting one service center with at most two servers provides a better solution.

(2) In almost all cases with fixed values of $k_{\max}$ and $\bar{k}$, increasing p improves the solution. The same is true for fixed values of p and $\bar{k}$, but with different values of $k_{\max}$.

(3) The results in tables 2 and 3 indicate that for fixed values of $k_{\max}$ and p, increasing $\bar{k}$ improves the solutions most of the time. But in some cases, percentages of population covered do not change, and sometimes even decrease. Note that by increasing $\bar{k}$, μ decreases; hence, the system can not cover more demand points.

(4) Comparison of the results in tables 2 and 3 shows that increasing Fs_j, results in a decrease in the percentage of population covered.

(5) Using a local search can improve the solutions, but does not guarantee optimality. For instance, in Table 3, 93.83% coverage is obtained for $n = 100$, with $k_{\max} = 3$ and $\bar{k} = 15^*$. This is bigger than the results obtained with $k_{\max} = 5$ and $\bar{k} = 15^*$.

Table 2. Computational results for single and multi-server systems with $Fs_j = 100$

Number of nodes	$k_{\max}=1$						$k_{\max}=3$					$k_{\max}=5$					
	$\bar{k}=3$	$\bar{k}=5$	$\bar{k}=10$	$\bar{k}=15$	$\bar{k}=20$	CPU	$\bar{k}=3$	$\bar{k}=5$	$\bar{k}=10$	$\bar{k}=15^*$	CPU	$\bar{k}=5$	$\bar{k}=10$	$\bar{k}=15$	$\bar{k}=15^*$	$\bar{k}=20^*$	CPU
100	78.01	74.20	34.86	16.40	9.62	0.6	77.42	77.42	92.42	96.53	1.2	77.42	92.42	92.42	97.24	96.88	2.4
	-	-	34.89	16.46	-	1.2	78.39	90.31	-	-	1.8	90.31	-	-	-	98.00	3.0
	-	-	-	-	-	0.6	85.52	90.10	-	-	4.8	90.10	-	-	-	-	4.8
200	78.57	82.22	49.43	25.34	15.51	2.5	71.46	81.47	81.47	94.13	4.3	81.47	81.47	81.47	93.12	93.12	7.4
	-	-	49.45	-	-	4.9	77.54	-	-	-	6.5	-	-	-	-	-	9.3
	-	-	-	-	-	2.5	82.14	-	-	-	36.5	-	-	-	-	-	15.5
324	23.14	36.23	51.66	47.08	33.30	5.0	23.14	23.14	23.14	36.23	5.7	23.14	23.14	23.14	36.23	36.23	7.0
	-	-	-	-	33.32	8.8	-	-	-	-	8.3	-	-	-	-	-	9.5
	-	-	-	-	-	5.0	-	-	-	-	14.0	-	-	-	-	-	19.2
402	87.46	94.63	76.47	45.59	31.33	12.8	83.10	83.10	92.26	100	64.9	83.10	92.26	92.26	100	100	113.5
	-	-	-	-	-	24.3	87.50	87.50	-	-	75.5	87.50	-	-	-	-	125.8
	-	-	-	-	-	12.8	92.39	96.85	-	-	1106.0	96.85	-	-	-	-	1087.8
500	87.22	92.29	73.48	43.08	29.18	19.7	87.22	87.22	84.52	94.71	73.8	87.22	87.22	87.22	99.48	99.48	99.5
	-	-	-	-	-	39.9	-	-	-	-	85.6	-	-	-	-	-	108.9
	-	-	-	-	-	19.8	-	88.65	-	-	90.3	88.65	-	-	-	-	154.3
708	80.28	90.67	80.24	48.97	34.32	44.5	80.28	80.28	80.28	93.73	85.5	80.28	80.28	80.28	93.73	93.73	177.1
	-	-	-	-	-	86.7	-	-	-	-	111.2	-	-	-	-	-	216.2
	-	-	-	-	-	44.5	-	-	-	-	135.0	-	-	-	-	-	333.5
818	64.77	82.38	79.89	53.17	38.22	60.5	64.77	64.77	64.77	85.56	96.1	64.77	64.77	64.77	85.56	85.56	128.4
	-	-	-	-	-	125.7	-	-	-	-	131.6	-	-	-	-	-	172.3
	-	-	-	-	-	60.6	-	-	-	-	172.5	-	-	-	-	-	368.4

* problems solved with p=5

Table 3. Computational results for single and multi-server systems with $Fs_j = 150$

Number of node	$k_{max}=1$						$k_{max}=3$					$k_{max}=5$					
	$\bar{k}=3$	$\bar{k}=5$	$\bar{k}=10$	$\bar{k}=15$	$\bar{k}=20$	CPU	$\bar{k}=3$	$\bar{k}=5$	$\bar{k}=10$	$\bar{k}=15^*$	CPU	$\bar{k}=5$	$\bar{k}=10$	$\bar{k}=15$	$\bar{k}=15^*$	$\bar{k}=20^*$	CPU
100	78.01	74.20	30.58	14.29	8.45	0.6	77.42	77.42	92.42	93.83	1.2	77.42	92.42	92.42	92.42	92.42	2.2
	-	-	30.67	14.35	-	1.2	78.39	90.31	-	-	1.5	90.31	-	-	93.39	92.51	3.2
	-	-	-	-	-	0.6	85.52	90.10	-	-	4.6	90.10	-	-	-	-	46.80
200	78.57	82.22	43.26	22.19	13.58	2.2	71.46	81.47	81.47	88.76	3.9	81.47	81.47	81.47	84.63	87.76	6.3
	-	-	-	-	-	4.3	77.54	-	-	-	5.9	-	-	-	84.85	-	8.6
	-	-	-	-	-	2.2	82.14	-	-	-	36.5	-	-	-	-	-	11.9
324	23.14	36.23	47.15	41.66	29.20	4.0	23.14	23.14	23.14	36.23	5.6	23.14	23.14	23.14	36.23	36.23	6.9
	-	-	-	-	29.21	8.2	-	-	-	-	8.0	-	-	-	-	-	9.8
	-	-	-	-	-	4.0	-	-	-	-	14.1	-	-	-	-	-	17.7
402	87.46	94.63	66.91	39.89	27.41	11.4	83.10	83.10	92.26	100	63.7	83.10	92.26	92.26	100	96.12	108.7
	-	-	-	-	-	22.2	87.50	87.50	-	-	73.3	87.50	-	-	-	96.15	124.5
	-	-	-	-	-	11.5	92.39	96.85	-	-	1092.5	96.85	-	-	-	-	130.3
500	87.22	92.29	64.30	37.69	25.53	17.4	87.22	87.22	84.52	89.48	70.3	87.22	87.22	87.22	87.22	90.87	95.4
	-	-	-	-	-	34.0	-	-	-	-	78.8	-	-	-	-	-	104.3
	-	-	-	-	-	17.5	-	88.65	-	-	75.6	88.65	-	-	-	-	125.8
708	80.28	90.67	70.22	42.85	30.03	38.8	80.28	80.28	80.28	87.74	75.8	80.28	80.28	80.28	88.98	88.98	167.3
	-	-	-	-	-	75.7	-	-	-	91.93	115.5	-	-	-	-	-	202.7
	-	-	-	-	-	38.9	-	-	-	92.32	9399.7	-	-	-	-	-	272.8
818	64.77	82.38	71.82	46.52	33.44	54.7	64.77	64.77	64.77	75.77	83.5	64.77	64.77	64.77	75.77	74.80	94.5
	-	-	-	53.17	-	121.5	-	-	-	-	139.6	-	-	-	-	-	155.8
	-	-	-	-	-	54.8	-	-	-	78.05	16625.	-	-	-	78.05	-	16735.

* problems solved with p=5

5. Conclusion

We considered the maximum covering location-allocation problem with an *M/M/k* queuing system with some constraints on the number of centers, total number of servers and on the number of servers in each center. Additional constraints were also considered on the average waiting time in each center, on the total costs of establishing service centers, and on locating servers. We first presented a mathematical model and noted that when the number of servers increased, the problem became highly nonlinear and was difficult to solve. To solve the problem, we proposed a heuristic procedure based on decomposing the problem into smaller sub-problems. Two local search algorithms were considered to improve the solutions. Finally, some problems with random data, as well as problems adopted from the literature were solved to test the efficiency of the proposed algorithms. The results indicated that an improvement in percentage of population covered could be obtained using the local search procedures, but this of course, at the expense of higher consumed CPU times.

Appendix

First, consider the following proposition.

Proposition. For any k $(k \geq 1)$, a_k can be calculated as follows:

$$a_k = \sum_{t=0}^{k-1} \frac{(k-1)!}{(k-t-1)!} \frac{\mu^t}{\lambda^t}.$$

Proof. We use induction.

For *k*=1, $a_1 = \sum_{t=0}^{0} \frac{(k-1)!}{(k-t-1)!} \frac{\mu^t}{\lambda^t} = 1$. Now, consider $a_k = \sum_{t=0}^{k-1} \frac{(k-1)!}{(k-t-1)!} \frac{\mu^t}{\lambda^t}$, for a given *k*, and show $a_{k+1} = \sum_{t=0}^{k} \frac{(k)!}{(k-t)!} \frac{\mu^t}{\lambda^t}$.

By definition of a_k, we have,

$$a_{k+1} = 1 + k\frac{\mu}{\lambda}(a_k) = 1 + k\frac{\mu}{\lambda}\left(\sum_{t=0}^{k-1} \frac{(k-1)!}{(k-t-1)!} \frac{\mu^t}{\lambda^t}\right)$$

$$= 1 + \sum_{t=0}^{k-1} \frac{k!}{(k-t-1)!} \frac{\mu^{t+1}}{\lambda^{t+1}} = 1 + \sum_{t=1}^{k} \frac{k!}{(k-t)!} \frac{\mu^t}{\lambda^t} = \sum_{t=0}^{k} \frac{k!}{(k-t)!} \frac{\mu^t}{\lambda^t} \ . \square$$

Proof of Lemma 3.1.

For *k*=1 and *k*=2 and if stability conditions ($\frac{\lambda}{\mu} < 1$ and $\frac{\lambda}{2\mu} < 1$, respectively) hold, we have

$$W(\lambda, \mu, 1) = \frac{\lambda}{(\mu - \lambda)^2\left(1 + \frac{\lambda}{\mu - \lambda}\right)} = \frac{\lambda}{(\mu - \lambda)(\mu - \lambda + \lambda)} = \frac{\lambda}{\mu(\mu - \lambda)} = \frac{\lambda^1}{\mu(1 \times 1! \mu^1 - \lambda^1)},$$

$$W(\lambda, \mu, 2) = \frac{\lambda}{(2\mu - \lambda)^2\left(1 + \frac{\mu}{\lambda} + \frac{\lambda}{2\mu - \lambda}\right)} = \frac{\lambda^2}{(2\mu - \lambda)\mu(2\mu + \lambda)} = \frac{\lambda^2}{\mu(2 \times 2! \mu^2 - \lambda^2)}.$$

Thus, the equality holds for these two cases. Now, for $k \geq 3$, $W(\lambda, \mu, k)$ can be written as:

$$W(\lambda,\mu,k)=\frac{\lambda}{(k\mu-\lambda)^2\left(\sum_{t=0}^{k-1}\frac{(k-1)!}{(k-t-1)!}\frac{\mu^t}{\lambda^t}+\frac{\lambda}{k\mu-\lambda}\right)}=\frac{\lambda}{(k\mu-\lambda)^2\left(\sum_{t=0}^{k-1}\frac{\frac{(k-1)!}{(k-t-1)!}\mu^t}{\lambda^t}+\frac{\lambda}{k\mu-\lambda}\right)}$$

$$D=(k\mu-\lambda)^2\left(\frac{\sum_{t=0}^{k-1}\frac{(k-1)!}{(k-t-1)!}\mu^t\lambda^{k-t-1}(k\mu-\lambda)+\lambda.\lambda^{k-1}}{\lambda^{k-1}(k\mu-\lambda)}\right)\qquad (k\mu-\lambda)\neq 0.$$

Then,

$$D=\frac{(k\mu-\lambda)}{\lambda^{k-1}}\left(\sum_{t=0}^{k-1}\frac{k!}{(k-t-1)!}\mu^{t+1}\lambda^{k-t-1}-\sum_{t=0}^{k-1}\frac{(k-1)!}{(k-t-1)!}\mu^t\lambda^{k-t}+\lambda^k\right)$$
$$=\frac{(k\mu-\lambda)}{\lambda^{k-1}}\left(\sum_{t=0}^{k-1}\frac{k!}{(k-t-1)!}\mu^{t+1}\lambda^{k-t-1}-\sum_{t=1}^{k-1}\frac{(k-1)!}{(k-t-1)!}\mu^t\lambda^{k-t}\right),$$

which implies:

$$D=\frac{1}{\lambda^{k-1}}\Big(\sum_{t=0}^{k-1}\frac{k.k!}{(k-t-1)!}\mu^{t+2}\lambda^{k-t-1}-\sum_{t=0}^{k-1}\frac{k!}{(k-t-1)!}\mu^{t+1}\lambda^{k-t}-$$
$$\sum_{t=1}^{k-1}\frac{k!}{(k-t-1)!}\mu^{t+1}\lambda^{k-t}+\sum_{t=1}^{k-1}\frac{(k-1)!}{(k-t-1)!}\mu^t\lambda^{k-t+1}\Big).$$

Setting aside the terms for $t=k-1$ and $t=k-2$ from the first sum, $t=0$ and $t=k-1$ from the second (one), $t=k-1$ from the third, and $t=1$ from the fourth, yields:

$$D=\frac{1}{\lambda^{k-1}}\left[\sum_{t=2}^{k-1}\left(\frac{k.k!}{(k-t+1)!}-\frac{2k!}{(k-t+1)!}+\frac{(k-1)!}{(k-t-1)!}\right)\mu^t\lambda^{k-t+1}+\right.$$
$$\left.+\left(\frac{\mathrm{k.k!}}{0!}\mu^{k+1}+\frac{k.k!}{1!}\mu^k\lambda^1\right)-2\left(\frac{k!}{0!}\mu^k\lambda^1\right)+\frac{(k-1)!}{(k-2)!}\mu\lambda^k\right]-\frac{k!}{(k-1)!}\mu\lambda^k.$$

By rearranging the terms, *D* can be written as:

$$D=\frac{1}{\lambda^{k-1}}\left[\sum_{t=2}^{k-1}\left(\frac{(k-2)k!}{(k-t+1)!}+\frac{(k-1)!}{(k-t+1)!}\right)\mu^t\lambda^{k-t+1}+\mu^{\mathrm{k}}\lambda^1(k.k!-2k!)+\mu\lambda^k((k-1)-k)\right.$$
$$\left.+k.k!\,\mu^{k+1}\right]=$$
$$=\frac{1}{\lambda^{k-1}}\left[\sum_{t=2}^{k-1}\left(\frac{(k-2)k!}{(k-t+1)!}+\frac{(k-1)!}{(k-t+1)!}\right)\mu^t\lambda^{k-t+1}+(k-2)k!\,\mu^k\lambda+\mu(k.k!\,\mu^k-\lambda^k)\right]$$
$$\geq\frac{1}{\lambda^{k-1}}\left[\mu(k.k!\,\mu^k-\lambda^k)\right].$$

And finally, if $k.k!\,\mu^k-\lambda^k>0$, then $\mathrm{W}(\lambda,\mu,\mathrm{k})\leq\frac{\lambda}{\frac{1}{\lambda^{k-1}}\left[\mu(k.k!\,\mu^k-\lambda^k)\right]}=\frac{\lambda^k}{\mu(k.k!\,\mu^k-\lambda^k)}$. □

Acknowledgement

The authors express their gratitude to the referees for their useful comments and suggestions.

References

[1] Aboolian, R., Berman, O. and Drezner, Z. (2009), The multiple server center location problem, *Annals of Operations Research,* 167(1), 337-352.

[2] Aboolian, R., Sun, Y. and Koehler, G.J. (2009), Production, manufacturing and logistics, a location–allocation problem for a web services provider in a competitive market, *European Journal of Operational Research*, 194, 64-77.

[3] Balas, E. (1965), An additive algorithm for solving linear programs with zero-one variables, *Operations Research*, 13(4), 517-546.

[4] Baldacci, R., Hadjiconstantinou, E., Maniezzo, V. and Mingozzi, A. (2002), A new method for solving capacitated location problems based on a set partitioning approach, *Computers & Operations Research*, 29(4), 365-386.

[5] Batta, R. and Berman, O. (1989), A location model for a facility operating as an M/G/k queue, *networks*, 19, 717-728.

[6] Berman, O. and Drezner, Z. (2007), The multiple server location problem, *Journal of the Operational Research Society*, 58, 91-99.

[7] Berman, O., Larson, R.C. and Chiu, S.S. (1985), Optimal server location on a network operating as an M/G/1 queue, *Operations Research*, 33 (4), 746-771.

[8] Berman, O. and Krass, D. (2004), Facility location problems with stochastic demands and congestion. In: Drezner, Z. and Hamacher, H.W. (Eds.), Facility Location: Application and Theory, Springer, pp. 329-371.

[9] Beasley, J.E. (1990), OR-Library: distributing test problems by electronic mail, *Journal of the Operational Research Society*, 41(11), 1069-1072.

[10] Boffey, B., Galvao, R. and Marianov, V. (2009), Location of single-server immobile facilities subject to a loss constraint, *Journal of the Operational Research Society*, 61, 987-999.

[11] Corrêa, F.D.A., Chaves, A.A. and Lorena, L.A.N. (2008), Hybrid heuristics for the probabilistic maximal covering location-allocation problem, *Operational Research, An International Journal*, 7(3), 323-344.

[12] Kleinrock, L. (1975), *Queuing Systems: Theory,* Vol. 1, John Wiley & Sons, New York.

[13] Lorena, L.A.N. and Senne, E.L.F. (2003), Local search heuristic for capacitated *p*-median problems, *Networks and Spatial Economics*, 3(4), 407-419.

[14] Marianov, V. (2003), Location of multiple-server congestible facilities for maximizing expected demand, when services are non-essential, *Annals of Operations Research*, 123, 125-141.

[15] Marianov, V., Boffey, T.B. and Galvao, R.D. (2008), Optimal location of multi-server congestible facilities operating as M/ E_r /m/N queues, *Journal of the Operational Research Society*, 60(5), 674-684.

[16] Marianov, V. and ReVelle, C. (1996), Theory and methodology: the queuing maximal availability location problem: A model for the siting of emergency vehicles, *European Journal of Operational Research*, 93, 110-120.

[17] Marianov, V. and Rios, M. (2000), A probabilistic quality of service constraint for a location model of switches in ATM communications networks, *Annals of Operations Research*, 96, 237-243.

[18] Marianov, V. and Serra, D. (1998), Probabilistic, maximal covering location-allocation models for congested systems, *Journal of Regional Science*, 38(3), 401-424.
[19] Marianov, V. and Serra, D. (2002), Location–allocation of multiple-server service centers with constrained queues or waiting times, *Annals of Operations Research*, 111, 35-50.
[20] Marianov, V. and Serra, D. (2003), Location models for airline hubs behaving as M/D/c queues, *Computers & Operations Research*, 30, 983-1003.
[21] Martello, S. and Toth, P. (1990), Knapsack Problems, *Algorithms and Computer Implementations,* John Wiley & Sons, New York.
[22] Pasternack, B.A. and Drezner, Z. (1998), A note on calculating steady state results for an M/M/k queuing system when the ratio of the arrival rate to the service rate is large, *Journal of Applied Mathematics & Decision Sciences*, 2(2), 201-203.
[23] Shavandi, H. and Mahlooji, H. (2006), A fuzzy queuing location model with a genetic algorithm for congested systems, *Applied Mathematics and Computation*, 181, 440-456.
[24] Silva, F. and Serra D. (2008), Locating emergency services with different priorities: the priority queuing covering location problem, *Journal of the Operational Research Society*, 59, 1229-1238.
[25] Wang, Q., Batta, R. and Rump, C.M. (2002), Algorithms for a facility location problem with stochastic customer demand and immobile servers, *Annals of Operations Research*, 111, 17-34.