Annals of Operations Research 111, 17–34, 2002

Algorithms for a Facility Location Problem with Stochastic Customer Demand and Immobile Servers

QIAN WANG, RAJAN BATTA and CHRISTOPHER M. RUMP *
Department of Industrial Engineering, University at Buffalo, State University of New York, 342 Lawrence D. Bell Hall, Buffalo, NY 14260-2050, USA

Abstract. This paper studies a facility location problem with stochastic customer demand and immobile servers. Motivated by applications to locating bank automated teller machines (ATMs) or Internet mirror sites, these models are developed for situations in which immobile service facilities are congested by stochastic demand originating from nearby customer locations. Customers are assumed to visit the closest open facility. The objective of this problem is to minimize customers' total traveling cost and waiting cost. In addition, there is a restriction on the number of facilities that may be opened and an upper bound on the allowable expected waiting time at a facility. Three heuristic algorithms are developed, including a greedy-dropping procedure, a tabu search approach and an ε-optimal branch-and-bound method. These methods are compared computationally on a bank location data set from Amherst, New York.

Keywords: facility location, queueing, heuristics, tabu search, branch and bound

1. Introduction and motivation

In this paper we consider the problem of locating automated teller machines (ATMs), Internet mirror sites, or other immobile (permanent location) service systems of limited service capacity. Typically, such systems process a variety of service requests (e.g., bank deposits, cash withdrawals, fund transfers, etc.) that require random amounts of service time. These service requests arrive at random points in time, originating from customers who usually travel to the closest facility location.

We model these service facilities as simple M/M/1 queueing systems. In locating these facilities, this paper considers the customer perspective, i.e., a desire to limit the time spent traveling to and waiting at a site. For balance, both of these components of the customer's objective are also separately constrained. Customers are assumed to travel to the closest open facility. Further, a constraint is placed on the maximum expected waiting time at any open facility.

The model presented in this paper is related to work in the area of facility location and server allocation. Particularly relevant is a recent paper on the optimal placement of web proxies in the Internet by Li et al. [14]. Here, a dynamic programming algorithm is developed to find the minimum cost (in terms of latency, hops, etc.) location for web proxies, where demand flows along a tree rooted by the original web site. Each proxy

* Corresponding author.

location services the demand emanating from its subtree. Thus, this model differs from ours in that the demand on a facility is not necessarily from the closest customers, but rather those children below on its subtree.

Also closely related is the flow-capturing model (or discretionary service facility) introduced independently in Hodgson [12] and Berman et al. [5]. The flow-capturing problem is motivated by applications such as locating gas stations, convenience stores and automatic teller machines [3]. Some variations of this problem include application to billboard locations [13]. Here, like our problem, the service systems have permanent locations on a transportation network. In [2] these servers may be congested. The problem assumes that there exist customer flows along pre-planned paths in the network, and a customer will possibly utilize a facility only if the facility is on or close to his pre-planned path. In contrast, our model assumes that a customer seeks out the closest service node for the sole purpose of receiving service rather than as a diversion from their pre-defined travel plans. Hence, our objectives are not ones of maximizing server utilization through demand capture, but rather ones of balancing service costs against service quality, as measured through travel and service time delays.

The stochastic queue median (SQM) model due to Berman et al. [4] considers a mobile server such as an emergency response unit. Like other models for siting emergency services [16], customers (e.g., patients) are located at nodes and request on-scene service from the service provider (e.g., ambulance). In response to each demand call, the server (if available) travels to the demand to provide on-scene service. When not busy, the server is located at a point on the network (either at a node or on the interior of a link). Although stochastic in nature, the SQM problem is much different from our problem due to the fact that the server is mobile, whereas in our problem there are many immobile server locations.

The literature in siting emergency services also includes many coverage models [16]. Through probabilistic constraints, these models attempt to ensure a minimum level of server availability within a time standard of each demand node. Extending the probabilistic location set covering problem (PLSCP) [19], Marianov and ReVelle [15] developed the Q-PLSCP, which models each geographic region as a multi-server queueing loss system. These nonlinear chance constraints utilize region-specific server utilization ratios, which can be incorporated the optimization model through numerical deterministic equivalents. In a similar way, we include a linear constraint that ensures that the nonlinear expected waiting time objective does not exceed a permissible level.

Server allocation models have also been studied in the manufacturing domain [20]. In these models, there already exists a pre-defined network of queues representing work center locations. Each work center has a fixed workload based on pre-defined random customer flows. The problem is to optimally allocate a number of servers to the work centers so that the throughput of the queueing network is maximized. Some variations of this problem include [6] and [10]. Since the queueing network (save for the number of servers at each station) is pre-defined, these models ignore the location problem, the subject of our paper.

Although these models are similar to our problem, they do not address the problem of locating servers in service systems (e.g., an ATM system) in which server locations are not pre-defined, servers cannot move, and customers travel to the closest available server rather than along a pre-defined path or a tree. We refer to our class of models as facility location problems with stochastic customer demand and immobile servers.

The rest of this paper is organized as follows. The next section introduces our model. Some preliminary analysis is presented in section 3. Three heuristics are then developed in section 4. Computational testing of these heuristics is performed in section 5. We close with a summary of our conclusions.

2. Model

Consider a service system in which the server is immobile and the customer must travel to a facility to receive service. We make the following general assumptions: (i) customers go to the closest open facility to receive service, (ii) requests for service from each customer node form an independent Poisson stream, (iii) each open facility site has only one server with exponential service times, (iv) there is an upper bound on the permissible expected waiting time of customers.

To model this situation, we denote:

- $M = \{1, 2, \ldots, m\}$: set of customer nodes;
- $N = \{1, 2, \ldots, n\}$: set of potential facility nodes;
- $D = (d_{ij})$: distance matrix from customer node i to facility node j;
- Λ: total demand rate of service requests in the system;
- λ_i: demand rate of service requests from customer node $i \in M$;
- γ_j: demand rate at open facility site $j \in N$;
- μ: common service rate for each server;
- $W_j = (\mu - \gamma_j)^{-1}$: expected waiting time of customers assigned to facility node $j \in N$;
- $\overline{W}$: upper bound on the permissible expected waiting time for customers;
- $\nu = 1/\overline{W}$: surplus service capacity to insure $W_j \leqslant \overline{W}$;
- p: number of facilities actually opened;
- $\bar{p}$: maximum number of facilities that can be opened.

The problem can be stated as follows: given a customer set with demand rates denoted by λ_i's, a potential facility location set with common service capacities (rates) μ, a positive integer $\bar{p}$, and a positive number $\overline{W}$, find a set of facility locations consisting of no more than $\bar{p}$ facilities that minimizes the expected total number of customers traveling to and waiting at their closest facility such that their expected waiting time at each open facility is not greater than $\overline{W}$.

Let v be the travel speed and

$$y_j = \begin{cases} 1 & \text{if a facility is opened at node } j, \\ 0 & \text{otherwise;} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{if customer } i \text{ is assigned to facility } j, \\ 0 & \text{otherwise.} \end{cases}$$

Then, the aggregate travel time of customers per unit time is

$$T = \sum_{i \in M} \sum_{j \in N} \lambda_i d_{ij} x_{ij} / v.$$

Since each open facility behaves as an M/M/1 queue, the expected waiting time at open facility site j is $W_j = 1/(\mu - \gamma_j)$ where $\gamma_j = \sum_{i \in M} \lambda_i x_{ij}$. Thus, the aggregate waiting time of customers per unit time is

$$V = \sum_{i \in M} \sum_{j \in N} \lambda_i x_{ij} W_j = \sum_{j \in N} \frac{\gamma_j}{\mu - \gamma_j}.$$

By Little's Law [11], T represents the average number of traveling customers and V represents the average number of waiting customers.

To guarantee that the customer goes to his or her closest open facility site, we require

$$\sum_{k \in N} d_{ik} x_{ik} \leqslant (d_{ij} - \Delta) y_j + \Delta, \quad \forall i \in M, j \in N,$$

where Δ is a large positive number (e.g., $\Delta = \max\{d_{ij}\colon i \in M, j \in N\}$). When $y_j = 0$, this constraint is not in effect because Δ is large. When $y_j = 1$, customer i cannot be assigned to a facility which is farther than j, otherwise, the constraint will be violated.

We arrive at the following mathematical programming formulation:

$$\text{(P)} \quad \min \sum_{i \in M} \sum_{j \in N} \lambda_i x_{ij} \left(\frac{d_{ij}}{v} + \frac{1}{\mu - \sum_{k \in M} \lambda_k x_{kj}} \right),$$

$$\text{subject to} \quad \sum_{j \in N} y_j \leqslant \bar{p}, \tag{1}$$

$$\sum_{j \in N} x_{ij} = 1, \qquad \forall i \in M, \tag{2}$$

$$x_{ij} \leqslant y_j, \qquad \forall i \in M, j \in N, \tag{3}$$

$$\sum_{k \in N} d_{ik} x_{ik} \leqslant (d_{ij} - \Delta) y_j + \Delta, \quad \forall i \in M, j \in N, \tag{4}$$

$$\sum_{i \in M} \lambda_i x_{ij} \leqslant \mu - v, \qquad \forall j \in N, \tag{5}$$

$$y_j \in \{0, 1\}, \quad x_{ij} \in \{0, 1\}, \qquad \forall i \in M, j \in N.$$

Here, the objective is to minimize the sum of the expected traveling and waiting customers, T and V, subject to a constraint (1) on the maximum number of facilities that may be opened. Constraints (2) and (3) ensure that each customer demand is assigned, and only to an open facility. Constraint (4) ensures that this assignment is done to the closest facility. Finally, constraint (5) ensures that the expected waiting time at any facility does not exceed $\overline{W}$.

Remark 1. Removing the queueing component, V, from the objective function yields a capacitated facility location problem (CFLP) with a closest-facility constraint (4) [17]. Therefore, any algorithm for CFLP with a closest-facility rule, such as Lagrangian relaxation, can be used to solve this somewhat easier problem.

Remark 2. Notice that constraint (1) is an inequality rather than an equality as in other problems such as the well-known p-median problem. The reason for this is that it may be desirable to open fewer than $\bar{p}$ facilities. For example, it may be desirable to close a heavily congested facility if that facility's demand splits off to several other slightly farther away but less-congested sites.

Factors that contribute to the hardness of problem (P) include the nonlinear objective function, the large number of constraints, and the binary nature of the x_{ij} variables. Intuitively, we can make the following general observations:

- There is a conflict between T and V; if we want to decrease V, T may be increased and vice versa;
- For a given value of $\bar{p}$, both T and V decrease when the value of μ increases;
- For a given value of μ, both T and V decrease when the value of $\bar{p}$ increases.

3. Preliminary analysis

In analyzing this problem (P), notice that the constraint of $\overline{W}$ limits the queueing component of the objective function, viz.

$$V = \sum_{j \in N} \frac{\gamma_j}{\mu - \gamma_j} \leqslant \overline{W} \sum_{j \in N} \gamma_j = \overline{W} \Lambda.$$

Thus, an upper-bound feasibility problem of (P) can be stated as: for a constant $T_{\max}$ and $\bar{p}$, does there exist an assignment of customers (constraints (2) and (3)) to no more than $\bar{p}$ open facilities (constraint (1)) according to the closest rule (constraint 4)) so that the each customer's expected waiting time is no more than $\overline{W}$ (constraint (5)) and the total weighted travel time is no more than $T_{\max}$? It is easy to show – by reducing the well-known set-covering problem (Garey and Johnson [9]) – that the feasibility problem of (P) is NP-complete, and therefore conclude that (P) is NP-hard.

The waiting time component can also be bounded from below, due to the following theorem.

Theorem 1. Consider a set P of open facilities, where each of the $p = |P|$ facilities has capacity μ. The expected number of waiting customers, $V = \sum_{k\in P} \gamma_k/(\mu - \gamma_k)$, aggregated over these p facilities is minimized over the constraint set $\sum_{k\in P} \gamma_k = \Lambda < p\mu$ and $0 \leqslant \gamma_k < \mu$, $k \in P$, at the point where the aggregate demand Λ is allocated equally to the p facilities, i.e., $\gamma_k = \Lambda/p$, $k \in P$.

Proof. The function V is the sum of strictly convex functions, and is therefore strictly convex as well. Thus, it achieves a global minimum over the variable constraint set at the solution to the Karush–Kuhn–Tucker (KKT) conditions, namely the solution where, due to the sharing of a common multiplier for the coupling constraint $\sum_{k\in P} \gamma_k = \Lambda$, each variable takes the same value, Λ/p. □

From theorem 1, the minimum value of V when there are p open facilities is

$$V^* = \sum_{k\in P} \frac{\Lambda/p}{\mu - \Lambda/p} = p\frac{\Lambda/p}{\mu - \Lambda/p} = \frac{p\Lambda}{p\mu - \Lambda}. \tag{6}$$

Since V^* is decreasing in p, setting $p = \bar{p}$ in (6) gives the smallest number of waiting customers that can be expected. This provides a valid lower bound on the value of V for any solution having at most $\bar{p}$ open facilities, a bound that we utilize in a Lagrangian relaxation ε-optimal algorithm developed in the next section. We also point out that this bound V^* is unreachable for most problems. Even if it is possible to equally allocate demand for some specially structured problem, it may lead to excessive travel.

Remark 3. Although we assume that the service rates are the same for each server, all the heuristics developed in this paper can be applied to the situation where the service rates are different for each server. The only consideration is for the lower bound on the number of waiting customers. When the values of μ_j depend on j, V^* is no longer a valid lower bound for the expected number of waiting customers V.

We now introduce some terminology that will be useful in strengthening the bound developed from theorem 1 in the next section. For every facility node $j \in N$, let $M_j = \{i \in M\colon d_{ij} = \min\{d_{ik}\colon k \in N\}\}$ denote the set of customers that will utilize this facility when open, regardless of the open/closed status of the other sites. Then $\ell_j = \sum_{i\in M_j} \lambda_i$ is the minimum customer arrival rate to facility node $j \in N$ once j is open. Therefore, if $\ell_j \geqslant (\mu - \nu)$, potential facility site j should be eliminated from the facility nodes set because if a server is located at this node the queue formed at this node would be overcongested. Hereafter, we assume that all such potential facility site nodes have already been eliminated.

4. Development and testing of heuristics

In this section, three heuristic approaches are developed for the problem. The first approach is a construction algorithm, the greedy-dropping heuristic. The second approach

is an improvement algorithm using tabu search. The third approach applies a branch-and-bound scheme using Lagrangian relaxation to find a lower bound. Since the feasibility of this problem is NP-complete, all the heuristics presented in this paper may fail to find a feasible solution for the problem.

4.1. Greedy-dropping heuristic

There are two kinds of construction heuristics for location problems. One is a greedy-adding algorithm that adds facilities sequentially. Another is a greedy-dropping algorithm that closes facilities sequentially. If we apply a greedy-adding algorithm, we generally use the objective function value as the criteria to select the next facility to open. For the problem studied in this paper, if we only open p or fewer facilities, where p satisfies $p\mu < \Lambda$, at least one of the queues at an open facility will be unstable, and therefore the objective function is infinity. Hence, we cannot use the objective function value as the criteria to select the next open facility. Therefore, we focus on a greedy-dropping heuristic instead.

Initially, we suppose all the facility sites are open. Then we select one facility to close, step by step. Generally speaking, after we close a facility, the objective function value will increase since the travel time increases and the waiting time may also increase. Therefore, at each iteration of this heuristic, we close the facility that increases the objective value the least. The procedure is repeated until one of the following stopping conditions is met:

1. No greater than $\bar{p}$ open facilities are left, and further closing any open facility provides a worse objective value than the current one;
2. Greater than $\bar{p}$ facilities are left, and any further closing causes the arrival rate to at least one open facility to exceed the upper bound $(\mu - \nu)$.

In case 1 the procedure stops at a feasible solution, whereas in case 2 the procedure stops at an infeasible solution.

Let Q be the set of nodes where open facilities are located, and let $F(Q)$ be the corresponding objective value. ($F(Q) = \infty$ when Q is infeasible.)

Greedy-dropping (GD) heuristic.

Step 1 (Initialization).
Start by opening all potential facility sites, i.e., set $Q = \{1, \ldots, n\}$.
Assign the m customer to the n facility sites according to the closest rule.

Step 2 (Stopping).
We greedily close the facilities until a stopping conditions is met:
Compute $\rho_j = F(Q \setminus \{j\}), \forall j \in Q$. Find $j_0 = \arg\min\{\rho_j, j \in Q\}$, where ties are broken arbitrarily.
If $|Q| \leqslant \bar{p}$ and $\rho_{j_0} \geqslant F(Q)$, STOP, a feasible solution Q has been found.
Else if $|Q| > \bar{p}$ and $\rho_{j_0} = \infty$, STOP, no feasible solution is found.
Else, set $Q \leftarrow Q \setminus \{j_0\}$, go to step 2.

The heuristic algorithm is illustrated by the following example.

Example 1. Consider a network with $m = 3$ customer nodes and $n = 4$ potential facility sites, each with service capacity $\mu = 5$ and reserve capacity $\nu = 1/\overline{W} = 1$, a desired $\bar{p} = 2$ open facilities, demand vector $\boldsymbol{\lambda} = (2 \quad 2 \quad 2)$, and demand-weighted distance matrix (with travel speeds $v = 1$)

$$\boldsymbol{\lambda} D = \begin{pmatrix} 1 & 2 & 5 & 6 \\ 1 & 5 & 2 & 6 \\ 2 & 3 & 4 & 1 \end{pmatrix}.$$

Iteration 1. Initially, open all the potential facility sites so that $Q = \{1, 2, 3, 4\}$. Hence, since customer 1 is closest to site 1, this customer will travel to facility node 1. Similarly (scanning subsequent rows of the distance matrix), customer 2 will be assigned to site 1, and customer 3 will travel to site 4. The total cost is $F(Q) = 7.\bar{6}$.

Now, compute the costs if we close a facility. Consider closing site 1 (removing the first column of the distance matrix). Customers 1 and 2 will now be assigned to sites 2 and 3, respectively. This creates a demand of 2 at sites 2, 3 and 4. Hence, $\rho_1 = 7$. Similar calculations yield $\rho_2 = 7.\bar{6}$, $\rho_3 = 7.\bar{6}$ and $\rho_4 = \infty$. Since closing the facility at node 1 provides the best objective value, the open facility set is now $Q = \{2, \ 3, \ 4\}$.

Iteration 2. The objective values are $\rho_2 = 12.\bar{6}$, $\rho_3 = 12.\bar{6}$ and $\rho_4 = 11.\bar{6}$ if we close a facility at nodes 2, 3, and 4, respectively. Since closing facility 4 leads to the best objective value, it is selected to be closed, leading to $Q = \{2, 3\}$.

Iteration 3. $|Q| = 2 = \bar{p}$ means a feasible solution has been found. Further closing any one of the facilities will lead to an infinite cost, so the procedure is stopped. $Q = \{2, \ 3\}$ is the solution obtained by the greedy-dropping heuristic with an objective function value of $11.\bar{6}$.

4.2. Tabu search approach

In the greedy-dropping heuristic, the algorithm stops at the first local optimum that it reaches (when $|Q| \leqslant \bar{p}$) or at an infeasible solution (when $|Q| > \bar{p}$). In order to have the opportunity to investigate local optima other than the first one, or to possibly find a feasible solution in the absence of one, we now modify the greedy-dropping heuristic by applying a tabu search technique.

To apply tabu search, we need an initial solution. The greedy-dropping heuristic is a way to generate the initial solution. First, apply the greedy-dropping heuristic. If a feasible solution is found, it will serve as the initial solution for tabu search. Otherwise, a greedy-adding heuristic based on the traveling component is applied. The general idea of the greedy-adding procedure is to start with an empty set and add facilities sequentially. The criteria for selecting a facility to add is to choose the one providing the largest decrease in traveling customers. The procedure terminates once $\bar{p}$ facilities are open. The solution obtained by this greedy adding procedure may be infeasible in the sense

of violating the waiting-time upper bound. However, it has a small traveling component and has no more than $\bar{p}$ open facilities.

After an initial solution is found, tabu search is then implemented. We define the search neighborhood $N(Q)$ of a solution Q to be the set of all possible pairwise facility swaps involving the swapping of one facility in the solution with one not in the solution. At each iteration, we select the facility swap that provides the best objective function value. If all feasible swaps lead to an infeasible solution, we select the one with the least number of facilities that have a waiting time violation. The procedure will not terminate even when no swap improves the objective function value. Rather, it continues until K successive swaps show no improvement in the objective function value, where K is a pre-defined positive integer. Furthermore, a tabu list is generated that classifies a subset of moves in a neighborhood as forbidden (in order to prevent the search from repeating swaps tried in the recent past).

To present the algorithm we introduce the following additional notation:

- Q^*: the best solution found so far;
- L: the length of the tabu list;
- K: the maximum number of successive non-improvement swaps permitted;
- $TL(i, j)$: an integer used to record the tabu status of facility pair (i, j).

Note that if $TL(i, j) > 0$, pair (i, j) is in the tabu list and cannot be used in the next $TL(i, j)$ iterations. Using the defined notation, we define the neighborhood of Q, i.e., the set of feasible facility swaps, as $N(Q) = \{(i, j)\colon i \in Q, j \in N \setminus Q, TL(i, j) = 0\}$. A brief description of the tabu search algorithm is as follows:

Tabu search (TS) approach.

Step 1 (Initialization).
Let Q be the solution generated by the greedy-dropping heuristic.
If $|Q| > \bar{p}$ (the greedy-dropping solution is infeasible), then starting from $Q = \emptyset$, use the greedy-adding heuristic (involving only traveling time) to find a set of facilities with $|Q| = \bar{p}$.
Set $Q^* = Q$, $k = 0$, and $TL(i, j) = 0, \forall i, j \in N$.

Step 2 (Choice and termination).
If $N(Q)$ is empty, STOP; otherwise, let (s, t) be the swap in $N(Q)$ that provides the best objective function value.
If all the swaps in $N(Q)$ are infeasible, then select the one with the least number of facilities that have a waiting-time violation.

Step 3 (Update).
If $F(Q \cup \{t\} \setminus \{s\}) < F(Q)$ (the swap is an improvement), then set $k \leftarrow 0$, $Q^* \leftarrow Q \cup \{t\} \setminus \{s\}$; otherwise, set $k \leftarrow k + 1$.
If $k = K$, then we have performed K successive swaps that do not improve the objective function value, STOP; otherwise, update $Q \leftarrow Q \cup \{t\} \setminus \{s\}$, $TL(s, t) =$

$TL(t, s) = L$, $TL(i, j) = \min\{TL(i, j) - 1, 0\}, \forall (i, j) \neq (s, t)$ or (t, s). Go to step 2.

Since the GD heuristic is a sub-procedure in the TS approach, the TS approach will always take more time than the greedy-dropping heuristic; however, it will always provide no worse a solution than the GD heuristic. We illustrate an improvement by the TS approach on the solution found by the GD heuristic in example 1.

Example 2. Consider the network in example 1. Let $L = 3$ and $K = 2$.

Iteration 0. Initially, $Q = \{2, 3\}$ is the solution generated by the greedy-dropping heuristic. Set $k = 0$, and $TL(i, j) = 0, \forall i, j \in N$.

Iteration 1. In $N(Q)$, swapping between sites 2 and 4 and between sites 3 and 4 provide the best objective function value of $12.\bar{6}$. Breaking the tie arbitrarily, we update $Q = \{2, 4\}$ and $TL(3, 4) = L = 3$. Since the objective function value is not improved, $k = 1$.

Iteration 2. Now, swapping sites 1 and 2 generates the best objective function value of $7.\bar{6}$. Since $7.\bar{6} < 11.\bar{6}$, $Q^* = \{1, 4\}$ and $k = 0$. Update $Q = \{1, 4\}$, $TL(1, 2) = 3$ and $TL(3, 4) = 2$.

Iteration 3. Swapping between sites 1 and 2 and between sites 1 and 3 provide the best objective function value of $12.\bar{6}$. Since $TL(1, 2) = 3 > 0$, the former swap is prohibited and is not included in $N(Q)$. Hence, we perform the latter swap, yielding $Q = \{3, 4\}$, $TL(1, 3) = 3$, $TL(1, 2) = 2$, $TL(3, 4) = 1$ and $k = 1$.

Iteration 4. The best non-tabu swap in $N(Q)$ is between sites 2 and 3 with objective function value of $12.\bar{6}$. k is updated to 2 and therefore the procedure stops. $Q^* = \{1, 4\}$ is the solution obtained by the tabu search approach with an objective function value of $7.\bar{6}$.

4.3. Lagrangian relaxation algorithm

When the greedy-dropping heuristic and the tabu search approach are applied, no information about the solution quality is provided. In this subsection, a branch-and-bound scheme based on Lagrangian relaxation is developed to generate an ε-optimal solution to the problem (P). Thus, the "approximation" can be made as tight as desired, subject only to computation time limitations.

The upper bound for (P) is the objective value of the best solution found so far. The lower bounds for the traveling and waiting components are found separately. We will use V^* given by (6) as the lower bound of V. We obtain a lower bound on T by solving the following problem relaxation:

$$\text{(PT)} \quad \min \sum_{i \in M} \sum_{j \in N} \lambda_i x_{ij} d_{ij} / v,$$

subject to the constraints (1)–(3) and $y_j \in \{0, 1\}$, $0 \leqslant x_{ij} \leqslant 1, i \in M, j \in N$.

Remark 4. The solution of problem (PT) will always satisfy the closest assignment constraints (4) and provide the minimum number of the traveling customers. However, the solution of this relaxation may have a high number of waiting customers or even violate the waiting time constraints (5).

The above problem relaxation (PT) is a facility location problem ignoring set-up costs. (If (1) were constrained to be tight, then (PT) becomes the p-median problem for which Lagrangian relaxation techniques have been well studied [8]. See remark 2.) A lower bound for (PT) can be obtained by applying Lagrangian relaxation with respect to the assignment constraints (2). Lagrangian relaxation for a given set of multipliers u is:

$$\text{(PTLR)} \quad L(u) = \min \sum_{i\in M}\sum_{j\in N} \lambda_i d_{ij} x_{ij}/v + \sum_{i\in M} u_i \left(1 - \sum_{j\in N} x_{ij}\right),$$

subject to the constraints (1), (3) and $y_j \in \{0, 1\}$, $0 \leqslant x_{ij} \leqslant 1$, $i \in M$, $j \in N$.

The objective function of the Lagrangian relaxation can be rewritten as:

$$\sum_{i\in M}\sum_{j\in N} (\lambda_i d_{ij}/v - u_i) x_{ij} + \sum_{i\in M} u_i.$$

Define $a^- = \min\{0, a\}$ and let $\rho_j(u) = \sum_{i\in M} (\lambda_i d_{ij}/v - u_i)^-$, $j \in N$. Let $J \subset N$ be the subset of at most $\bar{p}$ facilities with the most negative $\rho_j(u)$'s. The optimal solution to the Lagrangian relaxation for a given u is then:

$$y_j = \begin{cases} 1, & j \in J, \\ 0, & \text{otherwise;} \end{cases} \tag{7}$$

$$x_{ij} = \begin{cases} y_j, & \lambda_i d_{ij}/v - u_i < 0, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

The corresponding Lagrangian dual is

$$\text{(PTLD)} \quad \max L(u).$$

The dual can be solved via the subgradient algorithm listed as follows:

Subgradient algorithm.

Step 1. Set $k = 1$ and initialize Lagrangian multipliers $u_i^1 = \min\{d_{ij}\colon\ j \in Q\}$ where Q is the best set of locations for open facilities found so far. Let β be the incumbent (the best objective function value of the Lagrangian dual found so far). Set $\beta = -\infty$.

Step 2. Find an optimal solution y_j^k, x_{ij}^k, $\forall i \in M$, $j \in N$, of the Lagrangian relaxation $L(u^k)$ via (7) and (8). Let $g_i = 1 - \sum_{j\in N} x_{ij}$, $\forall i \in M$. If $g_i = 0, \forall i \in M$, STOP (complementary slackness is satisfied); y_j^k, x_{ij}^k solves (PT).

Step 3. Update β if y_j^k, x_{ij}^k provides a better objective function value for the dual (PTLD).

Step 4. Define a step size α_k and update the Lagrangian multipliers by $u_i^{k+1} \leftarrow u_i^k + \alpha_k g_i, \forall i \in M$. Set $k \leftarrow k+1$. If k is greater than the number of iterations permitted, STOP; otherwise, go to step 2.

The Lagrangian relaxation algorithm for obtaining an ε-optimal solution is as follows:

Lagrangian relaxation (LR) algorithm.

Step 1 (Initial solution).
Solve the problem by the TS Approach to obtain an initial feasible solution and an upper bound.

Step 2 (Branch and bound).
Solve the problem (P) via a branch-and-bound technique.
A lower bound for (P) is the summation of the lower bounds on V and T, where the lower bound of V is V^*, and the lower bound of T is obtained by solving the Lagrangian dual (PTLD) via the subgradient algorithm.
Branch on the y_j with the largest customer demand. If z^* is the best objective function value of (P) found so far, a node with a lower bound no less than $z^*/(1+\varepsilon)$ will be pruned from the branch-and-bound tree since z^* is within $\varepsilon\%$ of any better solution value, $\hat{z}$ that may be pruned, i.e., $z^* \leqslant \hat{z}(1+\varepsilon)$.

The lower bound V^* given by (6) is rather weak and can be strengthened for use in the LR algorithm. To strengthen it, suppose that at some node of the branch-and-bound tree, $C = \{j \in N\colon\ y_j = 0\}$ is the set of facilities that are closed and will remain so upon further expansion of this node down the search tree. For every facility node $j \in N \setminus C$ (that may be potentially opened), redefine $M_j = \{i \in M\colon\ d_{ij} = \min\{d_{ik}\colon\ k \in N \setminus C\}\}$ as the set of customers that will utilize this facility when open, regardless of the open/closed status of the remaining sites in $N \setminus C$. Then $\ell_j = \sum_{i \in M_j} \lambda_i$, is the minimal customer demand to facility node $j \in N \setminus C$ once j is open. Among the potentially open nodes, let $P \subseteq N \setminus C$ be the subset that are definitely open at a node of the tree, i.e., $P = \{j \in N \setminus C\colon\ y_j = 1\}$.

The open facilities in P must receive their minimal demands, which may be more than an equitable split of the total demand as described in theorem 1. Let $R = \{j \in P\colon\ \ell_j > \Lambda/\bar{p}\}$ denote the subset of these open facilities that must receive more than an equitable split of the total demand Λ. Let $r = |R| \leqslant \bar{p}$ be the number of such facilities, and $\Gamma = \sum_{j \in R} \ell_j \leqslant \Lambda$ be their minimal aggregate demand. To strengthen the bound (6), we allocate these minimal demands to the facilities in R, and split the remaining demand

$\Lambda - \Gamma$ among at most $\bar{p} - r$ other facilities per the logic following theorem 1. This provides a valid lower bound on V for this node of the search tree given by

$$V^* = \frac{(\bar{p} - r)(\Lambda - \Gamma)}{(\bar{p} - r)\mu - (\Lambda - \Gamma)} + \sum_{j \in L} \frac{\ell_j}{\mu - \ell_j}. \tag{9}$$

Clearly, the computation time of the LR algorithm will be larger than that for the TS approach, since the latter is the first step of the LR algorithm. However, the benefit of the LR algorithm is that it always provides a solution that is no worse than the one provided by the TS, and it guarantees that the solution is within $\varepsilon\%$ of the optimum. We illustrate the procedure by continuing example 1.

Example 3. Consider example 1, where $\Lambda = 6$, $\mu = 5$, and $\bar{p} = 2$. The initial solution generated by the tabu search approach is $Q = \{1, 4\}$ with $F(Q) = 7.\bar{6}$. Thus, $UB(P) = 7.\bar{6}$. The lower bound on V is $V^*(P) = \bar{p}\Lambda/(\bar{p}\mu - \Lambda) = 3$. In order to get the lower bound on T, we apply the subgradient algorithm. The initial values of Lagrangian multipliers are $u_1^1 = u_2^1 = u_3^1 = 1$. $L(u^1) = 3$ with all y_j's and x_{ij}'s equal to 0. Since $T(Q) = 3$, we stop the subgradient algorithm and obtain $T^*(P) = 3$. Therefore, the lower bound of (P) is $LB(P) = V^*(P) + T^*(P) = 6$. Since all $y_j = 0$, breaking ties arbitrarily and branching on y_1 generates two sub-problems: $(P_1) = (P : y_1 = 0)$ and $(P_2) = (P : y_1 = 1)$.

For branch (P_1), $V^*(P_1) = 3$, $T^*(P_1) = 5$, and $LB(P_1) = 8 > F(Q)$. So, it is fathomed.

For branch (P_2), $V^*(P_2) = 4 + 2/3$ (via (9)), $T^*(P_2) \geqslant T^*(P) = 3$, and $LB(P_2) = 7.\bar{6} = UB$. Therefore, $Q = \{1, 4\}$ is the optimal solution.

5. Computational results

In this section, we test the performance of the heuristic algorithms. All the three heuristics were used to generate solutions for 90 problems. Test data are based on a network derived from the Town of Amherst, New York. It consists of 459 customer nodes and 84 potential facility sites. Heuristic algorithms were programmed in Microsoft Visual C++ 6.0. All the programs were run on a Dell OptoPlex GX1MTDS450 PC with 128 MB RAM and 450 MHz CPU. Computation times are in seconds.

In this computational experiment, two parameters are under study:

- the number of facilities to open $\bar{p}$, with three levels 7, 10, 15;
- the common service rate μ, with three levels 320, 340, 360.

Hence, we have a total of 3×3 treatments. For each treatment, ten problems are solved corresponding to ten distance matrices, which were generated via perturbing on the Euclidean distance between the customer nodes and the potential facility nodes. The

distances for each link were perturbed uniformly over the range (1.1, 1.4). The range for perturbation stems from the fact that road distances between two randomly selected points in the Town of Amherst tend to be mainly within 10–40% of the Euclidean distance between the points (based on some experimentation we performed).

The number of potential facility sites was chosen with a bank example in mind. Specifically, we identified commercially zoned land/properties at major street intersections in the town of Amherst. The values of $\bar{p}$ reflect the number of branches associated with the three largest banks in the Town of Amherst.

Since we do not have software to solve the mixed integer nonlinear programming problems, we use a branch-and-bound algorithm described in this section to find the optimal solution for the problems. For most of the problems run in the experiment, the branch-and-bound algorithm can find an optimal solution very quickly. However, for some problems in the set, it even takes a long time to generate an approximate solution. We have attempted to find the 10%, 5%, 2%, 1% and 0% (exact) approximate solutions for each problem. Once it takes more than 1,000 seconds to find the approximate solution for a certain percentage, we do not run the program for lower percentages. For example, if it takes more than 1,000 seconds to find a 5% approximate solution, we do not attempt to find 2%, 1% and 0% (exact) approximate solutions.

Based on initial experimentation with the TS approach, the length of the tabu list was set to $L = 3$, and the maximal number of non-improvement swaps allowed was set to $K = 5$. In the LR algorithm, the number of iterations permitted in the subgradient algorithm was set at 100. The step size α_k is defined by $\alpha_k = 40/(k \cdot \|g\|)$, where $\|g\| = (\sum_i g_i^2)^{1/2}$ is the norm of g, and $g_i = 1 - \sum_{j \in N} x_{ij}, \forall i \in M$.

The computation time of the algorithms is listed in table 1. Since we do not run the branch-and-bound algorithm for every problem, some groups may have incomplete information on computational times. In the table, we only list the computational times for those treatments in which every problem has been solved.

Table 1
The average computation time for the heuristics (seconds).

p	μ	GD	TS	BB(10)	BB(5)	BB(2)	BB(1)	BB
7	320	0.6	1.2	1035.6				
	340	0.5	1.4	2.5	6.4	346.7	1052.4	
	360	0.6	1.2	2.3	2340.8			
10	320	0.5	1.9	3.1	3.1	3.2	3.2	3.2
	340	0.5	1.8	3.0	4.8	1658.5		
	360	0.7	2.1	2.8	3	13.5	42	213.1
15	320	0.6	3.5	4.3	4.6	4.8	4.8	4.8
	340	0.5	4.4	5.5	5.5	5.6	5.6	5.6
	360	0.6	3.5	4.4	4.6	4.9	4.9	4.9
Average		0.567	2.333					

Table 2
The average errlb and errub values.

p	μ	errlb		errub	
		GD	TS	GD	TS
7	320	0.202	0.004	0.298	0.098
	340	0.151	0.005	0.155	0.009
	360	0.089	0.011	0.132	0.051
10	320	0.027	0	0.027	0
	340	0.028	0.007	0.040	0.020
	360	0.026	0.006	0.026	0.006
15	320	0.017	0	0.017	0
	340	0.021	0	0.021	0
	360	0.015	0	0.015	0

From table 1, we can see that

- On average: (i) the TS approach takes over four times as long as the GD heuristic; (ii) the LR algorithm takes more time than the TS approach; and (iii) the smaller the value of ε, the longer the LR algorithm.
- The CPU time of the GD heuristic is not significantly affected by the different values of p and μ. The CPU time of the TS approach increases in p, but is not significantly affected by the different μ values. The reason for this is that when p is larger, the number of possible swaps in each iteration of the interchange phase is larger.
- The branch-and-bound scheme is efficient for finding the optimal solution for problems with large p and μ values, i.e., with low server utilization rate. However, it may take much more time for the situation with high server utilization rate. The reason for this might be that the lower bound V^* is not good under this situation.

Since we cannot find optimal solutions for some problems within a reasonable time, we do not know the exact error of the GD and the TS heuristics on the objective function values of these problems. Instead, we list the upper bound (errub) and lower bound (errlb) for the errors corresponding to those unsolved problems. Let VLR(ε) be the objective value found by the LR algorithm for the $\varepsilon\%$ solution. The errub and errlb are defined as follows:

$$\text{errub} = \frac{\text{value found by heuristics} - (1-\varepsilon\%)\text{VLR}(\varepsilon)}{(1-\varepsilon\%)\text{VLR}(\varepsilon)},$$

$$\text{errlb} = \frac{\text{value found by heuristics} - \text{VLR}(\varepsilon)}{\text{VLR}(\varepsilon)},$$

where ε is the lowest ε value attempted. For example, if we have found a 2% approximate solution with more than 1,000 seconds, then we will not run it for 1%. Hence, the ε used in the equations would be 2. From the definitions of errub and errlb, we can see that errub is the most conservative estimate of the error, and errlb is the most optimistic estimate of the error. The exact error would fall in the interval of [errlb, errub].

Table 3
The maximum errlb and errub values.

	GD	TS
errlb	0.603	0.032
errub	0.781	0.142

Table 4
The maximum and minimum errors.

	GD	TS
Maximum error	0.0580	0.0147
Minimum error	0	0

As we mentioned before, the heuristics may fail to find a feasible solution. Among the 90 problems attempted, the GD heuristic only failed to find feasible solutions for five problems. Both the TS approach and the LR algorithm found feasible solution for all 90 problems. The average values of errlb and errub for each treatment are listed in table 2. The maximum errlb and errub values for the GD and the TS are listed in table 3. For those problems for which an optimal solution is found, the maximum and minimum errors for the GD and the TS are listed in table 4.

From tables 2–4, we can see that for the problems for which we have found optimal solutions (generally, these problems have large p and μ values, i.e., they have low server utilization rates which are proportional to $1/(p\mu)$), the errlb equals errub since $\varepsilon = 0$. The maximum error between the objective value found by the GD heuristic and the optimal value for these problems is 5.8%, whereas the minimum error is 0. The maximum error between the objective value found by the TS approach and the optimal value is 1.47%, whereas the minimum error is 0. From this we can conclude that the performance of the TS is more robust than that of the GD heuristic. With a little more computational time, TS always finds a better solution than the GD heuristic. For the problem with large p and μ values, it always generates an optimal solution.

For those problems in which we failed to find an optimal solution (generally, problems with high server utilization rates), the value of errlb is less than the value of errub. Since table 3 shows that the maximum errub of the TS is 14.2% (obtained from a problem which is only solved for a 10% approximate solution) and the maximum errlb of the TS is 3.2%, we can conclude that the TS approach is still a good heuristic for these instances. However, the performance of the GD heuristic for this situation is much worse.

6. Summary and future work

This paper focused on a facility location problem with stochastic customer demand and immobile servers. The service rates for all the servers were assumed the same, and the objective was to minimize the number of traveling and waiting customers. Three

heuristics were developed to solve this problem. Numerical experiments have been run to examine the quality of the heuristics, and the results were reported.

Based on the results of numerical experiments we can conclude that the performance of the greedy-dropping heuristic is not as good as the tabu search approach and the Lagrangian relaxation algorithm. For a few problems, the greedy heuristic generates a solution more than 20% away from the optimal value.

Although the Lagrangian relaxation ε-optimal algorithm can guarantee the quality of the solution generated, for some problems, especially those problems with high server utilization rates, it may take much more time than the tabu search approach. possible extensions to the basic model were presented. The reason behind this may be that the lower bound that we developed for the expected number of waiting customers is not very tight when the server utilization rate is high. The fact that a separate lower bound on traveling customers is found independently of the waiting bound via the problem (PT) may also be a source of weakness. One referee has suggested solving a problem like (PT) with a waiting time objective in order to improve the waiting bound. We leave this interesting idea for future study.

Acknowledgments

We wish to thank the anonymous referees and editors of this special issue for their many helpful comments and suggestions for improving this paper.

References

[1] I. Averbakh and O. Berman, Locating flow-capturing units on a network with multicounting and diminishing returns to scale, European Journal of Operational Research 91 (1996) 495–506.

[2] O. Berman, The maximizing market-size discretionary facility location problem with congestion, Socio-Economic Planning Sciences 29(1) (1995) 39–46.

[3] O. Berman, M.J. Hodgson and D. Krass, Flow-interception problems, in: *Facility Location: A Survey of Applications and Methods*, ed. Z. Drezner, Springer Series in Operations Research (Springer, New York, 1995).

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

[5] O. Berman, R.C. Larson and N. Fouska, Optimal location of discretionary service facilities, Transportation Science 26 (1992) 201–211.

[6] O. Boxma, J. Kan, A. Rinnooy and M. van Vliet, Machine allocation problems in manufacturing networks, European Journal of Operational Research 45 (1990) 47–54.

[7] M.L. Brandeau and S.S. Chiu, A unified family of single-server queueing location models, Operations Research 38 (1990) 1034–1044.

[8] M.S. Daskin, *Network and Discrete Location: Models, Algorithms, and Applications* (Wiley, New York, 1995).

[9] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, New York, 1979).

[10] L. Green and D. Guha, Note: On the efficiency of imbalance in multi-facility multi-server service systems, Management Science 41 (1995) 179–187.

[11] D. Gross and C.M. Harris, *Fundamentals of Queueing Theory* (Wiley, New York, 1998).
[12] M.J. Hodgson, A flow-capturing location-allocation model, Geographical Analysis 22 (1990) 270–279.
[13] M.J. Hodgson and O. Berman, A billboard location model, Geographical and Environmental Modeling 1 (1997) 25–43.
[14] B. Li, M.J. Golin, G.F. Italiano, X. Deng and K. Sohraby, On the optimal placement of web proxies in the Internet, in: *Proceedings of the INFOCOM '99* (1999) pp. 1282–1290.
[15] V. Marianov and C. ReVelle, The queueuing probabilistic location set covering problem and some extensions, Socio-Economic Planning Sciences 28 (1994) 167–178.
[16] V. Marianov and C. ReVelle, Siting emergency services, in: *Facility Location: A Survey of Applications and Methods*, ed. Z. Drezner, Springer Series in Operations Research (Springer, New York, 1995).
[17] P.B. Mirchandani and R.L. Francis, *Discrete Location Theory* (Wiley, New York, 1990).
[18] S.Y. Prasad and R. Batta, Efficient facility locations on a tree network operating as a FIFO M/G/1 queue, Networks 23 (1993) 597–603.
[19] C. ReVelle and K. Hogan, The maximum reliability location problem and α-reliable p-center Problem: Derivatives of the probabilistic location set covering problem, Annals of Operations Research (1989) 155–174.
[20] J.G. Shanthikumar and D.D. Yao, Optimal server allocation in a system of multi-server stations, Management Science 33 (1987) 1173–1191.
[21] Q. Wang, Discrete facility location design with stochastic customer demand and immobile servers, Ph.D. dissertation, University at Buffalo, State University of New York (2000).