THE UNIVERSITY OF TEXAS AT SAN ANTONIO, COLLEGE OF BUSINESS

Working Paper SERIES

May 29, 2008

Wp# 0050MSS-061-2008

A Tabu Search Heuristic Procedure for the Capacitated Facility Location Problem

Minghe Sun

College of Business
The University of Texas at San Antonio
San Antonio, TX 78249-0634
(210) 458-5777
msun@utsa.edu
http://faculty.business.utsa.edu/msun

Department of Management Science & Statistics,
University of Texas at San Antonio,
San Antonio, TX 78249, U.S.A

ONE UTSA CIRCLE
SAN ANTONIO, TEXAS 78249-0631
210 458-4317 | BUSINESS.UTSA.EDU

A Tabu Search Heuristic Procedure for the Capacitated Facility Location Problem

Minghe Sun

College of Business
The University of Texas at San Antonio
San Antonio, TX 78249-0634
(210) 458-5777
msun@utsa.edu
http://faculty.business.utsa.edu/msun

Abstract

A tabu search heuristic procedure for the capacitated facility location problem is developed, implemented and computationally tested. The heuristic procedure uses both short term and long term memories to perform the main search process as well as the diversification and intensification functions. Visited solutions are stored in a primogenitary linked quad tree as a long term memory. The recent iteration at which a facility changed its status is stored for each facility site as a short memory. Lower bounds on the decreases of total cost are used to measure the attractiveness of switching the status of facilities and are used to select a move in the main search process. A specialized transportation algorithm is developed and employed to exploit the problem structure in solving transportation problems. The performance of the heuristic procedure is tested through computational experiments using test problems from the literature and new test problems randomly generated. It found optimal solutions for almost all test problems used. As compared to the Lagrangean and the surrogate/Lagrangean heuristic methods, the tabu search heuristic procedure found much better solutions using much less CPU time.

Keywords: Capacitated facility location, Tabu search, Metaheuristics

JEL Code: C61

1. Introduction

Facility location is a large area with a vast literature and numerous applications in the private and public sectors (Mirchandani and Francis, 1990; Daskin, 1995; Owen and Daskin, 1998; Drezner and Hamacher, 2001). This study focuses on the capacitated facility location problem (CFLP). The CFLP involves the selection of sites where facilities with limited capacities are established and the assignment of clients to facilities so as to satisfy client demands while minimizing total operating and transportation costs.

A CFLP consists of a set of m potential sites where facilities can be established and n clients whose demands can be satisfied from any established facility. Let I denote the index set of all candidate facility sites, *i.e.*, $I = \{1, \cdots, m\}$ and J denote the index set of all clients, *i.e.*, $J = \{1, 2, \cdots, n\}$. Each facility $i \in I$ has a capacity a_i and each client $j \in J$ has a demand b_j. The fixed cost of operating facility i is represented by f_j and the unit transportation cost from facility i to client j is represented by c_{ij}. A binary variable y_i is used to represent the status of facility i, *i.e.*, $y_i = 1$ if it is open and $y_i = 0$ if it is closed. A continuous variable x_{ij} is used to represent the quantity supplied from facility i to client j. The CFLP is represented by the following mixed integer programming model.

$$z = \min \quad \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{i \in I} f_i y_i \qquad (1)$$

$$\text{s.t.} \quad \sum_{i \in I} x_{ij} = b_j \qquad \forall \ j \in J \qquad (2)$$

$$\sum_{j \in J} x_{ij} \le a_i y_i \qquad \forall \ i \in I \qquad (3)$$

$$x_{ij} \ge 0 \qquad \forall \ i \in I \text{ and } j \in J \qquad (4)$$

$$y_i \in \{0,1\} \qquad \forall \ i \in I . \qquad (5)$$

In the model, the objective function (1) represents the total cost, including the total transportation cost and the total operating cost, to be minimized. Constraints (2) ensure that the demand of each client is satisfied and constraints (3) limit the amount of supplies to all clients from each facility i to be within its capacity $a_i y_i$. Constraints (4) and (5) define the values that the variables can assume. By assigning values to the binary variables, y_i, $\forall \ i \in I$, the resulting primal structure is a transportation problem. Setting $b_j = 1$ in (2) and replacing the constraint in (3) for each i with $x_{ij} \le y_i$, $\forall \ j \in J$, a CFLP becomes a uncapacitated facility location problem (UFLP).

In this study, a tabu search (TS) heuristic procedure is proposed for the CFLP. The short term memory structure records the recent iteration at which a facility changed status the last time. In addition to the recency based short term memory, a long term memory structure uses a primogenitary linked quad tree (PLQT) to store visited solutions. Once a solution is visited, it is memorized and its revisit is prohibited so as to prevent

repetition and cycling. The evaluation to find the exact total cost of a trial solution in the neighborhood of the current solution involves the solution of a transportation problem. Because many trial solutions need to be evaluated, solving these transportation problems takes too much computational time in a heuristic procedure. Therefore, upper bounds on the decrease of total costs are computed and used to evaluate the attractiveness of the trial solutions in the neighborhood. When solving transportation problems for the visited solutions, a special transportation algorithm is used to exploit the problem structure. The performance of the TS heuristic procedure is tested on test problems from the OR-Library (Beasley, 1990) and on test problems newly generated. The Lagrangean heuristic (LH) method and the surrogate/Lagrangean heuristic (SLH) methods, both implemented by Lorena and Senne (1999), are used as benchmarks to measure the effectiveness and efficiency of the TS procedure. The software CPLEX is used to find optimal solutions for easy problems.

The rest of the paper is organized as follows. A brief literature review is given in Section 2. The TS heuristic procedure is presented in Section 3. Computational results are reported in Section 4. Concluding remarks are given in Section 5.

2. Previous Work

The CFLP has been studied extensively. Many exact algorithms and heuristic methods have been developed to solve it in the last 40 years. Because UFLP and CFLP are closely related, many heuristic methods developed for the UFLP are also extended to the CFLP.

Kuehn and Hamburger (1963) developed the first heuristic method for the UFLP, which was later extended to the CFLP by Jacobsen (1983). This heuristic method consists of two phases. The first phase, called ADD, starts with all facilities closed and then the facility that causes the maximum total cost reduction is opened. This phase ends when no more facilities can be opened to further reduce the total cost. The second phase is a local search procedure in which an open facility and a closed facility exchange their status if this exchange reduces the total cost. Domschke and Drexl (1985) proposed priority rules for the ADD procedure to improve its performance in cases where the facilities have distinct capacities and/or distinct fixed operating costs. Feldman *et al*. (1966) proposed a different strategy for the first phase, named DROP, that was also extended to the CFLP by Jacobsen (1983). In DROP, all facilities are initially open and a facility is closed if closing it results in the maximum reduction in the total cost. This phase ends when closing a facility does not result in any further reduction in the total cost.

Lagrangean relaxation has been applied to several facility location problems. Cornuejols *et al*. (1991) presented an excellent theoretical analysis of all possible Lagrangean relaxations and the linear programming relaxation for the CFLP, and showed that only 7 relaxations yield distinct bounds. Dominance relations among the relaxations were also discussed. Beasley (1993) presented a unified framework of using the lagrangean Heuristic (LH) method to solve different facility location problems. In the proposed framework for the CFLP, constraints (2) and (3) are relaxed and the solution of the relaxed problem is trivial. Lorena and Senne (1999) proposed a LH method for the UFLP and the CFLP. Constraints (2) are relaxed and the resulting relaxed

problem decomposes into a continuous knapsack problem, which is solved in linear time (Martelo and Toth, 1990). In this LH method, local information from surrogate constraints is used to accelerate convergence of the subgradient method. Barahona and Chudak (2001) also proposed a LH method for the UFLP and the CFLP. Initially they considered the linear programming relaxation of the CFLP and then suggested the Lagrangean relaxation relative to constraints (2) for solving the linear programming problem. They used the volume algorithm (Barahona and Anbil, 2000) in order to maximize the dual objective function. The volume algorithm is an extension of the subgradient method and aims at generating good primal solutions. The name of the method comes from a theorem stating that a primal solution can be obtained from the volume under the faces of the piecewise linear and concave dual objective function.

Several exact algorithms based on branch-and-bound have been proposed. The major differences among these algorithms are in the types of relaxation, the methods of solving the relaxed problem and the strategies to improve the lower bound. Sa (1969) replaced the binary variable y_i with $(1/a_i)\sum_{j\in J} x_{ij}$ and the resulting relaxed problem became a transportation problem. Akinc and Khumawala (1977) applied the same relaxation and defined a "maximum useful capacity" in order to obtain tighter bounds. Geoffrion and McBride (1978) used the Lagrangean relaxation by relaxing constraints (2). Nauss (1978) used the same relaxation and included the inequality $\sum_{i\in I} a_i y_i \geq \sum_{j\in J} b_j$ in order to obtain tighter bounds. Christofides and Beasley (1983) also used this relaxation and introduced penalties for opening or closing a facility, also with the intent of improving the lower bound. Van Roy (1986) implemented the cross decomposition method that combines Benders decomposition and Lagrangean relaxation in order to exploit the primal and dual structures of the CFLP. Leung and Magnanti (1989) introduced a family of facets and valid inequalities for solving the CFLP with equal capacities. Aardal (1998) proposed new valid inequalities and implemented two branch-and-cut algorithms that are tested on small and medium test problems from the literature.

The TS metaheuristic has been successfully applied to a variety of combinatorial optimization problems, but not much research has been reported in using this metaheuristic to solve the CFLP. The TS heuristic procedure proposed by Grolimund and Ganascia (1997) was applied to the CFLP and limited computational results were reported. However, TS procedures have been developed for more complicated facility location problems, such as those by Delmaire *et al*. (1998), Ferreira Filho and Galvão (1998), França et al. (1999), and Tuzun and Burke (1999). Al-Sultan and Al-Fawzan (1999), Ghosh (2003), Hoefer (2003), Michel and Van Hentenryck (2004) and Sun (2005, 2006a) have applied TS to the UFLP.

3. The Tabu Search Heuristic Procedure

The TS metaheuristic (Glover, 1989, 1990; Glover and Laguna, 1997) uses responsive exploration and flexible memory to guide the search in the solution process. By responsive exploration, it determines a search direction based on the properties of the current solution and the search history. By flexible memory, it uses short term and longer term memory structures to record a selective search history. The memory structure used in this

study keeps track of solutions, as well as some of their attributes, visited in the search process. The short term memory records the recent time when a facility changed status and the long term memory memorizes visited solutions. This memory structure prevents solutions from being revisited and, therefore, prevents repetition and cycling.

The proposed TS heuristic procedure is composed of *search cycles*. Each search cycle, except for the first, comprises the diversification function, the main search process, and the diversification function, in that order. The components of the TS heuristic procedure are detailed first in the following and then a step-by-step description is given.

3.1. Move

For a given solution, I is partitioned into two subsets I_0 and I_1, where I_0 consists of the indices of the facilities that are closed and I_1 consists of the indices of the facilities that are open, respectively, *i.e.*, $I_0 = \{i \in I \mid y_i = 0\}$ and $I_1 = \{i \in I \mid y_i = 1\}$. Accordingly, m_0 and m_1, with $m_0 + m_1 = m$, represent the numbers of closed and open facilities, respectively, *i.e.*, $m_0 = \mid I_0 \mid$ and $m_1 = \mid I_1 \mid$.

A move is defined as the status change of any facility $i \in I$, *i.e.*, $y_i \leftarrow 1 - y_i$. Thus a move is a transition from the current partition of I to a new partition of I by taking one element from I_0 and placing it into I_1 or vice versa. We use k to count the number of moves, or iterations, made since the search started.

The minimum total cost, including the transportation cost and the operating cost, of the partition at iteration k is denoted by z^k .The cost of the best solution found in a single search cycle is denoted by z_0 and the iteration at which z_0 is found is denoted by k_0. The values of z_0 and k_0 are updated when a better solution is found and are reset when a new search cycle starts. The cost of the best solution found since the search started is denoted by z_{00}. The value of z_{00} is updated whenever a solution with total cost less than z_{00} is found.

3.2. Feasible and Infeasible Solutions

Let A denote the total capacity of all open facilities of a given partition and B the total demand of all clients, *i.e.*, $A = \sum_{i \in I_1} a_i$ and $B = \sum_{j \in J} b_j$. The solution corresponding to a partition is feasible if and only if $A \geq B$. This TS procedure searches only the set of feasible solutions. A facility $i' \in I_1$ of a feasible solution can be closed and the resulting solution is still feasible only if

$$A - a_{i'} \geq B . \tag{6}$$

Otherwise, the resulting solution is infeasible and facility $i' \in I_1$ cannot be closed. However, if the current solution is feasible, the resulting solution is always feasible when a currently closed facility $i' \in I_0$ becomes open. After the status of any facility i' changes, the value of A is updated accordingly, *i.e.*,

$$A \leftarrow A + a_{i'}, \text{ if } i' \in I_0 \qquad \text{or} \qquad A \leftarrow A - a_{i'}, \text{ if } i' \in I_1 . \tag{7}$$

Domschke and Drexl (1985) point out that the ADD method may lead to bad solutions when the facility capacities are distinct. They proposed three priority rules of opening facilities in order to overcome this difficulty. One priority rule is to move facilities $i \in I_0$ to I_1 in the order of increasing values of

$$P_i^2 = f_i - \frac{\sum_{j=1}^{n} b_j}{\sum_{j=1}^{n} c_{ij} b_j} a_i \tag{8}$$

until the feasibility condition $A \geq B$ is satisfied. Another priority rule is to move facilities $i \in I_0$ to I_1 in the order of increasing values of

$$P_i^3 = \frac{\sum_{j=1}^{\lfloor n/3 \rfloor} \overline{c}_{ij}}{\lfloor n/3 \rfloor} + \frac{f_i}{a_i} \tag{9}$$

until the feasibility condition $A \geq B$ is satisfied. Among the three rules, they showed that the rule of using P_i^3 in (9) is the most efficient. Therefore, this rule is used in this study to find an initial feasible solution.

3.3. Solution of Transportation Problems

When the variables y_i are fixed to their binary values for a given partition of I, the CFLP (1)–(5) reduces to a transportation problem with m_1 supply nodes and n demand nodes. In the solution process, one transportation problem needs to be solved for each move whether a facility is opened or closed. A network algorithm is used to solve these transportation problems. Although the transportation problem has only m_1 supply nodes, all m facilities are stored as supply nodes in the data structure of the solution algorithm. Keeping all m facilities as supply nodes allows the current transportation problem to be solved starting from the optimal solution of the previous one.

Facilities and clients are represented by nodes and roads from facilities to clients are represented by arcs. The arc from facility $i \in I$ to client $j \in J$ is represented by (i, j). One dummy demand node, numbered $n_1 = n + 1$, is used to absorb slack supplies. Hence, the demand at client n_1 is $b_{n_1} = A - B$ and the flow from a facility $i \in I$ to client n_1, denoted by x_{in_1}, represents the slack supply at facility i. A spanning tree, denoted by $\mathcal{T}$, is used to represent the current basic solution in the solution process. The fact that the arc (i, j) is in the spanning tree is denoted by $(i, j) \in \mathcal{T}$. As in any network algorithm, together with other data, the values of the dual variables associated with the nodes are stored in the spanning tree. These dual variables are used for pricing purpose, *i.e.*, for determining if the current solution is optimal and if a non-basic arc should be selected to enter the spanning tree.

Let $\overline{c}$ be a very large unit shipping cost, substantially larger than any c_{ij}, $\forall$ $i \in I$ and $j \in J$. If facility $i' \in I_1$ needs to be closed, then $\overline{c}$ is added to $c_{i'j}$, $\forall$ $j \in J$, to make the use of the arc (i', j) too costly. After $\overline{c}$ is added to $c_{i'j}$, $\overline{c}$ is also added to the dual variable associated with node i'. In this way, arcs (i', j),

$\forall\ j \in J$, are forced out of the spanning tree and the supply at facility i' will not be used, which is equivalent to closing facility i'. If facility $i' \in I_0$ needs to be opened, the current value of $c_{i'j}$ for each $j \in J$ is over its actual value by $\bar{c}$. Hence, $\bar{c}$ is subtracted from the current value of each $c_{i'j}$ to restore its original value. After $\bar{c}$ is subtracted from the current value of each $c_{i'j}$, $\bar{c}$ is also subtracted from the value of the dual variable associated with node i'. In this way, an arc (i', j) becomes more likely to be selected to enter the spanning tree and the supply at facility i' will be used, which is equivalent to opening facility i'. By imposing a large unit transportation cost to the arcs outgoing from a closed facility, rather than changing the capacity of the closed facility, the optimal solution of the previous transportation problem is still primally feasible in the new transportation problem.

In the implementation, a special purpose algorithm was developed to solve such transportation problems to exploit the problem structure. This special algorithm differs from standard network algorithms (Kennington and Helgason, 1980) in the selection of a non-basic arc to enter the spanning tree. Because none of the arcs outgoing from any closed facility $i' \in I_0$ should be in the spanning tree of the optimal solution, these arcs are skipped in the pricing process. Substantial computation time is saved by skipping these arcs, especially when m_0 is relatively large as compared with m_1.

3.4. Visited Solutions

Because the solution process of transportation problems uses most of the computation time, the transportation problem for any given partition of I only needs to be solved at most once. A solution is visited if it is the partition of I of any move. When visited, the transportation problem corresponding to the partition is solved, the partition and the minimum cost of the partition are saved.

For a problem with m candidate facility sites, there are fewer than 2^m feasible solutions. These solutions can be naturally ordered from 0 to $2^m - 1$. Only a tiny portion of these solutions is visited in the solution process. Therefore, it is not necessary and not feasible to store the visited solutions in an array with 2^m elements. However, searching these visited solutions may need too much computation time if the solutions are not stored in a specific order. Storing them in a specific order may take even a much longer computation time if they are stored in an ordinary array.

In this study, visited solutions are stored in a PLQT. The PLQT is very efficient in storing and retrieving these visited solutions (Sun, 2007). The CPU time used to manage the PLQT is negligible as compared to that used to solve transportation problems. Sun (2007) provided detailed descriptions about the way a solution is encoded and represented in a PLQT and about the algorithms of storing and retrieving solutions. Hashing has been used for this purpose in the literature (Woodruff and Zemel, 1993). However, hashing may cause difficulties such as collision and excessive amount of memory space (Carlton and Barnes, 1996; Sun, 2007). The use of a PLQT completely eliminates these difficulties.

3.5. Neighborhood

The neighborhood of a feasible solution is the set of m distinct solutions that can be reached by making one move from the current solution. Let $\Delta z_{i'}^k$ represent the decrease in total cost resulting from a move changing the status of facility $i' \in I$ at iteration k. The value of $\Delta z_{i'}^k$ measures the attractiveness of the move, the smaller the $\Delta z_{i'}^k$ is the more attractive the move is. Before a facility is selected to switch status for the next move, a number (up to *m*) of solutions in the neighborhood need to be evaluated to find or estimate $\Delta z_{i'}^k$ for some $i' \in I$.

If $i' \in I_1$, the feasibility condition (6) of the resulting solution is checked first. If (6) is not satisfied, a $\Delta z_{i'}^k = \infty$ is assigned to prevent $i' \in I_1$ from being closed. In the solution process, a solution is only expected to be visited at most once. If a solution is visited the second time, repeated visiting of a subset of solutions may occur. For each $i' \in I$ to be evaluated, the PLQT is searched. If the resulting solution is in the PLQT, the solution has been visited already. A $\Delta z_{i'}^k = \infty$ is then assigned to prevent the solution from being visited again. If the resulting solution is feasible but has not been visited, $\Delta z_{i'}^k$ is then estimated.

To find the exact value of $\Delta z_{i'}^k$ whether $i' \in I_0$ or $i' \in I_1$, a transportation problem needs to be solved. Solving up to m transportation problems before each move is relatively a very time consuming process. We found at the early stage of developing this TS procedure that exactly solving these transportation problems is impractical. Therefore, an upper bound on $\Delta z_{i'}^k$, rather than its exact value, is computed and used in this TS procedure.

A facility $i' \in I_0$ is a candidate to open if the resulting solution has not been visited already. The ADD-LO procedure proposed by Jacobsen (1983) is used to estimate an upper bound on $\Delta z_{i'}^k$ for $i' \in I_0$. When facility $i' \in I_0$ opens, the current flow from an $i \in I_1$ to a $j \in J$ may be shifted to a flow from i' to j if $c_{i'j} < c_{ij}$. Let $\mathcal{A} = \{(i, j) \in \mathcal{T} \mid c_{i'j} < c_{ij}\}$, *i.e.*, the set of basic arcs with $c_{i'j} < c_{ij}$. Let x_{ij}^* be the current flow on arc (i, j) and let $x_{i'j}^i$ be the resulting flow shifted from arc (i, j) to arc (i', j). Solve the following continuous knapsack problem:

$$\delta_{i'} = \min \sum_{(i,j)\in\mathcal{A}} (c_{i'j} - c_{ij}) x_{i'j}^i \qquad (10)$$

$$\text{s.t.} \sum_{(i,j)\in\mathcal{A}} x_{i'j}^i \le a_{i'} \qquad (11)$$

$$0 \le x_{i'j}^i \le x_{ij}^* \qquad \forall \ (i, j) \in \mathcal{A}. \qquad (12)$$

Then

$$\Delta^u z_{i'}^k = \delta_{i'} + f_{i'} \qquad (13)$$

is an upper bound on $\Delta z_{i'}^k$ (Jacobsen, 1983) and $z^k + \Delta^u z_{i'}^k$ is an upper bound of the minimum total cost of the resulting partition. Although the solution process of the continuous knapsack problem in (10)-(12) requires a sorting of all arcs $(i, j) \in \mathcal{A}$ in the ascending order of $c_{i'j} - c_{ij}$, the problem is easy to solve because the number of arcs in $\mathcal{A}$ is usually substantially smaller than that in $\mathcal{T}$.

A facility $i' \in I_1$ is a candidate to close in the next move only if it satisfies the feasibility condition (6) and the resulting solution has not been visited. When facility $i' \in I_1$ is closed, all flows from i' must be supplied from other facilities. To find an upper bond on $\Delta z_{i'}^k$, we assume these flows are supplied from the facilities $i \in I_1$ with positive slack supplies, *i.e.*, with $x_{in_1} > 0$. Let $I_S = \{i \in I_1 \mid x_{in_1} > 0, i \neq i'\}$ represent the set of facilities, not including facility i', with positive slack supplies in the current solution. Let $J_D = \{j \in J \mid x_{i'j} > 0\}$ represent the set of clients receiving shipments from facility i' in the current solution. Let $x_{i'j}^*$ be the current flow on arc (i', j), $x_{in_1}^*$ be the slack supply of facility $i \in I_S$, and $x_{ij}^{i'}$ be the resulting flow on the arc (i, j) for $i \in I_S$ and $j \in J_D$. Solve the following transportation problem:

$$\delta_{i'} = \min \quad \sum_{i \in I_S} \sum_{j \in J_D} (c_{ij} - c_{i'j}) x_{ij}^{i'} \qquad (14)$$

$$\text{s.t.} \quad \sum_{i \in I_S} x_{ij}^{i'} = x_{i'j}^* \qquad \forall\ j \in J_D \qquad (15)$$

$$\sum_{j \in J_D} x_{ij}^{i'} \leq x_{in_1}^* \qquad \forall\ i \in I_S \qquad (16)$$

$$x_{ij}^{i'} \geq 0 \qquad \forall\ i \in I_S \text{ and } j \in J_D. \qquad (17)$$

Then $\delta_{i'} - f_{i'}$ is an upper bound on $\Delta z_{i'}^k$ (Jacobsen, 1983). This transportation problem is usually much smaller than the original transportation problem and is much easier to solve. Jacobsen (1983) mentioned this procedure and called it DROP-LO but did not implement it. Instead of obtaining $\delta_{i'}$ by solving the transportation problem in (14)-(17) to optimality, the value of the objective function of an initial solution obtained with the greedy method, denoted by $\bar{\delta}_{i'}$, is used as an upper bound of $\delta_{i'}$ in this study. Then

$$\Delta^u z_{i'}^k = \bar{\delta}_{i'} - f_{i'} \qquad (18)$$

is also an upper bound on $\Delta z_{i'}^k$ and $z^k + \Delta^u z_{i'}^k$ is an upper bound of total cost of the resulting partition after closing facility i'. Although $\Delta^u z_{i'}^k$ is a looser bound than $\delta_{i'} - f_{i'}$, it is sufficient for the purpose of this TS heuristic. The greedy method requires a sorting of all arcs (i, j), $\forall\ i \in I_S$ and $j \in J_D$, in the ascending order of $c_{ij} - c_{i'j}$. However, such an initial solution is easy to find because the number of arcs involved is usually substantially smaller than that in $\mathcal{T}$.

3.6. The Main Search Process

The tabu tenure for each $i \in I_0$ is denoted by l_0 and that for each $i \in I_1$ is denoted by l_1. Hence, a facility is restricted to stay closed for l_0 moves after being closed and is restricted to stay open for l_1 moves after being opened unless the aspiration criterion is satisfied. The value of l_0 is randomly selected from the integers in the interval $[l_0^l, l_0^u]$ and that of l_1 is randomly selected from the integers in the interval $[l_1^l, l_1^u]$. The interval limits are related to m and the initial values of m_0 and m_1. After an initial feasible solution is found, these interval limits are set to $l_1^l = \lceil m/8 \rceil$, $l_1^u = \lfloor m/4 \rfloor$, $l_0^l = l_1^l (m_0 / m_1)$ and $l_0^u = l_1^u (m_0 / m_1)$. Care is taken to make sure that these interval limits satisfy the restrictions $l_0^l < l_0^u$, $l_1^l < l_1^u$, $l_0^l > 0$ and $l_0^u \le m$. If any of these restrictions is violated, some of the interval limits are reset to reasonable values. New values for l_0 and l_1 are selected from their respective intervals when the short term memory process restarts in a new search cycle. The tabu list is implemented through the integer vector $\mathbf{t}$. The element t_i of $\mathbf{t}$ represents the iteration number at which facility i changed its status for the last time.

At iteration k, a move switching the status of facility i' is chosen, such that,

$$\Delta^u z_{i'}^k = \min \{ \Delta^u z_i^k \mid i = 1, \cdots, m \}. \tag{19}$$

The following tabu condition is then checked

$$k - t_{i'} \le l_0 \text{ if } i' \in I_0 \text{ or } k - t_{i'} \le l_1 \text{ if } i' \in I_1. \tag{20}$$

The selected move is not tabu if (20) is not satisfied. In this case, the move is made. Otherwise, if (20) is satisfied, the move is tabu and is made only if the following aspiration criterion is satisfied

$$z^k + \Delta^u z_{i'}^k < z_0. \tag{21}$$

If tabu and (21) is not satisfied, $\Delta^u z_{i'}^k = \infty$ is set, another move is selected according to (19), its tabu condition is checked according to (20), and so on. This process continues until a move that is not tabu or tabu but satisfies the aspiration criterion is found and the move is made. The main search process stops if z_0 is not improved after $\alpha_1 m$ moves, where $\alpha_1 > 0$ is a parameter of the TS heuristic procedure, *i.e.*, when the condition $k - k_0 > \alpha_1 m$ is satisfied.

3.7. Intensification and Diversification

Each time after the main search process ends, the intensification function starts. The priority rules P_i^2 in (8) and P_i^3 in (9) (Domschke and Drexl, 1985) are used alternately for this purpose. If P_i^2 is used in the current search cycle, P_i^3 will be used in the next search cycle. Because no solution is allowed to be visited the second time, the intensification function will never end up at a previously visited solution, including the initial solution.

Let P_i^* be P_i^2 or P_i^3 whichever is used in the current search cycle. A facility $i' \in I_1$ is selected such that

$$P_{i'}^* = \max\{P_i^* \mid i \in I_1\}. \tag{22}$$

Facility i' is closed if the feasibility condition (6) is satisfied and the resulting solution has not been visited already. If the resulting solution has been visited already, i' is skipped and another $i' \in I_1$ is selected according to (22). Otherwise if the feasibility condition (6) is not satisfied, another facility $i' \in I_0$ is selected to open such that

$$P_{i'}^* = \min\{P_i^* \mid i \in I_0\}. \tag{23}$$

Facility i' is opened if the resulting solution has not been visited already, or i' is skipped and another $i' \in I_0$ is selected according to (23) otherwise. This process continues until there are not any $i' \in I_0$ and $i'' \in I_1$ such that $P_{i'}^* < P_{i''}^*$ that have not been checked. In the implementation, the values of P_i^2 and P_i^3 are sorted and stored in their respective ascending orders.

The recency based memory, *i.e.*, the integer vector $\mathbf{t}$, is also used for the diversification function. The diversification function is performed at the beginning of each search cycle, except for the first. A facility i' is selected such that

$$t_{i'} = \max\{t_i \mid i \in I\}. \tag{24}$$

A move is then made to open the facility if $i' \in I_0$ proved the resulting solution has not been visited already, or to close it if $i' \in I_1$ proved the feasibility condition (6) is satisfied and the resulting solution has not been visited already. If a move cannot be made, facility i' is skipped and another facility is selected according to (24).

This process is continued until c moves are made in the c th search cycle. In this way, the search will explore a new area in the feasible region to achieve diversification purpose. The search process terminates after C search cycles have been executed.

3.8. The TS Procedure

A step-by-step description of the TS heuristic procedure is given in the following. These steps are roughly grouped into different sections according to their functions.

Initialization

Step 1 Find an initial solution with a total cost z^1 using the ADD method. Let $z_0 \leftarrow z^1$ and $z_{00} \leftarrow z_0$. Determine m_1. Choose values for l_0^l and l_0^u and select an integer for l_0 such that $l_0 \in [l_0^l, l_0^u]$. Let $t_i \leftarrow -l_0, \forall\ i \in I_0$. Choose values for l_1^l and l_1^u and select an integer for l_1 such that $l_1 \in [l_1^l, l_1^u]$. Let $t_i \leftarrow -l_1, \forall\ i \in I_1$. Select values for the parameters α_1 and C. Let $k \leftarrow 1$, $k_0 \leftarrow 1$, $c_0 \leftarrow 0$, $c \leftarrow 1$.

Main Search Process

Step 2 Obtain $\Delta^u z_i^k$ for some $i \in I$. For $i \in I_0$, if the resulting solution is not in the PLQT already, solve the continuous knapsack problem in (10)-(12) and compute $\Delta^u z_i^k$ using (13). For $i \in I_1$, if the feasibility condition (6) is satisfied and the resulting solution is not in the PLQT already, find an initial solution for the transportation problem in (14)-(17) and compute $\Delta^u z_i^k$ using (18). Otherwise, let $\Delta^u z_i^k = \infty$.

Step 3 Select a move switching the status of facility i' according to (19). Check the tabu status of the selected move according to (20). If the move is tabu, go to Step 4; otherwise, go to Step 14.

Step 4 Check the aspiration criterion of the selected move according to (21). If (21) is not satisfied, set $\Delta^u z_{i'}^k \leftarrow \infty$ and go to Step 3; otherwise, go to Step 14.

Intensification

Step 5 If all i' have been checked, go to Step 9; otherwise, select a facility i' according to (22).

Step 6 If the feasibility condition (6) is not satisfied, go to step 7. If the resulting solution is in the PLQT already, skip i' and go to Step 5; otherwise, go to Step 14.

Step 7 If all i' have been checked, go to Step 9; otherwise, select a facility i' according to (23).

Step 8 If the resulting solution is in the PLQT already, skip i' and go to Step 7; otherwise, go to Step 14.

Diversification

Step 9 If $c > C$, Stop. Let $c_0 \leftarrow c_0 + 1$. If $c_0 > c$, go to Step 13.

Step 10 Select a facility i' according to (24). If $i' \in I_0$, go to Step 11, otherwise, go to Step 12.

Step 11 If the resulting solution is not in the PLQT already, go to Step 14; otherwise, skip i' and go to Step 10.

Step 12 If the feasibility condition (6) is satisfied and the resulting solution is not in the PLQT already, go to Step 14; otherwise, skip i' and go to Step 10.

Step 13 Let $c_0 \leftarrow 0$, $c \leftarrow c + 1$, $z_0 \leftarrow z$, $k_0 \leftarrow k$. Reset the value of l_0 such that $l_0 \in [l_0^l, l_0^u]$ and that of l_1 such that $l_1 \in [l_1^l, l_1^u]$ and then go to Step 2.

Move Execution

Step 14 Update A according to (7). If $i' \in I_0$, let $m_1 \leftarrow m_1 + 1$; otherwise, let $m_1 \leftarrow m_1 - 1$. Change the status of facility i', *i.e.*, let $y_{i'} \leftarrow 1 - y_{i'}$. Let $t_{i'} \leftarrow k$ and $k \leftarrow k + 1$. Solve the resulting transportation problem and let z^k be the minimum value of the total cost of the partition. If $z^k < z_0$, let $z_0 \leftarrow z^k$ and $k_0 \leftarrow k$. If $z_0 < z_{00}$, let $z_{00} \leftarrow z_0$.

Step 15 If $k - k_0 \le \alpha_1 m$, go to Step 2; otherwise, go to Step 5.

4. Computational Experiments

The proposed TS heuristic procedure was coded in C. The C code for the LH/SLH methods (Lorena and Senne, 1999) was obtained from the original authors. The computational experiments were conducted on a Sun Enterprise 450 computer with two 400 Mhz processors (only one is used) and 1.5 GB RAM.

In the following tables, computational results are reported. For each group of test problems, solution quality of a heuristic method is measured by the average gap. The gap in percentage for a test problem is the relative deviation between the total cost of the final solution obtained by a heuristic method (*i.e.*, the final value of z_{00} in the TS heuristic procedure) and that of the optimal solution. Using z_{final} to denote the total cost of the final solution obtained with a heuristic method, z_{opt} to denote the total cost of the optimal solution, and g to denote the gap, then g is defined as

$$g = \frac{z_{final} - z_{opt}}{z_{opt}} \times 100\% . \tag{25}$$

For each group of test problems, the average CPU time in seconds taken by each solution method is reported. For the TS heuristic procedure, the average CPU time in seconds needed to reach the best solution is also reported. For the TS procedure, the CPU time for each problem includes the time needed for data input and results output. For the LH/SLH methods, the CPU time does not include the time needed for data input and model setup (Lorena and Senne, 1999).

4.1. Test Problems from the Literature

The 49 test problems with known optimal solutions in the OR-Library (Beasley, 1990) are used first. These test problems are divided into 13 groups with four problems in each, except the second. The names of the problems are used in the OR-Library. The size of the problems are measured by $m \times n$. The values of the parameters in the TS procedure are set to $\alpha_1 = 1.0$ and $C = 5$ for smaller problems ($m \le 50$) or $C = 8$ for larger problems ($m = 100$).

Computational results of these test problems are presented in Table 1. These results were obtained with a single run. Slightly different results may be obtained if different values for the parameters in the TS procedure are used. The TS heuristic procedure found optimal solutions for all problems with $m \times n = 16 \times 50$ and $m \times n = 25 \times 50$. However, it missed the optimal solution for 2 out of the 12 problems with $m \times n = 50 \times 50$. The solutions it found for problems with $m \times n = 100 \times 1000$ are all very close to the optimal solutions. As compared to the LH/SLH methods, the TS procedure found much better solutions in much less CPU time for all problems. The TS procedure also found the best solutions relatively early in the search process.

However, for small problems, heuristic procedures don't have much computational advantage over exact algorithms. For problems with up to $m \times n = 50 \times 50$, the CPU time used by CPLEX is only a few times of that used by the TS procedure. For these problems, the LH/SLH methods even use much more CPU time than CPLEX. However, for problems with $m \times n = 100 \times 1000$, the CPU time used by heuristic methods are negligible

as compared to that used by CPLEX. The average CPU time of 236,383.95 seconds used by CPLEX for problems "capa*" is the average of problems "capa1" and "capa2", the only $m \times n = 100 \times 1000$ problems that can be solved so far on the computer used for this study. Although the LH/SLH methods use much more CPU time than the TS procedure does, the CPU time they use are still within the rounding error of that used by CPLEX. Therefore, heuristic methods are aimed for large problems.

Table 1 approximately here

4.2. New Test Problems

Randomly generated new test problems were used in order to evaluate the performance of the TS heuristic procedure with a wide range of test problems. Test problems were divided into groups. Test problems in each group have similar properties, such as the size of the problem, ranges of unit transportation costs, ranges of fixed operating costs, ranges of facility capacities and ranges of client demands. The notation $U[l_1, l_2]$ is used to denote a randomly generated number from the integer uniform distribution with a lower limit l_1 and an upper limit l_2.

Metric test problems are used. These metric test problems were generated in a scheme similar to that proposed by Cornuejols *et al*. (1991). Five problem groups, each with 30 problems, are used. The differences among these five groups are in the ratios of total capacity to total demand and in the fixed costs. The ratios for the five problem groups are listed Table 2 in the column with a heading R. Coordinates of facility and client locations were randomly generated in a unit square with a uniform distribution. The transportation cost c_{ij} is then 10 times of the Euclidean distance between facility i and client j. Fixed costs were generated according to the expression $f_i = r\left(U[0,90] + U[100,110]\sqrt{a_i}\right)$. The value of r is given in Table 2 for each problem group. Client demands were generated from $U[5,35]$, *i.e.*, $b_j = U[5,35]$. Facility capacities were first generated from $U[10,160]$, *i.e.*, $a_i = U[10,160]$, and then rescaled for each problem to have the total capacity to total demand ratio given in Table 2. All these problems have the same size $m \times n = 50 \times 50$. The values of the parameters in the TS procedure are set to $\alpha_1 = 1.0$ and $C = 10$.

Table 2 approximately here

Computational results of these test problems are reported in Table 2. The TS heuristic procedure found very good solutions for most of these problems using a small fraction of the time taken by CPLEX. For real life problems, the deviation from the optimal solution is much smaller than the imprecision in the input data. Among the five groups, problems in groups 4 and 5 are more difficult to solve than others. TS solutions have larger deviations from the optimal solutions and CPLEX takes more CPU time for these test problems. These problems have larger total capacity to total demand ratios than others. For the easier problems, the TS procedure found the best solutions relatively early in the search process, while for more difficult problems, it found the best solution relatively late. All these new test problems are more difficult to solve than the test problems in the OR-Library. Compared to the $m \times n = 50 \times 50$ problems (cap11*, cap12* and cap13*) in Table 1, CPLEX used much more

CPU times, while the TS procedure found solutions that are not as close to the optimal solutions although used more CPU times, for these new test problems. It could be interesting to compare the TS results with those obtained with the LH/SLH methods. Unfortunately, the LH/SLH code generated running time errors on these test problems.

5. Conclusions

This paper presents a TS heuristic procedure for the CFLP. In addition to recency based short term memory, it employs a PLQT to store all visited solution as a long term memory. Before moving to a solution, it checks to make sure that the solution has not been visited already. In this way, it explicitly prohibits the revisit of any previously visited solutions. Therefore, it completely eliminates repetition and cycling of a subset of solutions. Because the PLQT is very efficient in storing and retrieving solutions, the computational time used to manage the PLQT is negligible as compared to the total CPU time used by the TS procedure. This is the first time for the PLQT to be used in a heuristic procedure. It has the potential to be used in other heuristic procedures where visited solutions need to be stored.

Computational results on test problems from the literature and on test problems newly generated show that this TS heuristic procedure is very effective and efficient in finding good solutions. It finds optimal solutions for almost all test problems in the literature. As compared to the LH/SLH methods, it can find much better solutions using much less CPU time.

With modifications, this TS procedure may be applied to other facility location problems, such as the capacitated *p*-median problem and the single source capacitated facility location problem. Although other researchers have worked on such problems, it might be possible to find room for further improvement.

Acknowledgments

This research was partially funded by a 2007 Faculty Summer Research Grant from the College of Business, the University of Texas at San Antonio. The author is very grateful to Professor Edson L. F. Senne, at Faculdade de Engenharia, Departamento de Matemática Av. Ariberto Pereira da Cunha, 333 12516-410 Guaratinguetá, SP, for his generosity in sending his C code to the author and letting the author use it in the computational experiments.

References

Aardal, K. (1998). "Capacitated facility location: Separation algorithm and computational experience," *Mathematical Programming*, 81, 149-175.

Akinc, U. and Khumawala, M. (1977). "An efficient branch and bound algorithm for the capacitated warehouse location problem," *Management Science*, 23, 585-594.

Al-Sultan, K. S. and Al-Fawzan, M. A. (1999). "A Tabu search approach to the uncapacitated facility location problem," *Annals of Operations Research*, 86, 91-103.

Barahona, F. and Anbil, R. (2000). "The volume algorithm: producing primal solutions with a subgradient method," *Mathematical Programming*, 87, 385-399.

Barahona, F. and Chudak, F. (2001). "Near-optimal solutions to large scale facility location problems," Technical Report, www.ifor.math.ethz.ch/ staff/chudak.

Beasley, J. E. (1990). "OR-Library: Distributing test problems by electronic mail," *Journal of the Operational Research Society*, 41, 1069-1072.

Beasley, J. E. (1993). "Lagrangian heuristics for location problems," *European Journal of Operational Research*, 65, 383-399.

Carlton, W. B. and Barnes, J. W. (1996). "A note on hashing functions and tabu search algorithms," *European Journal of Operational Research*, 95, 237-239.

Christofides, N. and Beasley, J. E. (1983). "Extensions to a Lagrangean location problem," *European Journal of Operational Research*, 12, 19-28.

Cornuejols, G., Sridharan, R. and Thizy, J. M. (1991). "A comparison of heuristics and relaxations for the capacitated plant location problem," *European Journal of Operational Research*, 50, 280-297.

Daskin, M. S. (1995). Network and Discrete Location, Models, Algorithms, and Applications, Wiley, New York.

Delmaire, H. J., Diaz, J. A., Fernandez, E. and Ortega, M. (1998). "Reactive grasp and tabu search based heuristics for the single source capacitated plant location problem," *Information Systems and Operations Research*, 37, 194-225.

Domschke, W. and Drexl, A. (1985). "Add-heuristics starting procedures for capacitated plant location models," *European Journal of Operational Research*, 21, 47-53.

Drezner, Z. and Hamacher, H. W. (2001). *Facility Location: Theory and Algorithms*. Springer Verlag.

Feldman, E., Lehrer, F. A. and Ray, T. L. (1966). "Warehouse location under continuous economies of scale," *Management Science*, 12, 670-684.

Ferreira Filho, V. J. M. and Galvão, R. D. (1998). "A tabu search heuristic for the concentrator location problem," *Location Science*, 6, 189-209.

França, P. M., Sosa, N. G. and Pureza, V. M. (1999). "An adaptive tabu search approach for solving the capacitated clustering problem," *International Transactions in Operational Research*, 6, 665-678.

Geoffrion, A. M. and McBride, R. (1978). "Lagrangian relaxation applied to capacitated facility location problems," *AIIE Transactions*, 10, 40-47.

Ghosh, D. (2003). "Neighborhood search heuristics for the uncapacitated facility location problem," *European Journal of Operational Research*, 150, 150-162.

Glover, F. (1989). "Tabu Search, Part I," *ORSA Journal on Computing*, 1, 190-206.

Glover, F. (1990). "Tabu Search, Part II," *ORSA Journal on Computing*, 2, 4-32.

Glover, F. and Laguna, M. (1997). *Tabu Search*, Kluwer Academic Publishers, Hingham, MA.

Grolimund, S. and Ganascia, J. G. (1997). "Driving tabu search with case-based reasoning," *European Journal of Operational Research*, 103, 326-338.

Hoefer, M. (2003). Experimental comparison of heuristic and approximation algorithms for uncapacitated facility location," in K. Jansen *et al*. (Eds.): WEA 2003, LNCS 2647, 165-178.

Jacobsen, S. K. (1983). "Heuristics for the capacitated plant location model," *European Journal of Operational Research*, 12, 253-261.

Kennington, J. L. and Helgason, R. V. (1980). *Algorithms for Network Programming*, John Wiley and Sons, Inc., New York, NY.

Klein, R. (2000). "Project scheduling with time-varying resources constraints," *International Journal of Production Research*, 38, 3937-3952.

Kuehn, A. A. and Hamburger, M. J. (1963). "A heuristic program for locating warehouses," *Management Science*, 9, 643-666.

Leung, J. M. Y. and Magnanti, T. L. (1989). "Valid inequalities and facets of the capacitated plant location problem," *Mathematical Programming*, 44, 271-291.

Lorena, L. A. N. and Senne, E. L. F. (1999). "Improving traditional subgradient scheme for Lagrangean relaxation: an application to location problems," *International Journal of Mathematical Algorithms*, 1, 133-151.

Martello, S. and Toth, P. (1990). Knapsack problems: algorithms and computer implementations, Wiley, New York.

Michel, L. and Van Hentenryck, P. (2004). "A simple tabu search for warehouse location," *European Journal of Operational Research*, 157, 576-591.

Mirchandani, P. B. and Francis, R. L. (1990). *Discrete Location Theory*, Wiley, New York.

Nauss, R. M. (1978). "An improved algorithm for the capacitated facility location problem," *Journal of the Operational Research Society*, 29, 1195-1201.

Owen, S. H. and Daskin, M. S. (1998). "Strategic facility location: a review," *European Journal of Operational Research*, 111, 423-447.

Sa, G. (1969). "Branch and bound and approximate solutions to the capacitated plant location problem," *Operations Research*, 17, 1005-1016.

Sun, M. (2005). "A tabu search heuristic procedure for the uncapacitated facility location problem," in C. Rego and B. Alidaee (eds.), *Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search*, Kluwer Academic Publishers, Boston, MA, 191-211.

Sun, M. (2006a). "Solving uncapacitated facility location problems using tabu search," *Computers and Operations Research*, 33(9), 2563-2589.

Sun, M. (2006b). "A primogenitary linked quad tree data structure and its application to discrete multiple criteria optimization," *Annals of Operations Research*, 147(1), 87-107.

Sun, M. (2007). "A primogenitary linked quad tree approach for solution storage and retrieval in heuristic combinatorial optimization," Working Paper, the University of Texas at San Antonio.

Tuzun, D. and Burke, L. I. (1999). "A two-phase tabu search approach to the location routing problem," *European Journal of Operational Research*, 116(1), 87-99.

Van Roy, T. J. (1986). "A cross decomposition algorithm for capacitated facility location," *Operations Research*, 34, 145-163.

Woodruff, D. L. and Zemel, E. (1993). "Hashing vector for tabu search," *Annals of Operations Research*, 41, 123-137.

Table 1. Results of Test Problems from the Literature

Problem	TS					LH			SLH			CPLEX
Name	Size	Mean Gap	CPU Time to Best	CPU Time	Iterations	Mean Gap	CPU Time	Iterations	Mean Gap	CPU Time	Iterations	CPU Time
cap41*	16 × 50	0.000	0.02	0.05	62.0	1.223	1.83	583.5	1.223	1.23	426.8	0.17
cap51	16 × 50	0.000	0.05	0.09	125.0	0.397	1.75	555.0	0.562	0.92	288.0	0.36
cap61*	16 × 50	0.000	0.03	0.09	119.0	0.188	0.86	284.3	0.314	0.67	245.0	0.20
cap71*	16 × 50	0.000	0.02	0.07	126.8	0.087	0.28	105.8	0.000	0.27	108.8	0.16
cap81*	25 × 50	0.000	0.07	0.21	173.0	1.517	5.55	773.8	1.940	3.13	496.5	0.48
cap91*	25 × 50	0.000	0.05	0.16	175.0	0.106	2.49	464.8	0.122	1.26	234.3	0.47
cap101*	25 × 50	0.000	0.07	0.15	193.0	0.000	0.76	164.5	0.000	0.64	138.8	0.30
cap111*	50 × 50	0.158	0.14	0.52	298.5	0.591	11.13	636.5	0.637	7.24	455.5	1.71
cap121*	50 × 50	0.000	0.11	0.41	302.0	0.125	6.32	455.3	0.204	4.27	351.3	1.66
cap131*	50 × 50	0.000	0.13	0.42	322.5	0.171	3.22	265.0	0.353	2.52	193.8	1.26
capa*	10 0 × 1000	0.023	18.89	53.36	956.0	1.400	497.20	697.3	2.263	262.05	411.5	236383.95
capb*	10 0 × 1000	0.425	48.97	66.65	1047.0	2.030	669.62	722.3	3.099	394.90	574.0	—
capc*	10 0 × 1000	0.310	33.85	53.87	978.3	0.466	607.51	809.3	0.630	368.16	560.3	—

Table 2. Results of Randomly Generated Metric Test Problems

Problem	R	r	TS CPLEX				
			Gap	CPU Time to Best	CPU Time	Iterations	CPU Time
1	1.5	2.0	0.322	0.30	0.91	688.9	3.22
2	2.0	2.0	0.655	0.38	0.95	714.9	5.60
3	3.0	1.0	1.235	0.51	0.96	752.0	6.59
4	5.0	1.0	2.163	0.57	0.96	782.6	7.57
5	10.0	1.0	1.679	0.55	0.93	751.8	9.03