An-Approach to Solve a Bi-Objective Un-Capacitated Facility Location Problem

#Balakrishna G #Dr. Deepak Garg and #Dr. Mahesh Kumar Sharma

Abstract:

The problem of selecting up to a fixed number of customers from among a given number of potential sites for clustering a given number of sites to them subject to several constraints with two objectives, is considered. One of the constraints is that each site should be clustered to a unique customer which is selected for locating a site at it; however there is no restriction on the number of sites to be clustered to a selected site. The two objectives are to minimize the total cost and duration of meeting requirements of all the sites from their assigned customers at the selected sites. A net benefit heuristic algorithm (NBHA) incorporating tabu search is developed to find the set of efficient solutions of this problem.

Keywords: Efficient solution, Heuristic programming, Tabu search, clustering, facility location, heuristics.

1. Introduction

In this paper, we consider the un-capacitated facility location problem. In this problem, facilities are placed among *n* possible sites with the objective of minimizing the total cost for satisfying all demands at *m* given locations. The total cost consists of fixed charges for establishing facilities and costs for fulfilling the demands (distribution costs).

The un-capacitated facility location problem (UFLP) has been dealt with in the literature under several titles. This includes (un-capacitated, simple, optimal) followed by (plant, warehouse, facility, site), followed by the word (location) (Krarup and Pruzan [24]). UFLP is a simple mixed integer programming problem, yet it exhibits all the typical combinatorial difficulties of mixed (0–1) programming, but its special structure allows the development of specialized techniques for solving it (Guignard and Spielberg [17]). UFLP is an NP-hard problem (Garey and Johnson [12], Lenstra and Rinnooy Kan [25]).

The un-capacitated facility location problem has been extensively considered in the literature (see Krarup and Pruzan [24], Cornuéjols et al. [7] for comprehensive literature reviews). The first explicit formulation of UFLP is attributed to Balinski [1]; however, there seems to be some confusion on the part of some researchers which resulted in an obscure history

of UFLP and incorrect references to it. Krarup and Pruzan [24] very carefully followed the origin of UFLP. Efroymson and Ray [8] proposed a modified formulation of the problem and the use of a branch and bound algorithm to solve it. Khumawala [22] utilized the special structure of UFLP to improve the branch and bound algorithm of Efroymson and Ray. Bilde and Krarup [4] and Erlenkotter [9] developed a dual-based branch and bound procedure for the problem. Their procedure has been widely accepted as the most efficient known procedure for solving UFLP. Guignard [16] proposed to strengthen the separable Lagrangian relaxation of UFLP by using Benders inequalities generated during a Lagrangian dual ascent procedure. He showed that coupling his technique with a good primal heuristic can substantially reduce the integrality gap. Tcha et al. [29] developed a dual-based heuristic for this problem which is similar to Erlenkotter's procedure, and claimed that it yields solutions which are, in most of the cases, superior to those achieved by the dual ascent procedure with only a slight increase in computation time. Galvao and Raggi [10] proposed an optimal procedure for solving a more general formulation which has UFLP as a special case. Körkel [23] showed how to modify a primal–dual version of Erlenkotter's exact algorithm to get an improved procedure. He claimed that an implementation of the modified algorithm is faster than Erlenkotter's code by more than one order of magnitude. Simao and Thizy [26] developed a dual simplex algorithm for the canonical representation of this problem. Conn and Cornuéjols [6] suggested a new method for solving UFLP based on the exact solution of the condensed dual of the strong linear programming relaxation for UFLP via orthogonal projections. Beasley [3] proposed Lagrangian heuristics for several location problems which includes UFLP. His results indicated that his proposed approach is robust for solving different location problems. Gao and Robinson [11] presented a general model and dual-based branch and bound solution procedure to find optimal solutions for several un- capacitated location problems that include UFLP. They claimed that their proposed solution procedure effectively solves realistically sized UFLP.

In this paper, we present a tabu search based algorithm for solving the problem discussed above. The paper is organized as follows: the problem statement is given in section 2. In section 3, we give a brief introduction to tabu search. The proposed algorithm is stated in section 4. Computational results and discussion are presented in section 5. Finally, conclusions are given in section 6.

2. Problem statement

The un-capacitated facility location problem (UFLP) can be stated as follows:

Given *n* possible sites and demands at *m* locations, determine the optimal location of facilities to fulfil all demands such that the total cost of establishing the facilities and fulfilling the demands (distribution cost) is minimized and duration is maximized and vice versa.

Formulation of problem:

Suppose that there are N possible sites and demands at M locations, every site can be selected from among the N potential sites for locating customers at them. The N sites are to be clustered up to one or more sites in such a way that each shop is assigned to a unique site which is selected for locating a customer. There is no restriction on the number of site to be clustered to a selected customer. Let c_{ij} and t_{ij} $(i = 1,2,\ldots,M; j = 1,2,\ldots,N)$ units be the cost and time respectively of meeting requirements of site i from customer j. Let c_j units be the setup cost of a customer at the customer j only. Let x_{ij} be the variable assuming value 0 or 1 according as site i is not assigned or assigned to customer j, and y_j be the variable assuming the value 0 or 1 according as customer j is not selected or selected for locating a customer at it . Let C and T denote the total cost and duration respectively of meeting requirements of all the sites from their assigned customers. The mathematical formulation of this problem is as follows. The two objective functions which are sought to be minimized as well as maximized.

Minimized $\sum_{i=1}^{n}\sum_{j=1}^{m} C_{ij} + \sum_{i=1}^{n} f_i y_i$ .....(1)

$$T = \max\{t_{ij} x_{ij} : i = 1,2,\ldots,M; j = 1,2,\ldots,N\} \ldots\ldots\ldots\ldots\ldots\ldots..(2)$$

Constraints of the problem are $\sum_{i=1}^{n} x_{ij} = 1$, j=1..m .....(3)

$$x_{ij} \leq y_i \quad i=1,\ldots\ldots n,\ j=1\ldots\ldots m \quad \ldots\ldots\ldots\ldots\ldots\ldots..(4)$$

$$x_{ij},\ y_i = 0 \text{or} 1,\ i=1,\ldots n,\ j=1,\ldots m \quad \ldots\ldots\ldots\ldots\ldots\ldots..(5)$$

Where C_{ij} = *the cost meeting customer j's demand from facility i ;*

f_i = *the cost is established a facility at site i ;*

$$x_{ij} = \begin{cases} 1 if\ customer\ j\ is\ served\ from\ site\ i, \\ 0 \qquad\qquad otherwise; \end{cases}$$

$$y_i = \begin{cases} 1\ if\ a\ facility\ is\ established\ at\ i, \\ 0 \qquad\qquad otherwise\ ; \end{cases}$$

Note that the objective functions C provided by (1) and T provided by (2) are not accorded priorities. The constraint (3) ensures that one site can be selected from among the N possible sites while constraints (4) and (5) ensure that each site is assigned to a unique site which is selected for locating a customer. Note that x_{ij} is a binary variable because it is assumed that the demand of customer j, $j = 1, 2,\ldots,m$, is fulfilled by one and only one facility, say k (i.e. no partial fulfillment of demand is allowed), in which case $x_{kj} = 1$, $x_{ij} = 0$, for all $i = 1, 2,\ldots, n$, $i \neq k$. y_i is also a binary variable since a facility i is either established ($y_i = 1$), in which case the fixed charge f_i is incurred, or it is not established ($y_i = 0$) and in which case no fixed charge is incurred.

The UFLP can mathematically be decomposed into two interdependent subproblems:

1. Location. The facilities to be established are determined here (*i.e.*, y_i*'s*).
2. Allocation. For those established facilities, determines the distribution pattern (*i.e.*, x_{ij}*'s*).

For any solution of the location sub problem, an optimal solution of the allocation sub problem is easily obtained. More specifically, for any given y vector, an optimal assignment of the x_{ij}*'s* can obtained by using the following formula:

k *:* min C_{kj} , k= 1, ...n.

$$x_{ij} = \begin{cases} 1 & if\ i = k \\ 0 & otherwise \end{cases}$$

Therefore, if y_i*'s* are known, the x_{ij}*'s* can be optimally determined using the above argument. Krarup and pruzan[24] presented using a similer argument. We propose using the tabu search approach to find the optimal set of y_i*'s* .

3. The tabu search scheme

Tabu search is a Meta heuristic that guides local heuristic search procedures to explore the solution space beyond local optimality. It was introduced by Glover [13–15] specifically for combinatorial problems. Its basic ideas have also been proposed by Hansen [18] and Hansen and

Jaumard [19] with another name, "steepest ascent mildest descent". Since then, tabu search has successfully been applied to a wide range of problem settings. For example, tabu search has been applied to flow shop scheduling (Widmer and Hertz [30], Taillard [28]), architectural design (Bland and Dawson [5]), the time tabling problem (Hertz [20]), the interacting hub facilities location problem (Skorin-Kapov and Skorin-Kapov [27]), among others.

Tabu search is different from well-known hill-climbing local search techniques in the sense that it does not become trapped in local optimal solutions. This is due to the fact that tabu search allows moves out of a current solution which makes the objective function worse, in the hope that they might lead to a better solution.

The tabu search technique requires the following basic elements to be defined:

• *Configuration* is a solution or an assignment of values to variables.

• *A move* describes the process of generating a feasible solution to the combinatorial problem that is related to the current solution (i.e. a move is a procedure that describes how a new (trial) solution can be generated from the current one).

• *Set of candidate moves* is the set of all possible moves out of a current configuration.

• *Tabu restrictions*: This element is what distinguishes tabu search from other techniques. Tabu restrictions can be described as certain conditions imposed on moves which make some of them forbidden. These forbidden moves are known as *tabu*. It is done by forming a list of a certain size (called *tabu list*) that records these forbidden moves.

• *Aspiration criteria*: These are rules that override tabu restrictions, i.e. if a certain move is forbidden by tabu restriction, then the aspiration criterion, when satisfied, can make this move allowable .

Given the above basic elements, the tabu search scheme can be described as follows: start with a certain (current) configuration (i.e. initial solution), and evaluate the objective function for that configuration. Then, generate a certain set of candidate moves from the current configuration (if this set is too large, one could operate with a subset of it). If the best of these moves is not tabu or if the best is tabu but satisfies the aspiration criterion, then pick that move and consider it to be the new current configuration; otherwise, pick the best move that is not tabu and consider it to be the new current configuration. Repeat the procedure for a certain number of iterations. On termination, the best solution obtained so far is the solution obtained by the algorithm.

Note that the move that is picked at certain iteration is put on the tabu list so that it is not allowed to be reversed in the next little iteration. The taboo list has a certain size, and when the length of the tabu list reaches that size and a new move enters that list, then the first move on the tabu list is freed from being tabu and the process continues (i.e. the tabu list is circular). The size of the tabu list can be used to control

the intensity of the search, as will be seen later in section 5.3. The aspiration criterion can be taken as the value of the objective function, i.e., if the tabu move results in a value of the objective function that is better than the best known so far, then the aspiration criterion is satisfied and the tabu restriction is overridden by this.

In the above paragraph, we have outlined the basic steps of the tabu search procedure for solving combinatorial optimization problems. For a more complete description of this method, interested readers can refer to the papers of Glover [14, 15].

4. The proposed algorithm

As explained in section 2, we will concentrate on the location sub problem since it is the hard part of the problem to solve. Our proposed algorithm uses the tabu search technique to generate a set of y's and for every y, we use the argument in section 2 to get the optimal allocation (i.e., x_{ij}'s). The algorithm can be summarized as follows: use any heuristic to get an initial solution (i.e., y) (one can also use a random starting solution, or use the net benefit heuristic proposed below). Compute its total cost, and store it as the best solution so far. Using this solution, generate a set of neighboring y's. For every y, find the optimal allocation (i.e. x_{ij}'s) by the argument in section 2, and find the resulting total cost. Then the best improving y (or best non-Tabu if no improving y is found) is selected and its associated attribute is stored in the Tabu list. This procedure is repeated and controlled by tabu search until a predetermined number of iterations have been performed.

4.1. The net benefit heuristic *(NBH)*

To start the Tabu search algorithm, one needs to have an initial starting solution. It is preferred that this initial solution is of good quality and requires little computation time.

The NBH consists of two phases: the initial phase and the refinement phase. The *initial phase* proceeds as follows: for every demand j, $j = 1, 2,\ldots,m$, find the facility that can supply this demand at the cheapest transportation cost (i.e. find i^* such that $c_{i^*j} = \min_{1 \le i \le n} c_{ij}$, and establish this facility if it has not already been established). The result of the initial phase is the establishment of a certain number of facilities that will satisfy all demands (this of course constitutes an upper bound on the optimal solution).

The refinement phase is a procedure in which each facility that was established in the initial phase is investigated for possible closure, whereby a facility is closed if this can result in a net saving. This happens when the saving due to the closure of the facility (i.e. the fixed charge) is higher than the extra distribution cost incurred by allocating the demand originally allocated to this facility to the next cheapest established facility.

The NBH as shown in section 5.5 is very good in generating a good initial solution (i.e. 18.6% above the optimal on average) using a quite small running time. The NBH is stated below.

The NBH algorithm

Initial phase

Step 1:For each demand j, j = 1, 2,…, m, find the facility that can supply this demand at the cheapest distribution cost and denote it by i *.

For all j, $j = 1,\ldots, m$,

let $C_{i^*j} = \min_{1 \le i \le n} C_{ij}$, and form the set V of ordered pairs, $V = \{(i^*, j)\}$.

Step 2: Form the set I from V by extracting the first index. This is the list of facilities suggested for establishment.

Let I = {i * : (i*, j) ϵ V}.

Step 3: Evaluate the current solution which constitutes an upper bound one (UB1) on the optimal solution.

Calculate UB 1 = $\sum_{(i,j)\epsilon V} C_{ij} x_{ij} + \sum_{i\epsilon I} f_i y_i$

Step4: Evaluate the current solution which constitutes an upper bound two (UB2) on the optimal solution

Calculate UB2 = $\sum_{(i,j)\epsilon V} t_{ij} x_{ij}$

Refinement phase

Step 1: Set $k = 1$, let $m^* = |I|$ or m^* is the number of established facilities at the Initial phase.

Step 2: Consider the kth facility established in the initial phase.

Let i be the kth element in I.

Step 3: Let the set J be the indices of all those demands currently satisfied by facility i.

Let J = { j | (i, j) ϵV}.

Step 4: Repeat for all facilities established step 5, 6, 7

Step 5: For each demand currently satisfied by facility i, find which established facility Other than i can satisfy this demand at minimum distribution cost and compute the extra cost due to this reallocation.

For each $j \in J$, calculate $d_j = C_{lj} - C_{ij}$, where $C_{lj} = \min C_{tj}$, t ϵ I and t $\neq i$.

Step 6: Compute the extra cost which is equal to the cost of reallocation of demands Currently satisfied by facility i, less the saving due to the closure of facility i.

Calculate $\delta = \sum_{j \epsilon J} d_j - f_i$ create the array and calculate the maximum in the array.

Max = | δ[] |

Step 7: If the extra cost computed in step 5 is negative (i.e. benefit), then it is better To close the facility (step 8), otherwise, consider the next facility (step 9).

If $\delta < 0$, go to step 8; otherwise go to step 9.

Step 8(a): $I = I - \{i\}$. UB1 = UB1 – | δ |. If | I | = 1, stop; otherwise go to step 1.

Step 8(b): $I = I - \{i\}$. UB2 = UB2 – | δ |. If | I | = 1, stop; otherwise go to step 1.

Step 9: If $k = m^*$, stop; otherwise, set $k = k + 1$ and go to step 2.

Next, we formally present the proposed algorithm.

Statement of the un-capacitated facility location tabu search algorithm (UFLTSA)

Initialization step

Choose *nbhsize* (the size of the neighborhood). Choose a suitable size for the *tabu list*, *TL*. Choose *ITERMAX* (the maximum number of non improving iterations) (see section 5.3 for a description of these parameters as well as suitable values for them).

Use the NBH algorithm to get an initial *y*. Let $y_{current} = y$, $y_{min} = y$, $y_{best} = y$.

Let *totcost*(*y*) be the total cost of *y*.

Let *TL* =∅ , *BV* = *totcost*(*y*).

Let *k* = 1 and go to the main step.

Main step

Step 1 . Generate nbhsize random solutions from $y_{current}$. Each solution is evaluated and the solution with the minimum total cost among the generated solutions is selected as y_{min}.

min = ∞.

For *h* = 1 to *nbhsize*, do

Obtain *y* from $y_{current}$ (see section 5.1).

For every *j*, *j* = 1,…, *m*, find *k* : $\min_k C_{kj}$, *k* = 1,…, *n*.

Let $x_{kj} = 1$, $x_{ij} = 0$, for *i* =1,…, *n*, *i* ≠ *k*.

Evaluate *totcost*(*y*).

If *totcost*(*y*) < *min*, then *min* = *totcost*(*y*), $y_{min} = y$.

Step 2 . Check tabu status.

Check whether or not the solution ymin found in step 1 is in the tabu list.

2.1. l = 1.

2.2 If ($y_{min} \notin TL$) or ($y_{min} \in TL$ and *min* < *BV*), then go to step 3; otherwise,

replace *l* by *l* + 1. Let the *l*th best solution be y_{min} (i.e. this is the best

solution among all generated solutions in this iteration excluding those

Considered earlier in this step) and repeat this step.

Step 3: Update current solution.

Replace the current solution by the new solution ymin, and the best objective function (BV) by min. Store ymin in the tabu list.

Let *ycurrent = ymin*.

Store *ymin* in *TL* (see section 5.2).

If $min \geq BV$, then go to step 4; otherwise, $BV = min$, $k = 0$, and go to step 4.

Step 4: Check stopping criterion.

If the stopping criterion is satisfied, stop; otherwise, perform anotheriteration.

If k = ITERMAX, go to step 5; otherwise, replace k by k + 1, and go to step 1.

Step 5: Stop, and report the results.

Stop. y_{best} is the best solution found, and the corresponding total cost is BV.

Step 6: find t for y_{best} , x_{best}

5. Computational results and discussion

In this section, we discuss various implementations details and results of our computational experiments.

5.1. Generation of neighbors

In our experiments, we used the following scheme for generating a neighbor $y^{\wedge}$ for a given location vector y.

Let y_i be the ith component of the given vector y, where $1 \leq i \leq n$ and n is the number of facilities. To generate a neighbor y^, perform the following steps:

Step 1 . Let a be a given constant, where $0 < a < 1$. A is considered a parameter of this scheme, and will be called *the probability threshold*.

Step 2. Set $k = 1$.

Step 3 .Generate a random number, b ÎU(0, 1), where *U*(0, 1) is the uniform distribution between 0 and 1.

Step 4 . If b > a, go to step 5; otherwise, go to step 6.

Step 5. If *YK* = 1, set *y*ˆ *k* = 0, and if *yk* = 0, set *y*ˆ *k* = 1; stop.

Step 6 . If *k* = *n*, stop; otherwise set *k* = *k* + 1 and go to step 3.

5.2 Storing in the tabu list

In our algorithm, the chosen neighbor (i.e. the move) has to be stored in the tabu list, *TL*, in step 3. This can be achieved by storing the value of the index *k* at the termination of the above scheme for generation of neighbors.

5.3. Parameter setting

Tabu search is a parameter-sensitive technique similar to simulated annealing and genetic algorithms. As can be seen from the statement of the algorithm, it has the following parameters:

• The number of random solutions to be generated from the current one, *nbhsize*. Clearly, the larger the value of *nbhsize*, the better the quality of the solution, but the more computation time is needed. Hence, one has to balance these two conflicting objectives.

• The tabu list size, *TL*. If *TL* is large, the many moves are made tabu and hence their reversal is forbidden, which makes the search emphasize diversification. When the tabu list size *TL* is small, moves that are made tabu at a certain iteration are allowed to be reversed after few iterations, which makes the search emphasize intensification.

• The probability threshold, α. The value of α affects the probability assigned to every facility to change its status. The higher the value of α, the more even is the chance given to facilities to change their status.

• Maximum number of non improving iterations, *ITERMAX*. This is the maximum number of consecutive non improving iterations that the algorithm is allowed to go through before termination. Of course, the higher the value of *ITERMAX*, the better the quality of the solution one expects, but of course at the expense of more computation time. We have conducted an extensive parametric study on the proposed algorithm. We started this study by testing the performance of the algorithm at different values of the four parameters *nbhsize*, α, *TL*, and *ITERMAX*. Clearly, as one increases the values of *nbhsize*, α and *ITERMAX*, the quality of the solution is expected to improve but, of course, at the expense of more computation time. Therefore, one has to strike a balance between these conflicting objectives.

The results of this study show that the following set of values give the best compromise between n the quality of the solution and the computation time. The higher range values are needed for larger problems. They definitely work for all other problems, but they take a little extra time compared to using lower range values.

nbhsize = 5*n*,

$\alpha \in [0.9,0.95]$,

tabu list size, $TL \in [7,15]$,

$ITERMAX \in [10,50]$.

5.4. Results and discussion

Tables 1–3 show the results of our experiments. We coded the proposed algorithm in C++ and used an DELL personal computer equipped with intel microprocessor with a speed of 2GHzThe following notation is used in tables 1–3.

Data file = the name of the data file as in the OR Library (Satya Prakash [31]).

n = number of facilities.

m = number of potential locations.

NBH% = the percentage deviation of the NBH solution from optimal.

NBH-T = the CPU time needed by NBH.

UFLTSA% = the percentage deviation of the UFLTSA solution from optimal.

UFLTSA-T = the CPU time needed by UFLTSA.

Data File	n	m	NBH	NBH-T	UFLTSA	UFLTSA-T	Optimal	Time
input.txt	7	5	811.76	0.25	0.00	1.00	340.00	11.00
input.txt	7	5	155.00	7.00	0.00	7.00	830.00	6.00

Table-1

Data File	n	m	NBH	NBH-T	UFLTSA	UFLTSA-T	Optimal	Time
input.txt	7	5	474.07	0.25	0.00	1.00	540.00	11.00
input.txt	7	5	118.00	8.00	0.00	8.00	830.00	6.00

Table-2

Data File	n	m	NBH	NBH-T	UFLTSA	UFLTSA-T	Optimal	Time
input.txt	7	5	302.60	0.23	0.00	1.00	770.00	9.00
input.txt	7	5	86.00	7.00	0.00	7.00	830.00	6.00

Table-3

Conclusion:

A new algorithm for the un-capacitated facility location problem has been presented. The algorithm is based on the tabu search technique. Computational results show that our algorithm found the optimal solutions for all problems tested. One can extend the work by investigating other features for generating the random sequences and by using other elements of tabu search such as long-term memory, strategic oscillation, path re linking, among others. One could also improve the performance of the proposed heuristic for getting a good initial solution.

Future scope:

We can extend the objective function to multi objectives and improve the steps to solve the problem to add budget constraint.

Complexity: Complexity of this tabu search algorithm is $o(n^2)$.

References:

[1] M.L.Balinski, On finding integer solutions to linear programs, *Proceedings of IBM Scientific Symposium on Combinatorial Problems*, IBM, White Plains, NY, 1966, pp. 225–248.

[2] J.E. Beasley, OR-Library: *Distribution test problems by electronic mail*, Journal of perational Research Society 41(1990)1069–1072.

[3] J.E. Beasley, *Lagrangian heuristics for location problems*, European Journal of Operational Research 65(1993)383–399.

[4] O. Bilde and J. Krarup, *Sharp lower bounds and efficient algorithms for the simple plant location problem*, Annals of Discrete Mathematics 1(1977)79–97.

[5] J.A. Bland and G.P. Dawson, *Tabu search and design optimization, Computer-Aided Design* 23 (1991)195–201.

[6] A.R. Conn and G. Cornuéjols, *A projection method for the uncapacitated facility location problem, Mathematical Programming* 46(1990)273–298.

[7] G. Cornuéjols, G.L. Nemhauser and L.A. Wolsey, *The uncapacitated facility location problem, in: Discrete Location Theory*, eds. P.B. Mirchandani and R.L. Francis, Wiley, New York, 1990, pp.119–171.
[8] M.A. Efroymson and T.L. Ray, *A branch and bound algorithm for plant location*, Operations Research 14(1966)361–368.
[9] D. Erlenkotter, *A dual-based procedure for uncapacitated facility location*, Operations Research 14(1978)361–368.
[10] R.D. Galvao and L.A. Raggi, *A method for solving to optimality uncapacitated location problems*, Annals of Operations Research 18(1989)225–244.

[11] L.L. Gao and E.P. Robinson, Jr., *Uncapacitated facility location: General solution procedure and computational experience*, European Journal of Operational Research6(1994)410–427.
[12] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NPCompleteness*, Free man, San Francisco, 1979.
[13] F. Glover, *Future paths for integer programming and linkage to artificial intelligence*, Computers & Operations Research 13(1986)533–549.
[14] F. Glover, Tabu search – Part I, ORSA Journal on Computing 1(1989)190–206.
[15] F. Glover, Tabu Search – Part II, ORSA Journal on Computing 2(1990)4–32.
[16] M. Guignard, *A Lagrangian dual ascent algorithm for simple plant location problems*, European Journal of Operational Research 35(1977)193–200.
[17] M. Guignard and K. Spielberg, *Algorithms for exploiting the structure of the simple plant location problem*, Annals of Discrete Mathematics 1(1977)247–271.
[18] P. Hansen, *The steepest ascent mildest descent heuristic for combinatorial programming*, *Congress on Numerical Methods in Combinatorial Optimization*, Capri, Italy, 1986.
[19] P. Hansen and B. Jaumard, *Algorithms for the maximum satisfiability problem*, RUTCOR Research Report RR#43-87, Rutgers, New Brunswick, NJ, 1987.
[20] A. Hertz, Tabu search for large scale time tabling problems, European Journal of Operational Research 51(1991)39–47.
[21] R.L. Karg and G.L. Thompson, *A heuristic approach to solving travelling salesman problems*, Management Science 10(1964)225–248.

[22] B.M. Khumawala, *An efficient branch and bound algorithm for the warehouse location problem*, Management Science 18(1972)718–731.
[23] M. Körkel, *On the exact solution of large-scale simple plant location problems*, European Journal of Operational Research 39(1989)157–173.
[24] J. Krarup and P.M. Pruzan, *The simple plant location problem: Survey and synthesis*, European Journal of Operational Research 12(1983)36–81.
[25] J.K. Lenstra and A.H.G. Rinnooy Kan, *Complexity of packing, covering and partitioning problems*, in: *Packing and Covering Combinatorics*, ed. A. Schrijver, Mathematical Centre Tracts 106, Mathematical Centre, Amsterdam, 1979, pp. 257–291.
[26] H.P. Simao and J.M. Thizy, *A dual simplex algorithm for the canonical representation of the uncapacitated facility location problem*, Operations Research Letters 8(1989)279–286.
[27] D. Skorin-Kapov and J. Skorin-Kapov, *On tabu search for the location of interacting hub facilities*, European Journal of Operational Research 73(1994)502–509.
[28] E. Taillard, *Some efficient heuristic methods for the flow shop sequencing problem*, European Journal of Operational Research 47(1990)65–74.
[29] D.W. Tcha, H.B. Ro and C.B. Yoo, *A dual-based add heuristic for uncapacitated facility location*, Journal of the Operational Research Society 39(1988)873–878.
[30] M. Widmer and A. Hertz, *A new heuristic method for the flow shop sequencing problem*, European Journal of Operational Research 41(1989)186–193.
[31*]* Satya Prakash *Selection of warehouse sites for clustering ration shops to them with two objectives througha heuristic algorithm incorporating tabu search*