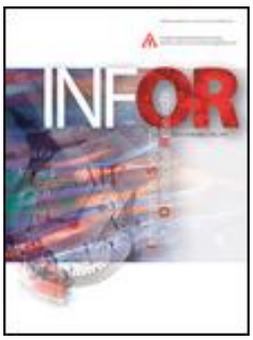

INFOR: Information Systems and Operational Research

ISSN: 0315-5986 (Print) 1916-0615 (Online) Journal homepage: http://www.tandfonline.com/loi/tinf20

A Fast Algorithm For The Greedy Interchange For Large-Scale Clustering And Median Location Problems

R.A. Whitaker

To cite this article: R.A. Whitaker (1983) A Fast Algorithm For The Greedy Interchange For Large-Scale Clustering And Median Location Problems, INFOR: Information Systems and Operational Research, 21:2, 95-108, DOI: 10.1080/03155986.1983.11731889

To link to this article: http://dx.doi.org/10.1080/03155986.1983.11731889

Published online: 25 May 2016.

A FAST ALGORITHM FOR THE GREEDY INTERCHANGE FOR LARGE-SCALE CLUSTERING AND MEDIAN LOCATION PROBLEMS*

R.A. WHITAKER

Central Statistics Bureau, Ministry of Industry and Small Business Development, Victoria, B.C.

ABSTRACT

A fast algorithm is described for implementing the greedy interchange heuristic for use in solving large scale clustering and uncapacitated median location problems. Computational experience is reported for these algorithms on a number of large randomly generated networks and on some difficult problem sets, and comparisons with some other implementations are made. An additional heuristic is proposed for solving these set partitioning problems based on an efficient procedure for achieving the interchange.

RÉSUMÉ

Cette étude décrit un algorithme très efficace qui fait intervenir une méthode heuristique connu sous le nom "greedy interchange." Ceci est utilisé dans la résolution de problèmes impliquant la recherche de groupement pour des populations de grande taille et par les problèmes où l'on cherche à situer des entrepôts. On possède une assez grande ensemble de résultats informatiques sur l'efficacité de ces algorithmes à résoudre de nombreux problèmes de graphes selectionnées aléatoirement et sur des ensembles de problèmes plutôt complexes. On a aussi procédé à des comparaisons avec quelques autres applications qui ont été faites. Une autre méthode heuristique est proposée pour solutionner des problèmes de partition d'ensembles selon la procédure performante pour accomplir "l'interchange."

In this paper an efficient algorithm is described for implementing the greedy interchange heuristic for solving large-scale clustering and p median location problems. The problem is of the form

$$\text{minimize } z = \sum_I \sum_J b_i d_{ij} x_{ij} \tag{1}$$

$$\text{subject to} \sum_J x_{ij} = 1 \text{ all } i, \tag{2}$$

$$\sum_J jj = p \tag{3}$$

$$x_{ij} \leqslant x_{jj} \text{ all } i, j, \tag{4}$$

$$x_{ij} = (0, 1) \text{ all } i, j, \tag{5}$$

where d_{ij} is a measure of the distance or dissimilarity between data point

*Received March 1981; revised April 1982

(node) i and j; b_i is a weight at the node (data point) i; $x_{jj} = 1$ signifies a median at point (node) j; I is the set of all n nodes or data points; J is the set of possible median locations; and p is the number of medians (or clusters) required. The weight b_i at the nodes is usually 1 for clustering problems but may be > 1 for median location problems. The objective of this model is to select p medians (locations) from among the set J so that the sum of the aggregate (weighted) distances from all n points to their respective medians is minimized. Assigning each data point to its nearest median defines the clusters for clustering problems and the service areas for median location problems.

Recent optimizing algorithms for solving this problem as reported by Cornuejols, Fisher, and Nemhauser[2] and Mulvey and Crowder[7], using Lagrangian relaxation, and by Nauss and Markland[8], using a derivative of Erlenkotter's[3] very effective dual based procedure, have greatly expanded the problem size for which optimal solutions can now be found. The computational experience reported by these various authors, however, suggests that problems with many hundreds or thousands of data points (as is not uncommon especially in clustering problems) remain beyond the computational reach of these optimizing methods, leaving heuristic algorithms as the only viable solution option currently available for these larger data sets. In this context the greedy interchange algorithm has been shown to be a useful and effective solution procedure. The greedy algorithm has been proposed for median location problems by Kuehn and Hamburger[6] and Church and Revelle[1], and for clustering problems by Mulvey and Crowder[7]. Cornuejols et al.[2] have provided a formal statement of this algorithm. The interchange heuristic that attempts to improve the greedy or any other initial solution by a systematic interchange of locations not in solution with those in the median set has been applied to p median problems by Teitz and Bart[11], Rushton and Kohler[10], Hodgson[4], Rosing et al.[9], and others.

Cornuejols et al.[2] (hereafter CFN) have provided an excellent theoretical discussion of the greedy interchange heuristic, including worse-case analysis and computational results. In extensive empirical testing with this heuristic Whitaker[12] found that the greedy interchange algorithm generated optimal solutions to 47% of more than 300 problems solved with a worst case result of 8.3%, but a mean percentage error of less than 1% over all the problems under study. These results suggest that the upper bounds provided by this heuristic are generally, though not invariably good. In the discussion that follows, implementation of the greedy interchange algorithms are compared to the formulations of some other researchers in the particular context of the p median location problem.

NOTATION

For purposes of exposition let the following definitions pertain:

G = a network
n = the number of network nodes
m = the number of possible facility sites
p = the number of facilities required
M = the set of all possible facility sites
P^* = the median set, a subset of M
P = the set of possible facility sites not in solution, that is, that subset of M not in P^*
S^* = the minimum cost solution for supplying the network G (objective function)
k = an iteration parameter
∞ = an arbitrarily large number
d_{ij} = the weighted distance (time, cost) from node i to potential facility j
u_i^k = the weighted distance from node i to nearest facility j, $j \in P^*$ during iteration k
w_i^k = the weighted distance from node i to second nearest facility j, $j \in P^*$ during iteration k
c_j^k = the potential value of S* if facility j is added to P^* during iteration k
S_r = the value of S^* resulting from the addition to P^* of node r
S_{rt} = the potential increase (decrease) to S^* resulting from the interchange of node r with facility t
I, J, K_j = subsets of the n nodes as defined
a, b, z, q = indexes and parameters as defined
$\emptyset$ = an empty set.

THE GREEDY ALGORITHM

The execution of this algorithm is achieved by means of a simple recursion. Locations are added sequentially, one per iteration until $k = p$ facilities have been chosen. On each iteration the criterion for choice is to pick that location that maximizes the incremental decrease to the current solution value. Stated in the context of the p median problem this algorithm is as follows:

ALGORITHM 1. *Greedy following* CFN.
Step 0. Initialization. Set $P^* = \emptyset$, $k = 1$ and let $u_i^1 = \infty$ for all $i = 1, 2, \ldots, n$.

Step 1. Let

$$c_j^k = \sum_{i=1}^{n} \min\,(d_{ij},\, u_i^k)$$

for all $j \in M$, $j \notin P^*$.
Step 2. Find some vertex r, $r \in M$, $r \notin P^*$ such that

$$S_r = \min_{j \notin P^*} c_j^k.$$

Step 3. Set $P^* = P^* \cup r$. If $k = p$, go to step 5; otherwise, if $k < p$, go to step 4.
Step 4. Set $k = k + 1$ and let $u_i^k = \min\,(d_{ir}, u_i^{k-1})$ for $i = 1, 2, \ldots, n$. Go to step 1.
Step 5. Set $S^* = S_r$. STOP.

This description parallels the notation and implementation of the related greedy maximizing problem considered by CFN[2]. To achieve an acceleration on this algorithm for rapid solution to problems where n, m, and p are large, we note that on all iterations where $k > 1$ only q nodes, where $q < n$, are reassigned as a result of the addition of the kth facility. Consequently the algorithm can be restructured such that on iteration k step 1 is executed with respect only to the set of q nodes reassigned on iteration $k - 1$. Since q decreases monotonically as k increases towards p, convergence will be rapid. This accelerated greedy algorithm can be stated as:

ALGORITHM 2. *Fast greedy.*
Step 0. Initialization. Set $P^* = \emptyset$, $I = \emptyset$ and $k = 1$. Set $c_j^0 = c_j^1 = 0$ for $j \in M$ and let $u_i^0 = 0$ and $u_i^1 = \infty$ for $i = 1, 2, \ldots, n$. Next set $I = I \cup i$ for $i = 1, 2, \ldots, n$.
Step 1. Let

$$c_j^k = [\sum_{i \in I} \min\,(d_{ij}, u_i^k)] + [c_j^{k-1} - \sum_{i \in I} \min\,(d_{ij}, u_i^{k-1})]$$

for all $j \in M$, $j \notin P^*$.
Step 2. Find vertex r, $r \in M$, $r \notin P^*$ such that

$$S_r = \min_{j \notin P^*} c_j^k.$$

Step 3. Set $P^* = P^* \cup r$. If $k = p$, go to step 5; otherwise, if $k < p$, set $I = \emptyset$ and go to step 4.
Step 4. Set $k = k + 1$. Update $u_i^k = \min\,(d_{ir}, u_i^{k-1})$ for $i = 1, 2, \ldots, n$. If $d_{ir} < u_i^{k-1}$ then set $I = I \cup i$ for all such $i \in G$. Go to step 1.
Step 5. Set $S^* = S_r$. STOP.

Comment. The set I defines the set of q nodes reassigned to the kth facility added during iteration k.

The Interchange Algorithm

This algorithm is initialized with any arbitrary or other starting solution such as the greedy method. If, by interchanging any location not in solution with some location that is in solution, a diminished value for S^* results, then an improved upper bound has been found. The interchange proceeds iteratively until no further improvements occur. Following Teitz and Bart[11] (hereafter T&B), the originators of this algorithm, it is usual to implement the interchange with respect to the first successful entering location, while deleting from solution the median in P^* that maximizes the overall improvement resulting from this switch. Simplifying the interchange relationships described by T&B (p. 959) and Hodgson[4], correcting for minor errors in the original description (see Rushton and Kohler[10]), and following Rushton and Kohler in their interpretation of the T&B procedures for updating u_i^k and w_i^k after each interchange, this algorithm can be stated as:

Algorithm 3. *The interchange algorithm of Teitz and Bart.*
Step 0. Initialization. Input start solution S^* and median vector P^*. Set $q = a = 0$, $k = 1$, $b = m - p$ and let $S = S^*$. Define $P = M - P^*$.
Step 1. Find vertices x and y, $x, y \in P^*$ such that

$$u_i^k = d_{ix} = \min_{j \in P^*} d_{ij} \text{ and } w_i^k = d_{iy} = \min_{j \in P^*, j \neq x} d_{ij} \text{ for } i = 1, 2, \ldots, n.$$

Step 2. If $q = b$, go to step 5; otherwise, if $q < b$, set $q = q + 1$ and go to step 3.
Step 3. *Set* $r = P_q$. For r, the qth node in P, find some vertex t, $t \in P^*$ such that

$$S_{rt} = \min_{j \in P^*} \Big[\sum_{i \in I} [\min (d_{ir}, u_i^k) - u_i^k] + \sum_{i \in J} [\min (d_{ir}, w_i^k) - u_i^k] \Big],$$

where

$$I = \{ \text{all } i \in G: d_{ij} > u_i^k \}$$

and

$$J = \{ \text{all } i \in G: d_{ij} = u_i^k \}$$

Step 4. If $S_{rt} \geq 0$, go to step 2; otherwise, if $S_{rt} < 0$, set $k = k + 1$ and let $S^* = S^* + S_{rt}$ Interchange r and t such that $r \in P^*$, $r \notin P$ and $t \in P$, $t \notin P^*$. Set $P_q = t$ and go to step 1.

Step 5. Set $a = a + 1$. If $S > S^*$, set $S = S^*$, $q = 0$ and go to step 2; otherwise, if $S = S^*$, STOP.

The parameter a defines the number of interchange cycles and there are $k - 1$ interchanges in all. A fast algorithm for implementing this interchange can be achieved by restructuring steps 1 and 3. First the interchange test phase of the Teitz-Bart algorithm (step 3) requires p distinct loops to evaluate all p interchange tests and possible switches for each of the $m - p$ entering locations. Each such loop or test requires $2n$ comparisons and $2n$ additions and subtractions. It is possible to structure the interchange test such that all p possible interchanges with respect to some entering location are evaluated within a single loop requiring no more than $2n$ additions and subtractions and $3n + p$ comparisons in all (algorithm 4, step 1). These savings are attractive, since the computation of this interchange test remains essentially invariant as the size of p increases. Secondly the procedures of T&B for updating the nearest and second nearest medians in P^* for each network node following an interchange (algorithm 3, step 1), and as implemented by Rushton and Kohler[10] require $3np$ comparisons. In particular, the vector of second nearest medians is recalculated for all nodes with respect to the set P^*, which leads almost invariably to some unnecessary calculation. Such redundancies can be eliminated and the updates performed with far fewer comparisons on average by considering all possible relationships for some node i between its nearest and second nearest medians and those of the interchanged vertices r and t (algorithm 4, step 3). In general, actual update changes will be restricted to those localized areas on the network in the neighbourhood of the interchanged vertices, and the more extended search among P^* for a second-nearest median is restricted to a small subset of the n network nodes when p is > 2. Finally, we note that a cycle as defined by T&B is a complete pass through the set P containing all possible facility sites not currently in solution. Any interchange effected during the course of some cycle gives rise to a new and exhaustive search through P on the subsequent cycle. However, the algorithm can be terminated following any $m - p$ consecutive iterations of the interchange step without an interchange, regardless of whether the current cycle is complete or not.

ALGORITHM 4. *Fast interchange.*
Step 0. Initialization. Input start solutions S^* and median vector P^*. Set $k = z = q = 1$, and let $b = m - p$. Define $P = M - P^*$. Find vertices x and y, $x, y \in P^*$ such that

$$u_i^1 = d_{ix} = \min_{j \in P^*} d_{ij} \text{ and } w_i^1 = d_{iy} = \min_{j \in P^*, j \neq x} d_{ij} \text{ for } i = 1, 2, \ldots n.$$

Step 1. Set $r = P_q$. For r, the qth node in P, find some vertex t, $t \in P^*$ such that

$$S_{rt} = [\sum_{i \in I} (d_{ir} - u_i^k)] + \min_{j \in P^*} [\sum_{i \in K_j} [\min (d_{ir}, w_i^k) - u_i^k]]$$

where

$$I = \{\text{all } i \in G\colon d_{ir} < u_i^k\}$$

and

$$K_j = \{\text{all } i \in G\colon d_{ir} \geq u_i^k \text{ and } d_{ij} = u_i^k\}.$$

Step 2. If $S_{rt} \geq 0$, go to step 4; otherwise if $S_{rt} < 0$, set $k = k + 1$ and let $S^* = S^* + S_{rt}$. Interchange r and t such that $r \in P^*$, $r \notin P$, and $t \in P$, $t \notin P^*$. Set $P_q = t$, $z = 0$ and go to step 3.

Step 3. Update u_i^k and w_i^k the nearest and second nearest medians for node i as follows. If $d_{it} > u_i^{k-1}$, then update

a. $u_i^k = \min (d_{ir}, u_k^{k-1})$. Next let
b. s, $s \in P^*$ be the median for node i such that $d_{is} = u_i^k$. Then set
c. $w_i^k = u_i^{k-1}$, if $d_{ir} \leq u_i^{k-1}$, or set
d. $w_i^k = \min (d_{ir}, w_i^{k-1})$, if $d_{ir} > u_i^{k-1}$ and $d_{it} > w_i^{k-1}$, or set
e. $w_i^k = \min_{j \in P^*, j \neq s} d_{ij}$, if $d_{ir} > u_i^{k-1}$ and $d_{it} = w_i^{k-1}$.

Otherwise, if $d_{it} = u_i^{k-1}$, then update

a. $u_i^k = \min d_{ir}, w_i^{k-1})$. Next let
b. s, $s \in P^*$ be the median for node i such that $d_{is} = u_i^k$. Then set
c. $w_i^k = w_i^{k-1}$ if $d_{ir} \leq w_i^{k-1}$ or set
d. $w_i^k = \min_{j \in P^*, j \notin s} d_{ij}$ if $d_{ir} > w_i^{k-1}$

Repeat step 3 for all $i = 1, 2, \ldots, n$.

Step 4. If $z = b$, go to step 5; otherwise if $z < b$, set $z = z + 1$, $q = q + 1$ and go to step 1.

Step 5. Set $a = q/b$ and STOP.

In implementing the interchange phase of the algorithm (step 1) we note that since

$$|I| + \sum_{j=1}^{p} |K_j| = n,$$

some $2n$ additions and subtractions and less than $3n + p$ comparisons are required to evaluate all p possible interchanges with respect to some entering location. Hence this interchange procedure is much faster than step 3 algorithm 3 in executing the interchange test. In updating u_i^k and

w_i^k, the nearest and second nearest distance vectors for each $i \in G$ following an interchange (step 3), some $2n$ comparisons are required for all u_i^k and less than $3n + 2\propto p$ comparisons for all w_i^k where $\propto$ is an indeterminate but small fraction of n. For $p > 2$ these results compare favourably with the $3np$ comparisons required for the update phase of the Teitz and Bart interchange algorithm.

Comparative Analysis

In order to compare the computational attributes of these alternatives computer codes were written and and the algorithms variously tested on some randomly generated networks with $n = m = 100$, 200, and 500 nodes. To construct each network n random x, y co-ordinates were generated and the Euclidean distances calculated between every pair of points. The weights (demands) at the nodes were assumed to be identical in each case. For each of the networks problems were solved by each algorithm for a variety of values of p. Summarizing the results of these experiments (table 1) we note that:

1. The fast greedy interchange algorithm (fast interchange initialized by fast greedy) was in excess of an order of magnitude faster than the CFN greedy – Teitz and Bait interchange heuristic for solving a number of the problems under study. The fast greedy algorithm was nearly six times faster than the greedy implementation of CFN for the largest problem solved by both procedures ($n = 500$, $p = 50$). The interchange algorithm itself, excluding the greedy initialization, was at least ten and in some instances more than thirty times faster than T&B in effecting the interchange for all problems with $p > 10$. Since the computational advantages of these methods increase with each increase in the sizes of n, m, and p ($p < m/2$), these new procedures appear well suited for solving clustering or p median problems on networks which are large.
2. The relative timings for the interchange test and update phases of the interchange algorithms (steps 3 and 1 in algorithm 3 and steps 1 and 3 in algorithm 4) are shown in columns 5 and 6. For given values of n and m computation times per cycle for the interchange test phase of the fast interchange algorithm remain relatively stable or decrease slightly as the value of p increases. Computationally, this phase of the heuristic reduces to a quadratic function of the parameters n, m, and p for each complete cycle of the algorithm, so that doubling the size of these parameters requires only a fourfold increase in execution time. With the methods of T&B cycle times for the interchange test phase rise as p increases to peak at $p \simeq m/2$, at which point the number of possible locations in solution is balanced by those not in the median set. Overall,

computation times per cycle for the fast interchange algorithm were far less variable for a given data set with respect to the size of p than were the procedures of T&B.

3. It is interesting to note that the fast interchange algorithm required less time than the fast greedy initialization for all but four of the problems solved. Unlike the greedy algorithm, there is no known polynomial time running bound for the interchange heuristic, since the number of cycles required for convergence is not known a priori at the outset of any run. For the problems under study, however, no more than four complete cycles were required to reach a stable solution regardless of the size of n and p.
4. In conducting these experiments on the randomly generated networks execution of the fast interchange algorithm was not terminated until completion of the final cyle; that is, the notion of terminating during this cycle following $m - p$ consecutive iterations of the interchange step without an interchange was not implemented for these particular test results. Hence, it is probable that for most of the problems solved, the results presented in table 1 significantly overstate both the number of cycles and times required for convergence of the fast interchange as described in algorithm 4.

To test the impact of this further additional saving and at the same time to investigate the sensitivity of the algorithm to changes in the structure of some of the coefficients of the model, some additional experiments were subsequently conducted, using the 100-node network problem described by Krolak et al.[5]. For this network 80 median problems were solved in the range of $p = 2$ to 81, using the fast greedy interchange algorithm. In the first run the weights (demands) were assumed to be identical at each node. For the second run weights in the range of 10–99 were randomly generated for each node. In each case execution of the algorithm was terminated following $m - p$ consecutive iterations of the interchange step without an interchange, and in each of the 160 problems solved this condition occurred before completion of the final cycle.

The results of these runs are shown in tables 2 and 3. Summarizing these results we note that average and maximum running times and numbers of cycles were greater for the problem set with variable weights at the nodes, but these differences were not judged to be of particular significance, since all problems were solved in less than two-tenths of a second. In only two of the 160 problems solved were execution times greater for the interchange algorithm than for the greedy initialization. Overall, timings for this set of tests were less than for the randomly generated network with $n = 100$ for comparable values of p, despite greater average numbers of interchanges. These results likely reflect the additional savings gained by terminating execution before completion of the final cycle.

TABLE 1

GREEDY-INTERCHANGE ALGORITHMS – COMPUTATIONAL EXPERIENCE*

(A) Fast greedy–fast interchange algorithms

$n \times m$	p	Fast greedy	Fast greedy–fast interchange	Interchange test phase	Update phase	No. of interchanges	No. of cycles	Interchange time per cycle
100 × 100	5	0.088	0.186	0.094	0.003	4	3	0.033
	10	0.104	0.211	0.094	0.011	16	3	0.035
	20	0.123	0.197	0.059	0.011	15	2	0.037
	50	0.149	0.224	0.060	0.006	8	3	0.025
	75	0.162	0.185	0.011	0.000	0	1	0.022
200 × 200	5	0.401	0.680	0.272	0.005	3	2	0.140
	10	0.473	0.891	0.405	0.010	7	3	0.139
	20	0.544	0.852	0.265	0.035	25	2	0.154
	50	0.645	1.169	0.467	0.040	27	4	0.131
	75	0.691	1.047	0.305	0.027	17	3	0.119
500 × 500	5	2.376	5.010	2.574	0.053	15	3	0.878
	10	2.844	5.535	2.586	0.094	29	3	0.897
	20	3.268	6.860	3.419	0.151	45	4	0.898
	50	3.798	6.572	2.523	0.204	61	3	0.925
	75	4.063	6.746	2.425	0.190	56	3	0.894

TABLE 1 (concl.)

(B) CFN greedy–Teitz & Bart interchange algorithms

$n \times m$	p	CFN greedy	CFN greedy–T&B interchange	Interchange test phase	Update phase	No. of interchanges	No. of cycles	Interchange time per cycle
100 × 100	5	0.808	0.421	0.331	0.008	4	3	0.114
	10	0.157	0.831	0.644	0.058	16	3	0.224
	20	0.298	1.113	0.712	0.090	15	2	0.408
	50	0.625	2.392	1.635	0.124	8	3	0.589
	75	0.789	1.200	0.399	0.000	0	1	0.411
200 × 200	5	0.334	1.294	0.945	0.013	3	2	0.480
	10	0.658	3.392	2.678	0.052	7	3	0.911
	20	1.288	4.995	3.354	0.345	25	2	1.854
	50	2.964	17.037	13.200	0.855	27	4	3.518
	75	–	–	–	–	–	–	–
500 × 500	5	2.269	11.965	9.528	0.162	15	3	3.232
	10	4.500	23.359	18.302	0.546	29	3	6.286
	20	8.893	> 60.000	nc	nc			nc
	50	21.534	> 150.000	nc	nc			nc
	75	–	–	–	–	–	–	–

*Computations on Amdahl 470 V/6 in seconds excluding I/O with FortranH Compiler
nc: Problems not completed
– Problem not attempted

TABLE 2

COMPUTATIONAL RESULTS* – TIMINGS ON 100-NODE NETWORK

	Identical nodal weights			Random nodal weights		
	Greedy – interchange	Greedy	Interchange	Greedy – interchange	Greedy	Interchange
Average time	0.153	0.113	0.040	0.160	0.112	0.048
Maximum time	0.173	0.137	0.079	0.186	0.138	0.097
Minimum time	0.094	0.049	0.009	0.097	0.049	0.017
Selected p						
5	0.149	0.070	0.079	0.122	0.069	0.053
10	0.144	0.085	0.059	0.150	0.083	0.067
15	0.147	0.094	0.053	0.164	0.092	0.072
20	0.165	0.100	0.065	0.182	0.099	0.083
25	0.155	0.105	0.050	0.160	0.105	0.055
50	0.169	0.122	0.047	0.165	0.122	0.043
75	0.156	0.135	0.021	0.159	0.136	0.023

*In seconds on Amdahl 470 V/6. Number of problems = 80.

TABLE 3

COMPUTATIONAL RESULTS* – INTERCHANGE CHARACTERISTICS ON 100-NODE NETWORK

	Identical nodal weights		*Random nodal weights*	
	Number of interchanges	Number of cycles	Number of interchanges	Number of cycles
Average	9	1.91	15	2.13
Maximum	19	3.12	25	3.98
Minimum	1	1.07	4	1.64
Selected p				
5	16	3.12	10	2.07
10	12	2.28	10	2.64
15	15	1.96	20	2.63
20	19	2.46	22	3.16
25	12	1.96	13	2.31
50	12	2.33	20	1.80
75	5	1.84	7	1.80

*Number of problems = 80.

A STEPWISE GREEDY INTERCHANGE HEURISTIC

The attractive computational results of the fast interchange procedure prompted some further experiments in which the interchange test was systematically inserted following the addition of each new facility to the median set during execution of the greedy algorithm. This sequence of add location, interchange, add location gives rise to a procedure that can be called the stepwise greedy interchange heuristic. In solving for a given value of p with this algorithm all other solutions in the range of $p = 1$ to $p - 1$ are sequentially generated. Hence the initialization of any set of p

medians will generally be tighter than the starting solution given by the greedy algorithm alone. While a tight initialization does not necessarily lead to a lower solution value, it seems likely, in any large number of problems solved, that a significant number of better solutions will be generated. To test this assertion a comparative analysis of the greedy interchange and stepwise greedy interchange heuristics was conducted on the 33-, 57-, and 100-city test problems discussed by CFN. With identical weights at the nodes, successive p-median solutions were generated by each algorithm in the range $p = 2$ to 26 for the 33-node problem, $p = 2$ to 56 for the 57-city problem, and $p = 2$ to 81 for the 100-city problem. In all a total of 160 problems were solved by each heuristic. Overall, the stepwise greedy interchange heuristic found 77 better and 33 worse solutions than the greedy interchange algorithm. Computationally, timings for the stepwise greedy interchange algorithm ranged from 0.010 seconds ($p = 2, n = 100$) to 3.268 seconds with $n = 100$ and $= 80$. These results suggest that this algorithm is a useful additional heuristic for solving clustering and p-median location problems on networks that are large, particularly when p is small relative to n.

Conclusions

Some fast algorithms for implementing the greedy interchange heuristic have been described. Computational tests on networks of up to 500 nodes in size confirm the utility of these procedures for solving large-scale clustering and p median location problems. Utilizing the accelerated interchange, a stepwise greedy interchange algorithm has been proposed as a useful supplementary heuristic for solving these set partitioning problems where p is small and data sets are large.

References

(1) R.L. Church and C.S. Revelle, "Theoretical and computational links between the p-median, location set covering and the maximal covering location problem," Geographical Analysis, 8, 1976, 406–15.

(2) G. Cornuejols, M.L. Fisher, G.L. Nemhauser, "Location of bank accounts to optimize float: an analytical study of exact and approximate algorithms," Management Science, 23, 1977, 789–810.

(3) D. Erlenkotter, "A dual based procedure for uncapacitated facility location," Operations Research, 26, 1978, 992–1009.

(4) M.J. Hodgson, "Toward more realistic allocation in location-allocation models: an interaction approach," Environment and Planning A, 10, 1978, 1273–85.

(5) P. Krolak, W. Felts, and G. Marble, "A man machine approach towards solving the travelling salesman problem," Comm. ACM, 14, 1971, 327–34.

(6) A.A. Kuehn, and M.J. Hamburger, "A heuristic program for locating warehouses," Management Science, 9, 1963, 643–66.

(7) J.M. Mulvey, and H.P. Crowder, "Cluster analysis: an application of lagrangian relaxation," Management Science, 25, 1979, 329–40.

(8) R.M. Nauss, and R.E. Markland, "Theory and application of an optimizing procedure for lock box location analysis," Management Science, 27, 1981, 855–65.
(9) K.E. Rosing, E.L. Hillsman, and H. Rosing-Vogelaar, "The robustness of two common heuristics for the p median problem," Environment and Planning A, 11, 1979, 373–80.
(10) G. Rushton, and J.A. Kohler, "ALLOC: heuristic solutions to multi-facility location problems on a graph," in G. Rushton, M.C. Goodchild, and L.A. Ostresh, Jr, *Computer Programs for Location-Allocation Problems*. Department of Geography, Monograph No. 6, University of Iowa, 1973.
(11) M.B. Teitz, and P. Bart, "Heuristic methods for estimating the generalized vertex median of a weighted graph," Operations Research, 16, 1968, 955–61.
(12) R.A. Whitaker, "Some interchange algorithms for median location problems," Environment and Planning B, 9, 1982, 119–29.