Pergamon

Location Science, Vol. 5, No. 4, pp. 207–226, 1997
© 1998 Elsevier Science Ltd
All rights reserved. Printed in Great Britain
0966-8349/98 $19.00 + 0.00

PII:S0966-8349(98)00030-8

VARIABLE NEIGHBORHOOD SEARCH FOR THE *P*-MEDIAN

P. HANSEN* and N. MLADENOVIĆ

GERAD and École des Haute Études Commerciales, 3000, chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

Abstract—Consider a set L of potential locations for p facilities and a set U of locations of given users. The p-median problem is to locate simultaneously the p facilities at locations of L in order to minimize the total transportation cost for satisfying the demand of the users, each supplied from its closest facility. This model is a basic one in location theory and can also be interpreted in terms of cluster analysis where locations of users are then replaced by points in a given space. We propose several new Variable Neighborhood Search heuristics for the p-median problem and compare them with Greedy plus Interchange, and two Tabu Search heuristics. © 1998 Elsevier Science Ltd. All rights reserved

Key words: heuristics, variable neighborhood search, location, p-median.

1. INTRODUCTION

Consider a set L of m potential locations for p facilities and a set U of locations of n given users. The p-median problem is to locate simultaneously the p facilities at locations of L in order to minimize the total transportation cost for satisfying the demand of the users, each supplied from its closest facility. This model is a basic one in location theory; see Mirchandani and Francis, 1990, for an introduction to discrete location theory.

The p-median and extensions of it often model real world situations such as the location of industrial plants, warehouses and public facilities; see e.g., Christofides, 1975 for a list of applications. The p-median can also be interpreted in terms of cluster analysis: locations of users are then replaced by points in an m-dimensional space; see Hansen and Jaumard, 1997 for a survey of cluster analysis from a mathematical programming viewpoint. Moreover, p-median can be defined as a purely mathematical problem: given an $n \times m$ matrix D, select p columns of D in order that the sum of minimum coefficients in each line within these columns be the smallest possible.

The p-median problem is NP-hard (Kariv and Hakimi, 1979). Many heuristics and exact methods have been proposed for solving it. Exact methods were developed by Beasley (1985) and Hanjoul and Peeters (1985), among others. Numerous references to work on this and related problems are given in Cornuejols *et al.* (1977), Brandeau and Chiu (1989) and Mirchandani and Francis (1990).

Despite this extensive work, there is still room for new or improved heuristics and more efficient exact algorithms for p-median. Indeed, when instances become large, in terms of m, n and p, one observes a marked deterioration in performance of many heuristics, as well as a rapid increase in solution time of exact methods. In this paper, we consider heuristics and, in particular, study how the new Variable Neighborhood Search metaheuristic (Mladenović,

*Author to whom correspondence should be addressed.

1995; Mladenović and Hansen, 1997) can be applied. Variable Neighborhood Search proceeds by a descent method to a local minimum, then explores, systematically or at random, increasingly distant neighborhoods of this solution. Each time, one or several points within the current neighborhood are used as an initial solution for a local descent. One jumps from the current solution to a new one if and only if a better solution has been found. Variable Neighborhood Search is not a trajectory method (as simulated annealing or Tabu Search) and does not specify forbidden moves. Despite its simplicity, it proves to be effective.

Classical heuristics for p-median, often cited in the literature, are the construction method 'Greedy' (Kuehn and Hamburger, 1963), and the improvement methods 'Alternate' (Maranzana, 1964) and Interchange (Teitz and Bart, 1968). In the Greedy method, a first facility is located in such a way as to minimize the total cost for all users. Facilities are then added one by one until the number p is reached; each time the location that most reduces total cost is selected. In the first iteration of Alternate, facilities are located at p points chosen in L, users assigned to the closest facility and 1-median problem solved by enumeration for each facility's set of users. Then the procedure is iterated with the new locations of the facilities until no more changes in assignments occur. The name Alternate comes from the fact that this heuristic alternately locates the facilities and then allocates users to them. The Interchange procedure is commonly used as a standard to compare with other methods. Here a usually random pattern of p facilities is given initially; then, facilities are moved iteratively, one by one, to vacant sites with the objective of reducing total cost. This local search process is stopped when no movement of any single facility decreases the value of the objective. In the multistart versions of Alternate and Interchange, the procedure is repeated a given number of times and the best solution kept.

Several hybrids of the heuristics described above have been suggested in the literature. For example, in the GreedyG heuristic (Captivo, 1991), in each step of Greedy, the Alternate procedure is run. A combination of Alternate and Interchange heuristics has been suggested in Pizzolato (1994). In Moreno *et al.* (1991), a variant of reverse Greedy is compared with Greedy plus Alternate and with Multistart Alternate, etc. The combination of Greedy and Interchange, where the Greedy solution is chosen as the initial one for Interchange, has been most often used for comparison with other newly proposed methods; see for example, Voss (1996). In the computer results section below we will do the same.

Another type of heuristic suggested in the literature is based on the compact dual of the integer programming formulation of p-median and uses variants of the well-known Dual Ascent heuristic, DUALOC (Erlenkotter, 1978). Such heuristics for solving the p-median problem are proposed in Galvão (1980) and in Captivo (1991). However, computer results on randomly generated 400 customer problems, reported in Captivo (1991), show that the dual approach is not better on average than the Interchange heuristic, although it is twice as fast.

Three different Tabu Search (Glover, 1989, 1990; Hansen and Jaumard, 1990) heuristics have recently been proposed for solving$p$-median; see Glover and Laguna (1993) for an introduction to Tabu Search. In Mladenović *et al.* (1996) a 1-interchange move is extended into a so-called 1-chain-substitution move, where both facilities currently in the solution (L_{in}) and facilities out of it (L_{out}) are divided into non-tabu and tabu subsets (L_{in}^{n}, L_{in}^{t}, $L_{\text{out}}^{\text{n}}$, $L_{\text{out}}^{\text{t}}$). Interchange of two facilities is performed by changing positions of four of the facilities in the list: the facility that goes out is chosen from L_{in}^{n} and replaced with one that belongs to the solution as well (i.e., from L_{in}^{t}), but whose turn has come to change tabu status. In its place comes a facility from $L_{\text{out}}^{\text{n}}$, which is substituted by an element from $L_{\text{out}}^{\text{t}}$. Finally, the chain is closed by placing the facility from L_{in}^{n} in $L_{\text{out}}^{\text{t}}$. In that way, Tabu Search recency based memory is easily exploited, i.e. the possibility of getting out of the local optima trap is achieved without additional effort. Another Tabu Search heuristic is proposed by Voss (1996), where a few variants of the so-called reverse elimination method are discussed. In Rolland *et al.* (1996), a 1-interchange move is divided into add and drop moves which do not necessarily follow each other and so feasibility is not necessarily maintained during the search. This approach, within Tabu Search, is known as strategic oscillation; see Glover (1989, 1990).

Regarding comparison of classical heuristics, Whitaker (1983) made an important study, which does not seem to be very well known within the scientific community. Among other results reported is that Add and Interchange moves have similar complexity. Moreover, p times less operations are spent for one Fast Interchange move proposed there than for the one Interchange move of Teitz and Bart (1968). In several recent papers concerning Interchange, Whitaker's result is ignored. For example, Rolland *et al.* (1996) claim in their conclusion: 'first, the search only considers ADD and DROP moves; thereby, eliminating the more computationally burdensome SWAP moves included in traditional exchange heuristics for the p-median problem'. In addition, the computer results reported in Whitaker (1983) show that Fast Interchange is even faster than the Fast Greedy proposed in the same paper. This fact was not known to Captivo (1991), where, in several tables that compare p-median heuristics, Greedy is more than four times faster than Interchange. In the appendix of this paper we develop an efficient implementation of Whitaker's Fast Interchange method.

It is well known that p-median instances with $n \leq 500$ and $p \leq 50$ can be solved exactly in reasonable time, for example, by the algorithm of Hanjoul and Peeters (1985). Moreover, Beasley (1985), using a modern mainframe with vector processing capabilities, extended the size of exactly solvable problems to $n = 900$ and $p = 90$. It should be noted that exact solution methods are very time consuming when not only n, but also p, is large. Still, it appears that quite a few of the p-median problems reported in the literature could possibly be solved by exact methods; for example, instances reaching $n = 300$ and $p = 150$ (Mladenović *et al.* 1995), $n = 389$ and $p = 85$ (Pizzolato, 1994), $n = 400, p = 133$ (Captivo, 1991), $n = 500, p = 20$ (Rolland *et al.* 1996), $n = 500$, $p = 167$ (Voss, 1996), $n = 1000$, $p = 10$ (Moreno *et al.*, 1991). However, larger problems require good heuristics, and good heuristic solutions may be useful as initial solutions in exact methods.

In overview, the main objectives of our study are: (i) to propose a new heuristic for solving p-median, within the variable neighbourhood search metaheuristic framework (Mladenović, 1995; Mladenović and Hansen, 1997) and to compare it with some recent heuristics based on the Tabu Search approach; (ii) to make clear what benefits can be obtained from the 1-Interchange move, and to develop an efficient implementation of the Fast Interchange local descent method proposed by Whitaker (1983); (iii) to conduct an extensive empirical study with the new Variable Neighborhood Search heuristic and some of its extensions. Standard test problems from the ORLIB library will be used for that purpose but we will also consider much larger problem instances than previously reported in the literature. A detailed sensitivity analysis on several parameters will be done, in order to better understand why some heuristics, in particular Variable Neighborhood Search ones, are more efficient than others.

The remainder of this paper is organized as follows. In the next section, p-median is formulated mathematically. A new basic parameter-free Variable Neighborhood Search heuristic is developed in Section 3 and compared experimentally with two Tabu Search methods in Section 4. Larger problem instances than previously reported in the literature are taken from the TSP library (Reinelt, 1991). They have $n = 1400$ and $n = 3038$ with $p \in (10, 500)$. Extensive empirical analysis on 40 ORLIB test problems (Beasley, 1985) is reported in Section 5. Conclusions are drawn in Section 6.

In the Appendix we discuss advantages of using the Interchange neighborhood structure (or vertex substitution if p-median is expressed in terms of an underlying graph) in the search. Pseudo-code of an efficient implementation of this local search method is also given.

2. PROBLEM FORMULATION

Consider a set L of m facilities (or location points), a set U of n users (or customers or demand points) and a $n \times m$ matrix D with the distances traveled (or costs incurred) for satisfying the demand of the user located at i from the facility located at j, for all $j \in L$ and $i \in U$. The objective is to minimize the sum of these distances (or transportation costs), i.e.

$$\mathrm{Min} z = \sum_{i \in U} \min_{j \in J} d_{ij}. \qquad ()$$

where $J \subseteq L$ and $|J| = p$. Beside this combinatorial formulation, the p-median problem has an integer programming one:

$$\mathrm{Min} z = \sum_{i} \sum_{j} d_{ij} x_{ij} \qquad (1)$$

subject to

$$\sum_{j} x_{ij} = 1, \forall_{i}, \qquad (2)$$

$$x_{ij} \leq y_j, \ \forall i,j, \qquad (2)$$

$$\sum_{j} y_j = p, \qquad (4)$$

$$x_{ij}, \ y_j \in \{0,1\} \qquad (5)$$

where $y_j = 1$ signifies a facility is located at j and $y_j = 0$ otherwise; $x_{ij} = 1$ if user i is assigned to facility j and $x_{ij} = 0$ otherwise. Constraint (2) expresses that the demand of each user must be met. Constraint (3) prevents any user from being supplied from a site with no open facility. The total number of open facilities is set to p by (4).

In the remainder of the paper we discuss heuristics for problems (1) to (5) and compare them on large instances. These heuristics often use a basic move, i.e. 1-Interchange. As mentioned in the introduction, this move can be implemented efficiently, following the ideas of Whitaker (1983). We will work in a slightly different way, using a best improvement strategy instead of a first improvement one. As the analysis of this point is rather technical, it is given in the Appendix.

3. A VARIABLE NEIGHBORHOOD SEARCH HEURISTIC FOR THE *P*-MEDIAN

Variable Neighborhood Search is a recently proposed metaheuristic for solving combinatorial problems (Mladenović, 1995; Mladenović and Hansen, 1997). The basic idea is to proceed to a systematic change of neighborhood within a local search algorithm. Exploration of these neighborhoods can be done in two ways. The smallest ones, i.e. those closest to the current solution, may be explored systematically until a solution better than the incumbent is found. The largest ones, i.e. those far from the current solution, may be explored partially by drawing a solution at random and beginning a (variable neighborhood) local search from there. The algorithm remains at the same solution until another solution better than the incumbent is found and then jumps there. Neighborhoods are usually ranked in such a way that solutions are explored increasingly far from the current one. In this way, intensification of the search around the current solution is followed naturally by diversification. The level of intensification or diversification can be controlled through a few easy to set parameters. We may view Variable Neighborhood Search as a 'shaking' process, where movement to a neighborhood further from the current solution corresponds to a harder shake. Unlike random restart, Variable Neighborhood Search allows a controlled increase in the level of the shake.

Let us denote with $X = \{x|x = \text{set of } p \text{ (out of } m) \text{ locations of facilities}\}$ a solution space for the problem. We say that the distance between two solutions x_1 and x_2 ($x_1, x_2 \in X$) is equal to k if and only if they differ in k locations. Since X is a set of sets, a (symmetric) distance function ρ can be defined as

$$\rho(x_1, x_2) = |x_1 \backslash x_2| = |x_2 \backslash x_1|,\ \forall x_1, x_2 \in X. \tag{6}$$

It can easily be checked that ρ is a metric function in X, thus, X is a metric space. The neighborhood structures we use are induced by metric ρ, i.e., k locations of facilities ($k \le p$) from the current solution are replaced by k others. We denote with $\mathcal{N}_k$ $k = 1,\dots,k_{\max}$ ($k_{\max} \le p$) the set of such neighborhood structures and with $\mathcal{N}_k(x)$ the set of solutions forming neighborhood $\mathcal{N}_k$ of a current solution x. More formally

$$x' \in \mathcal{N}_k(x) \Leftrightarrow \rho(x_1, x_2) = k. \tag{7}$$

Note that the cardinality of $\mathcal{N}_k(x)$ is

$$|\mathcal{N}_k(x)| = \left(\frac{p}{k}\right)\left(\frac{m-p}{k}\right), \tag{7}$$

since k out of p facilities are dropped and k out of $m-p$ added into the solution. This number first increases then decreases with k. Note also that sets $\mathcal{N}_k(x)$ are disjoint ($\mathcal{N}_k(x) \cap \mathcal{N}_l(x) = \emptyset, \forall l,k, l \ne k$), and their union, together with x, gives X, i.e.

$$x \bigcup_{k=1}^{p} \mathcal{N}_k(x) = X,\ 1 + \sum_{k=1}^{p} |\mathcal{N}_k(x)| = \left(\frac{m}{p}\right) = |X|. \tag{7}$$

Pseudo-code for our Variable Neighborhood Search heuristic for the p-median problem is given in Fig. 1. In step (2a) the incumbent solution x is perturbed in such a way that $\rho(x,x') = k$. Then x' is used as initial solution for Fast 1-Interchange in step (2b). If a better solution than x is obtained, we move there and start again with small perturbations of this new best solution (i.e. $k \leftarrow 1$). Otherwise, we increase the distance between x and the new randomly generated point, i.e. we set $k \leftarrow k+1$. If k reaches $k_{\max}$ ($k_{\max}$ is the single parameter of the basic version of the heuristic), we stop. Alternatively, we return to Main step (1), i.e. the main step can be iterated until some other stopping condition is met (e.g. maximum number of iterations, maximum CPU time allowed, or maximum number of iterations between two improvements). Note that the point x' is generated at random in step (2a) in order to avoid cycling, which might occur if any deterministic rule was used.

In order to get a parameter-free version of the heuristic just described, we set the value of the single parameter $k_{\max}$ to p. We conjectured that a very simple Variable Neighborhood Search heuristic would give solutions of good quality, based on the fact that all good p-median solutions are 'relatively' close to each other (with respect to distance ρ) and this conjecture turned out to be true. A similar property is shown to hold for some other combinatorial optimization problems; see for example Boese *et al.* (1994), for the Traveling Salesman and Graph Bisection problems, and literature mentioned there. In other words, there exist locations for facilities that take part in almost all local minima obtained by the Interchange procedure. We can consider them 'good' locations. Note that such information is lost in Multistart Interchange. In Fig. 2 we show the existence of good locations on test problem #15 from the ORLIB library Beasley (1985). Each point (u_i, v_i) on the graph in Fig.

```
Initialization
    (1) Find arrays $x_{opt}$, c1 and c2 and $f_{opt}$ as in initialization of the Interchange heuristic (see
    Appendix); (2) the set of neighbourhood structures $\mathcal{N}_k$ $k = 1, \ldots, k_{max}$ is induced by distance
    function $\rho$; (3) copy initial solution into the current one, i.e., copy $f_{opt}, x_{opt}$, c1 and c2 into
    $f_{cur}, x_{cur}, c1_{cur}$ and $c2_{cur}$ respectively.
Main step.
    . (1) $k \leftarrow 1$;
    . (2) Until $k = k_{max}$, repeat the following steps:
        (2a) Shaking
            (* Generate a solution at random from the $k^{th}$ neighbourhood *)
            For $j = 1$ to $k$, do the following:
                . Take facility to be inserted (goin) at random;
                . Find facility to be deleted (goout) by using procedure
                  Move $(c1_{cur}, c2_{cur}, d, x_{cur}, goin, n, p, u, goout)$ (see Appendix);
                . Find $c1_{cur}$ and $c2_{cur}$ for such interchange, i.e., run subroutine
                  Update $(d, goin, goout, n, p, c1_{cur}, c2_{cur})$ (see Appendix),
                . Update $x_{cur}$ and $f_{cur}$ accordingly;
            End for $j$
        (2b) Local search.
            Apply the Interchange heuristic (without Initialization step), with $x_{cur}$, $c1_{cur}$
            and $c2_{cur}$ as input and output values (see Appendix); denote the corresponding
            objective function value with $f_{cur}$;
        (2c) Move or not.
            If $f_{cur} < f_{opt}$ then
                (* Save current solution to be incumbent; return to $\mathcal{N}_1$ *)
                $f_{opt} \leftarrow f_{cur}$; $x_{opt} \leftarrow x_{cur}$; c1 $\leftarrow c1_{cur}$; c2 $\leftarrow c2_{cur}$; and set $k \leftarrow 1$
            else
                (* Current solution is incumbent one; change the neighbourhood *)
                $f_{cur} \leftarrow f_{opt}$; $x_{cur} \leftarrow x_{opt}$; $c1_{cur} \leftarrow$ c1; $c2_{cur} \leftarrow$ c2; and set $k \leftarrow k+1$
            End if
    End for $k$
```

Fig. 1. Pseudo-code for Variable Neighborhood Search heuristic for p-median.

2 refers to (at least) one local minimum x_i. They are obtained by the Interchange routine, which was run 10,000 times. The origin represents the known optimal solution. The symbol u_i represents the distance between the optimal solution and the local minimum ($u_i = \rho(x_{\text{opt}}, x_i)$), while v_i means the difference in their objective function values [$v_i = f(x_i) - f(x_{\text{opt}})$]. It appears that the local minimum farthest from the global one is at distance 33. As this is 1/3 of the maximal distance, we conclude that many locations are good and that all local minima are relatively close to one another.

Exploiting the property of closeness of local minima, the basic Variable Neighborhood Search heuristic secures two important advantages: (i) by staying in the incumbent the search is continued in an attractive area of the solution space X; (ii) as some of the locations are good (i.e., they are sites for facilities as in the optimal solution), local search in step (2b) uses several times less iterations than in the initialization step (or in Multistart Interchange); consequently, we may visit several high quality local optima in the same CPU time that one descent Interchange spends to visit only one when started from a random initial solution in X.

4. COMPUTATIONAL RESULTS

The tests on all problems described in this section are done in the same way: (i) the problem is first solved by the Greedy heuristic (G); (ii) the solution so obtained is used as the initial

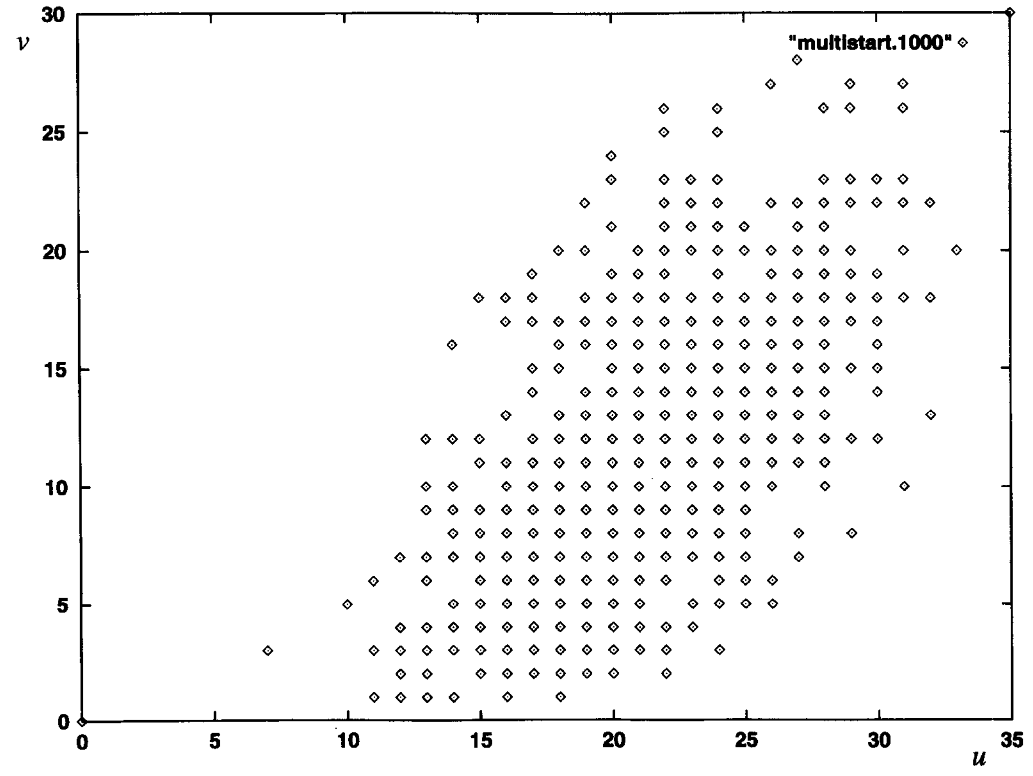

Fig. 2. Local minima obtained by 10,000 restarts of Interchange on test problem #15 (n = 300, p = 100).

solution for the Fast Interchange heuristic (G+I) explained in the Appendix; (iii) the solution so obtained is used as the initial solution for all methods tested. However, if a known optimal solution is obtained by Greedy or Greedy plus Fast Interchange, the comparisons are not performed.

The basic Variable Neighborhood Search heuristic is compared with two recent methods based on the Tabu Search approach: Tabu Search where the reverse elimination method is used for diversifying the search (Voss, 1996), which is called Tabu Search 1 (TS-1) and Tabu Search where the so-called 1-chain-substitution move is used in the search (Mladenović *et al.*, 1996); which is called Tabu Search 2 (TS-2).

All methods compared (except Tabu Search 1) are coded in FORTRAN and run on a SUN station 10. In order to decrease running time, compiling is done with the optimizing option, i.e., with 'f77 -cg92 -O4'.

4.1. ORLIB test problems

We first tested the basic Variable Neighborhood Search heuristic on 40 ORLIB problems from Beasley (1985), where the set of facilities is equal to the set of users. The problem parameters range from instances with $n = 900$ and $p = 5$, 10, 90. All these test problems are solved exactly (Beasley, 1985) on a modern mainframe with vector processing capabilities (i.e., on a Cray-1S computer), which makes them suitable for computational comparisons. In order to obtain matrix D that is used by the Interchange procedure, an All Shortest Paths algorithm is run first (the CPU time for this $O(n^3)$ procedure is not included in the tables below). Table 1 gives results of comparing Greedy (G), Greedy plus Fast Interchange (G+I), Tabu Search 1 (the best results reported in Voss, 1996), Tabu Search 2 and our basic Variable Neighborhood Search. All procedures stop when the number of iterations reaches 750, i.e., when 750 neighborhoods are visited. Therefore, computing time allocated to all methods is almost the same. In column 4, known optimal values (f_{opt}) are reported, followed by values obtained by Greedy and Greedy plus Fast Interchange. The % errors of the four methods are reported in columns 7–10. They are calculated as

$$\frac{f - f_{\text{opt}}}{f_{\text{opt}}} \cdot 100. \tag{7}$$

Columns 11–14 of Table 1 give the number of iterations when the best solution found by each method is obtained (and not when the algorithm stops). It appears that all three best methods improve significantly the quality of the solution obtained by Greedy plus Interchange. We first discuss total error for the first 25 test problems, as results for these problems are only reported in Voss (1996). All but two test problems are solved exactly by Variable Neighborhood Search with a total error of 0.07%; all but three by Tabu Search 2 with a total error of 0.57%; all but four by Tabu Search 1 with a total error of 0.78%. The last column in Table 1 shows the CPU time needed per iteration of the Interchange algorithm. Comparing minimum and maximum CPU time obtained (i.e. 0.01 and 0.55 s respectively) with 1 and 24 s (the minimum and maximum CPU time reported in Voss (1996), we conclude that not only the difference in computer speed (50 MHZ versus 33 MHZ) but also the Fast Interchange procedure causes such a big difference in CPU time. It should be noted that the basic version of Variable Neighborhood Search is compared with the best Tabu Search versions proposed in Mladenović *et al.* (1996) and Voss (1996).

Table 1. Comparison of methods for solving 40 ORLIB test problems. Results for 18 instances solved exactly by Greedy plus Fast Interchange are not reported (i.e. problems 1,3,5,6,...,38,39)

			Objective function			% Error				Iterations				
Pr. no.	n	p	f_{opt}	G	G+I	G+I	VNS	TS-1	TS-2	G+I	VNS	TS-1	TS-2	CPU/iter.
2	100	10	4093	4118	4105	0.29	0.00	0.00	0.00	2	95	76	73	0.01
4	100	20	3034	3088	3046	0.40	0.00	0.00	0.00	2	28	16	16	0.01
7	200	10	5631	5646	5645	0.25	0.00	0.00	0.00	1	100	—	10	0.08
8	200	20	4445	4472	4457	0.27	0.00	0.00	0.00	3	50	154	422	0.06
9	200	40	2734	2841	2753	0.69	0.00	0.00	0.00	7	221	301	748	0.05
10	200	67	1255	1295	1263	0.64	0.00	0.00	0.00	6	56	92	37	0.05
14	300	60	2968	3013	2970	0.07	0.03	0.00	0.00	10	146	373	518	0.11
15	300	100	1729	1761	1737	0.46	0.00	0.40	0.23	10	254	20	739	0.12
18	400	40	4809	4873	4811	0.04	0.00	0.00	0.00	6	158	652	446	0.22
19	400	80	2845	2899	2859	0.49	0.04	0.11	0.18	15	602	33	230	0.23
20	400	133	1789	1861	1802	0.73	0.00	0.11	0.00	19	350	57	48	0.23
22	500	10	8579	8670	8669	1.05	0.00	0.00	0.00	1	68	87	44	0.55
24	500	100	2961	3009	2967	0.20	0.00	0.00	0.00	12	272	571	250	0.39
25	500	167	1828	1896	1843	0.82	0.00	0.16	0.16	20	281	122	115	0.37
Total						6.62	0.07	0.78	0.57	114	2681	2554	3696	
28	600	60	4498	4580	4513	0.33	0.00	—	0.11	15	144	—	402	0.55
29	600	120	3033	3108	3039	0.20	0.00	—	0.03	24	543	—	586	0.57
30	600	200	1989	2037	2009	1.01	0.15	—	0.30	15	458	—	408	0.57
32	700	10	9297	9331	9301	0.04	0.00	—	0.00	3	24	—	5	0.74
33	700	70	4700	4798	4714	0.30	0.00	—	0.06	17	390	—	288	0.76
34	700	140	3013	3097	3034	0.70	0.00	—	0.27	22	493	—	264	0.86
37	800	80	5057	5121	5063	0.12	0.00	—	0.02	17	58	—	346	1.07
40	900	90	5128	5190	5141	0.25	0.04	—	0.06	19	699	—	226	1.47
Total						9.35	0.26	—	1.42	246	5325		6223	

4.2. *TSP-LIB test problems*

In the next table, Greedy plus Interchange, Variable Neighborhood Search and Tabu Search 2 are compared on large problem instances taken from TSPLIB (Reinelt, 1991). To the best of our knowledge, these are much larger problem instances for p-median than previously reported in the literature. Since optimal solutions are not known, the third column of Table 2 contains the best value found by the three methods. As a stopping condition for the two best methods, we used the maximum number of iterations, which are set to 20 and 5 times the number of iterations used by Interchange before stopping for FL1400 and PCB3038 respectively.

It appears that Variable Neighborhood Search is as good or better than Tabu Search 2 in all problems, and strictly better in all but three. Both of these algorithms are substantially better than Greedy plus Interchange, although it is faster.

5. EMPIRICAL ANALYSIS

In order to answer the question of why Variable Neighborhood Search works well, an extensive sensitivity analysis was performed on the 40 ORLIB (Beasley, 1985) test problems considered above.

5.1. *Effect of initial solution*

The choice of the initial solution is important for every local search heuristic method. In Table 1 the initial solution obtained by Greedy and then by Interchange (G + I) was used for all methods compared. In the next table, the total error (on all 40 ORLIB problems) is given when the initial solution is chosen as follows: (i) the same as in Table 1, i.e., Greedy plus Interchange; (ii) Greedy only (G); (iii) at random, followed by Interchange (RND + I) where the average total error for 10 random initial solutions are reported. In each run the procedure stops when the number of iterations reaches 750. We observe the following:

(a) the Interchange method does not need to start with the solution obtained by Greedy (compare 9.35 with 9.41 total % error in Table 3);

(b) the quality of the solutions obtained by Tabu Search 2 depends significantly on the initial solution, i.e., the best choice is to start with the solution of Greedy plus Interchange;

(c) Variable Neighborhood Search outperforms Tabu Search 2, i.e. the worst average value obtained by Variable Neighborhood Search is better than the best average value found by Tabu Search 2.

Table 2. Comparison in solving FL1400 and PCB3038 problems from TSPLIB

		Objective function values			% Error			# Iterations			
n	p	Best kn.	G	G+I	G+I	VNS	TS-2	G+I	VNS	TS-2	CPU/iter.
1400	10	101248.13	106529.47	101248.24	0.00	0.00	0.00	9	187	183	2.10
	20	57856.32	63280.80	57902.89	0.08	0.00	0.00	17	340	86	2.13
	30	44086.53	45891.75	44897.69	1.84	0.00	0.72	20	377	86	2.27
	40	35005.82	37026.66	35021.43	0.04	0.00	0.02	28	503	471	2.24
	50	29176.45	31088.08	29647.61	1.61	0.00	0.49	21	360	409	2.60
	60	25176.47	26626.12	25469.12	1.16	0.00	0.49	31	596	476	2.48
	70	22186.14	23402.51	22430.20	1.10	0.00	0.07	33	642	564	2.59
	80	19900.66	20905.06	19991.94	0.46	0.00	0.02	35	688	479	2.67
	90	18055.94	18858.27	18196.59	0.78	0.00	0.35	38	713	305	2.73
	100	16551.20	17385.65	16582.50	0.19	0.00	0.00	46	875	510	2.69
	150	12035.56	12592.97	12120.59	0.71	0.00	0.31	83	1585	1526	2.71
	200	9362.99	9901.69	9454.08	0.97	0.00	0.36	92	1738	1308	2.98
	250	7746.96	8147.45	7790.34	0.56	0.00	0.23	89	1703	1515	2.35
	300	6628.92	6886.06	6667.50	0.58	0.00	0.32	68	1312	1205	3.12
	350	5739.28	5935.70	5780.22	0.71	0.00	0.44	55	1021	584	3.21
	400	5045.84	5225.32	5096.78	1.01	0.00	0.36	48	931	471	3.87
	450	4489.93	4621.35	4511.76	0.49	0.00	0.08	58	1108	681	3.64
	500	4062.86	4139.91	4082.05	0.47	0.00	0.24	50	970	928	3.51
Total					12.76	0.00	4.50	821	15649	11787	
3038	50	507809.5	540848.5	509167.2	0.27	0.00	0.24	76	362	283	11.34
	100	354488.7	370042.3	357924.9	0.97	0.00	0.35	106	480	468	11.90
	150	281911.9	297022.2	283559.8	0.58	0.00	0.34	141	684	529	12.34
	200	239086.4	252445.2	240831.6	0.73	0.00	0.40	160	777	604	12.97
	250	209718.0	222100.3	210663.0	0.45	0.00	0.21	177	892	627	13.63
	300	188142.3	198937.2	189249.2	0.59	0.00	0.11	176	853	854	14.22
	350	171726.8	180777.9	172838.7	0.65	0.00	0.09	204	1002	919	14.97
	400	157910.1	166024.7	158835.5	0.59	0.00	0.03	215	1039	1023	14.54
	450	146087.8	153593.6	146724.6	0.44	0.00	0.07	259	1281	1139	15.55
	500	136081.7	142890.9	136711.2	0.46	0.00	0.17	266	1280	1329	16.07
Total					5.73	0.00	2.01	1780	8650	7775	

Table 3. Comparison of three methods by total % error on all 40 ORLIB problem instances, when applied to different initial solutions

	% Error			
Initial solution	I	VNS	TS-2	Number of trials
G+I	9.35	1.04	1.42	40
G	9.35	1.26	4.99	40
RND+I	9.41	0.53	4.27	400
Total	28.11	2.83	10.68	480

We then analyzed further the results obtained by Variable Neighborhood Search (column 3 in Table 3) to see whether the difference in error for various initial solutions (1.04, 1.26 and 0.53%) occurs because Variable Neighborhood Search has difficulties in getting out of a deep local minimum reached by Greedy. Starting from the initial solution obtained by Greedy plus Interchange, we ran all 40 test problems 10 times, but using different seed values for the pseudo-random number generator (the one we used is the so-called GGUBFS generator; see Carraghan and Pardalos, 1990). In that way, different solutions are generated around the same initial solution. The average error (on 400 tests) was 0.51%. Minimal and maximal error were 0.26% (reported in Table 1 as well) and 1.04% (reported in Table 3 and Table 4), respectively. Comparing 0.51% with 0.53% (the average error on 400 tests also obtained using Random plus Interchange to set an initial solution), we conclude that the quality of the solution obtained by Variable Neighborhood Search does not depend significantly on an initial solution, on average. However, the number of iterations needed to get to the optimal solution is very sensitive to both the seed value and initial solution for each particular problem, but again there is no significant difference in average results. The same tests were performed on Tabu Search 2, where a 1.60% average error is obtained (1.26% and 2.04% were minimal and maximal error respectively). Comparing 1.60% with 4.27% leads to conclusion (b) above.

5.2. *Effect of the number of iterations*

In previous tests, the stopping condition for all methods (except Interchange) was set to 750 iterations. In the next series of experiments we wanted to check how the number of iterations

Table 4. Comparison of Multistart Interchange, Variable Neighborhood Search and Tabu Search 2 methods when different numbers of iterations are used as stopping criteria

Maximum number of iterations	% Error			Number of problems solved (out of 40)		
	MI	VNS	TS-2	MI	VNS	TS-2
100	5.50	6.86	6.10	23	21	23
250	4.94	2.20	2.38	24	26	27
500	3.15	1.36	1.75	27	28	29
750	2.57	1.04	1.56	28	33	30
1,000	2.39	0.24	1.56	28	36	30
3,000	1.79	0.06	0.83	31	38	31
5,000	1.51	0.04	0.15	32	39	36
10,000	1.20	0.02	0.06	32	39	38

influenced the quality of the solution. For that purpose, the maximum number of 1-interchanges was set to 10,000. In Table 4, % error and the number of problems solved exactly (out of 40) are reported for different numbers of iterations as stopping criteria. Three methods are compared: Multistart Interchange (MI), Variable Neighborhood Search (VNS) and Tabu Search 2 (TS-2).

First we briefly report on the results obtained by Multistart Interchange (the detailed table is omitted to save space). The average number of iterations per random initial solution of Multistart Interchange ranges from 5 (in nine test problems where $p = 5$) to 115 (in problem #30 where $p = 200$). The number of iterations per call directly correlates with p. The quality of the solution obtained by Multistart Interchange depends on p as well. Eight problems that Multistart Interchange did not solve after 10,000 iterations are: 15, 19, 25, 29, 30, 34, 37 and 40. In all of them p is large. This suggests the existence of the so-called central limit catastrophe (Baum, 1986): when problems grow large, random local minima are almost surely of 'average' quality and increasing the number of restarts does not help. In order to determine whether this holds for Multistart Interchange, 100,000 iterations were allowed in solving equation (15) and equation (19), but the solution was not improved. This weakness of the multi-start approach has been observed earlier in some other combinatorial problems; see, for example, Boese *et al.*, 1994, for Traveling Salesman and Graph Bisection problems.

From Table 4, the following observations follow:

(a) Objective function values obtained by Multistart Interchange decrease faster than those obtained by both Variable Neighborhood Search and Tabu Search 2 when the maximum number of iterations is set to 100; this corresponds to around 9 restarts on average, as the total number of iterations Interchange used for 22 test problems reported in Table 1 was 246, i.e., $246/2 \approx 11$ and $11.9 \approx 100$;

(b) Multistart Interchange suffers from the central limit catastrophe as increasing the number of iterations does not improve solutions enough to make them competitive with those obtained by the two other methods;

(c) Tabu Search 2 continues to improve the quality of the solution even when the number of iterations becomes very large (but results remain worse than with Variable Neighborhood Search).

5.3. *Sensitivity to k_{max}*

As noted before, $k_{\max}$ is the single parameter used in the basic version of Variable Neighborhood Search. In order to get a parameter-free Variable Neighborhood Search, $k_{\max}$ was set to p in all previous experiments. In Table 5, different $k_{\max}$ values are chosen and used in all 40 ORLIB test problems. The stopping condition is again set to 750 iterations, while the initial solution corresponds to the first p users.

Table 5. Sensitivity to $k_{\max}$

	$k_{\max}$									
	1	2	4	8	10	15	20	25	p	m
% Error	2.95	1.26	1.23	0.79	0.21	0.37	0.48	0.57	0.25	0.24
# Exact	26	29	30	32	34	34	34	31	36	36
# Iterat.	6692	7040	6869	4901	7975	6160	6043	5325	6018	6989

Table 6. Sensitivity to b

	b							
	1	2	3	4	5	10	20	50
% Error	0.25	0.25	0.46	0.32	0.54	0.41	0.64	0.97
# Exact	36	35	34	33	33	33	31	30
# Iterat.	6018	7047	6564	7522	7219	7391	8507	10073

It appears that the worst results are obtained when only one neighborhood is used in the search ($k_{\max} = 1$), i.e., only 26 (out of 40) test problems are solved exactly with a total error of 2.95%. When $k_{\max}$ is increased, results are better up to $k_{\max} = 10$; increasing $k_{\max}$ over 10, the % error oscillates, but is never above 0.60%.

5.4. *Sensitivity to b*

In the basic Variable Neighborhood Search heuristic only one random solution from the kth neighborhood was chosen. Obviously, b (a parameter) points could be selected at random in Step (2a) of the algorithm as initial ones for the local search procedure, i.e. steps (2a) and (2b) could be iterated b times and the best solution retained. In the next series of experiments on 40 test problems, b is set to 1, 2, 3, 4, 5, 10, 20 and 50. Results are reported in Table 6. The initial solution as well as the stopping condition are the same as in the previous table.

From Table 6 one can see that the best average results are obtained for $b = 1$, and that the quality of the solution decreases with b. However, even for $b = 50$, the % error is below 1%. Thus, selection of a point from different neighborhoods is more powerful than considering a few from the same neighborhood.

5.5. *Sensitivity to k_l and k_{step}*

Another easy extension of the basic Variable Neighborhood Search heuristic is to introduce k_l and k_{step}, two parameters that control the change of neighborhood process, i.e., in the basic Variable Neighborhood Search heuristic set $k \leftarrow 1$ and $k \leftarrow k + k_{\text{step}}$ instead of $k \leftarrow k + 1$. In this way, intensification and diversification of the search is achieved in an easy and natural way. Large values for k_l and/or k_{step} ensure diversification while small ones for k_{step} entail intensification.

We found that the choice of both k_l and k_{step} may influence solution quality for each particular problem, but on average (results reported in Table 7 and Table 8) it appears that small values of these parameters give better results. However, some randomness remains among successive values. This can be explained by the fact that each 'jump' from x_{cur} to some

Table 7. Sensitivity to k_l

	k_l			
	1	2	4	8
% Error	0.25	0.31	0.33	0.26
# Exact	36	36	34	34
# Iterat.	6018	4711	5075	4983

Table 8. Sensitivity to k_{step}

	k_{step}							
	1	2	3	4	5	6	7	8
% Error	0.25	0.42	0.32	0.24	0.56	0.43	0.33	0.53
# Exact	36	33	34	35	32	32	33	31
# Iterat.	6018	5633	4985	5472	4930	4989	5825	4412

solution $x' \in \mathcal{N}_k[\rho(x_{\text{cur}},x') = k]$ is followed by a local search which needs a few (let us say q) steps (interchanges) to reach a local minimum x^*, i.e. $\rho(x',x^*) = q$. Therefore x^* could be closer to x_{cur} than x', and thus, there is not much difference among solutions chosen from close neighborhoods.

Note that if $k < q$, the search is intensified; if $k > q$, the search is diversified. Note also that the current value of k depends on $k_{\max}$ as well. To conclude, for comparison purposes, there is no need to apply this extended version of Variable Neighborhood Search (with two parameters k_1 and k_{step}). However, for each particular problem, intensification and diversification of the search can easily be controlled by $k_{\max}$, k_{step} and k_1.

5.6. *Why Variable Neighborhood Search is Better than Tabu Search*

Although both Tabu Search and Variable Neighborhood Search perform the search in an attractive area of the solution space (in its 'deep valley'), the possible reason why Tabu Search 2 is not as efficient as Variable Neighborhood Search could be the fact that it does not visit real but pseudo local minima. The solutions on the path from pseudo to real local minima might be tabu (i.e. in forbidden areas of the solution space).

This happens because Tabu Search 2 puts in the Tabu List attributes of the solutions, not solutions themselves (to save memory and time). If the length of the Tabu List is small, a few solutions are tabu, but the probability of cycling is large. If this length is large, many solutions from the neighborhood are forbidden. Note that the Aspiration function does not usually help here, because it considers only solutions from the closest neighborhood, i.e. only one point from the path from pseudo to real local minimum; that path contains several solutions and if at least one of them is tabu, we are not able to reach the real local minimum along it.

Solutions of the largest problem from ORLIB (#40, $n = 900, p = 90$) are illustrated in Fig. 3 and Fig. 4, where local minima are generated by Variable Neighborhood Search and Tabu Search 2, respectively. This example is chosen because the optimal solution has not been found by either of these methods. It appears that: (i) the number of 'real' local minima of Variable Neighborhood Search (represented in Fig. 3) is less than the number of 'pseudo' local minima (represented in Fig. 4); (ii) solutions obtained by Variable Neighborhood Search are of better quality and four solutions whose objective function value is only 1 larger than the optimal value are found, while the best Tabu Search 2 value is 3 larger than the optimal one.

6. CONCLUSIONS

A simple parameter-free Variable Neighborhood Search heuristic for solving the p-median problem has been proposed in this paper. Since the quality of the solution obtained by

Variable Neighborhood Search depends in general on the local search subroutine used, an efficient implementation of the Fast Interchange (or vertex substitution) method (Whitaker, 1983) has been developed. Different neighborhood structures for Variable Neighborhood Search are induced by a single metric function introduced in the solution space.

A basic version of a Variable Neighborhood Search heuristic is proposed for p-median. Then various studies are done on the effect of parameters. We have determined the best number b of solutions to be randomly selected as initial points for descent in each neighborhood. Furthermore, we have shown that the index $k_{\max}$ of the largest neighborhood should not be more than $n/3$. Jumping over several neighborhoods or considering each one in turn does not affect performance. The choice of the initial solution is best when one uses Random solution followed by Interchange (and not the solution of Greedy or Greedy plus Interchange). The best solutions of both Variable Neighborhood Search and Tabu Search improve when more computer time is given, whereas this is not the case for Multistart Interchange.

The proposed Variable Neighborhood Search heuristic has been extensively compared with Greedy plus Interchange and with two recent Tabu Search heuristics, on standard test problems from the literature, as well as on very large problem instances (up to 3038 nodes and 500 facilities). Variable Neighborhood Search gave consistently better results on average and for the larger instances, with 1400 and 3038 nodes, always at least as good as the other methods. Average improvements for these problems over Greedy plus Interchange was about

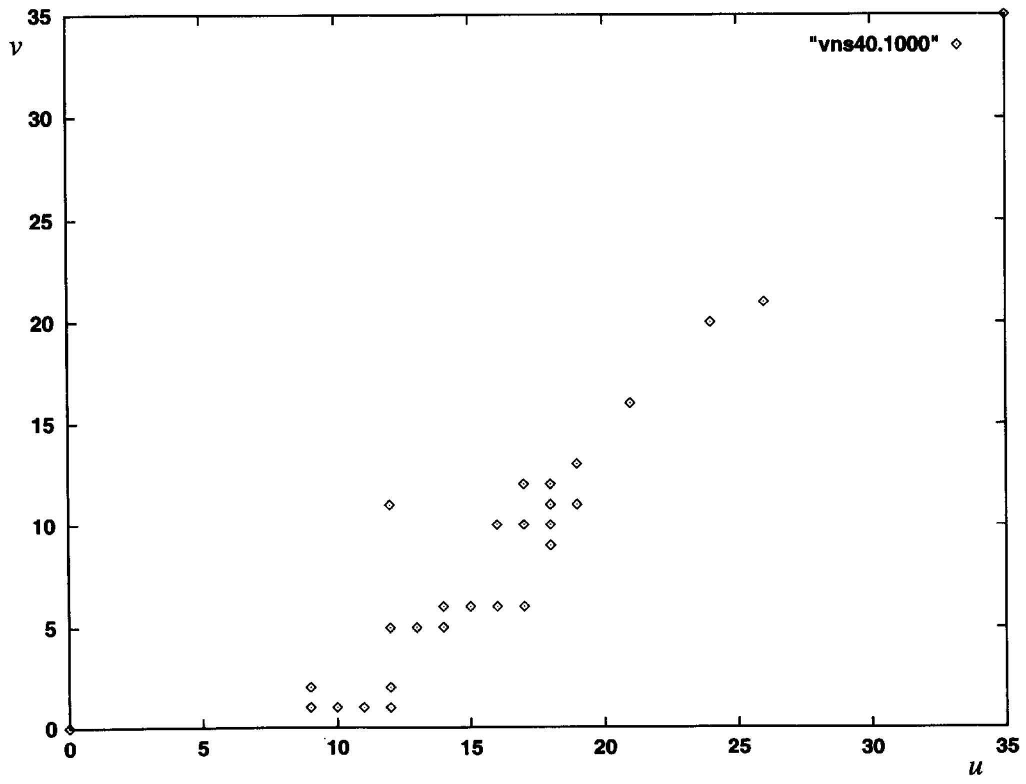

Fig. 3. Local minima obtained after 1000 iterations by Variable Neighborhood Search on test problem #40 ($n = 900, p = 90$); optimal solution has not been found.

0.7% and 0.5%, respectively, and over Tabu Search about 0.2% in both cases. The reason why Variable Neighborhood Search outperforms Tabu Search appears to be the effect of Tabu restrictions; many pseudo local minima are created by them and they make it more difficult to reach the global optimum.

Computing times went up to several hours for the largest instances. This suggests that decomposition might be worthwhile in heuristics for very large p-median problems, and this topic is currently under study. When the optimum solution was known, Variable Neighborhood Search almost always found it. So the proposed heuristic appears most often to squeeze out, with some computational effort, the last benefit over solutions given by other heuristics. This is particularly worthwhile when a very good initial solution is needed for an exact method such as stabilized column generation (du Merle *et al.*, 1997) or when costs are high.

APPENDIX

An efficient implementation of the fast interchange heuristic

We discuss here an efficient implementation of the basic move of many heuristics for p-median, i.e., interchange (or change of location for one facility). In particular, using this move only from a random initial solution gives a fairly good heuristic. As noted in the

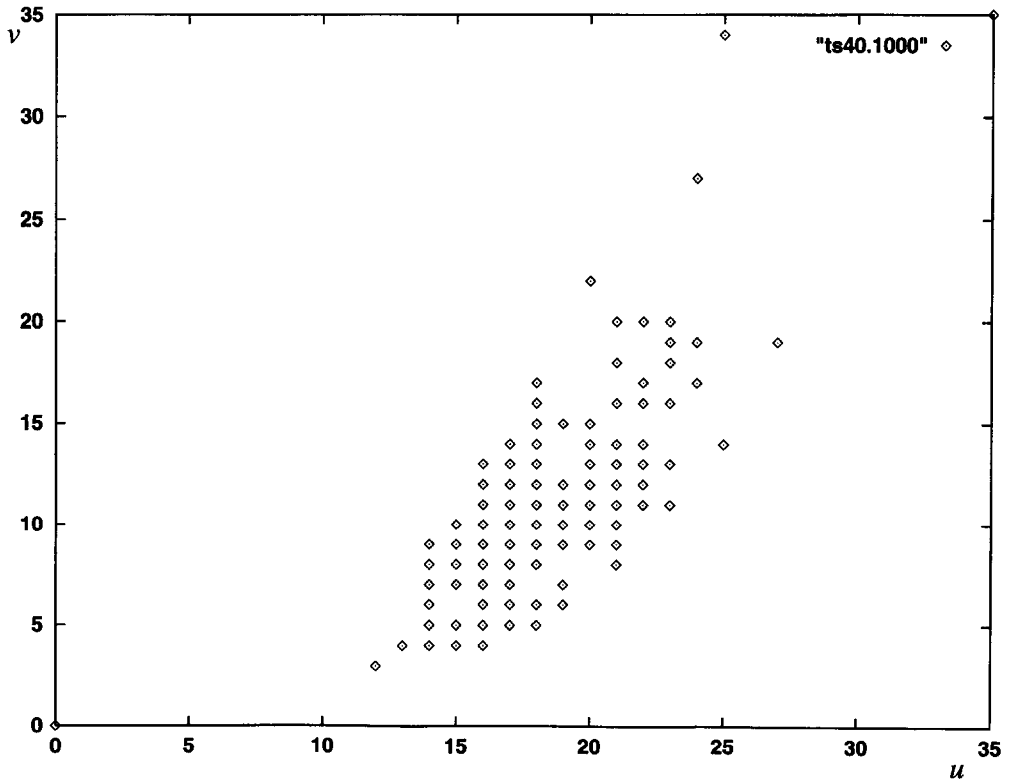

Fig. 4. Local and 'pseudo' local minima obtained after 1000 iterations by Tabu Search 2 on test problem #40 ($n = 900, p = 90$); optimal solution has not been found.

introduction, the benefit that could be derived from an efficient implementation of 1-interchange does not appear to be well-known. In Whitaker (1983), three efficient ingredients are incorporated in the interchange heuristic: (i) move evaluation, where a best removal of a facility is found when the facility to be added is known; (ii) updating of the first and the second closest facility of each user; (iii) restricted first improvement strategy, where each facility is considered to be added only once. In our implementation of the interchange algorithm only (i) and (ii) are used, i.e., instead of (iii) a best improvement strategy is applied. Moreover, the complexity of steps (i) and (ii) is evaluated.

A.1. Move evaluation. In the next procedure called Move, the change w in the objective function value is evaluated when the facility that is added (denoted with goin) to the current solution is known, while the best one to be dropped (denoted with goout) is to be found. In the description of the heuristic we use the following notation:

c1(i) — median (closest facility) of user i, $i = 1,\dots,n$;
c2(i) — second closest facility of user i, $i = 1,\dots,n$;
d(i,j) — distance (or cost) between user i and facility j, $i = 1,\dots,n$; $j = 1,\dots,m$;
goin — index of inserted facility (input value);
w — change in the objective function value obtained by the best interchange;
$x_{\mathrm{cur}}(i)$, $i = 1,\dots,p$ — current solution (indices of medians);
$v(j)$ — change in the objective function value obtained by deleting each facility currently in the solution, ($j = x_{\mathrm{cur}}(l)$, $l = 1,\dots,p$);
goout — index of deleted facility (output value).

```
Algorithm Move (c1,c2,$d$,$x_{\mathrm{cur}}$,goin,$n$,$p$,$w$,goout).
Initialization.
  Set $w \leftarrow 0$ and $v(x_{\mathrm{cur}}(l)) \leftarrow 0$, for all $l = 1,\dots,p$.
Best deletion
  For each user $i$ ($i = 1,\dots,n$) do the following:
    if $d(i,\mathrm{goin}) < d(i,\mathrm{c2}(i))$, then set $w \leftarrow w + d(i,\mathrm{goin}) - d[i,\mathrm{c1}(i)]$.
    otherwise, update the change in value obtained by deleting, c1($i$), i.e.
    $v(\mathrm{c1}(i)) \leftarrow v(\mathrm{c1}(i)) + \min\{d(i,\mathrm{goin}), d[i,\mathrm{c2}(i)]\} - d[i,\mathrm{c1}(i)]$.
  End for $i$.
  Find $g = \min\{v[x_{\mathrm{cur}}(l)], l = 1,\dots,p\}$ and facility goout [index $x_{\mathrm{cur}}(l)$ where this minimum is
  reached].
  $w \leftarrow w + g$.
```

Using algorithm Move, p different solutions from the solution space X are visited. Its worst-case complexity is O(n). Note that in Voss (1996) the same complexity is used to find w, when both goin and goout facilities are known. Thus, the number of operations needed for our Move evaluation procedure is p times less, as p possible vertices (facilities) could be removed.

A.2. Updating first and second closest facilities. In the Move evaluation procedure, both the closest [c1(i)] and second closest facility [c2(i)] for each user i must be known in advance. Among formal variables in the description of algorithm Update that follows, arrays c1(i) and c2(i), $i = 1,\dots,n$ are both input and output variables. All other formal variables in the list are input only.

```
Algorithm Update ($d$,goin,goout,$n$,$p$,c1,c2)
For each user $i$ ($i = 1$ to $n$) do the following
```

```
  (* For users whose center is deleted, find new one *)
  if c1(i) = goout then
    if $d(i,\text{goin}) \leq d(i,\text{c2}[i])$ then
      c1(i) = goin
    else
      c1(i) = c2(i)
      (* Find second closest facility for user i *)
      find center $l^*$ where $d(i,l)$ is minimum (for $l = 1,\dots,p,\ l \neq \text{c1}(i)$);
      $\text{c2}(i) \leftarrow l^*$
    endif
  else
    if $d[i,\text{c1}(i)] > d(i,\text{goin})$ then
      $\text{c2}(i) \leftarrow \text{c1}(i)$ and $\text{c1}(i) \leftarrow \text{goin}$
    else
      if $d(i,\text{goin}) < d(i,\text{c2}(i))$ then
        c2(i) = goin
    else
      if c2(i) = goout then
        find center $l^*$ where $d(i,l)$ is minimum (for $l = 1,\dots,p,\ l \neq \text{c1}(i)$);
        $\text{c2}(i) \leftarrow l^*$
      endif
    endif
  endif
end for i
```

The worst case complexity of the procedure Update is O($n\log n$) when a heap data structure is used for updating the second closest facility c2(i).

A.3. Fast interchange heuristic. Fast Interchange heuristic, which uses procedures Move and Update described before, is as follows:

```
Initialization
  Denote a random permutation of facilities $\{1,\dots,m\}$ with $x_{\text{opt}}(j), j = 1,\dots,m$. Let the first
  $p$ of them represent an initial solution; find the corresponding objective function value $f_{\text{opt}}$;
  find the closest and second closest facility for each user $i$, i.e., find arrays c1($i$), c2($i$),
  $i = 1,\dots,n$;
Iteration step
  $w^* \leftarrow \infty$
  For goin $= x_{\text{opt}}(p+1)$ to $x_{\text{opt}}(m)$
    (* Add facility goin not in the solution into it and find the best deletion *)
    Run procedure Move(c1,c2,$d$,goin,$n$,$p$,$w$,goout);
    (* Keep the best pair of facilities to be interchanged *)
    If $w < w^*$ then $w^* \leftarrow w$, goin* $\leftarrow$ goin, goout* $\leftarrow$ goout
  End for goin
Termination
  if $w^* \geq 0$ Stop; (* If no improvement in the neighborhood, Stop *)
Updating
  Update objective function value: $f_{\text{opt}} \leftarrow f_{\text{opt}} + w^*$;
Update $x_{opt}$: interchange position of $x_{opt}$(goout*) with $x_{opt}$(goin*);
Update closest and second closest facilities: Update($d$,goin*,goout*,$n$,$p$,c1,c2);
Return to Iteration step.
```

The complexity of one iteration of algorithm Fast interchange is $O[n(m-p)]$. This follows from the fact that procedure Move is used $m-p$ times, its complexity is $O(n)$ and the complexity of Update is less than $n(m-p)$.

Acknowledgements—Research supported by ONR grant N00014-95-1-0917, NSERC (Natural Scientific Research and Engineering Council of Canada) grant GPO105574 and FCAR (Fonds pour la Formation de Chercheurs et l'Aide à la Recherche) grant 95ER1048.

REFERENCES

Baum, E. B. (1986) Toward practical 'neural' computation for combinatorial optimization problems. In: *Neural networks for computing*, (Ed. J. Denker). American Institute of Physics.

Beasley, J. E. (1985) A note on solving large p-median problems. *European Journal of Operational Research*, **21**, 270–273.

Boese, K. D., Kahng, A. B. & Muddu, S. (1994) A new adaptive multi-start technique for combinatorial global optimization. *Operations Research Letters*, **16**, 101–113.

Brandeau, M. L. & Chiu, S. S. (1989) An overview of representative problems in location research. *Management Science*, **35**, 6 645–674.

Captivo, E. M. (1991) Fast primal and dual heuristics for the p-median location problem. *European Journal of Operational Research*, **52**, 65–74.

Carraghan, R. & Pardalos, P. M. (1990) An exact algorithm for the maximum clique problem. *Operations Research Letters*, **9**, 375–382.

Christofides, N. (1975) Graph Theory: An Algorithmic Approach. Academic Press, New York.

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

du Merle, O., Villeneuve, D., Desrosiers, J. & Hansen, P. (1997) Stabilisation dans le cadre de la génération de colonnes. *Les Cahiers du GERAD, G-97-08*, Montreal.

Erlennkoter, D. A dual-based procedure for uncapacitated facility location, *Operations Research*, **26**, 992–1009.

Galvão, R. D. (1980) A dual-bounded algorithm for the p-median problem. *Operations Research*, **28**, 1112–1121.

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. & Laguna, M. (1993) Tabu Search. In: *Modern Heuristic Techniques for Combinatorial Problems* (Ed. C. Reeves). Oxford: Blackwell.

Hanjoul, P. & Peeters, D. (1985) A comparison of two dual-based procedures for solving the p-median problem. *European Journal of Operational Research*, **20**, 387–396.

Hansen, P. & Jaumard, B. (1990) Algorithms for the maximum satisfiability problem. *Computing.*, **44**, 279–303.

Hansen, P. & Jaumard, B. (1997) Cluster analysis and mathematical programming. *Mathematical Programming*, **79**, 191–215.

Kariv, O. & Hakimi, S. L. (1979) An algorithmic approach to network location problems: part 2. The p-medians. *SIAM, Journal on Applied Mathematics*, **37**, 539–60.

Kuehn, A. A. & Hamburger, M. J. (1963) A heuristic program for locating warehouse. *Management Science*, **9**, 4 643–666.

Maranzana, F. E. (1964) On the location of supply points to minimize transportation costs. *Operations Research Quarterly*, **12**, 138–139.

Mirchandani, P. & Francis, R. (1990) *Discrete Location Theory.* Wiley-Interscience, New York.

Mladenović, N. (1995) A variable neighbourhood algorithm — a new metaheuristic for combinatorial optimization. Presented at Optimization Days, Montreal.

Mladenović, N. & Hansen, P. (1997) Variable neighbourhood search. *Computers and Operations Research*, **24**, 1097–1100.

Mladenović, N., Moreno, J. P. & Moreno-Vega, J. (1995) Tabu search in solving p-facility location–allocation problems. *Les Cahiers du GERAD*, G-95-38, Montreal.

Mladenović, N., Moreno, J. P. & Moreno-Vega, J. (1996) A chain-interchange heuristic method. *Yugoslav Journal of Operations Research.*, **6**, 1 41–54.

Moreno, J., Rodrigez, C. & Jimenez, N. (1990) Heuristic cluster algorithm for multiple facility location-allocation problem. *RAIRO — Recherche Opérationnelle/Operations Research*, **25**, 97–107.

Pizzolato, N. D. (1994) A heuristic for large-size p-median location problems with application to school location. *Annals of Operations Research*, **50**, 473–485.

Reinelt, G. (1991) TSPLIB — A traveling salesman problem library. *ORSA Journal on Computing*, **3**, 376–84.

Rolland, E., Schilling, D. A. & Current, J. R. (1996) An efficient tabu search procedure for the p-median problem. *European Journal of Operational Research*, **96**, 329–342.

Teitz, M. B. & Bart, P. (1968) Heuristic methods for estimating the generalized vertex median of a weighted graph. *Operations Research*, **16**, 5 955–961.

Voss, S. (1996) A reverse elimination approach for the p-median problem. *Studies in Locational Analysis*, **8**, 49–58.

Whitaker, R. (1983) A fast algorithm for the greedy interchange for large-scale clustering and median location problems *INFOR*, **21**, 95–108.