Journal of Heuristics, 7: 335–350, 2001

Variable Neighborhood Decomposition Search

PIERRE HANSEN AND NENAD MLADENOVIĆ
Gerad, Ecole des Hautes Etudes Commerciales, Montreal, Canada

DIONISIO PEREZ-BRITO
University "La Laguna", Tencrife, Spain

Abstract

The recent Variable Neighborhood Search (VNS) metaheuristic combines local search with systematic changes of neighborhood in the descent and escape from local optimum phases. When solving large instances of various problems, its efficiency may be enhanced through decomposition. The resulting two level VNS, called Variable Neighborhood Decomposition Search (VNDS), is presented and illustrated on the p-median problem. Results on 1400, 3038 and 5934 node instances from the TSP library show VNDS improves notably upon VNS in less computing time, and gives much better results than Fast Interchange (FI), in the same time that FI takes for a single descent. Moreover, Reduced VNS (RVNS), which does not use a descent phase, gives results similar to those of FI in much less computing time.

Key Words: y-median, metaheuristic, variable neighborhood search, decomposition

1. Introduction

Consider a finite but large set S. Combinatorial optimization problem consist in finding $x_{\text{opt}} \in X \subseteq S$ such that some objective function f is minimized,

$$\min\{f(x) : x \in X, X \subseteq S\}. \tag{1}$$

S, X, x and f are solution space, feasible set, feasible solution, and real valued function, respectively. Most combinatorial optimization problems are *NP-hard* and heuristic (sub-optimal) solution methods are needed to solve them (at least for large instances or as an initial solution for some exact procedure).

In local (or neighborhood) search heuristic methods the set of neighborhood solutions $\mathcal{N}(x)$ of any solution x is defined. The procedure starts with a feasible solution x_0; then, at iteration k, the objective function value of each $x'_k \in V_k \subseteq \mathcal{N}(x_k)$ is evaluated. If a better solution $x_{k+1} \in V_k$ is found the procedure continues with x_{k+1} as a new current solution. Otherwise, it stops in a local minimum, whose objective function value could be much larger than the globally minimum one.

Several ways to do better within the local search framework have been suggested in the literature. The easiest is the Multistart approach, where the same procedure is restarted a given number of times, from different initial solutions and the best local minimum is retained. Simulated Annealing (Kirkpatrick et al., 1983), Tabu Search (Glover, 1989, 1990; Hansen and Jaumard, 1990; Glover and Laguna, 1993, 1997) and Variable Neighborhood

Search (Mladenović, 1995; Mladenović and Hansen, 1997; Hansen and Mladenović, 1998) methods, explore the vicinity of the local minima, as they are found. The first two methods follow trajectories, accept ascent moves by different means, and most often use a single neighborhood structure in the search. On the other hand, the third method explores increasingly far neighborhoods of the current local minimum, with a descent method, and re-centers the search if a better solution than the incumbent is found.

Note that on occasion, use of several types of moves was proposed by various authors in heuristics for specific problems, without however this being the main idea or principle advocated. For instance, it was suggested long ago to use the interchange heuristic after the ATTILA or BABEL (stingy or greedy) heuristics for the simple plant location problem (e. g., Kaufman and Hansen, 1972). Another example is an informal discussion of "managerial robots" for employee scheduling problems by Glover, McMillan, and Glover (1984), who consider adding a tour (for a particular person on a particular day) deleting a tour, modifying a tour by changing its lunch period or breaks, etc.

Performance of all local search metaheuristics such as those above-mentioned commonly depends on the efficiency of their descent phase. For small and medium size problems, descent local searches are very fast and general heuristics usually use much longer CPU times. But, for very large problem instances, local search algorithms often require substantial amounts of running time. One way to reduce the running time is by using parallelism, another by using decomposition.

In this paper we propose a new heuristic decomposition method for combinatorial optimization problems. We call it Variable Neighborhood Decomposition Search (VNDS) as the method follows a basic VNS scheme. In the next section both the basic VNS and VNDS heuristics are described. Section 3 presents our implementation of VNDS for solving the p-Median problem. Computer results are reported in Section 4, while Section 5 concludes the paper.

2. Variable neighborhood decomposition search

Let us denote a finite set of pre-selected neighborhood structures with $\mathcal{N}_k$, $(k = 1, \ldots, k_{\max})$, and with $\mathcal{N}_k(x)$ the set of solutions in the kth neighborhood of x. Note that local search heuristics usually use one neighborhood structure, i. e., $k_{\max} = 1$. The basic VNS heuristic comprises the steps given in figure 1.

The stopping condition may be e.g., maximum number of iterations, maximum number of iterations between two improvements or maximum CPU time allowed. Often successive neighborhoods $\mathcal{N}_k$ will be nested. 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.

It is worth stressing the ease of implementation of both the basic version of VNS (with only one parameter $k_{\max}$) and various simple extensions discussed below. Even this single parameter can be disposed of by taking nested neighborhoods which partition the solution space, e.g. the sets of boolean n-vectors at Hamming distance $1, 2, \ldots, n$ of the incumbent solution when solving a problem in n 0-1 variables. Step 2a is easy to program. For example, if $\mathcal{N}_k$ is obtained by k-interchanges of solution attributes, one need only add a few lines to an existing code for a local search method (Step 2b).

Initialization. Select the set of neighborhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$, that will be used in the search; find an initial solution x; choose a stopping condition;

Repeat the following sequence until the stopping condition is met:

(1) Set $k \leftarrow 1$;

(2) Until $k = k_{max}$, repeat the following steps:

(a) Shaking. Generate a point x' at random from the k^{th} neighborhood of x $(x' \in \mathcal{N}_k(x))$;

(b) Local search. Apply some local search method with x' as initial solution; denote with x'' the so obtained local optimum;

(c) Move or not. If this local optimum is better than the incumbent, move there $(x \leftarrow x'')$, and continue the search with $\mathcal{N}_1$ $(k \leftarrow 1)$; otherwise, set $k \leftarrow k + 1$;

Figure 1. Steps of the basic VNS.

Initialization. Select the set of neighborhood structures $\mathcal{N}'_k$, $k = 1, \ldots, k'_{max}$, that will be used in the descent; find an initial solution x (or apply the rules to a given x);

Repeat the following sequence until no improvement is obtained:

(1) Set $k \leftarrow 1$;

(2) Until $k = k'_{max}$, repeat the following steps:

(a) Exploration of neighborhood. Find the best neighbor x' of x $(x' \in \mathcal{N}'_k(x))$;

(b) Move or not. If the solution thus obtained x' is better than x, set $x \leftarrow x'$ and $k \leftarrow 1$; otherwise, set $k \leftarrow k + 1$.

Figure 2. Steps of the basic VND.

As a local optimum within a given neighborhood is not necessarily one within another, change of neighborhoods can be performed during the local search phase too. In some cases, as when applying VNS to graph theory, the use of many neighborhoods in the local search is crucial. This local search is then called Variable neighborhood Descent (VND) and its steps are given in figure 2.

The basic VNS is in fact a descent, first improvement method. Without much additional effort it could be transformed into a descent-ascent method: in Step 2c set also $x \leftarrow x''$ with some probability even if the solution is worse than the incumbent, and/or a best improvement method: make a move to the best neighborhood k^* among all $k_{\max}$ of them. Other variants of the basic VNS are described in Hansen and Mladenović (1998). We only recall here one of them, called Reduced VNS (RVNS). It results from an elementary to implement but quite drastic change: *in the basic scheme remove the local search Step* (2b).

RVNS is useful for very large instances for which local search is costly. It is akin to a Monte-Carlo method, but more systematic. Its relationship to the Monte-Carlo method is the same as that of VNS to Multistart local search.

We now describe Variable Neighborhood Decomposition Search. This method follows a basic VNS scheme within a successive approximations decomposition method. For a given solution x, all but k attributes (or variables or subset of variables) are fixed in the local

Initialization. Select the set of neighborhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$, that will be used in the search; find an initial solution x; choose a stopping condition;

Repeat the following sequence until the stopping condition is met:

(1) Set $k \leftarrow 1$;

(2) Until $k = k_{max}$, repeat the following steps:

(a) Shaking. Generate a point x' at random from the k^{th} neighborhood of x ($x' \in \mathcal{N}_k(x)$); in other words, let y be a set of k solution attributes present in x' but not in x ($y = x' \setminus x$).

(b) Local search. Find the local optimum in the space of y either by inspection or by some heuristic; denote the best solution found with y' and with x'' the corresponding solution in the whole space S ($x'' = (x' \setminus y) \cup y'$);

(c) Move or not. If the solution thus obtained is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with $\mathcal{N}_1$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$;

Figure 3. Steps of the basic VNDS.

search phase. All possible such fixations define a neighborhood $\mathcal{N}_k(x)$. As in VNS, we start with a random solution x' from $\mathcal{N}_1(x)$. But, instead of performing local search in the whole solution space S with x' as a starting point, we solve a one-dimensional problem in the space of the unfixed variable (or subset of variables) that has been chosen at random. We then return a new value for this variable into the solution and compute (or update) the objective function value. The other steps are the same as in VNS: if the new solution is not better than the incumbent, we set $k = k + 1$, i.e., we look for improvement in the subspace where all but two variables are fixed, etc.; otherwise a move has been made and we set $k \leftarrow 1$ again. The steps of the basic VNDS are given in figure 3.

Note that the only difference between the basic VNS and VNDS is in Step 2b: instead of applying some local search method in the whole solution space $\mathcal{S}$ (starting from $x' \in \mathcal{N}_k(x)$), in VNDS we solve at each iteration a subproblem in some subspace $V_k \subseteq \mathcal{N}_k(x)$ with $x' \in V_k$. If a given local search heuristic is used for solving this subproblem (in Step 2b of VNDS), then VNDS uses a single parameter, $k_{\max}$. However, we can use some better heuristic, such as VNS, and then an additional problem specific parameter, say b, can be considered. Its aim is to strike a balance between the number of subproblems solved and the desired quality of the solution of each subproblem. Parameter b could represent, e.g., maximum CPU time allowed for solving each subproblem, or maximum size of the subproblem we are willing to solve. If the CPU time or the size of the subproblem exceeds its limit, we set $k \leftarrow 1$, i.e., we continue the decomposition by solving smaller subproblems. In this case we have in fact a recursive, two level, VNS heuristic. Note that it may be worthwhile to consider neighborhoods built from close smaller ones if there is a significant proximity relation between them. This happens in the p-median problem next discussed.

Fixing temporarily some variables in an optimization problem, improving the resulting solution and iterating is, as mentioned above, an application of the idea of successive approximations, which is well-known in mathematics at least since the 19th century. Its use in combinatorial optimization goes back to the beginnings of dynamic programming (e.g., Bellman and Dreyfus, 1962) and perhaps earlier. Another early application is the *alternate*

heuristic for bilinear programming (Griffith and Stewart, 1961). Although from its definition (as far as one is given), the idea of *referent domain optimization* (Glover and Laguna, 1997, pp. 354–356) appears to be quite different, one of the examples given there can also be viewed as an application of successive approximations. Of course, no claim is made in this paper to have discovered (or rediscovered) the idea of successive approximations. What it does is exploit this idea together with the VNS principle to build a two-level VNS, or VNDS framework, and apply it to the p-median problem.

3. VNDS for the p-median problem

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 (PM) 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 e.g., Mirchandani and Francis, 1990, for an introduction to discrete location theory).

The p-median problem and its extensions are useful to model many real word situations, such as the location of industrial plants, warehouses and public facilities (see for example (Christofides, 1975), for a list of applications). PM 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, PM 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 smallest possible.

Consider a set L of m potential facilities (or location points or medians), a set U of n users (or customers or demand points) and a $n \times m$ matrix D of 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.,

$$(\min) \sum_{i \in U} \min_{j \in J} d_{ij}.$$

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

$$(\min) \sum_{i=1}^{n} \sum_{j=1}^{m} d_{ij}\, x_{ij} \tag{2}$$

subject to

$$\sum_{j=1}^{m} x_{ij} = 1, \quad \forall i, \tag{3}$$

$$x_{ij} \leq y_j, \quad \forall i, j \tag{4}$$

$$\sum_{j=1}^{m} y_j = p, \tag{5}$$

$$x_{ij}, y_j \in \{0, 1\} \tag{6}$$

where $y_j = 1$ signifies a facility is located at j; $x_{ij} = 1$ if user i is assigned to facility j (and 0 otherwise).

Constraints (2) express that the demand of each user must be met. Constraints (3) prevent any user from being supplied from a site with no open facility. The total number of open facilities is set to p by constraint (4).

The p-median problem is NP-hard (Kariv and Hakimi, 1969). Many heuristics and exact methods have been proposed for solving it. Exact algorithms were developed by Beasley (1985) and Hanjoul and Peeters (1985), among others. Extensive references to work on this and related problems are contained in Cornuejols et al. (1977), Brandeau and Chiu (1989), Mirchandani and Francis (1990) and Drezner (1995).

Classical heuristics for PM often cited in the literature, are *Greedy* (Kuehn and Hamburger, 1963), *Alternate* (Maranzana, 1964) and *Interchange* (Teitz and Bart, 1968). Several hybrids of these heuristics have been suggested. 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* + *Alternate* and *Multi-start Alternate*, etc. The combination of *Greedy* and *Interchange*, where the *Greedy* solution is chosen as initial one for *Interchange*, has been most often used for comparison with other newly proposed methods (see for example Voss (1996) and Hansen and Mladenović (1997)).

Another type of heuristics suggested in the literature is based on the relaxed dual of the integer programming formulation of PM and uses the well-known *Dual ascent* heuristic DUALOC (Erlenkotter, 1978). Such heuristics for solving the p-median problem are proposed in Galvao (1980) and in Captivo (1991).

Three different Tabu Search (Glover, 1989, 1990; Hansen and Jaumard, 1990) methods have recently been proposed for solving PM (see Glover and Laguna (1993, 1997) for an introduction to Tabu Search). In Mladenovic et al. (1995) a 1-interchange move is extended into a so-called 1-*chain-substitution* move. Another TS heuristic is suggested 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 necessary follow each other and so feasibility is not necessary maintained during the search; this approach, within TS, is known as *strategic oscillation* (see Glover and Laguna (1993)).

A simple parameter free Variable Neighborhood Search (VNS) method for solving the p-median problem has been proposed in Hansen and Mladenović (1997). Since the quality of the solution obtained by VNS depends in general on the local search subroutine used, an efficient implementation of the *Fast Interchange* (FI) (or vertex substitution) method (Whitaker, 1983) has been developed. Different neighborhood structures for VNS are induced by a single metric function introduced in the solution space. The proposed method has been extensively compared with *Greedy* + *Fast Interchange* and with two Tabu search

methods, on standard test problems from the literature, as well as on very large problem instances (up to 3038 nodes and 500 facilities). Results obtained by VNS were favorable. However, 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.

Let us denote with $X = \{x \mid x = \{m_1, m_2, \ldots, m_p\}, m_i \in L, |L| = m\}$ a solution space of the problem. The cardinality of X is obviously $\binom{m}{p}$. Note that each choice of p (out of m) medians uniquely defines a partition of the users set U into p disjoint subsets $\{U_1, \ldots, U_p\}$, because each user should be served by its closest facility m_i. Note also that the opposite does not hold, i.e., $|X|$ is less than the number of all partitions of U. Thus, solution of the p-median problem can be represented as a set of users subsets, $\{U_1, \ldots, U_p\}$. 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

$$\rho(x_1, x_2) = |x_1 \backslash x_2| = |x_2 \backslash x_1|, \quad \forall x_1, x_2 \in X. \tag{7}$$

It can easily be checked that ρ is a metric function in X, thus, X is a metric space. The neighborhood structures used in Hansen and Mladenović (1997) were induced by metric ρ, i.e., k locations of facilities ($k \leq p$) from the current solution are replaced by k others in $\mathcal{N}_k$. The same metric will be used in VNDS below. We denote with $\mathcal{N}_k$, $k = 1, \ldots, k_{\max}$ ($k_{\max} \leq p$) 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

$$x_2 \in \mathcal{N}_k(x_1) \Longleftrightarrow \rho(x_1, x_2) = k. \tag{8}$$

For solving large problem instances by VNDS, it is important to get an initial solution quickly. Moreover, it needs to be of good quality. As mentioned before, greedy heuristics do not match both of these goals. Here we use reduced VNS (or RVNS for short), initially proposed by Mladenović (1995). The difference between VNS for the p-median problem suggested by Hansen and Mladenović (1997), and RVNS which we are using here is that RVNS does not have Step 2b: the local search is simply skipped. The steps of the VNDS method for solving large p-median problems are given in figure 4.

4. Computer results

The sets of problem instances used in testing are: (i) 40 ORLIB problems from Beasley (1985); (ii) four problems from Rolland et al. (1996); (iii) three large problems from the TSP library (Reinelt, 1991) with $n = 1400$, $n = 3038$ and $n = 5934$ with $m = n$ and $p \in [10, 1500]$.

All methods compared are coded in FORTRAN 77 and run on a SUN Ultra I System (143-MHz). In order to decrease running time, compiling is done with the optimizing option, i.e., with '$f77$-$cg92$-$O4$'. The CPU times reported in the tables below are in seconds, while the % error are calculated as $\frac{f - f_{\text{opt}}}{f_{\text{opt}}} \cdot 100$, where f denotes the best solution found by the heuristic.

Initialization. (1) Rank locations for potential facilities from L, i.e., for each potential facility, find its closest, second closest, etc. potential facility; (2) Neighborhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$ are induced by distance as defined in formula (2); (3) Find an initial solution x by RVNS: in the shaking step k facilities are added at random and the best deletion found; RVNS stops when the number of unsuccessful attempts at improvement is greater than a given value r_{max}; use $k'_{max} = 2$ in RVNS. (4) Set $k_{max} = p$ and choose parameter b (i.e., the size of the subproblem we are willing to solve by VNS cannot have more than b users, $|U_k| \leq b$); choose stopping conditions for both VNDS and VNS (that will be used in step 2b below);

Repeat the following until the stopping condition is met:

(1) Set $k \leftarrow 1$;

(2) Until $k = k_{max}$, repeat the following steps:

(a) Choose at random one facility from the solution x and take its $k-1$ closest medians among those that belong to the solution x. (Note that the ranking of facilities is done in the Initialization step); denote the set of k facilities with y; (note also that now $x' \in \mathcal{N}_k(x)$ can be obtained by replacing y with any other subset of k facilities $y' \subseteq L$ not being in the solution).

(b) Denote with U_k the set of users whose closest facilities belong to y. If $|U_k| \leq b$, use VNS to solve the k-median subproblem, with U_k as set of users; if $|U_k| > b$, solve the k-median problem by RVNS; denote the solution obtained with y'; replace y by y' in x' to get the new solution $x'' \in \mathcal{N}_k(x)$ ($x'' = (x' \setminus y) \cup y'$);

(c) If the solution thus obtained is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with $\mathcal{N}_1$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$;

Figure 4. VNDS for solving large PM.

In all tests reported in this section the maximum number of neighborhoods is fixed at p, 2, and p for VNS, RVNS and VNDS respectively. For the second level of VNS (within VNDS), $k''_{\max}$ is fixed at 5, and $b_{\max}$ is set to 400. Parameter $r_{\max}$ for RVNS is set to 1000.

4.1. OR-LIB *test problems*

We first tested three methods (FI, RVNS and VNDS) on 40 ORLIB problems from Beasley (1985), where the set of facilities is equal to the set of users ($m = n$). The problem parameters range from instances with $n = 100$ nodes and $p = 5, 10, 20$ and 33 up to instances with $n = 900$ and $p = 5, 10, 90$. All these test problems were 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 get matrix D that is used by the FI procedure, an *all shortest paths* algorithm is run first (the CPU time for this $O(n^3)$ procedure is not included in the Table 1 below). We did not include basic VNS in the comparison, because the results of an extensive empirical analysis on them have already been reported in Hansen and Mladenović (1997) (38 among 40 test problems are solved exactly, with an average error of 0.01%).

In column 4 of Table 1, known optimal values (f_{opt}) are reported, followed by values obtained by FI, RVNS and VNDS. The CPU times (in seconds) when the best solution has been found by each method are given in columns 8–10. In column 11 maximum time

Table 1. 40 OR-LIB test problems.

Pr. No.	n	p	Objective values				CPU time				% Error		
			f_{opt}	FI	RVNS	VNDS	FI	RVNS	VNDS	VNDS	FI	RVNS	VNDS
1	100	5	5819	5819	5819	5819	0.02	0.09	0.09	0.14	0.00	0.00	0.00
2	100	10	4093	4105	4105	4105	0.03	0.08	0.08	0.11	0.29	0.29	0.29
3	100	10	4250	4250	4270	4270	0.04	0.08	0.08	0.11	0.00	0.47	0.47
4	100	20	3034	3034	3034	3034	0.04	0.08	0.08	0.12	0.00	0.00	0.00
5	100	33	1355	1361	1358	1358	0.05	0.08	0.08	0.12	0.44	0.22	0.22
6	200	5	7824	7824	7824	7824	0.11	0.41	0.41	2.06	0.00	0.00	0.00
7	200	10	5631	5645	5639	5639	0.15	0.51	0.51	2.10	0.25	0.14	0.14
8	200	20	4445	4445	4474	4454	0.30	0.39	1.15	2.28	0.00	0.65	0.20
9	200	40	2734	2738	2753	2753	0.36	0.70	0.70	2.33	0.15	0.69	0.69
10	200	67	1255	1256	1288	1259	0.47	0.47	1.93	2.42	0.08	2.63	0.32
11	300	5	7696	7696	7696	7696	0.24	1.10	1.10	8.18	0.00	0.00	0.00
12	300	10	6634	6634	6634	6634	0.46	0.65	0.65	8.39	0.00	0.00	0.00
13	300	30	4374	4387	4374	4374	0.92	0.95	0.95	8.85	0.30	0.00	0.00
14	300	60	2968	2969	2980	2969	1.72	1.13	5.77	9.48	0.03	0.40	0.03
15	300	100	1729	1752	1742	1731	2.51	0.95	5.13	10.08	1.33	0.75	0.12
16	400	5	8162	8162	8162	8162	0.58	1.30	1.30	20.63	0.00	0.00	0.00
17	400	10	6999	6999	7010	7009	0.73	1.59	20.37	20.65	0.00	0.16	0.14
18	400	40	4809	4813	4815	4811	2.31	1.88	11.49	22.06	0.08	0.12	0.04
19	400	80	2845	2852	2862	2849	4.88	2.09	16.45	24.32	0.25	0.60	0.14
20	400	133	1789	1797	1796	1789	7.12	2.22	25.11	26.13	0.45	0.39	0.00
21	500	5	9138	9138	9138	9138	0.57	1.08	1.08	42.26	0.00	0.00	0.00
22	500	10	8579	8579	8592	8579	1.40	1.57	15.20	43.07	0.00	0.15	0.00
23	500	50	4619	4641	4623	4623	5.24	4.27	4.27	46.44	0.48	0.09	0.09
24	500	100	2961	2993	2968	2961	12.52	3.99	9.68	53.47	1.08	0.24	0.00
25	500	167	1828	1841	1842	1830	14.73	2.90	54.27	54.94	0.71	0.77	0.11
26	600	5	9917	9917	9924	9924	0.85	1.93	1.93	79.48	0.00	0.07	0.07
27	600	10	8307	8310	8314	8310	2.05	2.08	3.65	80.34	0.04	0.08	0.04
28	600	60	4498	4505	4506	4505	9.73	3.54	9.10	87.24	0.16	0.18	0.16
29	600	120	3033	3045	3042	3039	20.28	3.97	17.39	96.53	0.40	0.30	0.20
30	600	200	1989	2021	2004	1990	26.00	4.30	29.24	101.45	1.61	0.75	0.05
31	700	5	10086	10086	10086	10086	1.57	2.45	2.45	132.87	0.00	0.00	0.00
32	700	10	9297	9301	9326	9297	3.31	2.73	7.93	133.94	0.04	0.31	0.00
33	700	70	4700	4723	4707	4703	19.33	5.37	39.30	148.44	0.49	0.15	0.06
34	700	140	3013	3037	3021	3016	40.49	9.23	163.74	166.46	0.80	0.27	0.10
35	800	5	10400	10400	10400	10400	2.13	3.33	3.33	197.65	0.00	0.00	0.00
36	800	10	9934	9934	9989	9988	3.87	3.48	4.85	199.45	0.00	0.55	0.54

(*Continued on next page*.)

Table 1. (*Continued*).

Pr. No.	n	p	Objective values				CPU time				% Error		
			f_{opt}	FI	RVNS	VNDS	FI	RVNS	VNDS	VNDS	FI	RVNS	VNDS
37	800	80	5057	5076	5074	5066	34.44	5.70	46.44	226.32	0.38	0.34	0.18
38	900	5	11060	11060	11071	11071	2.74	4.63	4.63	283.27	0.00	0.10	0.10
39	900	10	9423	9423	9423	9423	4.80	6.48	6.48	285.36	0.00	0.00	0.00
40	900	90	5128	5137	5134	5134	51.06	10.47	10.47	326.62	0.18	0.12	0.12
	Average						7.00	2.51	13.22	73.90	0.25	0.30	0.12

allowed for VNDS (VNDS*) is presented (it is obtained as a time for reading data + time of FI). The % errors of the three methods are reported in columns 12 to 14.

It appears that: (i) there is no need for decomposition for this relatively small set of instances, since VNS performed better than VNDS; (ii) the average % error of VNDS is less than that of FI (compare 0.12% error of VNDS with 0.25% of FI); (iii) results of similar quality are obtained by FI and RVNS (compare 0.25% with 0.30% error), but RVNS spent less CPU time on average (compare 7.0 with 2.5 seconds); (iv) although VNDS was allowed to run 73.9 seconds on average, it used only 13.22 seconds on average to find the best solution it obtained. This suggests better parameter settings for this small size set of instances. We changed the two parameters of RVNS to $k'_{\max} = 5$ and $r_{\max} = 2000$, i.e., we allowed longer running time for initial solution of VNDS, and thus less time for its iterations. Then % errors of 0.19, and 0.08 are obtained (instead of 0.30 and 0.12) for RVNS and VNDS respectively.

4.2. Rolland et al. test problems

In a recent paper by Roland et al. (1996), a Tabu search (TS) method for p-Median is proposed and tested on random graphs with $m = n = 200, 300, 400$ and 500 nodes, having $p = 10, 15$ and 20 facilities. The (symmetric) distance between any two nodes are generated at random from the integer interval [0, 100], thus, the triangular inequality is not satisfied. In other words, the d_{ij} can be larger than the distance on the shortest path between nodes i and j. In addition, since the weights are associated to users, asymmetric distances are involved in the model as well (obtained as $d_{ij} \leftarrow w_i d_{ij}$). These test instances are called 'large' problems, and since the optimal solutions were not found by available software, the best known solutions are reported. The proposed TS method was compared with two local search procedures: classical Interchange (called Node substitution) (Teitz and Bart, 1968), and Global/Regional interchange algorithm (Densham and Rushton, 1992). In 11 among 12 large test problems, the best known solutions reported were obtained by TS. More recently, those results were compared with the Heuristic Concentration (HC) method (Rosing et al., 1998), but results for $n = 200$ only were reported. However, in Table 2, we present results obtained by HC for all test instances, kindly communicated by Rosing (1998).

Here we compare the quality of the solutions obtained by several heuristics, on the 'large' test instances from Roland et al. (1996): (i) Fast Interchange (FI) (Whitaker, 1983; Hansen and Mladenović, 1997); (ii) Basic Variable Neighborhood Search (VNS) (Hansen

Table 2. Results for test problems from Rolland et al. (1997).

			% Error							CPU time					
n	p	Best known	FI	VNS	HC	CSTS	TS	RVNS	VNDS	FI	VNS	CSTS	TS	RVNS	VNDS
200	10	48912.	1.21	0.00	0.00	0.02	0.68	0.00	0.00	5.2	80.3	59.1	381.9	31.1	43.2
	15	31153.	1.29	0.00	0.00	0.00	2.80	1.00	1.00	7.5	121.2	118.9	401.1	29.8	44.7
	20	23323.	1.96	0.00	0.65	0.00	0.74	2.42	2.32	9.6	161.6	163.2	416.6	32.8	45.3
300	10	82664.	3.58	0.00	0.00	1.08	0.47	1.91	1.91	12.3	248.2	179.2	1241.0	53.1	60.7
	15	52685.	4.47	0.00	0.17	0.50	1.98	3.07	3.07	19.4	373.3	311.8	1321.6	47.3	81.7
	20	38244.	3.34	0.00	1.35	0.72	2.49	3.80	3.23	24.4	475.8	467.0	1378.3	56.5	92.7
400	10	123464.	0.19	0.00	0.00	0.12	3.79	3.68	2.99	24.5	463.4	361.3	2910.6	76.1	109.3
	15	79872.	5.20	0.00	0.75	2.50	5.15	5.77	5.36	32.0	631.5	463.1	3096.8	74.1	130.4
	20	58459.	7.08	0.46	0.00	0.92	1.17	5.66	5.66	45.4	958.6	653.3	3218.3	83.5	103.1
500	10	150112.	2.03	0.00	0.00	0.14	1.52	2.94	2.94	40.6	864.9	474.3	9732.2	92.1	175.2
	15	97624.	3.54	0.00	0.00	1.55	0.79	1.49	1.49	54.7	1164.2	887.2	9731.1	121.6	157.9
	20	72856.	4.86	0.00	0.41	1.46	0.41	1.92	1.92	74.5	1593.4	1350.0	9748.4	117.8	161.9
	Average		3.23	0.04	0.27	0.75	1.83	2.80	2.66	29.2	594.7	457.4	3631.5	67.9	100.5

Maximum time allowed for CSTS and VNS is set to be 30 times those of FI; the best solution found in 50 trials of FI, CSTS, VNS, RVNS and VNDS are reported.

and Mladenović, 1997); (iii) Heuristic Concentration (HC) (Rosing and ReVelle, 1996; Rosing, 1998); (iv) Chain Substitution Tabu Search (CSTS) (Mladenović et al., 1995); (v) Tabu Search (TS) (Roland et al., 1996); (vi) Reduced Variable Neighborhood Search (RVNS); and (vii) Variable Neighborhood Decomposition Search (VNDS).

Although the problems are not very large, their 'unpleasant' structure (distances are not symmetric and do not satisfy the triangular inequality), did not allow us to get global optimal solutions with the exact codes we have. Thus, in column 3 of Table 2, the best known solutions are reported. In columns 4 to 10 the % error of each method compared with the best known solution are given, while columns 11–16 consist of corresponding computing times (the times for HC were not available (Rosing, 1998)). All methods, except TS and HC, are restarted 50 times (from different initial solutions). In each run VNS and CSTS started with FI solution and terminate the search when CPU time exceeds 30 times those of FI. In each run RVNS stops after $r_{\max} = 1000$ unsuccessful trials, while the VNDS starts with RVNS solution and works until time for reading data + time for one FI descent is not exceeded.

From Tables 2, the following observations can be derived: (i) the best known solutions obtained by TS are improved upon in all cases; (ii) VNS and HC outperform other heuristics. VNS reaches the best known solution 11 times with an average error of 0.04%, while HC reaches the best known solutions 7 times with an average error of 0.27%; (iii) both structure and size of the problem instances are not convenient for the decomposition method. Most of the time the solutions obtained by RVNS have not been improved by VNDS; (iii) among the two Tabu search methods (CSTS and TS), CSTS gives the better results (compare 0.75% error of CSTS with 1.83% error of TS) and takes the least time.

4.3. TSP-LIB *test problems*

In the next tables, FI, VNS, RVNS and VNDS are compared on three large problem instances taken from TSPLIB (Reinelt, 1991): (i) RL1400 ($n = m = 1400$); (ii) PCB3038 ($n = m = 3038$); (iii) RL5934 ($n = m = 5934$). The tests on all problems are done in the same way: (i) initial solution is generated at random; (ii) the solution so obtained is used as initial one for the *fast interchange* heuristic (FI) and for RVNS; (iii) the solution obtained by FI is used as initial one for the basic VNS, allowing five times longer CPU running time for VNS than for FI; (iv) the solution obtained by RVNS is used as initial one for VNDS; maximum running time for VNDS is set to be equal to the CPU time FI spends for a single descent. Since CPU time can be very long for some test examples, each instance is run only once, i.e., we do not report the average results for each n and p.

Some computer results on the first two problems have already been reported in Hansen and Mladenović (1997), where *Greedy*, *Greedy* + FI, CSTS and basic VNS were compared. Since the best known solution had always been obtained by VNS, the second columns of Tables 3 and 4 contain those values. Using the method proposed in du Merle et al. (1998), the RL1400 problem was solved exactly for $p = 10, \ldots, 70$ and for $p = 90$. We indicate in bold

Table 3. 1400-customer problem.

	Objective values				CPU time			% Error r.t VNS		
p	VNS	FI	RVNS	VNDS	FI	RVNS	VNDS	FI	RVNS	VNDS
10	**101249.47**	101941.88	101276.11	**101249.47**	14.78	6.34	9.25	0.69	0.03	0.00
20	**57857.55**	58644.50	**57857.55**	**57857.55**	25.60	13.55	13.55	1.36	0.00	0.00
30	44086.53	44389.07	44099.32	44087.78	35.61	15.93	18.65	0.69	0.03	0.00
40	35005.82	35031.07	35037.22	35012.53	45.62	17.59	25.73	0.07	0.09	0.02
50	29130.10	29130.10	29228.42	**29089.78**	65.61	19.49	21.71	0.00	0.34	−0.14
60	25176.47	25335.92	25501.19	25166.15	90.05	15.90	31.41	0.63	1.29	−0.04
70	22186.14	22383.90	22205.34	**22125.53**	101.05	23.87	96.96	0.89	0.09	−0.27
80	19900.66	20016.74	19990.21	19877.88	110.07	17.46	50.09	0.58	0.45	−0.11
90	18055.94	18210.47	18129.06	**17987.94**	137.52	18.63	46.78	0.86	0.40	−0.38
100	16551.20	16718.28	16642.21	16586.68	165.53	20.17	39.41	1.01	0.55	0.21
150	12035.56	12121.79	12120.68	12032.65	215.90	35.04	150.26	0.72	0.71	−0.02
200	9362.99	9466.94	9421.32	9360.01	283.13	42.93	148.83	1.11	0.62	−0.03
250	7746.96	7794.77	7772.68	7742.70	362.21	31.75	122.69	0.62	0.33	−0.06
300	6628.92	6681.50	6675.19	6624.52	389.38	43.47	366.37	0.79	0.70	−0.07
350	5739.28	5832.61	5800.82	5727.02	407.88	38.97	360.74	1.63	1.07	−0.21
400	5045.84	5084.86	5078.97	5020.50	479.77	49.42	136.51	0.77	0.66	−0.50
450	4489.93	4514.54	4521.20	4487.73	467.14	33.58	77.58	0.55	0.70	−0.05
500	4062.86	4094.67	4080.59	4049.03	423.53	27.94	285.66	0.78	0.44	−0.34
	Average				212.24	26.22	111.23	0.76	0.47	−0.11

Bold represent that optimal solution is reached.

Table 4. 3038-customer problem.

	Objective values				CPU time			% Error r.t VNS		
p	VNS	FI	RVNS	VNDS	FI	RVNS	VNDS	FI	RVNS	VNDS
10	1213082.12	1216205.62	1213641.62	1213506.75	116.90	43.79	43.79	0.26	0.05	0.04
20	841560.25	846910.62	841432.00	841349.12	316.35	45.78	70.05	0.64	−0.02	−0.03
30	680540.06	686703.31	683510.25	683168.56	257.06	52.83	243.12	0.91	0.44	0.39
40	574575.25	576184.38	576375.50	573407.44	394.22	87.68	290.20	0.28	0.31	−0.20
50	507809.50	510330.19	510216.38	507655.19	612.87	60.71	311.15	0.50	0.47	−0.03
60	462293.53	465578.47	462794.34	462232.94	702.79	88.18	477.86	0.71	0.11	−0.01
70	428474.06	429152.31	429556.72	428062.66	902.57	73.81	679.82	0.16	0.25	−0.10
80	398081.28	401627.03	398617.22	397990.28	944.54	120.71	732.84	0.89	0.13	−0.02
90	375110.69	375737.53	374995.34	373846.97	1164.15	94.28	663.62	01.17	−0.03	−0.34
100	354488.69	356005.06	356666.34	353255.22	1248.77	158.95	1062.94	0.43	0.61	−0.35
150	281911.91	284158.97	283024.56	281772.09	1896.87	167.12	1862.07	0.80	0.39	−0.05
200	239086.41	240646.23	241355.64	238622.98	2526.70	139.85	2335.87	0.65	0.95	−0.19
250	209718.00	210612.94	210727.70	209343.34	3114.28	195.36	2846.61	0.43	0.48	−0.18
300	188142.30	189467.47	188709.30	187807.06	3358.40	169.75	1913.19	0.70	0.30	−0.18
350	171726.81	172668.55	172388.47	171009.30	3645.59	199.01	2951.38	0.55	0.39	−0.42
400	157910.08	158549.50	158805.00	157079.67	5308.28	206.29	4772.22	0.40	0.57	−0.53
450	146087.80	146727.20	147061.95	145448.98	5491.01	233.34	2148.47	0.44	0.67	−0.44
500	136081.72	136680.48	136664.97	135467.97	6044.12	272.65	3379.78	0.44	0.43	−0.45
550	127029.12	127804.91	127989.73	126867.38	6407.13	318.19	2883.01	0.61	0.76	−0.13
600	119554.00	120330.77	120408.42	119107.99	4845.88	196.81	3392.02	0.65	0.71	−0.37
650	112516.82	113386.11	113190.49	112090.28	5275.87	243.98	5160.90	0.77	0.60	−0.38
700	106194.02	107065.33	106960.07	105893.39	5346.42	188.82	3421.69	0.82	0.72	−0.28
750	100744.98	101590.45	101512.59	100362.55	5108.20	202.88	5095.70	0.84	0.76	−0.38
800	95832.92	96599.56	96681.89	95445.06	5024.91	164.18	4143.99	0.80	0.89	−0.40
850	91452.21	92021.92	92317.28	91023.87	5023.56	157.50	4733.62	0.62	0.95	−0.47
900	87337.32	88051.39	88149.26	87041.84	5656.95	193.38	5549.94	0.82	0.93	−0.34
950	83654.05	84202.81	84217.28	83310.19	5079.25	238.98	4328.75	0.66	0.67	−0.41
1000	80213.42	80692.07	80662.78	79900.52	4883.87	205.55	4879.89	0.60	0.56	−0.39
	Average				3239.20	161.44	2513.37	0.59	0.50	−0.24

the values in Table 3 where the heuristic solution is equal to the optimal one. The RL5934 problem appears to be larger than p-Median problems previously reported in the literature. The second column of Table 5 contains the best value found by the three methods compared.

It appears that: (i) solutions of similar quality are obtained by FI and RVNS, despite the fact that this last method uses no local search, but RVNS is about 8, 20 and 40 times faster then FI in Tables 3–5 respectively; (ii) VNDS outperforms FI within similar CPU times; (iii) VNDS is 0.11% and 0.24% better on average than VNS, using five times less

Table 5. 5934-customer problem.

	Objective value	CPU times			% Error		
p	Best known	FI	RVNS	VNDS	FI	RVNS	VNDS
10	9794951.00	642.83	85.56	85.56	0.00	0.01	0.01
20	6729282.50	2143.16	66.73	66.73	0.23	0.00	0.00
30	5405661.50	2061.82	183.99	275.59	0.25	0.01	0.00
40	4574374.00	2915.30	331.86	854.07	0.51	0.23	0.00
50	4053917.75	4257.26	418.05	550.18	0.72	0.06	0.00
60	3655898.75	4640.17	241.67	516.79	1.03	0.08	0.00
70	3353885.00	5847.18	296.33	1264.27	0.58	0.08	0.00
80	3104877.75	5451.85	501.01	891.03	0.63	0.11	0.00
90	2903895.25	5722.39	420.01	1453.24	0.39	0.44	0.00
100	2733817.25	6637.48	510.20	6087.75	0.36	0.15	0.00
150	2151018.50	9971.28	373.54	2657.35	0.90	0.86	0.00
200	1809064.38	14966.05	663.69	14948.37	0.79	0.36	0.00
250	1571813.50	17118.27	620.14	11042.62	0.73	0.40	0.00
300	1394715.12	20127.91	541.76	17477.51	0.65	0.51	0.00
350	1257900.00	22003.88	868.85	21769.56	0.82	0.55	0.00
400	1145669.38	23630.95	618.62	22283.04	0.82	0.59	0.00
450	1053450.88	66890.41	1898.83	21683.38	0.90	0.79	0.00
500	974275.31	29441.97	954.10	10979.77	0.98	0.51	0.00
600	848459.38	32957.36	768.95	18996.37	0.78	0.47	0.00
700	752068.38	36159.45	768.84	32249.00	0.64	0.50	0.00
800	676846.12	38887.40	813.38	20371.81	0.61	0.53	0.00
900	613367.44	41607.78	731.71	27060.09	0.55	0.53	0.00
1000	558802.38	44176.27	742.70	26616.96	0.73	0.66	0.00
1100	511813.19	45763.18	740.33	28740.89	0.90	0.63	0.00
1200	470295.38	46387.06	674.87	15886.20	0.99	0.91	0.00
1300	433597.44	46803.63	740.48	26150.85	1.00	0.82	0.00
1400	401853.00	47184.10	708.90	44944.41	0.87	1.18	0.00
1500	374061.41	47835.35	823.95	44727.36	0.98	0.93	0.00
	Average	24008.30	611.04	15026.60	0.68	0.47	0.00

CPU times. Moreover, VNDS solved exactly 5 among 8 problems from Table 3, where the optimal solution is known.

5. Conclusions

A new decomposition heuristic method for solving combinatorial and global optimization problems, named Variable Neighborhood Decomposition Search (VNDS) is proposed. It

follows the rules of the recent Variable Neighborhood Search metaheuristic and combines them with a successive approximation decomposition method. The sequence of subproblems (problems of smaller sizes than the initial one) are generated from the different preselected set of neighborhoods. If the solution of the subproblem does not lead to an improvement in the whole space, the neighborhood is changed. Otherwise, the search continues from the incumbent in the first pre-selected neighborhood. The process is iterated until some stopping condition is met.

A VNDS heuristic is presented and illustrated on the p-Median problem. It is shown that for medium size instances, VNDS does not always give satisfactory results (compared with basic VNS), but for very large problems it can be very useful. Results on 1400, 3038 and 5934 node instances from the TSP library show VNDS improves notably upon VNS in less computing time, and gives much better results than Fast Interchange (FI), in the same time that FI takes for a single descent.

References

Baum, E.B. (1986). "Toward Practical 'Neural' Computation for Combinatorial Optimization Problems." In J. Denker (ed.), *Neural Networks for Computing.* New York: American Institute of Physics.

Beasley, J.E. (1985). "A Note on Solving Large p-Median Problems." *European Journal of Operational Research* 21, 270–273.

Bellman, R. and S. Dreyfuss. (1962). *Applied Dynamic Programming*, Princeton: Princeton University Press.

Boese, K.D., A.B. Kahng, and S. Muddu. (1994). "A New Adaptive Multi-Start Technique for Combinatorial Global Optimizations." *Operations Research Letters* 16, 101–113.

Brandeau, M.L. and S.S. Chiu. (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. and P.M. Pardalos. (1990). "An Exact Algorithm for the Maximum Clique Problem." *Operations Research Letters* 9, 375–382.

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

Cornuejols, G., M.L. Fisher, and G.L. Nemhauser. (1977). "Location of Bank Accounts to Optimize Float: An Analytic Study of Exact and Approximate Algorithms." *Management Science* 23, 789–810.

Densham, P.J. and G. Rushton. (1992). "A More Efficient Heuristic for Solving Large p-Median Problems." *Papers in Regional Science* 71(3), 307–329.

Drezner, Z. (ed.). (1995). *Facility Location. A survey of Applications and Methods.* New York: Springer.

du Merle, O., D. Villeneuve, J. Desrosiers, and P. Hansen. (1999). "Stabilized Column Generation." *Discrete Mathematics* 194, 229–237.

Erlenkotter, D. (1978). "A Dual-Based Procedure for Uncapacitated Facility Location." *Operations Research* 26, 992–1009.

Galvao, 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. and M. Laguna. (1993). "Tabu Search." In C. Reeves (ed.), *Modern Heuristic Techniques for Combinatorial Problems*, Ch. 3, Oxford: Blackwell.

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

Glover, F., C. McMillan, and R. Glover. (1984). "A Heuristic Approach to the Employee Scheduling Problem and Some Thoughts on "Managerial Robots." *Journal of Operations Management* 4, 113–128.

Griffith, R.E. and R.A. Stewart. (1961). "A Nonlinear Programming Technique for the Optimization of Continuous Processing Systems." *Management Science* 7, 379–392.

Hanjoul, P. and D. Peeters. (1985). "A Comparison of Two Dual-Based Procedures for Solving the p-Median Problem." *European Journal of Operational Research* 20, 387–396.

Hansen, P. and B. Jaumard. (1990). "Algorithms for the Maximum Satisfiability Problem." *Computing* 44, 279–303.

Hansen, P. and B. Jaumard. (1997). "Cluster Analysis and Mathematical Programming." *Mathematical Programming* 79, 191–215.

Hansen, P. and L. Kaufman. (1972). "Comparaison d'Algorithmes Pour le Problème de la Localisation des Enterpôts." In J. Brennan (ed.), *Operational Research in Industrial Systems*. London: English University Press, pp. 281–294.

Hansen, P. and N. Mladenović. (1997). "Variable Neighborhood Search for the p-Median." *Location Science* 5, 207–226.

Hansen, P. and N. Mladenović. (1998). "An Introduction to Variable Neighborhood Search." In S. Voss et al. (eds.), *Metaheuristics, Advances and Trends in Local Search Paradigms for Optimization*. Dordrecht: Kluwer, pp. 433–458.

Kariv, O. and S.L. Hakimi. (1969). "An Algorithmic Approach to Network Location Problems; Part 2. The p-Medians." *SIAM Journal on Applied Mathematics* 37, 539–560.

Kirkpatrick, S., C.D. Gelatt, and M.P. Vecchi. (1983). "Optimization by Simulated Annealing." *Science* 220, 671–680.

Kuehn, A.A. and M.J. Hamburger. (1963). "A Heuristic Program for Locating Warehouses." *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. and R. Francis (eds.). (1990). *Discrete Location Theory*. Wiley-Interscience.

Mladenović, N. (1995). "A Variable Neighborhood Algorithm—A New Metaheuristic for Combinatorial Optimization." Presented at Optimization Days, Montreal.

Mladenović, N. and P. Hansen. (1997). "Variable Neighborhood Search." *Computers and Operations Research* 24, 1097–1100.

Mladenović, N., J.P. Moreno, and J. Moreno-Vega. (1995). "Tabu Search in Solving p-Facility Location - Allocation Problems." *Les Cahiers du GERAD, G-95–38*, Montreal.

Mladenović, N., J.P. Moreno, and J. Moreno-Vega. (1996). "A Chain-Interchange Heuristic Method." *Yugoslav Journal of Operations Research* 6(1), 41–54.

Moreno, J., C. Rodrigez, and N. Jimenez. (1991). "Heuristic Cluster Algorithm for Multiple Facility Location - Allocation Problem." *RAIRO—Recherche Operationnelle/Operations Research* 25, 97–107.

Osman, I. and N. Christofides. (1994). "Capacitated Clustering Problems by Hybrid Simulated Annealing and Tabu Search." *Int. Trans. Oper. Res.* 1(3), 317–336.

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, 376–384.

Rolland, E., D.A. Schilling, and J.R. Current. (1996). "An Efficient Tabu Search Procedure for the p-Median Problem." *European Journal of Operational Research* 96, 329–342.

Rosing, K.E. and C.S. ReVelle. (1997). "Heuristic Concentration: Two Stage Solution Construction." *European Journal of Operational Research* 97, 75–86.

Rosing, K.E., C.S. ReVelle, E. Rolland, D.A. Schilling, and J.R. Current. (1998). "Heuristic Concentration and Tabu Search: A Head to Head Comparison." *European Journal of Operational Research* 104, 93–99.

Rosing, K.E. (1998). Private communication.

Teitz, M.B. and P. Bart. (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.