Chapter 12
Variable Neighborhood Search

Pierre Hansen and Nenad Mladenović

12.1 Introduction

Variable neighborhood search (VNS) is a metaheuristic, or framework for building heuristics, which exploits systematically the idea of neighborhood change, both in the descent to local minima and in the escape from the valleys which contain them. In this tutorial we first present the ingredients of VNS, i.e. variable neighborhood descent (VND) and Reduced VNS (RVNS) followed by the basic and then the general scheme of VNS itself which contain both of them. Extensions are presented, in particular Skewed VNS which enhances exploration of far away valleys and variable neighborhood decomposition search (VNDS), a two-level scheme for solution of large instances of various problems. In each case, we present the scheme, some illustrative examples and questions to be addressed in order to obtain an efficient implementation.

Let us consider a combinatorial or global optimization problem

$$\min_{x \in X} f(x) \tag{12.1}$$

where $f(x)$ is the objective function to be minimized and X the set of feasible solutions. A solution $x^* \in X$ is *optimal* if

$$f(x^*) \leq f(x), \ \forall x \in X. \tag{12.2}$$

An *exact algorithm* for problem (12.1) and (12.2), if one exists, finds an optimal solution x^*, together with the proof of its optimality, or shows that there is no feasible solution, that is $X = \emptyset$. Moreover, in practice, the time to do so should be finite

P. Hansen (✉)
GERAD and HEC Montreal, Montréal, Québec, Canada
e-mail: Pierre.Hansen@gerad.ca

N. Mladenović
School of Mathematics, Brunel University, Uxbridge, Middlesex UB8 3PH, UK

E.K. Burke and G. Kendall (eds.), *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, DOI 10.1007/978-1-4614-6940-7_12,

(and not too large); if one deals with a continuous function one must admit a degree of tolerance i.e. stop when a feasible solution x^* has been found such that

$$f(x^*) < f(x) + \varepsilon, \ \forall x \in X \tag{12.3}$$

or

$$\frac{f(x^*) - f(x)}{f(x^*)} < \varepsilon, \ \forall x \in X \tag{12.4}$$

for some small positive ε.

Numerous instances of problems of the form (12.1) and (12.2), arising in Operational Research and other fields, are too large for an exact solution to be found in reasonable time. It is well known from complexity theory (Garey and Johnson 1979; Papadimitriou 1994) that thousands of problems are *NP-hard*, that no algorithm with a number of steps polynomial in the size of the instances is known and that finding one for any such problem would entail obtaining one for any and all of them. Moreover, in some cases where a problem admits a polynomial algorithm, the power of this polynomial may be so large that realistic size instances cannot be solved in reasonable time in worst case, and sometimes also in the average case or most of the time.

So one is often forced to resort to heuristics, which yield quickly an approximate solution, or sometimes an optimal solution but without proof of its optimality. Some of these heuristics have a worst-case guarantee, i.e. the solution x_h obtained satisfies

$$\frac{f(x_h) - f(x)}{f(x_h)} \leq \varepsilon, \ \forall x \in X \tag{12.5}$$

for some ε, which is however rarely small. Moreover, this ε is usually much larger than the error observed in practice and may therefore be a bad guide in selecting a heuristic. In addition to avoiding excessive computing time, heuristics address another problem: local optima. A local optimum x_L of (1) and (2) is such that

$$f(x_L) \leq f(x), \ \forall x \in N(x_L) \cap X \tag{12.6}$$

where $N(x_L)$ denotes a neighborhood of x_L (ways to define such a neighborhood will be discussed below). If there are many local minima, the range of values they span may be large. Moreover, the globally optimum value $f(x^*)$ may differ substantially from the average value of a local minimum, or even from the best such value among many, obtained by some simple heuristic (a phenomenon called the Tchebycheff catastrophe by Baum 1986). There are, however, many ways to get out of local optima and, more precisely, the valleys which contain them (or set of solutions from which the descent method under consideration leads to them).

Metaheuristics are general frameworks to build heuristics for combinatorial and global optimization problems. For a discussion of the best-known of them the reader is referred to the books of surveys edited by Reeves (1993) and Glover and Kochenberger (2003) as well as to other chapters of the present volume. Some of the many successful applications of metaheuristics are also mentioned there.

VNS (Mladenović and Hansen 1997; Hansen and Mladenović 1999, 2001c, 2003) is a recent metaheuristic which exploits systematically the idea of neighborhood change, both in descent to local minima and in escape from the valleys which contain them. VNS exploits systematically the following observations:

Fact 1 *A local minimum with respect to one neighborhood structure is not necessarily so for another.*

Fact 2 *A global minimum is a local minimum with respect to all possible neighborhood structures.*

Fact 3 *For many problems local minima with respect to one or several neighborhoods are relatively close to each other.*

This last observation, which is empirical, implies that a local optimum often provides some information about the global one. This may for instance be several variables with the same value in both. However, it is usually not known which ones are such. An organized study of the neighborhood of this local optimum is therefore in order, until a better one is found.

Unlike many other metaheuristics, the basic schemes of VNS and its extensions are simple and require few, and sometimes no parameters. Therefore, in addition to providing very good solutions, often in simpler ways than other methods, VNS gives insight into the reasons for such a performance, which in turn can lead to more efficient and sophisticated implementations.

The chapter is organized as follows. In the next section, we examine the preliminary problem of gathering information about the problem under study, and evaluating it. In Sect. 12.3 the first ingredient of VNS, i.e. VND, which is mostly or entirely deterministic, is studied. Section 12.4 is devoted to the second ingredient, RVNS, which is stochastic. Both ingredients are merged in the basic and the general VNS schemes, described in Sect. 12.5. Extensions are then considered. Skewed VNS, which addresses the problem of getting out of very large valleys, is discussed in Sect. 12.6. Very large instances of many problems cannot be solved globally in reasonable time; VNDS studied in Sect. 12.7 is a two-level scheme which merges VNS with successive approximation (including a two-level VNS). Various tools for analyzing in detail the performance of a VNS heuristic, and then streamlining it are presented in Sect. 12.8. They include *distance-to-target diagrams* and *valley profiles*. In each of these sections basic schemes, or tools, are illustrated by examples from papers by a variety of authors. Questions to be considered in order to get an efficient implementation of VNS are also systematically listed. Promising areas of research are outlined in Sect. 12.9. Brief conclusions complete the chapter in Sect. 12.10. Finally, sources of further information are suggested.

12.2 Preliminaries: Documentation

Once a problem of the form (12.1) and (12.2) has been selected for study and approximate solution by VNS, a preliminary step is to gather in a thorough way the papers written about it or closely related problems. Note that this may be a difficult task as papers are often numerous, dispersed among many journals and volumes of proceedings and the problem may appear (usually under different names) in several fields. Tools such as the *ISI Web of Knowledge*, *NEC Research's Citeseer* or even general web browsers such as *Google* may prove to be very useful.

There are several reasons for studying the literature on the selected problem:

(i) *Evaluating its difficulty*. Is it NP-hard? Is it strongly NP-hard? (and hence admits no fully polynomial approximation scheme). If it is in P, what is the complexity of the best-known exact algorithm, and is it sufficiently low for realistic instances to be solvable in reasonable time?

(ii) *Evaluating the performance of previous algorithms*. Are there some instances of (preferably real-word) data for the problem available (e.g. at http://www.informs.org/Resources/Resources/Problem_Instances/)? What are the largest instances solved exactly?

(iii) *Evaluating the performance of previous heuristics*. Which metaheuristics have been applied to this problem? What are the performances of the resulting heuristics, in terms of size of problems solved, error and computing time (assuming comparison among computing environments, if needed, can be done in a fairly realistic way)?

(iv) *What steps are used* in the heuristics already proposed? What are the corresponding neighborhoods of the current solution? Are codes for these heuristics available? Are codes for simple descent methods available?

Question (i)'s role is to help to assess the need for a VNS (or others) heuristic for the problem considered. Questions (ii) and (iii) aim at obtaining a benchmark to evaluate the performance of the VNS heuristic when it will be designed and implemented: a good heuristic should obtain optimal solutions for most and preferably all instances solved by an exact algorithm (which suffers from the additional burden of having to prove optimality). Moreover, the new heuristic should do as well as previous ones on most or all instances and substantially better than them on quite a few instances to be viewed as a real progress (doing slightly better on a few instances is just not sufficient).

Question (iv) aims at providing ingredients for the VNS heuristic, notably in its VND component; it also inquires indirectly about directions not yet explored. Incidentally, it raises the question of possible re-use of software, which is reasonable for standard steps, e.g. a descent with Newton's method or a variant thereof.

Initialization.
Choose f, X, neighborhood structure $N(x)$, initial solution x;
Current step (Repeat).
(1) Find $x' = \arg\min_{x \in N(x)} f(x)$;
(2) If $f(x') < f(x)$ set $x' \leftarrow x''$ and iterate; otherwise, stop.

Fig. 12.1 Steepest-descent heuristic

Initialization.
Choose f, X, neighborhood structure $N(x)$, initial solution x;
Current step (Repeat).
(1) Find first solution $x' \in N(x)$;
(2) If $f(x') > f(x)$, find next solution $x'' \in N(x)$; set $x' \leftarrow x''$ and iterate (2); otherwise, set $x \leftarrow x'$ and iterate (1);
(3) If all solutions of $N(x)$ have been considered, stop.

Fig. 12.2 First-descent heuristic

12.3 Variable Neighborhood Descent

A *steepest-descent* heuristic (known also as *best improvement* local search) consists of choosing an initial solution x, finding a direction of steepest descent from x, within a neighborhood $N(x)$, and moving to the minimum of $f(x)$ within $N(x)$ along that direction; if there is no direction of descent, the heuristic stops, and otherwise it is iterated. This set of rules is summarized in Fig. 12.1.

Observe that a neighborhood structure $N(x)$ is defined for all $x \in X$; in discrete optimization problems it usually consists of all vectors obtained from x by some simple modification, e.g. complementing one or two components of a 0–1 vector. Then, at each step, the neighborhood $N(x)$ of x is explored completely. As this may be time-consuming, an alternative is to use the *first-descent* heuristic. Vectors $x' \in N(x)$ are then enumerated systematically and a move is made as soon as a descent direction is found. This is summarized in Fig. 12.2.

VND is based on Fact 1 of the Introduction, i.e. *a local optimum for a first type of move* $x \leftarrow x'$ (or heuristic, or within the neighborhood $N_1(x)$) *is not necessary one for another type of move* $x \leftarrow \tilde{x}$ (within neighborhood $N_2(x)$)). It may thus be advantageous to combine descent heuristics. This leads to the basic VND scheme presented in Fig. 12.3.

Caution should be exercised when applying that scheme. In particular one should consider the following questions:

(i) What complexity do the different moves have?
(ii) What is the best order in applying them?

Initialization. Select the set of neighborhood structures N_ℓ, for $\ell = 1, \ldots, \ell_{\max}$, 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 $\ell \leftarrow 1$;
(2) *Repeat* the following steps until $\ell = \ell_{\max}$:
(a) Exploration of neighborhood. Find the best neighbor x' of x ($x' \in N_\ell(x)$);
(b) Move or not. If the solution x' thus obtained is better than x, set $x \leftarrow x'$ and $\ell \leftarrow 1$; otherwise, set $\ell \leftarrow \ell + 1$;

Fig. 12.3 Steps of the basic VND

(iii) Are the moves considered sufficient to ensure a thorough exploration of the region containing x?
(iv) How precise a solution is desired?

Question (i) aims at selecting and ranking moves: if they involve too many elementary changes (e.g. complementing three components or more of a 0–1 vector), the resulting heuristic may be very slow and often takes more time than an exact algorithm on small or medium size examples.

Question (ii) also bears upon computing times in relation to the quality of solutions obtained. A frequent implementation consists of ranking moves by order of complexity of their application (which is often synonymous with by size of their neighborhoods $|N_\ell(x)|$), and returning to the first one each time a direction of descent is found and a step made in that direction. Alternatively, all moves may be applied in sequence as long as descent is made for some neighborhood in the series.

Question (iii) is a crucial one: for some problems elementary moves are not sufficient to leave a narrow valley, and heuristics using them only can give very poor results. This is illustrated in Example 12.2.

Finally, the precision desired, as asked for in question (iv) will depend upon whether VND is used alone or within some larger framework, such as VNS itself. In the former case, one will strive to obtain the best solution possible within the allocated computing time; in the latter, one may prefer to get a good solution fairly quickly by the deterministic VND and to improve it later by faster stochastic search in VNS.

Example 12.1 (Simple plant location (see Cornuejols et al. 1990, for a survey)). The simple (or uncapacitated) plant location problem consists of locating a set of facilities i among a given set I of m potential locations, with fixed costs f_i, in order to minimize total costs for satisfying the demand of a given set of users J with delivery costs $c_{ij}, i \in I, j \in J$. It is expressed as follows:

$$\min_{x,y} z_P = \sum_{i=1}^{m} f_i y_i + \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{12.7}$$

s.t.

$$\sum_{i=1}^{m} x_{ij} = 1,\ \forall j \in J \tag{12.8}$$

$$y_i - x_{ij} \geq 0,\ \forall i \in I,\ \forall j \in J \tag{12.9}$$

$$y_i \in \{0,1\},\ \forall i \in I \tag{12.10}$$

$$x_{ij} \geq 0,\ \forall i \in I,\ \forall j \in J, \tag{12.11}$$

where $y_i = 1$ if a facility is located at i, and 0 otherwise; $x_{ij} = 1$ if demand of user j is satisfied from facility i and 0 otherwise. Note that for fixed y_i, the best solution is defined by

$$x_{ij} = \begin{cases} 1 \text{ if } c_{ij} = \min_{\ell | y_\ell = 1} c_{\ell j} \text{ (with minimum index } \ell \text{ in case of ties);} \\ 0 \text{ otherwise.} \end{cases}$$

Therefore neighborhoods can be defined on the y_i, e.g. by Hamming distance (or number of components with complementary values). A first heuristic, Greedy, proceeds by opening a facility ℓ with minimum total cost:

$$f_\ell + \sum_j c_{\ell j} = \min_i \left\{ f_i + \sum_j c_{ij} \right\} \tag{12.12}$$

then letting

$$c_{rj} = \min_{i | y_i = 1} c_{ij},\ \forall j \tag{12.13}$$

computing the gains g_i obtained by opening a facility at i

$$g_i = \sum_j \max\{c_{rj} - c_{ij}, 0\} - f_i \tag{12.14}$$

and iteratively opening the facility for which the gain is larger, as long as it is positive. Each iteration takes $O(mn)$ time.

Once the Greedy heuristic has been applied, an improved solution may be obtained by the Interchange heuristic which proceeds iteratively to the relocation of one facility at a time in the most profitable way. With an efficient implementation, the idea of which was suggested by Whitaker (1983) for the closely related p-median problem, an iteration of interchange can also be made in $O(mn)$ time.

Applying in turn Greedy and Interchange is a simple case of VND. Further moves in which one facility would be closed and two opened, or two closed and one opened, or two opened and two closed would be too costly if all possible exchanges are examined.

Example 12.2 (Minimum sum-of-squares clustering, MSSC). Given N points $a_\ell \in \mathbf{R}^p$, the MSSC problem consists of partitioning them in M classes (or clusters) C_j such as to minimize the sum of squared distances between the points and the centroids $\bar{x}_i$ of their clusters:

$$\min \sum_{i=1}^{m} \sum_{\ell : a_\ell \in C_i} \|a_\ell - \bar{x}_i\|^2 \tag{12.15}$$

where

$$\bar{x}_i = \frac{1}{|C_i|} \sum_{\ell: a_\ell \in C_i} a_\ell \tag{12.16}$$

and $\|.\|$ denotes the Euclidean norm.

Traditional heuristics for MSSC are: (i) H-means, which proceeds from an initial partition by moving one entity x_ℓ from its cluster to another one, in a greedy way, until no further move decreases the objective function value, and (ii) K-means, which proceeds from an initial partition by, alternatingly, finding the centroids of its clusters, and reassigning entities to the closest centroid, until stability is attained.

Computational experiments (Hansen and Mladenović 2001) show that both H-means and K-means may lead to very poor results for instances with large M and N (the relative error being sometimes greater than 100 %). This is due to bad exploration of X, or in other words, to difficulties in leaving valleys. A new *jump* move, defined as the displacement of a centroid to a point a_ℓ which does not coincide with a centroid, leads to a new VND heuristic, called J-means, which improves very substantially on both H-means and K-means.

12.4 Reduced VNS

Assume a local minimum x of f has been reached. One would then like to leave its valley, and find another deeper one. In the standard versions of VNS, no previous knowledge of the landscape is assumed, or exploited. (Note that interesting hybrids could be built, using also values of $f(x)$ at previous iteration points x). Then, the questions to be asked are

(i) Which direction to go?
(ii) How far?
(iii) How should one modify moves if they are not successful?

Question (i) bears upon the possibility of reaching any feasible point $x \in X$, or every valley; the simplest answer is to choose a direction at random. For problems in 0–1 variables this will amount to complementing some variables; for continuous Euclidean problems, drawing angular coefficients at random (or, in other words, choosing at random a point on the unit ball around x) takes all points of X into account.

Question (ii) is crucial. Indeed one wants to exploit to the limit Fact 2 of the Introduction, i.e. in many combinatorial and global optimization problems, local optima tend to be close one to another and situated in one (or sometimes several) small parts of X. So once a local optimum has been reached, it contains implicit information about close better, and perhaps globally optimum, ones. It is then natural to explore first its vicinity. But, if the valley surrounding the local optimum x is large, this may not be sufficient, and what to do next is asked for in question (iii). Again a natural answer is to go further.

Initialization. Select the set of neighborhood structures $\mathcal{N}_k$, for $k = 1, \ldots, k_{\max}$, 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) *Repeat* the following steps until $k = k_{\max}$:
(a) Shaking. Generate a point x' at random from the kth neighborhood of x ($x' \in \mathcal{N}_k(x)$);
(b) Move or not. If this point is better than the incumbent, move there ($x \leftarrow x'$), and continue the search with $\mathcal{N}_1$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$;

Fig. 12.4 Steps of the Reduced VNS

These aims are pursued in the RVNS scheme, presented in Fig. 12.4. A set of neighborhoods $N_1(x), N_2(x), \ldots, N_{k_{\max}}(x)$ will be considered around the current point x (which may be or not a local optimum). Usually, these neighborhoods will be nested, i.e. each one contains the previous. Then a point is chosen at random in the first neighborhood. If its value is better than that of the incumbent (i.e. $f(x') < f(x)$), the search is re-centered there ($x \leftarrow x'$). Otherwise, one proceeds to the next neighborhood. After all neighborhoods have been considered, one begins again with the first, until a stopping condition is satisfied (usually it will be maximum computing time since the last improvement, or maximum number of iterations).

Due to the nestedness property the size of successive neighborhoods will be increasing. Therefore, one will explore more thoroughly close neighborhoods of x than farther ones, but nevertheless search within these when no further improvements are observed within the first, smaller ones.

Example 12.3 (p-median (see Labbé et al. 1995 for a survey)). This is a location problem very close to that of Simple Plant Location. The differences are that there are no fixed costs, and that the number of facilities to be opened is set at a given value p. It is expressed as follows:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{12.17}$$

subject to

$$\sum_{i=1}^{m} x_{ij} = 1, \quad \forall j \tag{12.18}$$

$$y_i - x_{ij} \geq 0, \quad \forall i, j \tag{12.19}$$

$$\sum_{i=1}^{m} y_i = p \tag{12.20}$$

$$x_{ij}, y_i \in \{0, 1\}. \tag{12.21}$$

The Greedy and Interchange heuristics described above for Simple Plant Location are easily adapted to the p-median problem and, in fact, the latter was early proposed by Teitz and Bart (1967).

Table 12.1 5934-customer p-median problem

	Obj. value	CPU times			% Error		
p	*Best known*	FI	RVNS	VNDS	FI	RVNS	VNDS
100	27,33,817.25	6,637.48	510.20	6,087.75	0.36	0.15	0.00
200	18,09,064.38	14,966.05	663.69	14,948.37	0.79	0.36	0.00
300	13,94,715.12	20,127.91	541.76	17,477.51	0.65	0.51	0.00
400	11,45,669.38	23,630.95	618.62	22,283.04	0.82	0.59	0.00
500	9,74,275.31	29,441.97	954.10	10,979.77	0.98	0.51	0.00
700	7,52,068.38	36,159.45	768.84	32,249.00	0.64	0.50	0.00
800	6,76,846.12	38,887.40	813.38	20,371.81	0.61	0.53	0.00
900	6,13,367.44	41,607.78	731.71	27,060.09	0.55	0.53	0.00
1,000	5,58,802.38	44,176.27	742.70	26,616.96	0.73	0.66	0.00
Average		28,403.90	705.00	19,786.00	0.68	0.48	0.00

Initialization. Select the set of neighborhood structures $\mathcal{N}_k$, for $k = 1, \ldots, k_{\max}$, 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) *Repeat* the following steps until $k = k_{\max}$:

(a) Shaking. Generate a point x' at random from the kth 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 the local optimum x'' is better than the incumbent x, move there ($x \leftarrow x''$), and continue the search with $\mathcal{N}_1$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$;

Fig. 12.5 Steps of the basic VNS

Fast interchange, using Whitaker's (1983) data structure applies here also (Hansen and Mladenović 1997). Refinements were proposed by Resende and Werneck (2002). A comparison between that approach and RVNS is made in Hansen et al. (2001), and the results are summarized in Table 12.1. It appears that RVNS gives better results than Fast Interchange in 2.5 % of the time.

12.5 Basic and General VNS

In the previous two sections, we examined how to use variable neighborhoods in descent to a local optimum and in finding promising regions for near-optimal solutions. Merging the tools for both tasks leads to the General VNS scheme. We first discuss how to combine a local search with systematic changes of neighborhoods around the local optimum found. We then obtain the Basic VNS scheme presented in Fig. 12.5.

According to this basic scheme, a series of neighborhood structures, which define neighborhoods around any point $x \in X$ of the solution space, are first selected. Then

Initialization. Select the set of neighborhood structures $\mathcal{N}_k$, for $k = 1, \ldots, k_{\max}$, that will be used in the shaking phase, and the set of neighborhood structures N_ℓ for $\ell = 1, \ldots, \ell_{\max}$ that will be used in the local search; find an initial solution x and improve it by using RVNS; choose a stopping condition;
Repeat the following sequence until the stopping condition is met:
(1) Set $k \leftarrow 1$;
(2) *Repeat* the following steps until $k = k_{\max}$:
(a) Shaking. Generate a point x' at random from the kth neighborhood $\mathcal{N}_k(x)$ of x;
(b) Local search by VND.
(b1) Set $\ell \leftarrow 1$;
(b2) Repeat the following steps until $\ell = \ell_{\max}$;
· *Exploration of neighborhood.* Find the best neighbor x'' of x' in $N_\ell(x')$;
· *Move or not.* If $f(x'') < f(x')$ set $x' \leftarrow x''$ and $\ell \leftarrow 1$; otherwise set $\ell \leftarrow \ell + 1$;
(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 $k \leftarrow k + 1$;

Fig. 12.6 Steps of the general VNS

the local search is used and leads to a local optimum x. A point x' is selected at random within the first neighborhood $\mathcal{N}_1(x)$ of x and a descent from x' is done with the local search routine. This leads to a new local minimum x''. At this point, three outcomes are possible: (i) $x'' = x$, i.e. one is again at the bottom of the same valley; in this case the procedure is iterated using the next neighborhood $\mathcal{N}_k(x)$, $k \geq 2$; (ii) $x'' \neq x$ but $f(x'') \geq f(x)$, i.e. another local optimum has been found, which is not better than the previous best solution (or incumbent); in this case too the procedure is iterated using the next neighborhood; (iii) $x'' \neq x$ and $f(x'') < f(x)$ i.e. another local optimum, better than the incumbent has been found; in this case the search is recentered around x'' and begins again with the first neighborhood. Should the last neighborhood be reached without a solution better than the incumbent being found, the search begins again at the first neighborhood $\mathcal{N}_1(x)$ until a stopping condition, e.g. a maximum time or maximum number of iterations or maximum number of iterations since the last improvement, is satisfied.

If instead of simple local search, one uses VND and if one improves the initial solution found by Reduced VNS, one obtains the General VNS scheme. This scheme is presented in Fig. 12.6.

Several questions about selection of neighborhood structures are in order:

(i) What properties of the neighborhoods are mandatory for the resulting scheme to be able to find a globally optimal or near-optimal solution?
(ii) What properties of the neighborhoods will favor finding a near-optimal solution?
(iii) Should neighborhoods be nested? Otherwise how should they be ordered?
(iv) What are desirable properties of the sizes of neighborhoods?

The first two questions bear upon the ability of the VNS heuristic to find the best valleys, and to do so fairly quickly. To avoid being blocked in a valley, while there

may be deeper ones, the union of the neighborhoods around any feasible solution x should contain the whole feasible set:

$$X \subseteq \mathcal{N}_1(x) \cup \mathcal{N}_2(x) \cup \cdots \cup \mathcal{N}_{k_{\max}}(x), \ \ \forall x \in X.$$

These sets may cover X without necessarily partitioning it, which is easier to implement, e.g. when using nested neighborhoods, i.e.

$$\mathcal{N}_1(x) \subset \mathcal{N}_2(x) \subset \cdots \subset \mathcal{N}_{k_{\max}}(x), \ \ X \subset \mathcal{N}_{k_{\max}}(x), \ \ \forall x \in X.$$

If these properties do not hold, one might still be able to explore X completely, by traversing small neighborhoods around points on some trajectory, but it is not guaranteed. To illustrate, as mentioned before in MSSC, the neighborhoods defined by moving an entity (or even a few entities) from one cluster to another one are insufficient to get out of many local optima. Moving centers of clusters does not pose a similar problem.

Nested neighborhoods are easily obtained for many combinatorial problems by defining a first neighborhood $\mathcal{N}_1(x)$ by a type of move—e.g. 2-opt in the traveling salesman problem (TSP)—and then iterating it k times to obtain neighborhoods $\mathcal{N}_k(x)$ for $k = 2, \ldots, k_{\max}$. They have the property that their sizes are increasing. Therefore if, as is often the case, one goes many times through the whole sequence of neighborhoods the first ones will be explored more thoroughly than the last ones. This is desirable in view of Fact 3 mentioned in the Introduction, i.e. that local optima tend to be close one from another.

Restricting moves to the feasible set X may be too constraining, particularly if this set is disconnected. Introducing some or all constraints in the objective function with Lagrangian multipliers, makes it possible to move to infeasible solutions. A variant of this idea is to penalize infeasibility, e.g. pairs of adjacent vertices to which the same color is assigned in graph coloring (see Zufferey et al. 2003).

Example 12.4 (Scheduling workover rigs for onshore oil production). Many oil wells in onshore fields rely on artificial lift methods. Maintenance services such as cleaning and others, which are essential to these wells, are performed by workover rigs. They are slow mobile units and, due to their high operation costs, there are relatively few workover rigs when compared with the number of wells demanding service. The problem of scheduling workover rigs consists in finding the best schedule S_i $(i = 1, \ldots, m)$ of the m workover rigs to attend all wells demanding maintenance services, so as to minimize the oil production loss (production before maintenance being reduced).

In Aloise et al. (2003) a basic VNS heuristic is developed for solving the Scheduling of Workover Rigs Problem. Initial schedule S_i (where S_i is an ordered set of wells serviced by workover rig i), is obtained by a *Greedy* constructive heuristic. For the shaking step $k_{\max} = 9$ neighborhoods are constructed: (1) *Swap routes* (SS): the wells and the associated routes assigned to two workover rigs are interchanged; (2) *Swap wells from the same workover rig* (SWSW): the order in which two wells are serviced by the same rig is swapped; (3) *Swap wells from different workover rig* (SWDW): two wells assigned to two different workover rigs are swapped;

Table 12.2 Average results with eight workover rigs over 20 runs of each synthetic test problem and three possible scenarios (from Aloise et al. 2003)

Problem	GA	GRASP	AS	MMAS	VNS
P-111	16,791.87	16,602.51	15,813.53	15,815.26	**15,449.50**
P-211	20,016.14	19,726.06	19,048.13	19,051.61	**18,580.64**
P-311	20,251.93	20,094.37	19,528.93	19,546.10	**19,434.97**

(4) *Add/Drop* (AD): a well assigned to a workover rig is reassigned to any position of the schedule of another workover rig; (5) $(SWSW)^2$: apply twice the SWSW move; (6) $(SWDW)^2$: apply twice the SWDW move; (7) $(SWDW)^3$: apply three times the SWDW move; (8) $(AD)^2$: successively apply two (AD) moves; (9) $(AD)^3$: successively apply three (AD) moves.

For local search, the neighborhood consists of all possible exchanges of pairs of wells, i.e. the union of (SWSW) and (SWDW) from above are used.

A basic VNS is compared with a genetic algorithm, the greedy randomized adaptive procedure (GRASP) and two ant colony methods (AS and MMAS) on synthetical and real-life problems from Brazilian onshore fields. Some results on synthetical data are given in Table 12.2. On 27 possible scenarios in generating data sets (denotedby P-111, P-112, P-113, P-121, ..., P333), VNS was better than others in 85 % of the cases and MMAS in 15 %. For real-life problems, results were much better than the gains expected. For example, a daily reduction of 109 m^3 (equivalent to 685.6 bbl) in the production losses over 15 days was obtained by VNS compared with Petrobras' previous solution. That leads to a total savings estimated at 6,600,000 US dollars a year.

12.6 Skewed VNS

VNS gives usually better (or as good) solutions than multistart, and much better ones when there are many local optima. This is due to Fact 3 of the Introduction: many problems have clustered local optima; often, their objective function is a globally convex one plus some noise. However, it may happen that some instances have several separated and possibly far apart valleys containing near-optimal solutions. If one considers larger and larger neighborhoods, the information related to the currently best local optimum dissolves and VNS degenerates into multistart. Moreover if the current best local optimum is not in the deepest valley this information is in part irrelevant. It is therefore of interest to modify VNS schemes in order to explore more fully valleys which are far away from the incumbent solution. This will be done by accepting to re-center the search when a solution close to the best one known, but not necessarily as good, is found, provided that it is far from this last solution. The modified VNS scheme for this variant, called Skewed VNS (SVNS) is

Initialization. Select the set of neighborhood structures $\mathcal{N}_k$, for $k = 1, \ldots, k_{\max}$, that will be used in the search; find an initial solution x and its value $f(x)$; set $x_{\text{opt}} \leftarrow x$, $f_{\text{opt}} \leftarrow f(x)$: choose a stopping condition and a parameter value α;
Repeat the following until the stopping condition is met:
(1) Set $k \leftarrow 1$;
(2) *Repeat* the following steps until $k = k_{\max}$:
(a) *Shaking*. Generate a point x' at random from the kth neighborhood of x;
(b) *Local search*. Apply some local search method with x' as initial solution; denote with x'' the so-obtained local optimum;
(c) *Improvement or not*. If $f(x'') < f_{\text{opt}}$ set $f_{\text{opt}} \leftarrow f(x)$ and $x_{\text{opt}} \leftarrow x''$;
(d) *Move or not*. If $f(x'') - \alpha\rho(x, x'') < f(x)$ set $x \leftarrow x''$ and $k \leftarrow 1$; otherwise set $k \leftarrow k+1$.

Fig. 12.7 Steps of the Skewed VNS

presented in Fig. 12.7. The relaxed rule for re-centering uses an evaluation function linear in the distance from the incumbent, i.e. $f(x'')$ is replaced by

$$f(x'') - \alpha\rho(x, x'')$$

where $\rho(x, x'')$ is the distance from x to x'' and α a parameter. A metric for distance between solutions is usually easy to find, e.g. the Hamming distance when solutions are described by Boolean vectors or the Euclidean distance in the continuous case. Clearly, more complicated formulas could be used for re-centering; possibly, one might take into account known values at points already visited in the valley being explored.

Questions to be answered when applying SVNS are the following:

(i) Does the problem under consideration have a roughly convex objective function, or are there several far apart deep valleys?
(ii) How should α be chosen?

These questions can be answered, to some extent, by first using a multistart version of VNS, i.e. starting VNS from various random points and running it for a short time. Then one can look at the position of the best local optima found and see if they are clustered or dispersed. Further, one can plot values in function of distance from the corresponding local optima to the best known solution and choose α as a fraction of the average slope.

Example 12.5 (Weighted maximum satisfiability, WMAX-SAT). The satisfiability problem, in clausal form, consists in determining if a given set of m clauses (all in disjunctive or all in conjunctive form) built upon n logical variables has a solution or not. The maximum satisfiability problem consists in finding a solution satisfying the largest possible number of clauses. In the weighted maximum satisfiability problem (WMAXSAT) positive weights are assigned to the clauses and a solution maximizing the sum of weights of satisfied clauses is sought. Results of comparative experiments with VNS and tabu search (TS) heuristics on instances having 500

Table 12.3 Results for *GERAD* test problems for WMAXSAT ($n = 500$)

	VNS	VNS-low	SVNS-low	TS
Number of instances where best solution is found	6	4	23	5
% average error in 10 trials	0.2390	0.2702	0.0404	0.0630
% best error in 10 trials	0.0969	0.1077	0.0001	0.0457
Total number of instances	25	25	25	25

Initialization. Select the set of neighborhood structures $\mathcal{N}_k$, for $k = 1, \ldots, k_{\max}$, 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) *Repeat* the following steps until $k = k_{\max}$:

(a) Shaking. Generate a point x' at random from the kth 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 a 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 $k \leftarrow k + 1$;

Fig. 12.8 Steps of the basic VNDS

variables, 4,500 clauses and three variables per clause, in direct or complemented form, are given in Table 12.3 from Hansen et al. (2001). It appears that using a restricted neighborhood consisting of a few directions of steepest descent or mildest ascent in the Shaking step does not improve results, but using this idea in conjunction with SVNS improves notably upon results of basic VNS and also upon those of a TS heuristic.

12.7 Variable Neighborhood Decomposition Search

The VNDS method (Hansen et al. 2001) extends the basic VNS into a two-level VNS scheme based upon decomposition of the problem. Its steps are presented in Fig. 12.8.

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 $x' \in V_k$. When the local search used in this step is also VNS, the two-level VNS scheme arises.

VNDS can be viewed as embedding the classical successive approximation scheme in the VNS framework.

12.8 Analyzing Performance

When a first VNS heuristic has been obtained and tested, the effort should not stop there. Indeed, it is often at this point that the most creative part of the development process takes place. It exploits systematically Fact 2 of the Introduction, that global minima are local minima for all possible neighborhoods simultaneously. The contrapositive is that if a solution $x \in X$ is a local minimum (for the current set of neighborhoods) and not a global one there are one or several neighborhoods (or moves) to be found, which will bring it to this global optimum.

The study then focuses on instances for which an optimal solution is known (or, if none or very few are available, on instances with a presumably optimal solutions, i.e. the best one found by several heuristics) and compares it with the heuristic solution obtained. Visualization is helpful and may take the form of a *distance-to-target diagram* (Hansen and Mladenović 2003). Then, the heuristic solutions, the optimal one and their symmetric difference (e.g. for the TSP) are represented on screen. Moreover, an interactive feature makes it possible to follow how the heuristic works step by step. The information thus gathered is much more detailed than one would get just from objective values and computer times if, as is often the case, the heuristic is viewed as a black box. For instance, this clearly shows that 2-opt is not sufficient to get a good solution for the TSP, that moves involving three or four edges are needed and that those edges leading to an improvement may be far apart along the tour. For another application of VNS to the TSP see Burke et al. (1999).

Similarly, for location problems, one can focus on those facilities which are not at their optimal location and study why, in terms of distributions of nearby users.

Another point is to study how to get out of a large valley if there exists another promising one. *Valley* (or *mountain*) *profiles* are then useful (Hansen et al. 2001). They are obtained by drawing many points x' at random within nested neighborhoods $\mathcal{N}_1(x), \mathcal{N}_2(x), \ldots$ (or, which is equivalent, at increasing distance of a local minimum x) then performing one VND descent and plotting probabilities to get back to x, to get to another local minimum x'' with a value $f(x'') \geq f(x)$ or to get to an improved local minimum x' with $f(x'') < f(x)$. Alternately one may also study the probabilities to go in the direction of x, i.e. $\rho(x,x'') \leq \rho(x,x')$ or towards another valley i.e. $\rho(x,x'') > \rho(x,x')$.

12.9 Promising Areas of Research

Research on Variable Neighborhood Search and its applications is currently very active. We review some of the promising areas in this section; these include a few which are barely explored yet.

A first set of areas concerns enhancements of the VNS basic scheme and ways to make various steps more efficient.

(a) *Initialization.* Both VND and VNS, as many other heuristics, require an initial solution. Two questions then arise: *How best to choose it?* and *Does it matter?* For instance, many initialization rules have been proposed for the k-means heuristic for minimum sum-of-squares clustering, described above, 25 such rules are compared in Hansen et al. (2003a). It appears that while sensitivity of k-means to the initial solution is considerable (best results being obtained with Ward's hierarchical clustering method), VNS results depend very little on the chosen rule. The simplest one is thus best. It would be interesting to extend and generalize this result by conducting similar experiments for other problems.

(b) *Inventory of neighborhoods.* As mentioned above, a VNS study begins by gathering material on neighborhoods used in previous heuristics for the problem under study. A systematic study of moves (or neighborhoods) used for heuristics for whole classes of problems (e.g. location, network design, routing, ...) together with the data structures most adequate for their implementation should be of basic interest for VNS as well as for other metaheuristics. Several researchers (e.g. Ahuja et al. 2000) are working in that direction.

(c) *Distribution of neighborhoods.* When applying a General VNS scheme, neighborhoods can be used in the local search phase, in the shaking phase or in both. A systematic study of their best distribution between phases can enhance performance and provide further insight in the solution process. In particular, the trade-off between increased work in the descent, which provides better local optima, and in shaking which leads to better valleys should be focused upon.

(d) *Ancillary tests.* VNS schemes use randomization in their attempts to find better solutions. This also avoids possible cycling. However, many moves may not lead to any improvement. This suggests adding an ancillary test (Hansen 1974, 1975) the role of which is to decide if a move should be used or not, in its general or in a restricted form. Considering again MMSC, one could try to select better the centroid to be removed from the current solution (a possible criterion being that its cluster contains a few entities only or is close to another centroid) as well as the position where it will be assigned (e.g. the location of an entity far from any other centroid and in a fairly dense region).

A second set of areas concerns changes to the Basic VNS scheme.

(e) *Use of memory.* VNS in its present form relies only on the best solutions currently known to center the search. Knowledge of previous good solutions is forgotten, but might be useful to indicate promising regions not much explored yet. Also, characteristics common to many or most good solutions, such as variables taking the same value in all or most such solutions could be used to better focus the shaking phase. Use of memory has been much studied in tabu search and other metaheuristics. The challenge for VNS is to introduce memory while keeping simplicity.

An interesting way to use memory to enhance performance is *Reactive VNS*, explored by Braisy (2001) for the vehicle routing problem with time windows. If some constraints are hard to satisfy, their violation may be penalized more frequently than for others in the solution process.

(f) *Parallel VNS.* Clearly, there are many natural ways to parallelize VNS schemes. A first one, within VND, is to perform local search in parallel. A second one, within VNS, is to assign the exploration of each neighborhood of the incumbent to a different processor. A third one, within VNDS, is to assign a different subproblem to each processor. Lopez et al. (2002) explore several options in designing a parallel VNS.

(g) *Hybrids.* Several researchers (e.g. Rodriguez et al. 1999; Festa et al. 2001; Ribeiro et al. 2001) have combined VNS with other metaheuristics for various problems. Again, this is not always easy to do without losing VNS's simplicity but may lead to excellent results, particulary if the other metaheuristics are very different from VNS.

At a more general level, one might wish to explore combinations of VNS with constraint programming, instead of its development within mathematical programming as in the applications described above. This could be done in two ways: on the one hand, techniques from constraint programming could be applied to enhance VND; on the other hand, VNS could be applied to constraint programming by minimizing a sum of artificial variables measuring infeasibility and possibly weighted by some estimate of the difficulty of satisfying the corresponding constraints.

A third set of areas concerns new aims for VNS, i.e. non-standard uses:

(h) *Solutions with bounds on the error.* VNS, as other metaheuristics, most often provides near-optimal solutions to combinatorial problems, without bounds on their error. So while such solutions may be optimal or very close to optimality, this fact cannot be recognized. One approach to obtain such bounds is to find with VNS a heuristic solution of the primal problem, deduce from it a solution to the dual (or its continuous relaxation) and then improve this dual solution by another application of VNS. Moreover, complementary slackness conditions can be used to simplify the dual. For problems with a small duality gap this may lead to a near-optimal solution, guaranteed to be very close to optimality. To illustrate, recent work of Hansen et al. (2003a) on the simple plant location problem (SPLP) gave solutions to instances with up to 15,000 users and 15,000 possible facilities with an error bounded by 0.05 %.

(i) *Using VNS within exact algorithms for mixed-integer programming.* Sophisticated algorithms for mixed-integer programming often contain various phases where heuristics are applied. This is illustrated, for example, by Desaulniers et al. (2001) for the airline crew scheduling problem.

Extending the results described in the previous section, in the branch-and-bound framework led to solve exactly SPLP instances with up to 7,000 users (Hansen et al. 2007).

A different approach, called *local branching*, has been recently proposed by Fischetti and Lodi (2003) and Fischetti et al. (2003), both for exact and approximate resolution of large mixed-integer programs. At various branches in the branch-and-bound tree, cuts (which are not valid in general) are added; they express that among a given set of 0–1 variables, already at an integer value, only a few may change their value. They thus correspond to neighborhoods defined by the Hamming distance.

Then CPLEX is used to find the optimal solution within the neighborhood and in this way feasible solutions are more easily obtained. Improved solutions were obtained for a series of large mixed-integer programming instances from various sources, when local branching idea is combined with VNS (Hansen et al. 2006; Lazic et al. 2010).

(j) *Artificial intelligence: enhancing graph theory with VNS.* VNS, as other metaheuristics, has been extensively used to solve a variety of optimization problems in graph theory. However, it may also be used to enhance graph theory per se, following an artificial intelligence approach. This is done by the AutoGraphiX (AGX) system developed by Caporossi and Hansen (2000, 2003). This system considers a graph invariant (i.e. a quantity defined for all graphs of the class under study and independent of vertex and edge labelings) or a formula involving several invariants (which is itself a graph invariant). Then AGX finds extremal or near-extremal graphs for that invariant parametrizing on a few variables, often the order n (or number of vertices) and the size m (of number of edges) of the graph. Analyzing automatically or interactively these graphs and the corresponding curves of invariant values leads to the finding of new conjectures, refuting, corroborating or strengthening existing ones, and giving hints about a possible proof from the minimal list of moves needed to find the extremal graphs. To illustrate, the *energy* E of a graph is the sum of absolute values of the eigenvalues of its adjacency matrix. The following relations were obtained by Caporossi et al. (1999) with AGX: $E \geq 2\sqrt{m}$ and $E \geq \frac{4m}{n}$ and were easily proved. Over 70 new relations have now been obtained, in mathematics and in chemistry. Three ways to attain full automation based on the mathematics of principal component analysis, linear programming and recognition of extremal graphs together with formula manipulations are currently being studied.

12.10 Tricks of the Trade

12.10.1 Getting Started

This purpose of this section is to help students in making a first very simple version of VNS, which would not necessary be competitive with later more sophisticated versions. Most of the steps are common to the implementation of other metaheuristics.

12.10.1.1 A Step-by-Step Procedure

1. *Become familiar with the problem.* Think about the problem at hand; in order to understand it better, make a simple numerical example and spend some time in trying to solve it by hand in your own way. Try to understand why the problem is hard and why a heuristic is needed.

2. *Read literature.* Read about the problem and solution methods in the literature.
3. *Test instances (read data).* Use your numerical example as a first instance for testing your future code, but if it is not large enough, take some from the web, or make a routine for generating random instances. In the second case, read how to generate events using uniformly distributed numbers from (0,1) interval (since each programming language has statement for getting such random numbers).
4. *Data structure.* Think about how the solution of the problem will be represented in the memory. Consider two or more presentations of the same solution if they can reduce the complexity of some routines, i.e. analyze the advantages and disadvantages of each possible presentation.
5. *Initial solution.* Having a routine for reading or generating input data of the problem, the next step is to get an initial solution. For the simple version, any random feasible solution may be used, but the usual way is to develop some *greedy* constructive heuristic, which should not be very hard to do.
6. *Objective value.* Make a procedure that calculates the objective function value for a given solution. Notice that at this stage we already have all the ingredients for the Monte Carlo method: generation of random solution and calculation of objective function value. Get a solution to your problem by Monte Carlo heuristic (i.e. repeat steps 5 and 6 many times and keep the best result).
7. *Shaking.* Make a procedure for shaking. This is a key step of VNS. However, it is easy to implement and usually has only a few lines of computer code. For example, in solving the multi-source Weber problem (see Example 12.2), the easiest perturbation of the current solution is to reallocate randomly chosen entity ℓ from its cluster to another one, also chosen at random. In fact, in this case, the shaking step (or jump in the kth neighborhood) would have only three lines of the computer code:

   For $i = 1$ to k
   $a(1 + n \cdot Rnd1) = 1 + m \cdot Rnd2$
   EndFor

   The solution is saved in the array $a(\ell) \in \{1, \ldots, m\}$ that denotes membership or allocation of entity ℓ ($\ell = 1, \ldots, n$); $Rnd1$ and $Rnd2$ denote random numbers uniformly distributed from the (0,1) interval. Compare the results of the obtained Reduced VNS (take $k_{\max} = 2$) with the Monte Carlo method.
8. *Local search.* Choose an off-the-shelf local search heuristic (or develop a new one). In building a new local search, consider several usual moves that define the neighborhood of the solution: *drop, add, swap, interchange*, etc. Also, for the efficiency (speed) of the method, it is very important to pay special attention to *updating* of the incumbent solution. In other words, it is not usually necessary to use a procedure for calculating objective function values for each point in the neighborhood, i.e. it is possible to get those values by very simple calculation.
9. *Comparison.* Include a local search routine in RVNS to get the basic VNS, and compare it with other methods from the literature.

12.10.2 More Tips

Sometimes basic VNS does not provide very good results.

1. *First versus best improvement.* Compare experimentally *first* and *best improvement* strategies within local search. Previous experience suggests the following: if your initial solution is chosen at random, use first improvement, but if some constructive heuristic is used, use best improvement rule.
2. *Reduce the neighborhood.* The reason for the bad behavior of any local search may be unnecessary visits to all solutions in the neighborhood. Try to identify a *promising* subset of the neighborhood and visit only those; ideally, find a rule that automatically selects solutions from the neighborhood whose objective values are not better than the current one.
3. *Intensified shaking.* In developing more effective VNS, one must spend some time in checking how sensitive the objective function is to small change (shake) of the solution. The trade-off between intensification and diversification of the search in VNS is balanced in the shaking procedure. For some problem instances completely random jump in the kth neighborhood is too diversified. In such cases, some *intensify shaking* procedure may increase intensification of the search. For example, k-interchange neighborhood may be reduced by repeating k times *random add* followed by *best drop* moves. A special case of intensified shaking is so-called large neighborhood search, where k randomly chosen attributes of the solutions are destroyed (dropped), and then the solution is re-built in the best way (by some constructive heuristic).
4. *VND.* Analyze several possible neighborhood structures, estimate their size, make order of them, i.e. develop VND and replace the local search routine with VND to get general VNS.
5. *Experiments with parameter settings.* The single parameter of VNS is $k_{\max}$, which should be estimated experimentally. However, the procedure is usually not very sensitive on $k_{\max}$. In order to make a parameter-free VNS, one can fix its value at the value of some input parameter: e.g. for the p-median (Example 12.3), $k_{\max} = p$; for the MSSC (Example 12.2), $k_{\max} = m$.

12.11 Conclusions

The general schemes of VNS have been presented, discussed and illustrated by examples. References to many further successful applications are given in the next section. In order to evaluate a VNS research program, one needs a list of desirable properties of metaheuristics. The following eight are identified by Hansen and Mladenović (2003):

1. *Simplicity.* The metaheuristic should be based on a simple and clear principle, which should be largely applicable;

2. *Precision.* Steps of the metaheuristic should be formulated in precise mathematical terms, independent from the possible physical or biological analogy which was an initial source of inspiration;
3. *Coherence.* All steps of heuristics for particular problems should follow naturally from the metaheuristic's principle;
4. *Efficiency.* Heuristics for particular problems should provide optimal or near-optimal solutions for all or at least most realistic instances. Preferably, they should find optimal solutions for most problems of benchmarks for which such solutions are known, when available;
5. *Effectiveness.* Heuristics for particular problems should take moderate computing time to provide optimal or near-optimal solutions;
6. *Robustness.* The performance of heuristics should be consistent over a variety of instances, i.e. not just fine-tuned to some training set and less good elsewhere;
7. *User-friendliness.* Heuristics should be clearly expressed, easy to understand and, most important, easy to use. This implies they should have as few parameters as possible and ideally none;
8. *Innovation.* Preferably, the metaheuristic's principle and/or the efficiency and effectiveness of the heuristics derived from it should lead to new types of applications.

VNS possesses, to a large extent, all of these properties. This has led to heuristics being among the very best ones for several problems, but more importantly to insight into the solution process and some innovative applications.

Sources of Additional Information

Some Web addresses with sources of information about VNS include

- http://scholar.google.co.uk/—"Variable neighborhood search" gets around 36,000 files.
- http://apps.isiknowledge.com/—"Variable neighborhood search" in the "topics" window finds 478 VNS papers, together with 4,081 citations and VNS h-index equal to 28 (on 30 November 2010).
- http://www.gerad.ca/en/publications/cahiers_rech.php—if one chooses the option "search for papers" and in the "Abstract" box types "Variable neighborhood search", 56 papers for downloading will appear at the screen; typing "Variable neighbourhood search" gets an additional eight papers.

Survey papers: Hansen and Mladenović (1999, 2001a,c, 2002a,b, 2003), Hansen et al. (2003a, 2008, 2010a,b) and Kochetov et al. (2003).

References

Ahuja RK, Orlin JB, Sharma D (2000) Very large-scale neighborhood search. Int Trans Oper Res 7:301–317

Aloise DJ, Aloise D, Rocha CTM, Ribeiro Filho JC, Moura LSS, Ribeiro CC (2006) Scheduling workover rigs for onshore oil production. Discr Appl Math 154:695–702

Baum EB (1986) Toward practical 'neural' computation for combinatorial optimization problems. In: Denker J (ed) Neural networks for computing. American Institute of Physics, New York

Braysy O (2001) Local search and variable neighborhood search algorithms for vehicle routing with time windows. Acta Wasaensia 87

Burke EK, Cowling P, Keuthen R (1999) Effective local and guided variable neighborhood search methods for the asymmetric traveling salesman problem. In: Proceedings of the Evo workshops. LNCS 2037. Springer, Berlin, pp. 203–212

Caporossi G, Hansen P (2000) Variable neighborhood search for extremal graphs 1. The AutoGraphiX system. Discr Math 212:29–44

Caporossi G, Hansen P (2003) Variable neighborhood search for extremal graphs 5. Three ways to automate conjecture finding. Discr Math 276:81–94

Caporossi G, Cvetković D, Gutman I, Hansen P (1999) Variable neighborhood search for extremal graphs 2. Finding graphs with extremal energy. J Chem Inf Comput Sci 39:984–996

Cornuejols G, Fisher M, Nemhauser G (1990) The uncapacitated facility location problem. In: Mirchandani P, Francis R (eds) Discrete location theory. Wiley, New York

Desaulniers G, Desrosiers J, Solomon MM (2001) Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems. In: Essays and surveys in metaheuristics. Kluwer, Dordrecht, pp. 309–324

Festa P, Pardalos P, Resende M, Ribeiro C (2001) GRASP and VNS for Max-cut. In: Proceedings of the MIC 2001, pp. 371–376

Fischetti M, Lodi A (2003) Local branching. Math Program B 98:23–47

Fischetti M, Polo C, Scantamburlo M (2003) A local branching heuristic for mixed-integer programs with 2-level variables. Research report, University of Padova

Garey MR, Johnson DS (1978) Computers and intractability: a guide to the theory of NP-completeness. Freeman, New York

Glover F, Laguna M (1997) Tabu search. Kluwer, Boston

Glover F, Kochenberger G (eds) (2003) Handbook of metaheuristics. Kluwer, Dordrecht

Hansen P (1974) Programmes mathématiques en variables 0–1. Thèse d'Agrégation de l'Enseignment Supérieur, Université Libre de Bruxelles

Hansen P (1975) Les procédures d'optimization et d'exploration par séparation et évaluation. In: Roy B (ed) Combinatorial programming. Reidel, Dordrecht, pp 19–65

Hansen P, Mladenović N (1997) Variable neighborhood search for the p-median. Locat Sci 5:207–226

Hansen, P, Mladenović N (1999) An introduction to variable neighborhood search. In: Voss S et al (eds) Metaheuristics, advances and trends in local search paradigms for optimization. Kluwer, Dordrecht, pp 433–458

Hansen P, Mladenović N (2001a) Variable neighborhood search: principles and applications. Eur J Oper Res 130:449–467

Hansen P, Mladenović N (2001b) J-Means: A new local search heuristic for minimum sum-of-squares clustering. Pattern Recognit 34:405–413

Hansen P, Mladenović N (2001c) Developments of variable neighborhood search. In: Ribeiro C, Hansen P (eds) Essays and surveys in metaheuristics. Kluwer, Dordrecht, pp. 415–440

Hansen P, Mladenović N (2002a) Variable neighborhood search. In: Pardalos P, Resende M (eds) Handbook of applied optimization. Oxford University Press, New York, pp. 221–234

Hansen P, Mladenović N (2002b) Recherche à voisinage variable. In: Teghem J, Pirlot M (eds) Optimisation approchée en recherche opérationnelle. Lavoisier Hermès, Paris, pp. 81–100

Hansen P, Mladenović N (2003) Variable neighborhood search. In: Glover F, Kochenberger G (eds) Handbook of metaheuristics. Kluwer, Dordrecht, pp 145–184

Hansen P, Jaumard B, Mladenović N, Parreira A (2000) Variable neighborhood search for weighted maximum satisfiability problem. Les Cahiers du GERAD G-2000-62

Hansen P, Mladenović N, Perez-Brito D (2001) Variable neighborhood decomposition search. J Heuristics 7:335–350

Hansen P, Mladenović N, Moreno-Pérez JA (2003a) Búsqueda de Entorno Variable (in Spanish). Inteligencia Artificial 19:77–92

Hansen P, Ngai E, Cheung B, Mladenović N (2003b) Survey and comparison of initialization methods for k-means clustering (in preparation)

Hansen P, Mladenovic N, Urosevic D (2006) Variable neighborhood search and local branching. Comput Oper Res 33:3034–3045

Hansen P, Brimberg J, Urosevic D, Mladenovic N (2007) Primal-dual variable neighborhood for the simple plant location problem. INFORMS J Comput 19:552–564

Hansen P, Mladenović N, Moreno-Pérez JA (2008) Variable neighborhood search. Eur J Oper Res 191:593–595

Hansen P, Mladenovic N, Brimberg J, Moreno Pérez JA (2010a) Variable neighbourhood search. In: Gendreau M, Potvin J-Y (eds) Handbook of metaheuristics, 2nd edn. Kluwer, Dordrecht, pp 61–86

Hansen P, Mladenović N, Moreno Pérez JA (2010b) Variable neighborhood search: algorithms and applications. Ann Oper Res 175:367–407

Kirkpatrick S, Gellatt CD Jr, Vecchi P (1983) Optimization by simulated annealing. Science 220:671–680

Kochetov Y, Mladenović N, Hansen P (2003) Lokalnii poisk s chereduyshimisy okrestnostyami (in Russian). Diskretaja Matematika 10:11–43

Labbé M, Peeters D, Thisse JF (1995) Location on networks. In: Ball M et al (eds) Network routing. North-Holland, Amsterdam, pp 551–624

Lazic J, Hanafi S, Mladenovic N, Urosevic D (2010) Variable neighborhood decomposition search for 0–1 mixed integer programs. Comput Oper Res 37:1055–1067

Lopez FG, Batista BM, Moreno Pérez JA, Moreno Vega JM (2002) The parallel variable neighborhood search for the *p*-median problem. J Heuristics 8:375–388

Mladenović N, Hansen P (1997) Variable neighborhood search. Comput Oper Res 24:1097–1100

Papadimitriou C (1994) Computational complexity. Addison-Wesley, Reading

Reeves CR (ed) (1993) Modern heuristic techniques for combinatorial problems. Blackwell, Oxford

Resende MGC, Werneck R (2003) On the implementation of a swap-based local search procedure for the *p*-median problem. In: Ladner RE (ed) Proceedings of the ALENEX 2003. SIAM, Philadelphia, pp 119–127

Ribeiro C, Uchoa E, Werneck R (2001) A hybrid GRASP with perturbations for the Steiner problem in graphs. Technical report, Computer Science Department, Catholic University of Rio de Janeiro

Rodriguez I, Moreno-Vega M, Moreno-Perez J (1999) Heuristics for routing-median problems. SMG report, Université Libre de Bruxelles

Teitz MB, Bart P (1968) Heuristic methods for estimating the generalized vertex median of a weighted graph. Oper Res 16:955–961

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

Zufferey N, Hertz A, Avanthay C (2003) Variable neighborhood search for graph colouring. Eur J Oper Res 151:379–388