Chapter 3
Variable Neighborhood Search

Pierre Hansen, Nenad Mladenović, Jack Brimberg and José A. Moreno Pérez

Abstract Variable neighborhood search (VNS) is a metaheuristic for solving combinatorial and global optimization problems whose basic idea is a systematic change of neighborhood both within a descent phase to find a local optimum and in a perturbation phase to get out of the corresponding valley. In this chapter we present the basic schemes of VNS and some of its extensions. We then describe a recent development, i.e., formulation space search. We then present five families of applications in which VNS has proven to be very successful: (i) exact solution of large-scale location problems by primal–dual VNS; (ii) generation of feasible solutions to large mixed integer linear programs by hybridization of VNS and local branching; (iii) generation of good feasible solutions to continuous nonlinear programs; (iv) generation of feasible solutions and/or improved local optima for mixed integer nonlinear programs by hybridization of sequential quadratic programming and branch and bound within a VNS framework, and (v) exploration of graph theory to find conjectures, refutations, and proofs or ideas of proofs.

Pierre Hansen
GERAD and Ecole des Hautes Etudes Commerciales, 3000 ch. de la Cote-Sainte-Catherine, Montréal H3T 2A7, QC, Canada
e-mail: pierreh@crt.umontreal.ca

Nenad Mladenović
School of Mathematics, Brunel University-West London, Uxbridge, UB8 3PH, UK
e-mail: nenad.mladenovic@brunel.ac.uk

Jack Brimberg
Department of Mathematics and Computer Science, Royal Military College of Canada, Kingston ON, Canada, K7K 7B4
e-mail: jack.brimberg@rmc.ca

José A. Moreno Pérez
IUDR and DEIOC, Universidad de La Laguna, 38271 La Laguna, Santa Cruz de Tenerife, Spain
e-mail: jamoreno@ull.es

3.1 Introduction

Optimization tools have greatly improved during the last two decades. This is due to several factors: (i) progress in mathematical programming theory and algorithmic design; (ii) rapid improvement in computer performances; (iii) better communication of new ideas and integration in widely used complex softwares. Consequently, many problems long viewed as out of reach are currently solved, sometimes in very moderate computing times. This success, however, has led researchers and practitioners to address much larger instances and more difficult classes of problems. Many of these may again only be solved heuristically. Therefore thousands of papers describing, evaluating, and comparing new heuristics appear each year. Keeping abreast of such a large literature is a challenge. Metaheuristics, or general frameworks for building heuristics, are therefore needed in order to organize the study of heuristics. As evidenced by this handbook, there are many of them. Some desirable properties of metaheuristics [64, 67, 68] are listed in the concluding section of this chapter.

Variable neighborhood search (VNS) is a metaheuristic proposed by some of the present authors a dozen years ago [85]. Earlier work that motivated this approach can be found in [28, 41, 44, 82]. It is based on the idea of a systematic change of neighborhood both in a descent phase to find a local optimum and in a perturbation phase to get out of the corresponding valley. Originally designed for approximate solution of combinatorial optimization problems, it was extended to address mixed integer programs, nonlinear programs, and recently mixed integer nonlinear programs. In addition VNS has been used as a tool for automated or computer-assisted graph theory. This led to the discovery of over 1500 conjectures in that field, the automated proof of more than half of them as well as the unassisted proof of about 400 of them by many mathematicians.

Applications are rapidly increasing in number and pertain to many fields: location theory, cluster analysis, scheduling, vehicle routing, network design, lot-sizing, artificial intelligence, engineering, pooling problems, biology, phylogeny, reliability, geometry, telecommunication design, etc. (see, e.g., [20, 21, 31, 38, 39, 66, 77, 99]). References are too numerous to be all listed here, but many others can be found in [69] and special issues of *IMA Journal of Management Mathematics* [81], *European Journal of Operational Research* [68], and *Journal of Heuristics* [89] are devoted to VNS.

This chapter is organized as follows. In the next section we present the basic schemes of VNS, i.e., variable neighborhood descent (VND), reduced VNS (RVNS), basic VNS (BVNS), and general VNS (GVNS). Two important extensions are presented in Section 3.3: skewed VNS and variable neighborhood decomposition search (VNDS). A further recent development called formulation space search (FSS) is discussed in Section 3.4. The remainder of this chapter describes applications of VNS to several classes of large scale and complex optimization problems for which it has proven to be particularly successful. Section 3.5 is devoted to primal–dual VNS (PD-VNS) and its application to location and clustering problems. Finding feasible solutions to large mixed integer linear programs with VNS is discussed in Section 3.6. Section 3.7 addresses ways to apply VNS in continuous global optimization. The more difficult case of solving mixed integer nonlinear programming by VNS is considered in Section 3.8. Applying VNS to graph theory per se (and not just to particular optimization problems defined on graphs) is discussed in Section 3.9. Brief conclusions are drawn in Section 3.10.

3.2 Basic Schemes

A deterministic optimization problem may be formulated as

$$\min\{f(x)|x \in X, X \subseteq \mathscr{S}\}, \tag{3.1}$$

where $\mathscr{S}$, X, x, and f denote the *solution space*, the *feasible set*, a *feasible solution*, and a real-valued *objective function*, respectively. If $\mathscr{S}$ is a finite but large set, a *combinatorial optimization* problem is defined. If $\mathscr{S} = \mathbb{R}^n$, we refer to *continuous optimization*. A solution $x^* \in X$ is *optimal* if

$$f(x^*) \leq f(x), \ \forall x \in X.$$

An *exact algorithm* for problem (3.1), if one exists, finds an optimal solution x^*, together with the proof of its optimality, or shows that there is no feasible solution, i.e., $X = \emptyset$, or the solution is unbounded. Moreover, in practice, the time needed to do so should be finite (and not too long). For continuous optimization, it is reasonable to allow for some degree of tolerance, i.e., to stop when sufficient convergence is detected.

Let us denote $\mathscr{N}_k$ ($k = 1, \ldots, k_{max}$), a finite set of pre-selected neighborhood structures, and with $\mathscr{N}_k(x)$ the set of solutions in the kth neighborhood of x. Most local search heuristics use only one neighborhood structure, i.e., $k_{\max} = 1$. Often successive neighborhoods $\mathscr{N}_k$ are nested and may be induced from one or more metric (or quasi-metric) functions introduced into a solution space S. An *optimal solution* x_{opt} (or global minimum) is a feasible solution where a minimum is reached. We call $x' \in X$ a *local minimum* of Equation (3.1) with respect to $\mathscr{N}_k$ (w.r.t. $\mathscr{N}_k$ for short), if there is no solution $x \in \mathscr{N}_k(x') \subseteq X$ such that $f(x) < f(x')$. Metaheuristics (based on local search procedures) try to continue the search by other means after finding the first local minimum. VNS is based on three simple facts:

Fact 1 *A local minimum w.r.t. one neighborhood structure is not necessarily so for another;*

Fact 2 *A global minimum is a local minimum w.r.t. all possible neighborhood structures;*

Fact 3 *For many problems, local minima w.r.t. one or several $\mathscr{N}_k$ 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. For instance, it might be several variables with the same value in both solutions. However, it is usually not known which variables are such. Since these variables usually cannot be identified in advance, one should conduct an organized study of the neighborhoods of the local optimum until a better solution is found.

In order to solve Equation (3.1) by using several neighborhoods, facts 1 to 3 can be used in three different ways: (i) deterministic, (ii) stochastic, (iii) both deterministic and stochastic.

We first give in Algorithm 1 the solution move and neighborhood change function that will be used later.

Algorithm 1 Neighborhood change

```
Function NeighborhoodChange ($x, x', k$)
1  if $f(x') < f(x)$ then
2      $x \leftarrow x'$ // Make a move
3      $k \leftarrow 1$ // Initial neighborhood
   else
4      $k \leftarrow k+1$ // Next neighborhood
return $x, k$
```

Function `NeighborhoodChange()` compares the incumbent value $f(x)$ with the new value $f(x')$ obtained from the kth neighborhood (line 1). If an improvement is obtained, the new incumbent is updated (line 2) and k is returned to its initial value (line 3). Otherwise, the next neighborhood is considered (line 4).

(i) The **variable neighborhood descent** (VND) method is obtained if a change of neighborhoods is performed in a deterministic way. It is presented in Algorithm 2, where neighborhoods are denoted as $N_k, k = 1, \ldots, k_{\max}$.

Algorithm 2 Variable neighborhood descent

```
Function VND ($x, k_{max}$)
1  $k \leftarrow 1$
2  repeat
3      $x' \leftarrow arg\min_{y \in N_k(x)} f(y)$ // Find the best neighbor in $N_k(x)$
4      $x, k \leftarrow$ NeighborhoodChange ($x, x', k$) // Change neighborhood
   until $k = k_{max}$
return $x$
```

Most local search heuristics use a single or sometimes two neighborhoods for improving the current solution (i.e., $k_{\max} \leq 2$). Note that the final solution should be a local minimum w.r.t. all $k_{\max}$ neighborhoods, and thus a global optimum is more likely to be reached than with a single structure. Beside this *sequential* order of neighborhood structures in VND, one can develop a *nested* strategy. Assume, for example, that $k_{\max} = 3$; then a possible nested strategy is to perform VND with Algorithm 2 for the first two neighborhoods from each point x' that belongs to the third one ($x' \in N_3(x)$). Such an approach is successfully applied in [22, 27, 65].

(ii) The **reduced VNS** (RVNS) method is obtained if random points are selected from $\mathcal{N}_k(x)$ and no descent is made. Rather, the values of these new points are compared with that of the incumbent and an update takes place in case of improvement. We also assume that a stopping condition has been chosen like the maximum CPU time allowed $t_{\max}$ or the maximum number of iterations between two improvements. To simplify the description of the algorithms we always use $t_{\max}$ below. Therefore, RVNS uses two parameters: $t_{\max}$ and $k_{\max}$. It is presented in Algorithm 3.

Algorithm 3 Reduced VNS

```
Function RVNS($x, k_{max}, t_{max}$)
1  repeat
2      $k \leftarrow 1$
3      repeat
4          $x' \leftarrow$ Shake($x, k$)
5          $x, k \leftarrow$ NeighborhoodChange ($x, x', k$)
       until $k = k_{max}$
6      $t \leftarrow$ CpuTime()
   until $t > t_{max}$
return $x$
```

The function Shake in line 4 generates a point x' at random from the kth neighborhood of x, i.e., $x' \in \mathcal{N}_k(x)$. It is given in Algorithm 4, where it is assumed that points from $\mathcal{N}_k(x)$ are $\{x^1, \ldots, x^{|\mathcal{N}_k(x)|}\}$. RVNS is useful for very large instances

Algorithm 4 Shaking function

```
Function Shake($x, k$)
1  $w \leftarrow \lfloor 1+$Rand$(0,1) \times |\mathcal{N}_k(x)| \rfloor$
2  $x' \leftarrow x^w$
return $x'$
```

for which local search is costly. It can be used as well for finding initial solutions for large problems before decomposition. It has been observed that the best value for the parameter $k_{\max}$ is often 2 or 3. In addition, a maximum number of iterations between two improvements is usually used as the stopping condition. RVNS is akin to a Monte Carlo method, but is more systematic (see, e.g., [86] where results obtained by RVNS were 30% better than those of the Monte Carlo method in solving a continuous min–max problem). When applied to the p-Median problem, RVNS gave equally good solutions as the *Fast Interchange* heuristic of [98] while being 20 to 40 times faster [70].

(iii) The **basic VNS** (BVNS) method [85] combines deterministic and stochastic changes of neighborhood. The deterministic part is represented by a local search heuristic. It consists in (i) choosing an initial solution x, (ii) finding a direction of descent from x (within a neighborhood $N(x)$), and (iii) moving to the minimum of $f(x)$ within $N(x)$ along that direction. If there is no direction of descent, the heuristic stops, otherwise it is iterated. Usually the steepest descent direction, also referred to as *best improvement*, is used. This is summarized in Algorithm 5, where we assume that an initial solution x is given. The output consists of a local minimum, also denoted by x, and its value. As the Steepest descent heuristic may be time-

Algorithm 5 Best improvement (steepest descent) heuristic

```
Function BestImprovement($x$)
1  repeat
2      $x' \leftarrow x$
3      $x \leftarrow arg\min_{y \in N(x)} f(y)$
   until ($f(x) \geq f(x')$)
return $x$
```

consuming, an alternative is to use the *first descent* heuristic. Vectors $x^i \in N(x)$ are then enumerated systematically and a move is made as soon as a direction for the descent is found. This is summarized in Algorithm 6.

Algorithm 6 First improvement (first descent) heuristic

```
Function FirstImprovement($x$)
1  repeat
2      $x' \leftarrow x$;  $i \leftarrow 0$
3      repeat
4          $i \leftarrow i + 1$
5          $x \leftarrow arg\min\{f(x), f(x^i)\}$,  $x^i \in N(x)$
       until ($f(x) < f(x')$ or $i = |N(x)|$)
   until ($f(x) \geq f(x')$)
return $x$
```

The stochastic phase is represented by the random selection of one point from the kth neighborhood. The BVNS is given in Algorithm 7.

Note that point x' is generated at random in step 5 in order to avoid cycling, which might occur with a deterministic rule.

Example. We illustrate the basic steps on a minimum k-cardinality tree instance taken from [76], see Figure 3.1. The minimum k-cardinality tree problem on graph G (k-card for short) consists in finding a subtree of G with exactly k edges whose sum of weights is minimum.

The steps of BVNS for solving the 4-card problem are illustrated in Figure 3.2. In step 0 the objective function value, i.e., the sum of edge weights, is equal to 40;

Algorithm 7 Basic VNS

```
Function BVNS($x, k_{max}, t_{max}$)
1  $t \leftarrow 0$
2  while $t < t_{max}$ do
3      $k \leftarrow 1$
4      repeat
5          $x' \leftarrow$ Shake($x,k$)                    // Shaking
6          $x'' \leftarrow$ BestImprovement($x'$)          // Local search
7          $x,k \leftarrow$ NeighborhoodChange($x,x'',k$) // Change neighborhood
       until $k = k_{max}$
8      $t \leftarrow$ CpuTime()
   return $x$
```

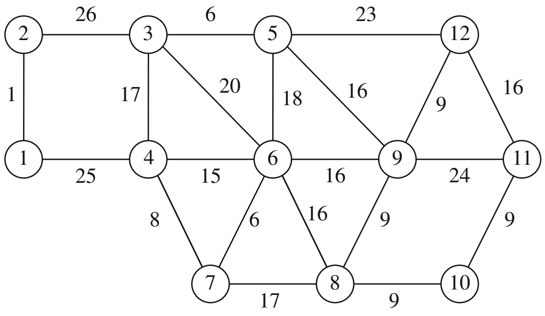

Fig. 3.1 4-cardinality tree problem.

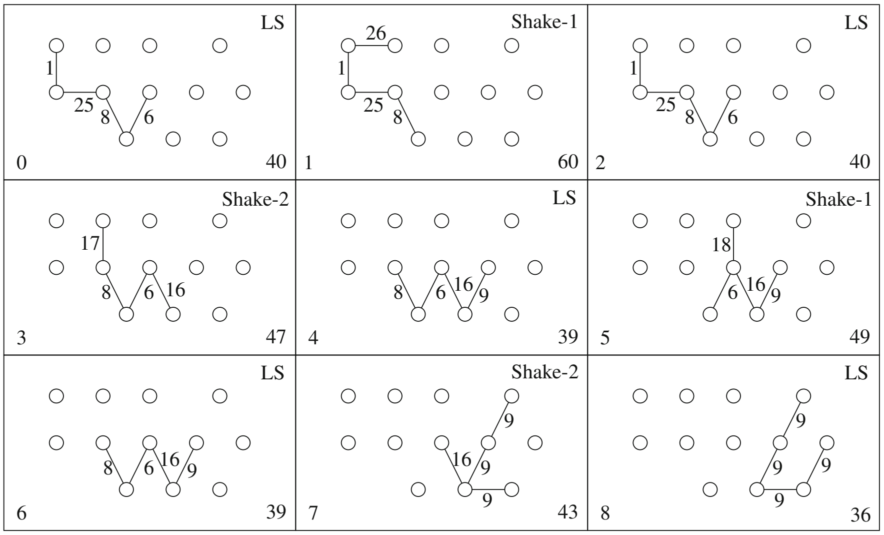

Fig. 3.2 Steps of the basic VNS for solving 4-card tree problem.

it is indicated in the right bottom corner of the figure. That first solution is a local minimum with respect to the edge-exchange neighborhood structure (one edge in, one out). After shaking, the objective function is 60, and after another local search, we are back to the same solution. Then, in step 3, we take out 2 edges and add another 2 at random, and after a local search, an improved solution is obtained with a value of 39. Continuing in that way, the optimal solution with an objective function value equal to 36 is obtained in step 8.

(iv) **General VNS.** Note that the local search step (line 6 in BVNS, Algorithm 7) may also be replaced by VND (Algorithm 2). This general VNS (VNS/VND) approach has led to some of the most successful applications reported in the literature (see, e.g., [2, 27, 30, 32, 34, 36, 37, 65, 71, 92, 93]). The general VNS (GVNS) is given in Algorithm 8 below.

Algorithm 8 General VNS

```
Function GVNS ($x, \ell_{max}, k_{max}, t_{max}$)
1  repeat
2      $k \leftarrow 1$
3      repeat
4          $x' \leftarrow$ Shake($x, k$)
5          $x'' \leftarrow$ VND($x', \ell_{max}$)
6          $x, k \leftarrow$ NeighborhoodChange($x, x'', k$)
       until $k = k_{max}$
7      $t \leftarrow$ CpuTime()
   until $t > t_{max}$
return $x$
```

3.3 Some Extensions

(i) The **skewed VNS** (SVNS) method [59] addresses the problem of exploring valleys far from the incumbent solution. Indeed, once the best solution in a large region has been found, it is necessary to go quite far to obtain an improved one. Solutions drawn at random in far-away neighborhoods may differ substantially from the incumbent, and VNS may then degenerate, to some extent, into the multistart heuristic (in which descents are made iteratively from solutions generated at random, which is known not to be very efficient). So some compensation for distance from the incumbent must be made and a scheme called skewed VNS is proposed for that purpose. Its steps are presented in Algorithms 9, 10, and 11. The `KeepBest`(x,x') function (Algorithm 9) in SVNS simply keeps the best of solutions x and x'. The `NeighborhoodChangeS` function (Algorithm 10) performs the move and neighborhood change for the SVNS.

SVNS makes use of a function $\rho(x,x'')$ to measure the distance between the incumbent solution x and the local optimum x''. The distance function used to define $\mathcal{N}_k$, as in the above examples, could be used also for this purpose. The parameter

Algorithm 9 Keep best solution

```
Function KeepBest($x$,$x'$)
1  if $f(x') < f(x)$ then
2      $x \leftarrow x'$
   return $x$
```

Algorithm 10 Neighborhood change for the skewed VNS

```
Function NeighborhoodChangeS($x$,$x'$,$k$,$\alpha$)
1  if $f(x') - \alpha\rho(x,x') < f(x)$ then
2      $x \leftarrow x'$
3      $k \leftarrow 1$
   else
4      $k \leftarrow k+1$
   return $x$,$k$
```

Algorithm 11 Skewed VNS

```
Function SVNS ($x, k_{max}, t_{max}, \alpha$)
1   $x_{best} \leftarrow x$
2   repeat
3       $k \leftarrow 1$
4       repeat
5           $x' \leftarrow$ Shake($x,k$)
6           $x'' \leftarrow$ FirstImprovement($x'$)
7           $x,k \leftarrow$ NeighborhoodChangeS($x,x'',k,\alpha$)
        until $k = k_{max}$
8       $x_{best} \leftarrow$ KeepBest ($x_{best}, x$)
9       $x \leftarrow x_{best}$
10      $t \leftarrow$ CpuTime()
    until $t > t_{max}$
    return $x$
```

α must be chosen to allow movement to valleys far away from x when $f(x'')$ is larger than $f(x)$ but not too much larger (otherwise one will always leave x). A good value for α is to be found experimentally in each case. Moreover, in order to avoid frequent moves from x to a close solution one may take a smaller value for α when $\rho(x,x'')$ is small. More sophisticated choices of a function of $\alpha\rho(x,x'')$ could be made through some learning process.

(ii) The **variable neighborhood decomposition search** (VNDS) method [70] extends the basic VNS into a two-level VNS scheme based on decomposition of the problem. It is presented in Algorithm 12, where t_d is an additional parameter that represents the running time allowed for solving decomposed (smaller sized) problems by basic VNS (line 5).

For ease of presentation, but without loss of generality, we assume that the solution x represents a set of some attributes. In step 4 we denote by y a set of k solution attributes present in x' but not in x ($y = x' \setminus x$). In step 5 we find the local optimum

Algorithm 12 Variable neighborhood decomposition search

```
Function VNDS ($x, k_{max}, t_{max}, t_d$)
1 repeat
2     $k \leftarrow 1$
3     repeat
4         $x' \leftarrow$ Shake $(x, k)$; $y \leftarrow x' \setminus x$
5         $y' \leftarrow$ BVNS$(y, k, t_d)$; $x'' = (x' \setminus y) \cup y'$
6         $x''' \leftarrow$ FirstImprovement$(x'')$
7         $x, k \leftarrow$ NeighborhoodChange$(x, x''', k)$
      until $k = k_{max}$
  until $t > t_{max}$
return $x$
```

y' in the space of y; then we denote with x'' the corresponding solution in the whole space S ($x'' = (x' \setminus y) \cup y'$). We notice that exploiting some *boundary effects* in a new solution can significantly improve solution quality. That is why, in step 6, the local optimum x''' is found in the whole space S using x'' as an initial solution. If this is time consuming, then at least a few local search iterations should be performed.

VNDS can be viewed as embedding the classical successive approximation scheme (which has been used in combinatorial optimization at least since the 1960s, see, e.g., [52]) in the VNS framework.

3.4 Variable Neighborhood Formulation Space Search

Traditional ways to tackle an optimization problem consider a given formulation and search in some way through its feasible set X. Given that the same problem can often be formulated in different ways, it is possible to extend search paradigms to include jumps from one formulation to another. Each formulation should lend itself to some traditional search method, its “local search” that works totally within this formulation, and yields a final solution when started from some initial solution. Any solution found in one formulation should easily be translatable to its equivalent solution in any other formulation. We may then move from one formulation to another by using the solution resulting from the local search of the former as an initial solution for the local search of the latter. Such a strategy will of course only be useful when local searches in different formulations behave differently.

This idea was recently investigated in [87] using an approach that systematically alternates between different formulations for solving the circle packing problem (CPP). It is shown there that a stationary point for a nonlinear programming formulation of CPP in Cartesian coordinates is not necessarily a stationary point in polar coordinates. A method called *Reformulation Descent* (RD) that alternates between these two formulations until the final solution is stationary with respect to both formulations is suggested. Results obtained were comparable with the best known values, but were achieved some 150 times faster than with an alternative single formulation approach. In the same paper the idea suggested above of *Formulation Space Search* (FSS) is also introduced, using more than two formulations. Some research in that direction has also been reported in [74, 83, 88, 90]. One methodology that uses the variable neighborhood idea when searching through the formulation space is given in Algorithms 13 and 14. Here ϕ (ϕ') denotes a formulation from a given space $\mathscr{F}$, x (x') denotes a solution in the feasible set defined with that formulation, and $\ell \leq \ell_{\max}$ is the formulation neighborhood index. Note that Algorithm 14 uses a reduced VNS strategy in the formulation space $\mathscr{F}$. Note also that the `ShakeFormulation()` function must provide a search through the solution space $\mathscr{S}'$ in order to get a new solution x'. Any appropriate method can be used for this purpose.

Algorithm 13 Formulation change

```
Function FormulationChange($x$,$x'$,$\phi$,$\phi'$,$\ell$)
1 Set $\ell_{min}$ and $\ell_{step}$
2 if $f(\phi',x') < f(\phi,x)$ then
3     $\phi \leftarrow \phi'$
4     $x \leftarrow x'$
5     $\ell \leftarrow \ell_{min}$
  else
6     $\ell \leftarrow \ell + \ell_{step}$
7 return $x, \phi, \ell$
```

Algorithm 14 Reduced variable neighborhood FSS

```
Function VNFSS($x$, $\phi$, $\ell_{max}$)
1 repeat
2     $\ell \leftarrow 1$                      // Initialize formulation in $\mathscr{F}$
3     while $\ell \leq \ell_{max}$ do
4         $x', \phi', \ell \leftarrow$ ShakeFormulation($x$,$x'$,$\phi$,$\phi'$,$\ell$) // $(\phi',x') \in (N_\ell(\phi),\mathscr{N}(x))$ at random
5         $x, \phi, \ell \leftarrow$ FormulationChange($x$,$x'$,$\phi$,$\phi'$,$\ell$) // Change formulation
  until some stopping condition is met
6 return $x$
```

3.5 Primal–Dual VNS

For most modern heuristics the difference in value between the optimal solution and the one obtained is completely unknown. Guaranteed performance of the primal heuristic may be determined if a lower bound on the objective function value is known. To this end, the standard approach is to relax the integrality condition on the primal variables, based on a mathematical programming formulation of the problem. However, when the dimension of the problem is large, even the relaxed problem may be impossible to solve exactly by standard commercial solvers. Therefore, it seems to be a good idea to solve dual relaxed problems heuristically as well. In this way we get guaranteed bounds on the primal heuristic's performance. The next problem arises if we want to get an exact solution within a branch and bound framework since having the approximate value of the relaxed dual does not allow us to branch in an easy way, for example, by exploiting complementary slackness conditions. Thus, the exact value of the dual is necessary.

In primal–dual VNS (PD-VNS) [58] one possible general way to get both the guaranteed bounds and the exact solution is proposed. It is given in Algorithm 15.

Algorithm 15 Basic PD-VNS

```
Function PD-VNS ($x, k_{max}, t_{max}$)
1 BVNS ($x, k_{max}, t_{max}$)    // Solve primal by VNS
2 DualFeasible($x, y$)            // Find (infeasible) dual such that $f_P = f_D$
3 DualVNS($y$)                    // Use VNS do decrease infeasibility
4 DualExact($y$)                  // Find exact (relaxed) dual
5 BandB($x, y$)                   // Apply branch-and-bound method
```

In the first stage a heuristic procedure based on VNS is used to obtain a near-optimal solution. In [58] it is shown that VNS with decomposition is a very powerful technique for large-scale simple plant location problems (SPLP) with up to 15,000 facilities and 15,000 users. In the second phase the objective is to find an exact solution of the relaxed dual problem. Solving the relaxed dual is accomplished in three stages: (i) find an initial dual solution (generally infeasible) using the primal heuristic solution and complementary slackness conditions; (ii) find a feasible solution by applying VNS to the unconstrained nonlinear form of the dual; and (iii) solve the dual exactly starting with the found initial feasible solution using a customized "sliding simplex" algorithm that applies "windows" on the dual variables, thus substantially reducing the problem size. On all problems tested, including instances much larger than those previously reported in the literature, the procedure was able to find the exact dual solution in reasonable computing time. In the third and final phase, armed with tight upper and lower bounds obtained from the heuristic primal solution in phase 1 and the exact dual solution in phase 2, respectively, a standard branch-and-bound algorithm is applied to find an optimal solution of the original problem. The lower bounds are updated with the dual sliding simplex method and the upper bounds whenever new integer solutions are obtained at the nodes of the branching tree. In this way it was possible to solve exactly problem instances of sizes up to $7\,000 \times 7\,000$ for uniform fixed costs and 15,000 $\times$ 15,000 otherwise.

3.6 Variable Neighborhood Branching—VNS for Mixed Integer Linear Programming

The mixed integer linear programming (MILP) problem consists of maximizing or minimizing a linear function, subject to equality or inequality constraints, and integrality restrictions on some of the variables. The mixed integer programming problem $(MILP)$ can be expressed as follows:

$$(MILP) \quad \left[\begin{array}{lll} \min & \sum_{j=1}^{n} c_j x_j & \\ \text{s.t.} & \sum_{j=1}^{n} a_{ij} x_j \geq b_i & \forall i \in M = \{1,2,\ldots,m\} \\ & x_j \in \{0,1\} & \forall j \in \mathscr{B} \neq \emptyset \\ & x_j \geq 0, \texttt{integer} & \forall j \in \mathscr{G} \\ & x_j \geq 0 & \forall j \in \mathscr{C} \end{array}\right. ,$$

where the set of indices $N = \{1,2,\ldots,n\}$ is partitioned into three subsets $\mathscr{B}, \mathscr{G}$, and $\mathscr{C}$, corresponding to binary, general integer, and continuous variables, respectively.

Numerous combinatorial optimization problems, including a wide range of practical problems in business, engineering and science, can be modeled as MILP problems. Several special cases, such as knapsack, set packing, cutting and packing, network design, protein alignment, traveling salesman, and other routing problems, are known to be NP-hard [48].

There are several commercial solvers such as CPLEX [75] for solving MILPs. Methods included in such software packages are usually of branch-and-bound (B&B) or of branch-and-cut (B&C) types. Basically, those methods enumerate all possible integer values in some order and perform some restrictions for the cases where such enumeration cannot improve the current best solution.

The connection between local search-based heuristics and exact solvers may be established by introducing the so-called *local branching constraint* [43]. By adding just one constraint into (MILP), the kth neighborhood of (MILP) is defined. This allows the use of all local search-based metaheuristics, such as tabu search, simulating annealing, VNS. More precisely, given two solutions x and y of the (MILP), the distance between x and y is defined as follows:

$$\delta(x,y) = \sum_{j \in \mathscr{B}} \mid x_j - y_j \mid .$$

Let X be the solution space of the (MILP) considered. The neighborhood structures $\{\mathscr{N}_k \mid k = 1,\ldots,k_{\max}\}$ can be defined, knowing the distance $\delta(x,y)$ between any two solutions $x,y \in X$. The set of all solutions in the kth neighborhood of $y \in X$ is denoted as $\mathscr{N}_k(y)$ where

$$\mathscr{N}_k(y) = \{x \in X \mid \delta(x,y) \ \leq \ k\}.$$

For the pure 0-1 MILP given above ($\mathscr{G} = \emptyset$), $\delta(.,.)$ represents the Hamming distance and $\mathscr{N}_k(y)$ may be expressed by the following *local branching constraint*

$$\delta(x,y) = \sum_{j \in S}(1 - x_j) + \sum_{j \in \mathscr{B} \setminus S} x_j \leq k, \tag{3.2}$$

where $S = \{j \in \mathscr{B} \mid y_j = 1\}$.

In [71] we developed a general VNS procedure for solving 0-1 MILPs. An exact MILP solver (CPLEX) is used as a black box for finding the best solution in the neighborhood, based on the given formulation (MILP) plus the added local

Algorithm 16 VNS branching

```
Function VnsBra(total_time_limit, node_time_limit, k_step, $x\_opt$)
1  TL := total_time_limit; UB := $\infty$; $first$ := true
2  $stat$ := MIPSOLVE(TL, UB, $first$, $x\_opt$, $f\_opt$)
3  $x\_cur$:=$x\_opt$; $f\_cur$:=$f\_opt$
4  while (elapsedtime < total_time_limit) do
5      $cont$ := true; $rhs$ := 1; $first$ := false
6      while ($cont$ or elapsedtime < total_time_limit) do
7          TL = min(node_time_limit, total_time_limit-elapsedtime)
8          add local br. constr. $\delta(x, x\_cur) \leq rhs$; UB := $f\_cur$
9          $stat$ := MIPSOLVE(TL, UB, $first$, $x\_next$, $f\_next$)
10         switch $stat$ do
11             case "opt_sol_found":
12                 reverse last local br. const. into $\delta(x, x\_cur) \geq rhs + 1$
13                 $x\_cur$ := $x\_next$; $f\_cur$ := $f\_next$; $rhs$ := 1;
14             case "feasible_sol_found":
15                 reverse last local br. constr. into $\delta(x, x\_cur) \geq 1$
16                 $x\_cur$ := $x\_next$; $f\_cur$ := $f\_next$; $rhs$ := 1;
17             case "proven_infeasible":
18                 remove last local br. constr.; $rhs$ := $rhs$+1;
19             case "no_feasible_sol_found":
20                 $cont$ := false
21     if $f\_cur < f\_opt$ then
22         $x\_opt$ := $x\_cur$; $f\_opt$ := $f\_cur$; $k\_cur$ := k_step;
       else
23         $k\_cur$ := $k\_cur$+k_step;
24     remove all added constraints; $cont$ := true
25     while $cont$ and ($elapsedtime$ < total_time_limit) do
26         add constraints $k\_cur \leq \delta(x, x\_opt) < k\_cur + \mathtt{k\_step}$
27         TL := total_time_limit-elapsedtime; UB := $\infty$; $first$ := true
28         $stat$ := MIPSOLVE(TL, UB, $first$, $x\_cur$, $f\_cur$)
29         remove last two added constraints; $cont$ =false
30         if $stat$ = "proven_infeasible" or "no_feasible" then
31             $cont$ :=true;  $k\_cur$ := $k\_cur$+k_step
```

branching constraints. Shaking is performed using the Hamming distance defined above. A detailed description of this VNS branching method is provided in Algorithm 16. The variables and constants used in the algorithm are defined as follows [71]:

. UB—input variable for CPLEX solver which represents the current upper bound.
. *first*—logical input variable for CPLEX solver which is `true` if the first solution lower than UB is asked for in the output; if *first* = `false`, CPLEX returns the best solution found so far.
. TL—maximum time allowed for running CPLEX.
. *rhs*—right-hand side of the local branching constraint; it defines the size of the neighborhood within the inner or VND loop.

. *cont*—logical variable which indicates if the inner loop continues (`true`) or not (`false`).
. *x_opt* and *f_opt*—incumbent solution and corresponding objective function value.
x_cur, *f_cur*, *k_cur*—current solution, objective function value and neighborhood from where VND local search starts.
. *x_next* and *f_next*—solution and corresponding objective function value obtained by CPLEX in inner loop.

3.7 Variable Neighborhood Search for Continuous Global Optimization

The general form of the continuous constrained nonlinear global optimization problem (GOP) is given as follows:

$$\text{(GOP)} \quad \left[\begin{array}{lll} \min & f(x) & \\ \text{s.t.} & g_i(x) \le 0 & \forall i \in \{1,2,\ldots,m\} \\ & h_i(x) = 0 & \forall i \in \{1,2,\ldots,r\} \\ & a_j \le x_j \le b_j & \forall j \in \{1,2,\ldots,n\} \end{array} \right. ,$$

where $x \in R^n$, $f : R^n \to R$, $g_i : R^n \to R$, $i = 1,2,\ldots,m$, and $h_i : R^n \to R$, $i = 1,2,\ldots,r$, are possibly nonlinear continuous functions, and $a, b \in R^n$ are the variable bounds. A box constraint GOP is defined when only the variable bound constraints are present in the model.

The GOP naturally arises in many applications, e.g., in advanced engineering design, data analysis, financial planning, risk management, scientific modeling. Most cases of practical interest are characterized by multiple local optima and, therefore, a search effort of global scope is needed to find the globally optimal solution.

If the feasible set X is convex and objective function f is convex, then (GOP) is relatively easy to solve, i.e., the Karush–Kuhn–Tucker conditions can be applied. However, if X is not a convex set or f is not a convex function, we can have many local optima and the problem may not be solved with classical techniques.

For solving (GOP), VNS has been used in two different ways: (i) with neighborhoods induced by using an ℓ_p norm and (ii) without using an ℓ_p norm.

(i) **VNS with ℓ_p norm neighborhoods** [42, 79, 84, 86]. A natural approach in applying VNS for solving GOPs is to induce neighborhood structures $\mathcal{N}_k(x)$ from an ℓ_p metric such as

$$\rho(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} \quad (1 \le p < \infty) \tag{3.3}$$

or

$$\rho(x,y) = \max_{1 \le i \le n} |x_i - y_i| \quad (p \to \infty). \tag{3.4}$$

The neighborhood $\mathcal{N}_k(x)$ denotes the set of solutions in the kth neighborhood of x, and using the metric ρ, it is defined as

$$\mathcal{N}_k(x) = \{y \in X \mid \rho(x,y) \le \rho_k\} \tag{3.5}$$

or

$$\mathcal{N}_k(x) = \{y \in X \mid \rho_{k-1} < \rho(x,y) \le \rho_k\}, \tag{3.6}$$

where ρ_k, known as the radius of $\mathcal{N}_k(x)$, is monotonically increasing with k.

For solving box constraint GOPs, both [42] and [79] use neighborhoods as defined in Equation (3.6). The basic differences between the two are as follows: (1) in the procedure suggested in [79] the ℓ_∞ norm is used, while in [42] the choice of metric is either left to the analyst or changed automatically in some predefined order; (2) the commercial solver SNOPT [49] is used as a local search procedure within VNS in [79], while in [42], the analyst may choose one out of six different convex minimizers. A VNS-based heuristic for solving the generally constrained GOP is suggested in [84]. There, the problem is first transformed into a sequence of box constrained problems within the well-known exterior point method:

$$\min_{a \le x \le b} F_{\mu,q}(x) = f(x) + \frac{1}{\mu}\sum_{i=1}^{m}(\max\{0, g_i(x)\})^q + \sum_{i=1}^{r}|h_i(x)|^q, \tag{3.7}$$

where μ and $q \ge 1$ are a positive penalty parameter and penalty exponent, respectively. Algorithm 17 outlines the steps for solving the box constraint subproblem as proposed in [84]:

Algorithm 17 VNS using a ℓ_p norm

```
Function Glob-VNS ($x^*, k_{max}, t_{max}$)
1  Select the set of neighborhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{\max}$
2  Select the array of random distributions types and an initial point $x^* \in X$
3  $x \leftarrow x^*$, $f^* \leftarrow f(x)$, $t \leftarrow 0$
4  while $t < t_{max}$ do
5      $k \leftarrow 1$
6      repeat
7          for all distribution types do
8              $y \leftarrow$ Shake($x^*, k$) // Get $y \in \mathcal{N}_k(x^*)$ at random
9              $y' \leftarrow$ BestImprovment($y$) // Apply LS to obtain a local minimum $y'$
10             if $f(y') < f^*$ then
11                 $x^* \leftarrow y'$, $f^* \leftarrow f(y')$, go to line 5
12         $k \leftarrow k+1$
       until $k = k_{\max}$
13     $t \leftarrow$ CpuTime()
```

The Glob-VNS procedure from Algorithm 17 contains the following parameters in addition to $k_{\max}$ and $t_{\max}$: (1) *Values of radii* ρ_k, $k = 1, \ldots, k_{\max}$. Those values may be defined by the user or calculated automatically in the minimizing process; (2) *Geometry* of neighborhood structures $\mathcal{N}_k$, defined by the choice of metric. Usual choices are the ℓ_1, ℓ_2, and ℓ_∞ norms; (3) *Distribution* used for obtaining the random point y from $\mathcal{N}_k$ in the *Shaking* step. Uniform distribution in $\mathcal{N}_k$ is the obvious choice, but other distributions may lead to much better performance on some problems. Different choices of geometric neighborhood shapes and random point distributions lead to different VNS-based heuristics.

(ii) **VNS without using ℓ_p norm**. Two different neighborhoods, $N_1(x)$ and $N_2(x)$, are used in the VNS-based heuristic suggested in [97]. In $N_1(x)$, r (a parameter) random directions from the current point x are generated and a one-dimensional search along each direction is performed. The best point (out of r) is selected as a new starting solution for the next iteration, if it is better than the current one. If not, as in VND, the search is continued within the next neighborhood $N_2(x)$. The new point in $N_2(x)$ is obtained as follows. The current solution is moved for each x_j $(j = 1,\ldots,n)$ by a value Δ_j, taken at random from interval $(-\alpha,\alpha)$, i.e., $x_j^{(\text{new})} = x_j + \Delta_j$ or $x_j^{(\text{new})} = x_j - \Delta_j$. Points obtained by the plus or minus sign for each variable define the neighborhood $N_2(x)$. If a relative increase of 1% in the value of $x_j^{(\text{new})}$ produces a better solution than $x^{(\text{new})}$, the + sign is chosen; otherwise the $-$ sign is chosen.

Neighborhoods N_1 and N_2 are used for designing two algorithms. The first, called VND, iterates over these neighborhoods until there is no improvement in the solution value. In the second variant, a local search is performed with N_2 and $k_{\max}$ is set to 2 for the shaking step. In other words, a point from the neighborhood N_2 is obtained by generating a random direction followed by a line search along it (as prescribed for N_1) and then by changing each of the variables (as prescribed for N_2).

It is interesting to note that computational results reported by all VNS-based heuristics were very promising. They usually outperformed other recent approaches from the literature.

3.8 Mixed Integer Nonlinear Programming (MINLP) Problem

The problems we address here are cast in the following general form:

$$(\mathtt{MINLP})\quad \left[\begin{array}{lll} \min & f(x) & \\ \text{s.t.} & \ell_i \le g_i(x) \le u_i & \forall i \in \{1,\ldots,m\} \\ & a_j \le x_j \le b_j & \forall j \in N \\ & x_j \in \{0,1\} & \forall j \in \mathcal{B} \neq \emptyset \\ & x_j \ge 0, \mathtt{integer} & \forall j \in \mathcal{G} \\ & x_j \ge 0 & \forall j \in \mathcal{C} \end{array}\right.,$$

where the set of indices $N = \{1,2,\ldots,n\}$, as in the formulation of MILP, is partitioned into three subsets $\mathcal{B},\mathcal{G}$ and $\mathcal{C}$, corresponding to binary, general integer and continuous variables, respectively. Therefore, in the above formulation, the x_j are the decision variables, $f : R^n \to R$ is a possibly nonlinear function, $g : R^n \to R^m$ is a vector of m possibly nonlinear functions (assumed to be differentiable), $\ell, u \in R^m$ are the constraint bounds (which may be set to $\pm\infty$), and $a, b \in R^n$ are the variable bounds.

In order to apply VNS for solving (MINLP), one needs to answer three questions: (i) how to define the set of neighborhoods around any solution x; (ii) how to perform (local) search starting from any point and finishing with a feasible solution; (iii) how to get a feasible solution starting from an infeasible point.

(i) **Neighborhoods.** Naturally, for the set of binary variables x_j, $j \in \mathscr{B}$, the Hamming distance, expressed by local branching constraints (3.2), can be used. For the set of continuous variables x_j, $j \in \mathscr{G}$ one can use the ℓ_p norm (3.3) or (3.4); for the set of integer variables, either an extension of formula (3.2) given in [43] or (3.6) can be used. The point $x' \in \mathscr{N}_k(x)$ denotes a kth neighborhood solution of combined binary, continuous, and integer parts.

(ii) **Local search.** The local search phase mainly depends on available software. The simplest way is just to use an existing commercial solver for MINLP by adding constraints that define neighborhood N_k. Such an approach for solving (MILP) is applied in the local branching [43] and VNS branching [71] methods explained earlier. Since such a solver for MINLP does not exist on the market, it becomes necessary to split the problem at any branching node into easier subproblems and alternately solve these subproblems until an improved feasible solution is hopefully found. For example, integrality conditions may be relaxed in one subproblem, and continuous variables fixed in the next. The partition into subproblems depends mostly on existing solvers and their qualities. By relaxing all binary and integer variables, a NLP problem is obtained, whose complexity depends on the properties of $f(x)$ and $g_i(x), i \in \{1, \ldots, m\}$. If all functions are convex, the problem may be much easier to solve. The relaxed solution is then used to get a lower bound within a branch and bound (BB) enumerative procedure. Thus, the quality of the solution obtained in this local search phase mostly depends on the way different solvers are combined and on the quality of such solvers.

(iii) **Feasible solution.** Realistically sized MINLPs can often have thousands (or tens of thousands) of variables (continuous and integer) and nonconvex constraints. With such sizes, it becomes a difficult challenge to even find a feasible solution, and BB algorithms become almost useless. Some good solvers targeting convex MINLPs exist in the literature [1, 24, 26, 45, 46, 78]; and although they can all be used on nonconvex MINLPs as well (forsaking the optimality guarantee), their quality varies wildly in practice with the instance of the problem being solved, resulting in a high fraction of "false negatives" (i.e., feasible problems for which no feasible solution was found). The feasibility pump (FP) idea was recently extended to convex MINLPs [25], but again this does not work so well when applied to unmodified nonconvex MINLPs.

In a recent paper [80] an effective and reliable MINLP heuristic based on VNS is suggested, called Relaxed-Exact Continuous-Integer Problem Exploration

(RECIPE for short). RECIPE puts together a global search phase based on VNS and a local search phase based on a BB-type heuristic. The VNS global phase relies on neighborhoods defined as hyperrectangles for the continuous and general integer variables and by local branching constraints for the binary variables. The local phase employs a BB solver for convex MINLPs [46], which is applied to nonconvex MINLPs heuristically. A local NLP solution using a sequential quadratic programming (SQP) algorithm [49] supplies an initial constraint-feasible solution to be employed by the BB as initial upper bound. RECIPE (see Algorithm 18) is an efficient, effective, and reliable general-purpose algorithm for solving complex MINLPs of small and medium scale. The original contribution of RECIPE is the particular combination of some well-known and well-tested tools to produce a very powerful global optimization method. It turns out that RECIPE, acting on the whole MINLPLib library [29], is able to find optima equal to or better than those reported in the literature for 55% of the instances. The closest competitor is SBB+CONOPT with 37%. The known optima are improved in 7% of the cases.

Algorithm 18 The RECIPE heuristic for solving MINLP

```
Function RECIPE ($a, b, k_{max}, t_{max}, x^*$)
1  $x^* = (a+b)/2; t \leftarrow 0$
2  while $t < t_{max}$ do
3      $k \leftarrow 1$
4      while $k \leq k_{max}$ do
5          $i \leftarrow 1$
6          while $i \leq b$ do
7              Sample $\bar{x} \in \mathcal{N}_k(x^*)$ at random
8              $x \leftarrow$ SQP($\bar{x}$)
9              if $x$ not feasible then
10                 $x \leftarrow \bar{x}$
11             $x' \leftarrow$ BB ($x, k, k_{max}$)
12             if $x'$ is better than $x^*$ then
13                 $x^* \leftarrow x'; k \leftarrow 0$; Exit loop $i$
14             $i \leftarrow i+1$
15         $k \leftarrow k+1$
16     $t \leftarrow$ CpuTime()
```

3.9 Discovery Science

In all the above applications, VNS is used as an optimization tool. It can also lead to results in "discovery science," i.e., help in the development of theories. This has been done for graph theory in a long series of papers with the common title "Variable neighborhood search for extremal graphs" that report on the development and applications of the AutoGraphiX (AGX) system [4, 36, 37]. This system addresses the following problems:

- Find a graph satisfying given constraints.
- Find optimal or near optimal graphs for an invariant subject to constraints.
- Refute a conjecture.
- Suggest a conjecture (or repair or sharpen one).
- Provide a proof (in simple cases) or suggest an idea of proof.

A basic idea is then to address all of these problems as parametric combinatorial optimization problems on the infinite set of all graphs (or in practice some smaller subset) using a generic heuristic to explore the solution space. This is done by applying VNS to find extremal graphs with a given number n of vertices (and possibly also a given number of edges). Extremal graphs may be viewed as a family of graphs that maximize some invariant such as the independence number or chromatic number, possibly subject to constraints. We may also be interested in finding lower and upper bounds on some invariant for a given graph G. Once an extremal graph is obtained, VND with many neighborhoods may be used to build other such graphs. Those neighborhoods are defined by modifications of the graphs such as the removal or addition of an edge, rotation of an edge. Once a set of extremal graphs, parameterized by their order, is found, their properties are explored with various data mining techniques, leading to conjectures, refutations, and simple proofs or ideas of proof.

The current list of references in the series "VNS for extremal graphs" is given by [3–8, 10–12, 17–19, 23, 32, 34, 36, 37, 40, 47, 53, 61–63, 72, 73, 94, 95]. Another list of papers, not included in this series is given in [9, 13–16, 33, 35, 54–57, 60, 96]. Papers in these two lists cover a variety of topics:

(i) **Principles of the approach** [36, 37] and its implementation [4];
(ii) **Applications to spectral graph theory**, e.g., bounds on the index for various families of graphs, graphs maximizing the index subject to some conditions [3, 17, 23, 40, 57];
(iii) **Studies of classical graph parameters**, e.g., independence, chromatic number, clique number, average distance [5, 9–12, 94, 95];
(iv) **Studies of little known or new parameters of graphs**, e.g., irregularity, proximity, and remoteness [13, 62];
(v) **New families of graphs discovered by AGX**, e.g., bags, which are obtained from complete graphs by replacing an edge by a path, and bugs, which are obtained by cutting the paths of a bag [8, 72];
(vi) **Applications to mathematical chemistry**, e.g., study of chemical graph energy, and of the Randić index [18, 19, 34, 47, 53–56, 61];
(vii) **Results of a systematic study of 20 graph invariants**, which led to almost 1500 new conjectures, more than half of which were proved by AGX and over 300 by various mathematicians [7];
(viii) **Refutation or strengthening of conjectures from the literature** [6, 33, 56];
(ix) **Surveys and discussions about various discovery systems in graph theory**, assessment of the state of the art and the forms of interesting conjectures together with proposals for the design of more powerful systems [35, 60].

3.10 Conclusions

The general schemes of variable neighborhood search have been presented and discussed. In order to evaluate research development related to VNS, one needs a list of the desirable properties of metaheuristics. Eight of these are presented in Hansen and Mladenović (2003):

(i) *Simplicity:* the metaheuristic should be based on a simple and clear principle, which should be widely applicable;
(ii) *Precision:* the steps of the metaheuristic should be formulated in precise mathematical terms, independent of possible physical or biological analogies which may have been the initial source of inspiration;
(iii) *Coherence:* all steps of the heuristics for solving a particular problem should follow naturally from the metaheuristic principles;
(iv) *Effectiveness:* 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 benchmark problems for which such solutions are known;
(v) *Efficiency:* heuristics for particular problems should take a moderate computing time to provide optimal or near-optimal solutions, or comparable or better solutions than the state of the art;
(vi) *Robustness:* the performance of the metaheuristics should be consistent over a variety of instances, i.e., not merely fine tuned to some training set and not so good elsewhere;
(vii) *User friendliness:* the metaheuristics should be clearly expressed, easy to understand and, most importantly, easy to use. This implies they should have as few parameters as possible, ideally none;
(viii) *Innovation:* the principle of the metaheuristic and/or the efficiency and effectiveness of the heuristics derived from it should lead to new types of application.

This list has been completed with three more items added by one member of the present team and his collaborators:

(ix) *Generality:* the metaheuristic should lead to good results for a wide variety of problems;
(x) *Interactivity:* the metaheuristic should allow the user to incorporate his knowledge to improve the resolution process;
(xi) *Multiplicity:* the metaheuristic should be able to produce several near-optimal solutions from which the user can choose.

As shown above, VNS possesses, to a great extent, all of the above properties. This has led to heuristics which are among the very best ones for many problems. Interest in VNS is growing quickly. This is evidenced by the increasing number of papers published each year on this topic (10 years ago, only a few; 5 years ago, about a dozen; and about 50 in 2007). Moreover, the 18th EURO mini-conference held in Tenerife in November 2005 was entirely devoted to VNS. It led to special issues of *IMA Journal of Management Mathematics* in 2007 [81], *European Journal of Operational Research* [68], and *Journal of Heuristics* [89] in 2008. In retrospect, it appears that the good shape of VNS research is due to the following perspectives, strongly influenced by Karl Popper's philosophy of science [91]: (i) in devising heuristics favor insight over efficiency (which comes later) and (ii) learn from heuristic failures.

References

1. Abhishek, K., Leyffer, S., Linderoth, J.: FilMINT: An outer-approximation based solver for nonlinear mixed-integer programs. Technical Report ANL/MCSP1374- 0906, Argonne National Laboratory, 2007
2. Aloise, D.J., Aloise, D., Rocha, C.T.M., Ribeiro, C.C., Ribeiro, J.C., Moura, L.S.S.: Scheduling workover rigs for onshore oil production. Discrete Appl. Math. **154**, 695–702 (2006)
3. Aouchiche, M., Bell, F.K., Cvetković, D., Hansen, P., Rowlinson, P., Simić, S.K., Stevanović, D.: Variable neighborhood search for extremal graphs 16. Some conjectures related to the largest eigenvalue of a graph. Eur. J. Oper. Res. **191**, 661–676 (2008)
4. Aouchiche, M., Bonnefoy, J.M., Fidahoussen, A., Caporossi, G., Hansen, P., Hiesse, L., Lacheré, J., Monhait, A.: Variable neighborhood search for extremal graphs 14. The AutoGraphiX 2 system. In: Liberti, L., Maculan, N. (eds.) Global Optimization: From Theory to Implementation, pp. 281–309. Springer, Berlin (2006)
5. Aouchiche, M., Brinkmann, G., Hansen, P.: Variable neighborhood search for extremal graphs 21. Conjectures and results about the independence number. Discrete Appl. Math. **156**, 2530–2542 (2009)
6. Aouchiche, M., Caporossi, G., Cvetković, D.: Variable neighborhood search for extremal graphs 8. Variations on Graffiti 105. Congressus Numerantium **148**, 129–144 (2001)
7. Aouchiche, M., Caporossi, G., Hansen, P.: Variable Neighborhood search for extremal graphs 20. Automated comparison of graph invariants. MATCH. Commun. Math. Comput. Chem. **58**, 365–384 (2007)
8. Aouchiche, M., Caporossi, G., Hansen, P.: Variable neighborhood search for extremal graphs 27. Families of extremal graphs. Les Cahiers du GERAD **G-2007-87** (2007)
9. Aouchiche, M., Caporossi, G., Hansen, P., Laffay, M.: AutoGraphiX: a survey. Electron. Notes Discrete Math. **22**, 515–520 (2005)
10. Aouchiche, M., Favaron, O., Hansen, P.: Variable neighborhood search for extremal graphs 22. Extending bounds for independence to upper irredundance. Discrete Appl. Math. **157**, 3497–3510 (2009)
11. Aouchiche, M., Favaron, O., Hansen, P.: Recherche à voisinage variable de graphes extrèmes 26. Nouveaux résultats sur la maille (French). Les Cahiers du GERAD **G-2007-55** (2007)
12. Aouchiche, M., Hansen, P.: Recherche à voisinage variable de graphes extrèmes 13. À propos de la maille (French). RAIRO Oper. Res. **39**, 275–293 (2005)
13. Aouchiche, M., Hansen, P.: Automated results and conjectures on average distance in graphs. Graph Theory in Paris, Trends Math. **6**, 21–36 (2007)
14. Aouchiche, M., Hansen, P.: On a conjecture about the Randic index. Discrete Math. **307**, 262–265 (2007)
15. Aouchiche, M., Hansen, P.: Bounding average distance using minimum degree. Graph Theory Notes of New York (to appear) (2009)
16. Aouchiche, M., Hansen, P.: Nordhaus-Gaddum relations for proximity and remoteness in graphs. Les Cahiers du GERAD **G-2008-36** (2008)
17. Aouchiche, M., Hansen, P., Stevanović, D.: Variable neighborhood search for extremal graphs 17. Further conjectures and results about the index. Discusiones Mathematicae: Graph Theory (to appear) (2009)

18. Aouchiche, M., Hansen, P., Zheng, M.: Variable neighborhood search for extremal graphs 18. Conjectures and results about the Randic index. MATCH. Commun. Math. Comput. Chem. **56**, 541–550 (2006)
19. Aouchiche, M., Hansen, P., Zheng, M.: Variable Neighborhood Search for Extremal Graphs 19. Further Conjectures and Results about the Randic Index. MATCH. Commun. Math. Comput. Chem. **58**, 83–102 (2007)
20. Audet, C., Báchard, V., Le, Digabel, S.: Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. J. Global Optim. **41**, 299–318 (2008)
21. Audet, C., Brimberg, J., Hansen, P., Mladenović, N.: Pooling problem: alternate formulation and solution methods. Manage. Sci. **50**, 761–776 (2004)
22. Belacel, N., Hansen, P., Mladenović, N.: Fuzzy J-means: a new heuristic for fuzzy clustering. Pattern Recognit. **35**, 2193–2200 (2002)
23. Belhaiza, S., de, Abreu, N., Hansen, P., Oliveira, C.: Variable neighborhood search for extremal graphs 11. Bounds on algebraic connectivity. In: Avis, D., Hertz, A., Marcotte, O. (eds.) Graph Theory and Combinatorial Optimization, pp. 1–16. (2007)
24. Bonami, P., Biegler, L.T., Conn, A.R., Cornuejols, G., Grossmann, I.E., Laird, C.D., Lee, J., Lodi, A., Margot, F., Sawaya, N., Wachter, A.: An algorithmic framework for convex mixed integer nonlinear programs. Technical Report RC23771, IBM Corporation (2005)
25. Bonami, P., Cornuejols, G., Lodi, A., Margot, F.A.: Feasibility pump for mixed integer nonlinear programs. Technical Report RC23862 (W0602-029), IBM Corporation (2006)
26. Bonami, P., Lee, J.: BONMIN User's manual. Technical Report, IBM Corporation (2007)
27. Brimberg, J., Hansen, P., Mladenović, N., Taillard, É.: Improvements and comparison of heuristics for solving the multisource Weber problem. Oper. Res. **48**, 444–460 (2000)
28. Brimberg, J., Mladenović, N.: A. variable neighborhood algorithm for solving the continuous location-allocation problem. Stud. Locat. Anal. **10**, 1–12 (1996)
29. Bussieck, M.R., Drud, A.S., Meeraus, A.: MINLPLib - A collection of test models for mixed-integer nonlinear programming. INFORMS J. Comput. **15**, 114–119 (2003)
30. Canuto, S., Resende, M., Ribeiro, C.: Local search with perturbations for the prize-collecting Steiner tree problem in graphs. Networks **31**, 201–206 (2001)
31. Caporossi, G., Alamargot, D., Chesnet, D.: Using the computer to study the dynamics of the handwriting processes. Lect. Notes Comput. Sci. **3245**, 242–254 (2004)
32. Caporossi, G., Cvetković, D., Gutman, I., Hansen, P.: Variable neighborhood search for extremal graphs 2. Finding graphs with extremal energy. J. Chem. Inf. Comput. Sci. **39**, 984–996 (1999)
33. Caporossi, G., , Dobrynin, A.A., Gutman, I., Hansen, P.: Trees with palindromic Hosoya polynomials. Graph Theory Notes New York **37**, 10–16 (1999)
34. Caporossi, G., Gutman, I., Hansen, P.: Variable neighborhood search for extremal graphs 4. Chemical trees with extremal connectivity index. Comput. Chem. **23**, 469–477 (1999)
35. Caporossi, G., Gutman, I., Hansen, P., Pavlović, L., Graphs with maximum connectivity index. Comput. Biol. Chem. **27**, 85–90 (2003)
36. Caporossi, G., Hansen, P.: Variable neighborhood search for extremal graphs 1. The AutoGraphiX system. Discrete Math. **212**, 29–44 (2000)
37. Caporossi, G., Hansen, P.: Variable neighborhood search for extremal graphs 5. Three ways to automate finding conjectures. Discrete Math. **276**, 81–94 (2004)
38. Carrabs, F., Cordeau, J.-F., Laporte, G.: Variable neighbourhood search for the pickup and delivery traveling salesman problem with LIFO loading. INFORMS J. Comput. **19**, 618–632 (2007)
39. Carrizosa, E., Martín-Barragán, B., Plastria, F., Romero, Morales, D.: On the selection of the globally optimal prototype subset for nearest-neighbor classification. INFORMS J. Comput. **19**, 470–479 (2007)
40. Cvetkovic, D., Simic, S., Caporossi, G., Hansen, P.: Variable neighborhood search for extremal graphs 3. On the largest eigenvalue of color-constrained trees. Linear Multilinear Algebra **49**, 143–160 (2001)

41. Davidon, W.C.: Variable metric algorithm for minimization. Argonne National Laboratory Report ANL-5990 (1959)
42. Dražić, M., Kovacevic-Vujcić, V., Cangalović, M., Mladenović, N.: GLOB - A. new VNS-based software for global optimization In: Liberti L., Maculan N. (eds.) Global Optimization: From Theory to Implementation, pp. 135–144, Springer, Berlin (2006)
43. Fischetti, M., Lodi, A.: Local branching. Math. Programming **98**, 23–47 (2003)
44. Fletcher, R., Powell, M.J.D.: Rapidly convergent descent method for minimization. Comput. J. **6**, 163–168 (1963)
45. Fletcher, R., Leyffer, S.: Solving mixed integer nonlinear programs by outer approximation. Math. Program. **66**, 327–349 (1994)
46. Fletcher, R, Leyffer, S.: Numerical experience with lower bounds for MIQP branch-and-bound. SIAM J. Optim. **8**, 604–616 (1998)
47. Fowler, P.W., Hansen, P., Caporossi, G., Soncini, A.: Variable neighborhood search for extremal graphs 7. Polyenes with maximum HOMO-LUMO gap. Chem. Phys. Lett. **49**, 143–146 (2001)
48. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, New York (1978)
49. Gill, P., Murray, W., Saunders, M.A.: SNOPT:An SQP algorithms for largescale constrained optimization. SIAM J. Optim. **12**, 979–1006 (2002)
50. Gill, P., Murray, W., Wright, M.: Practical Optimization. Academic Press, London (1981)
51. Glover, F., Kochenberger, G., (eds.) Handbook of Metaheuristics. Kluwer, Dorchester, London, New York (2003)
52. Griffith, R.E., Stewart, R.A.: A nonlinear programming technique for the optimization of continuous processing systems. Manage. Sci. **7**, 379–392 (1961)
53. Gutman, I., Hansen, P., Mélot, H., Variable neighborhood search for extremal graphs 10. Comparison of irregularity indices for chemical trees. J. Chem. Inf. Model. **45**, 222–230 (2005)
54. Gutman, I., Miljković, O., Caporossi, G., Hansen, P.: Alkanes with small and large Randić connectivity indices. Chem. Phys. Lett. **306**, 366–372 (1999)
55. Hansen, P.: Computers in Graph Theory. Graph Theory Notes New York **43**, 20–39 (2002)
56. Hansen, P.: How far is, should and could be conjecture-making in graph theory an automated process? Graph Discov., Dimacs Series Discrete Math. Theor. Comput. Sci. **69**, 189–229 (2005)
57. Hansen, P., Aouchiche, M., Caporossi, G., Mélot, H, Stevanović, D.: What forms do interesting conjectures have in graph theory? Graph Discov., Dimacs Ser. Discrete Math. Theor. Comput. Sci. **69**, 231–251 (2005)
58. Hansen, P., Brimberg, J., Uro̦ sević, D., Mladenović, N.: Primal-dual variable neighborhood search for the simple plant location problem. INFORMS J. Comput. **19**, 552–564 (2007)
59. Hansen, P., Jaumard, B., Mladenović, N., Parreira, A.: Variable neighborhood search for weighted maximum satisfiability problem. Les Cahiers du GERAD **G-2000-62** (2000)
60. Hansen, P., Mélot, H.: Computers and discovery in algebraic graph theory. Linear Algebra Appl. **356**, 211–230 (2002)
61. Hansen, P., Mélot, H.: Variable neighborhood search for extremal graphs 6. Analysing bounds for the connectivity index. J. Chem. Inf. Comput. Sci. **43**, 1–14 (2003)
62. Hansen, P., Mélot, H.: Variable neighborhood search for extremal graphs 9. Bounding the irregularity of a graph. Graph Discov., Dimacs Series Discrete Math. Theor. Comput. Sci. **69**, 253–264 (2005)
63. Hansen, P., Mélot, H., Gutman, I.: Variable neighborhood search for extremal graphs 12. A note on the variance of bounded degrees in graphs. MATCH Commun. Math. Comput. Chem. **54**, 221–232 (2005)
64. Hansen, P., Mladenović, N.: Variable neighborhood search: Principles and applications. Eur. J. Oper. Research **130**, 449–467 (2001)
65. Hansen, P., Mladenović, N.: J-Means: A. new local search heuristic for minimum sum-of-squares clustering. Pattern Recognit. **34**, 405–413 (2001)

66. Hansen, P., Mladenović, N.: Developments of variable neighborhood search. In: Ribeiro, C., Hansen, P. (eds.) Essays and Surveys in Metaheuristics, pp. 415–440, Kluwer, Dorchester, London, New York (2001)
67. Hansen, P., Mladenović, N.: Variable neighborhood search. In: Glover F., Kochenberger G. (eds.) Handbook of Metaheuristics, pp. 145–184, Kluwer, Dorchester, London, New York (2003)
68. Hansen, P., Mladenović, N., Moreno Pérez, J.A.: Variable neighborhood search. Eur. J. Oper. Res. **191**, 593–595 (2008)
69. Hansen, P., Mladenović, N., Moreno Pérez, J.A.: Variable neighborhood search: methods and applications. 4OR. Q. J. Oper. Res. **6**, 319–360 (2008)
70. Hansen, P., Mladenović, N., Pérez-Brito, D.: Variable neighborhood decomposition search. J. Heuristics **7**, 335–350 (2001)
71. Hansen, P., Mladenović, N., Urošević, D.: Variable neighborhood search and local branching. Comput. Oper. Res. **33**, 3034–3045 (2006)
72. Hansen, P., Stevanović, D.: Variable neighborhood search for extremal graphs 15. On bags and bugs, Discrete Appl. Math. **156**, 986–997 (2005)
73. Hansen, P., Vukičević, D.: Variable neighborhood search for extremal graphs 23. On the Randic index and the chromatic number. Discrete Math. **309**, 4228–4234 (2009)
74. Hertz, A., Plumettaz, M., Zufferey, N.: Variable space search for graph coloring. Discrete Appl. Math. **156**, 2551–2560 (2008)
75. ILOG CPLEX 10.1. User's Manual (2006)
76. Jornsten, K., Lokketangen, A.: Tabu search for weighted k-cardinality trees. Asia-Pacific J. Oper. Res. **14**, 9–26 (1997)
77. Lejeune, M.A.: A. variable neighborhood decomposition search method for supply chain management planning problems. Eur. J. Oper. Res. **175**, 959–976 (2006)
78. Leyffer, S., User, manual, for, , `minlp_bb` Technical Report, University of Dundee, UK (1999)
79. Liberti, L., Dražić, M.: Variable neighbourhood search for the global optimization of constrained NLPs. In: Proceedings of GO Workshop, Almeria, Spain (2005)
80. Liberti, L., Nannicini, G., Mladenović, N.: A good recipe for solving MINLPs. In: Maniezzo, V., Stuetze, T., Voss, S. (eds.) MATHEURISTICS: Hybridizing metaheuristics and mathematical programming, Operations Research/Computer Science Interface Series. Springer, Berlin (2008)
81. Melián, B., Mladenović, N.: (eds.) IMA J. Manage. Math. **18**, 99–100 (2007)
82. Mladenović, N.: A variable neighborhood algorithm—a new metaheuristic for combinatorial optimization. Abstracts of papers presented at Optimization Days, Montréal, p. 112 (1995)
83. Mladenović, N.: Formulation space search—a new approach to optimization (plenary talk). In: Vuleta, J. (ed.) Proceedings of XXXII SYMOPIS'05, pp. 3–5 Vrnjacka Banja, Serbia (2005)
84. Mladenović, N., Dražić, M., Kovačevic-Vujčić, V., Čangalović, M.: General variable neighborhood search for the continuous optimization. Eur. J. Oper. Res. **191**, 753–770 (2008)
85. Mladenović, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**, 1097–1100 (1997)
86. Mladenović, N., Petrović, J., Kovačević-Vujčić, V., Čangalović, M.: Solving spread spectrum radar polyphase code design problem by tabu search and variable neighborhood search. Eur. J. Oper. Res. **151**, 389–399 (2003)
87. Mladenović, N., Plastria, F., Urošević, D.: Reformulation descent applied to circle packing problems. Comput. Oper. Res. **32**, 2419–2434 (2005)
88. Mladenović, N., Plastria, F., Urošević, D.: Formulation space search for circle packing problems. Lect. Notes Comput. Sci. **4638**, 212–216 (2007)
89. Moreno-Vega, J.M., Melián, B.: Introduction to the special issue on variable neighborhood search. J. Heuristics **14**, 403–404 (2008)
90. Plastria, F., Mladenović, N., Urošević, D.: Variable neighborhood formulation space search for circle packing. 18th Mini Euro Conference VNS., Tenerife, Spain (2005)

91. Popper K.: The Logic of Scientific Discovery. London, Hutchinson (1959)
92. Ribeiro, C.C., de, Souza, M.C.: Variable neighborhood search for the degree-constrained minimum spanning tree problem. Discrete Appl. Math. **118**, 43–54 (2002)
93. Ribeiro, C.C., Uchoa, E., Werneck, R.: A hybrid GRASP with perturbations for the Steiner problem in graphs. INFORMS J. Comput. **14**, 228–246 (2002)
94. Sedlar, J., Vukicevic, D., Aouchiche, M., Hansen, P.: Variable neighborhood search for extremal graphs 24. Conjectures and results about the clique number. Les Cahiers du GERAD **G-2007-33** (2007)
95. Sedlar, J., Vukicevic, D., Aouchiche, M., Hansen, P.: Variable neighborhood search for extremal graphs 25. Products of connectivity and distance measures. Les Cahiers du GERAD **G-2007-47** (2007)
96. Stevanovic, D., Aouchiche, M., Hansen, P.: On the spectral radius of graphs with a given domination number. Linear Algebra Appl. **428**, 1854–1864 (2008)
97. Toksari, A.D., Güner, E.: Solving the unconstrained optimization problem by a variable neighborhood search. J. Math. Anal. Appli. **328**, 1178–1187 (2007)
98. Whitaker, R.: A fast algorithm for the greedy interchange of large-scale clustering and median location problems. INFOR **21**, 95–108 (1983)
99. Zhang, C., Lin, Z., Lin, Z.: Variable neighborhood search with permutation distance for QAP. Lect. Notes Comput. Sci. **3684**, 81–88 (2005)