ELSEVIER

European Journal of Operational Research 130 (2001) 449–467

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/dsw

Invited Review

Variable neighborhood search: Principles and applications

Pierre Hansen *, Nenad Mladenović

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

Received 1 July 1999; accepted 22 December 1999

Abstract

Systematic change of neighborhood within a possibly randomized local search algorithm yields a simple and effective metaheuristic for combinatorial and global optimization, called variable neighborhood search (VNS). We present a basic scheme for this purpose, which can easily be implemented using any local search algorithm as a subroutine. Its effectiveness is illustrated by solving several classical combinatorial or global optimization problems. Moreover, several extensions are proposed for solving large problem instances: using VNS within the successive approximation method yields a two-level VNS, called variable neighborhood decomposition search (VNDS); modifying the basic scheme to explore easily valleys far from the incumbent solution yields an efficient skewed VNS (SVNS) heuristic. Finally, we show how to stabilize column generation algorithms with help of VNS and discuss various ways to use VNS in graph theory, i.e., to suggest, disprove or give hints on how to prove conjectures, an area where metaheuristics do not appear to have been applied before.

Keywords: Heuristic; Metaheuristic; Variable neighborhood search; VNS

1. Introduction

Since the early years of O.R., heuristics have been widely used to solve a large variety of practical problems. Indeed, modeling of complex issues often leads to NP-hard problems. While such problems are intractable in worst case, they may not be so in average case or in the case under study. However, the large set of tools now available to solve them exactly (branch-and-bound, cutting planes, decomposition, Lagrangian relaxation, column generation, ...) may not suffice to solve very large instances, and thus one reverts to heuristics. Moreover, as will be illustrated below, heuristics may be very useful *within* exact algorithms.

One much used type of heuristic is *local search*. It proceeds from an initial solution by a sequence of local changes, which improve each time the value of the objective function until a local optimum is found. In recent years, several metaheuristics have been proposed, which extend in various ways this scheme and avoid being trapped in local optima with a poor value (see [56] for a multi-authored book-length survey and [54] for an extensive bibliography). These metaheuristics, e.g.,

* Corresponding author.
E-mail address: pierre.hansen@hec.ca (P. Hansen).

0377-2217/01/$ - see front matter
PII: S0377-2217(00)00100-4

adaptive multi-start [4], variable depth search [47,55], simulated annealing [42], Tabu search (TS) [23–27,30], GRASP [14] and others such as genetic search [38] and ant colonies [11], have led to much improved results in many practical contexts.

In this paper, we examine a relatively unexplored approach to the design of heuristics: change of neighborhood in the search. Using systematically this idea and very little more, i.e., only a local search routine, leads to a new metaheuristic, which is widely applicable. We call this approach variable neighborhood search (VNS). Contrary to other metaheuristics based on local search methods, VNS does not follow a trajectory but explores increasingly distant neighborhoods of the current incumbent solution, and jumps from this solution to a new one if and only if an improvement has been made. In this way often favorable characteristics of the incumbent solution, e.g., that many variables are already at their optimal value, will be kept and used to obtain promising neighboring solutions. Moreover, a local search routine is applied repeatedly to get from these neighboring solutions to local optima. This routine may also use several neighborhoods. Therefore, to construct different neighborhood structures and to perform a systematic search, one needs to have a way for finding the distance between any two solutions, i.e., one needs to supply the solution space with some metric (or quasi-metric) and then induce neighborhoods from it. In the application part of the next sections, we answer this problem-specific question for each problem considered.

The paper is organized as follows. Rules of the basic VNS are given in Section 2, followed by application to five combinatorial or global optimization problems. In Section 3, several extensions aimed at solving very large problem instances are presented. They include a variable neighborhood decomposition search (VNDS) heuristic and a modification of the basic scheme, called skewed VNS (SVNS), aimed at efficient exploration of valleys far from the incumbent solution. This technique is illustrated on one more combinatorial optimization problem. New types of applications of VNS are described in Section 4. Its role in making efficient stabilized column generation is first explained. It is then shown how VNS can be applied in graph theory, an area where (contrary to graph optimization) metaheuristics do not appear to have been used before. Section 5 concludes the paper with a brief discussion of criteria for evaluating metaheuristics with application to VNS.

2. Basic VNS

In this section, we describe in general terms the basic rules of the VNS metaheuristic. We then illustrate them on five problems:

1. a classical combinatorial optimization problem, i.e., the traveling salesman problem (TSP);
2. a central problem of discrete location theory, i.e., the p-median problem (PM);
3. a well-known problem of global optimization, which is the counterpart of the previous problem in continuous location theory, i.e., the multi-source Weber (MW) problem;
4. a much studied problem of cluster analysis, i.e., the minimum sum-of-squares clustering (or partitioning) problem (MSSC);
5. an important problem of structured global optimization, i.e., the bilinear programming problem with bilinear constraints (BBLP).

In each case, a comparison is made with results of other classical or recent state-of-the-art heuristics.

2.1. Rules

Let us denote with $\mathcal{N}_k$ ($k = 1, \ldots, k_{\max}$), a finite set of pre-selected neighborhood structures, and with $\mathcal{N}_k(x)$ the set of solutions in the kth neighborhood of x. (Most local search heuristics use one neighborhood structure, i.e., $k_{\max} = 1$.) Steps of the basic VNS are presented in Fig. 1. They make use of a local search routine discussed later.

The stopping condition may be e.g., maximum CPU time allowed, maximum number of iterations, or maximum number of iterations between two improvements. Often successive neighborhoods $\mathcal{N}_k$ will be nested. Observe that point x' is generated at random in step 2a in order to avoid

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

Repeat the following until the stopping condition is met:

(1) Set $k \leftarrow 1$; (2) Until $k = k_{max}$, repeat the following steps:

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

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

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

Fig. 1. Steps of the basic VNS.

cycling, which might occur if any deterministic rule was used.

As a local optimum within some neighborhood is not necessarily one within another, change of neighborhoods can be performed during the local search phase also. In some cases, as when applying VNS to graph theory, the use of many neighborhoods in the local search is crucial. This local search is then called variable neighborhood descent (VND) and its steps are presented in Fig. 2. (Again, most local search heuristics use in their descents a single or sometimes two neighbourhoods, i.e., $k'_{\max} \leqslant 2$.)

The basic VNS is a descent, first improvement method with randomization. Without much additional effort it could be transformed into a descent–ascent method: in step 2c set also $x \leftarrow x''$ with some probability even if the solution is worse than the incumbent. It could also be changed into a best improvement method: make a move to the best neighborhood k^* among all $k_{\max}$ of them. Of course, the former variant is reminiscent of simulated annealing [42] and the latter of variable depth search [47,55]. Other variants of the basic VNS could be to find solution x' in step 2a as the best among ℓ (a parameter) randomly generated solutions from the kth neighborhood, or to introduce $k_{\min}$ and k_{step}, two parameters that control the change of neighborhood process, i.e., in the previous algorithm instead of $k \leftarrow 1$ set $k \leftarrow k_{\min}$ and instead of $k \leftarrow k + 1$ set $k \leftarrow k + k_{\text{step}}$. Then intensification and diversification of the search is achieved in an easy and natural way: indeed, if $k_{\min}$ and/or k_{step} are set to some fairly large integer values, then the search continues in far away regions of the solution space, i.e., it is diversified; if $k_{\min}$ is small and $k_{\text{step}} = \lfloor \frac{k}{\ell} \rfloor + 1$, where $\lfloor b \rfloor$ is the largest integer not greater than b, then the search spends more time in the region close to the incumbent, i.e., it is intensified. Of course, many other ways to exploit intensification or diversification, including some fairly close ones, have been proposed within the TS research program [23–27].

Further modifications to basic VNS are discussed in Section 2.2. When using more than one neighborhood structure in the search, the following problem specific questions have to be answered:

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

Repeat the following until no improvement is obtained:

(1) Set $k \leftarrow 1$; (2) Until $k = k'_{max}$, repeat the following steps:

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

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

Fig. 2. Steps of the basic VND.

1. what $\mathcal{N}_k$ should be used and how many of them?
2. what should be their order in the search?
3. what strategy should be used in changing neighborhoods?

Similar questions apply to the $\mathcal{N}'_k$. Application of the VNS metaheuristic for solving each particular problem is based on answers to these questions.

2.2. TSP

Given n cities with intercity distances, the TSP is to find a minimum cost tour x (i.e., a permutation of the cities, which minimizes the sum of the n distances between adjacent cities in the tour). It is a classical NP-hard problem.

For a recent, detailed survey and comparison of heuristics for the TSP, see [40]. Probably the most popular heuristic for solving TSP is 2-OPT, where in turn two links between cities in the current tour x are removed and these cities reconnected by adding links in the only other way which gives a tour. Since 2-OPT is a descent local search heuristic, it stops in a local minimum. We first apply the basic VNS rules using 2-OPT (or a quicker variant in which only the edges of moderate length are used) as local search routine.

Let us denote with $\mathcal{S}$ the solution space (i.e., the set of all permutations) of the TSP ($x \in \mathcal{S}$). One may measure the distance ρ between two TSP tours by the cardinality of their symmetric difference: if tours x_1 and x_2 differ in k links, then $\rho(x_1, x_2) = k$. It is clear that ρ is a metric on $\mathcal{S}$. The neighborhoods induced by this metric are

$$\mathcal{N}_k(x) = \{x' | \rho(x, x') = k,\ x \in \mathcal{S}\}, \quad k = 2, \ldots, n, \tag{1}$$

that is the k-OPT neighborhoods [46]. We obtain a parameter-free VNS heuristic by setting the single parameter $k_{\max}$ to n. Then, all ingredients for application of the basic VNS (VNS-1 for short) are given. Another version (VNS-2) that uses a quicker 2-OPT in step 2b works as follows. Instead of deleting and reconnecting all possible pairs of links, the $O(n^2)$ neighborhood is reduced by neglecting the $r\%$ (a parameter) longest links from each city. Let (i, j) be the link to be deleted in the outer loop of 2-OPT. If city j does not belong to the $(1 - r)\%$ cities closest to city i, then the inner loop does not take place. Long links which might initially be included in the tour will be considered for deletion when selecting points in the $\mathcal{N}_k(x)$ and will thus progressively disappear.

In Table 1, average results for random Euclidean problems over 100 trials for $n = 100, \ldots, 500$ and 10 trials for $n = 600, \ldots, 1000$ are reported. Each problem instance is first solved twice by 2-OPT and the best solution value kept. The CPU time spent for these two independent calls of 2-OPT

Table 1
TSP: Average results for random Euclidean problems over 100 trials for $n = 100, \ldots, 500$ and 10 trials for $n = 600, \ldots, 1000$[a]

n	Best value found			% Improvement		CPU times		
	2-OPT	VNS-1	VNS-2	VNS-1	VNS-2	2-OPT	VNS-1	VNS-2
100	825.69	817.55	811.95	0.99	1.66	0.25	0.18	0.17
200	1156.98	1143.19	1132.63	1.19	2.10	3.88	3.21	2.82
300	1409.24	1398.16	1376.76	0.79	2.30	12.12	10.29	9.35
400	1623.60	1602.59	1577.42	1.29	2.84	46.13	40.03	34.37
500	1812.08	1794.59	1756.26	0.96	3.07	110.64	99.57	91.00
600	1991.56	1959.76	1925.51	0.97	3.32	204.60	191.85	173.07
700	2134.86	2120.59	2089.33	0.67	2.13	347.77	307.93	259.06
800	2279.18	2242.11	2190.83	1.63	3.88	539.94	480.50	462.23
900	2547.43	2399.52	2342.01	5.81	8.06	699.33	656.96	624.74
1000	2918.10	2555.56	2483.95	12.42	14.88	891.61	844.88	792.88
Average	1869.87	1803.36	1768.67	2.73	4.43	285.63	263.54	244.97

[a] Computing times in seconds CPU on a SUN SPARC 10, 135.5 Mips (as all other results in this paper).

is then used as the stopping criterion for both VNS variants. The first column of Table 1 gives the size of the problem, the next three, the average objective function values, while the last three columns report the average CPU time when the best solution was found by each method, (i.e., at the last iteration for 2-OPT). In VNS-2, the size of the 2-OPT neighborhood is reduced by taking $r = 40\%$ for $n = 100$, 60% for $n = 200$ and 80% for $n \geqslant 300$. Average improvements in value of 2.73% and 4.43% over the classical 2-OPT heuristic are obtained by VNS-1 and VNS-2, respectively. These improvements are significant when compared with those of other studies. For example, a 0.5% average improvement for the 2.5-OPT heuristic [3] over the 2-OPT heuristic (on random Euclidean instances as well) at the cost of a 30–40% increase in running time is reported in [40]. Improvements due to VNS-2 are more important than those of VNS-1 and occur earlier as shown in the last two columns of the table.

A heuristic for the Euclidean TSP called GENIUS, which is very different from 2-OPT, was developed in [21]. It is a sophisticated insertion followed by local deletion/insertion and correction procedure. The size of the neighborhood in GENIUS depends on a parameter p (the number of cities already in the tour closest to the city that is considered for possible deletion or insertion). We immediately get a set of neighborhood structures for VNS by denoting with $\mathcal{N}_p(x)$ all tours obtained by deletion/insertion with parameter value p. Details can be found in [52], where results on the same type of test problems as reported in [21] are given. VNS gives a 0.75% average improvement over GENIUS within a similar CPU time. Moreover, improvements are obtained for all problem sizes. The GENIUS heuristic was also applied by Gendreau et al. [22] to the TSP with back-hauls. In this problem, customers (or cities) are divided into three disjoint sets: depot, line-haul and back-haul customers. Starting from the depot, a tour must be designed such that all line-haul customers are visited before all back-haul customers. This time VNS gives a 0.40% average improvement over GENIUS with a 30% increase in computing time. Again, improvements are obtained for all problem sizes.

2.3. PM problem

Given a set L of m potential locations for p facilities and a set U of given locations for n users, the PM problem is to locate simultaneously the p facilities in order to minimize the total transportation distance (or costs) from users to facilities, each user being served by the closest facility.

This problem is NP-hard [41] and a central one in discrete location theory [49]. PM can also be expressed as a purely combinatorial problem: let $D = (d_{ij})$ be an $n \times m$ matrix of nonnegative numbers (associated with distances traveled or costs incurred to satisfy the demand of the user $i \in U$ from a facility $j \in L$). PM amounts to choosing p columns of D such that the sum of minima for all lines within these columns is minimum. It may be written concisely as

$$\min_{J} \sum_{i \in U} \min_{j \in J} d_{ij},$$

where $J \subseteq L$ and $|J| = p$. Let us denote with

$$\mathcal{S} = \{x \mid x = \text{set of } p \text{ (out of } m\text{) locations of } L\}$$

the solution space of the problem. The distance between two solutions x_1 and x_2 ($x_1, x_2 \in \mathcal{S}$) is equal to k, if and only if they differ in k locations. Since $\mathcal{S}$ is a set of sets, a (symmetric) distance function ρ can be defined as

$$\rho(x_1, x_2) = |x_1 \setminus x_2| = |x_2 \setminus x_1| \quad \forall x_1, x_2 \in \mathcal{S}. \tag{2}$$

It can easily be checked that ρ is a metric function in $\mathcal{S}$, thus, $\mathcal{S}$ is a metric space. The neighborhood structures $\mathcal{N}_k$ we use are induced by metric ρ, i.e., k locations of facilities ($k \leqslant p$) from the current solution are replaced by k others. Such moves have already been mentioned in the literature as λ-interchange mechanism [53] or k-substitution move [51].

PM is the first problem solved by VNS, in [50]. This work is discussed in [35]. Better results in solving PM by VNS are reported in [33], where an efficient implementation of fast interchange (FI) descent from [66] is developed and used in step 2b of the basic VNS. The same metric ρ is used for both shaking ($k = 1, \ldots, k_{\max}$) and local search ($k = 1$). In order to get a parameter free version of

VNS, we set the value of the single parameter k_{max} to p.

The difficulty of PM depends greatly on the way in which matrix D is obtained. In [59], test problems have values d_{ij} drawn at random from a uniform distribution over an interval $[0, 100]$. Thus, these values do not satisfy the triangle inequality, nor correspond even approximately to distances between points in a low-dimensional space (as with data based on road distances).

A sophisticated TS heuristic is developed in [59], which includes strategic oscillation [23–26]. VNS [36] improved best-known solution for all of the 12 largest test problems of [59]. Moreover, computing times are reduced by a factor of over 6 (for 50 VNS trials), which may be due to several reasons, but probably the most important is the fact that FI is not used in Rolland et al.'s computations. Results are given in Table 2, which also includes values obtained by heuristic concentration (HC) [61,62], supplied to us by the authors [60], and chain substitution TS (CSTS) [51]. It appears that VNS and HC outperform the two TS heuristics. The best-known solution is reached by VNS 11 times with an average error equal to 0.04, while seven times by HC and with an average error of 0.27. It should be noted that the quality of the solution obtained by CSTS depends much on its two parameters, i.e., lengths of the two tabu lists. Since variable length TS did not work well, we fixed the two parameters at 1 and 3, as suggested in [51].

Comparison of other TS implementations with VNS on problems from the ORLib data base gave more balanced results. The basic VNS heuristic is compared in [33] with FI and two recent methods based on the TS approach: TS where the reverse elimination method is used for diversifying the search [65], we denote it by TS-1; TS where the so-called 1-chain-substitution move is used in the search [51], we denote it by TS-2. Results reported are favorable for the basic VNS. For example, for the first 25 test problems from [2] (as results for them only are reported in [65]), all but two test problems are solved exactly by VNS, with a total error of 0.07%; all but three are solved exactly by TS-2, with a total error of 0.57%; all but four are solved exactly by TS-1, with a total error of 0.78%. An explanation for the better performance of VNS over TS, in terms of the effect of pseudo-local optima due to Tabu restrictions is given in [33] and briefly in [35].

2.4. MW problem

The MW problem (also known as continuous location-allocation problem) is the continuous

Table 2
PM: Results for test problems from [59]; maximum time allowed for CSTS [51] and VNS is set to be 30 times those of FI; the best solution found in 50 trials of FI, CSTS and VNS are reported[a]

n	p	Best known	% Error					CPU time			
			FI	VNS	HC	CSTS	TS	FI	VNS	CSTS	TS
200	10	48912	1.21	0.00	0.00	0.02	0.68	5.2	80.3	59.1	381.9
	15	31153	1.29	0.00	0.00	0.00	2.80	7.5	121.2	118.9	401.1
	20	23323	1.96	0.00	0.65	0.00	0.74	9.6	161.6	163.2	416.6
300	10	82664	3.58	0.00	0.00	1.08	0.47	12.3	248.2	179.2	1241.0
	15	52685	4.47	0.00	0.17	0.50	1.98	19.4	373.3	311.8	1321.6
	20	38244	3.34	0.00	1.35	0.72	2.49	24.4	475.8	467.0	1378.3
400	10	123464	0.19	0.00	0.00	0.12	3.79	24.5	463.4	361.3	2910.6
	15	79872	5.20	0.00	0.75	2.50	5.15	32.0	631.5	463.1	3096.8
	20	58459	7.08	0.46	0.00	0.92	1.17	45.4	958.6	653.3	3218.3
500	10	150112	2.03	0.00	0.00	0.14	1.52	40.6	864.9	474.3	9732.2
	15	97624	3.54	0.00	0.00	1.55	0.79	54.7	1164.2	887.2	9731.1
	20	72856	4.86	0.00	0.41	1.46	0.41	74.5	1593.4	1350.0	9748.4
Average			3.23	0.04	0.27	0.75	1.83	29.2	594.7	457.4	3631.5

[a] CPU times for HC method were not available to us.

counterpart of PM: instead of locating the p facilities at some locations of L, they can be located anywhere in the plane. MW can be formulated as follows [45]:

$$\min_{u_i,v_i,z_{ij}} \sum_{i=1}^{p} \sum_{j=1}^{n} z_{ij} \cdot w_j \cdot d_j(u_i, v_i)$$

s.t.

$$\sum_{i=1}^{p} z_{ij} = 1, \quad j = 1, 2, \ldots, n,$$

$$z_{ij} \in [0, 1], \quad i = 1, 2, \ldots, p, \ \ j = 1, 2, \ldots, n,$$

where p facilities must be located to satisfy the demand of n users, u_i, v_i denote the coordinates of the ith facility, d_j the Euclidean distance from (u_i, v_i) to the jth user, w_j the demand (or weight) of the jth user and z_{ij} the fraction of this demand, which is satisfied from the ith facility. It is easy to see that there is always an optimal solution with all $z_{ij} \in \{0, 1\}$, i.e., each user is satisfied from a single facility, which is the (or a) closest one.

An early application of VNS to MW is given in [5]. Several other ones are discussed at length in [6]. It appears that the choice of neighborhoods is crucial. Reassignment of customers to facilities a few at a time is a poor choice, as it entails only marginal changes in the solutions considered. Much better results are obtained when the facilities themselves are moved. As they may be located anywhere in the plane target locations are needed. An easy and efficient choice is locations of customers, where there is no facility as yet. Using this neighborhood structure, several basic TS and VNS heuristics were developed and an extensive empirical study carried out to evaluate various heuristics – old, recent, and new – in a unified setting. The different methods (i.e., Genetic search, three TS variants, four VNS variants, etc.) were compared on the basis of equivalent CPU times. Results of this study indicate that the VNS can be effectively used to obtain superior solutions. For instance on a series of 20 problems with 1060 users the average error (by comparison with best known solution) is of 0.02% only for the best VNS, while it can rise to more than 20% for some well-known heuristics of the literature. Average error for a Genetic algorithm was 1.27% and for the best TS 0.13%.

2.5. MSSC

Consider a set X of n entities in q-dimensional Euclidean space:

$$X = \{x_1, \ldots, x_n\}, \ x_j = (x_{1j}, \ldots, x_{qj}) \in \mathbb{R}^q.$$

The MSSC problem is to find a partition of X into m disjoint subsets (or clusters) C_i, such that the sum of squared distances from each entity to the centroid $\bar{x}_i$ of its cluster is minimum, i.e.,

$$\min \sum_{i=1}^{m} \sum_{\ell: x_\ell \in C_i} \|x_\ell - \bar{x}_i\|^2, \quad \bar{x}_i = \frac{1}{n_i} \sum_{\ell: x_\ell \in C_i} x_\ell, \tag{3}$$

where $\|\cdot\|$ denotes the Euclidean norm and $n_i = |C_i|$ $(n_1 + \cdots + n_m = n)$.

Among many heuristics for MSSC (see [28] or [63] for surveys), the K-MEANS local search algorithm [39,48] is the most popular. Given an initial partition, an entity x_j that belongs to the cluster C_ℓ in the current solution is assigned to some other cluster C_i, $i \neq \ell$. New centroids and objective function values are then updated by using simple formulae; a neighborhood of the current solution is defined by all possible such exchanges (i.e., for all i and j). A move is made if the best solution in the neighborhood is better than the current one. Otherwise, the procedure stops. Another popular heuristic is the so-called H-MEANS algorithm [1], which is very similar to Cooper's alternating heuristic [10] for the MW problem. A new descent local search heuristic, called J-MEANS, is proposed in [34], where the cluster centroid $\bar{x}_i$ is relocated at some entity which does not already coincide with a centroid. Since this move may be a large one and corresponds to n_i reallocations (or n_i K-MEANS moves), we refer to it as a *jump*, hence the name J-MEANS. Obviously, heuristic solutions obtained with the jump neighborhood could be improved by the H-MEANS and/or K-MEANS heuristics. Using them after J-MEANS gives a VND heuristic called J+H+K-MEANS.

Table 3
MSSC: Results for a 1060-entity problem; average and best results in 10 trials; stopping rule: 150 seconds for each trial

m	Best found	% Deviation from best found									
		K-MEANS		H-MEANS		H+K-MEANS		VNS-1		VNS-2	
		Av.	Best	Av.	Best	Av.	Best	Av.	Best	Av.	Best
10	1754840264.9	0.19	0.19	0.01	0.00	0.01	0.00	0.14	0.00	0.14	0.04
20	791925963.7	2.89	0.00	1.87	1.29	1.31	0.46	3.50	1.84	0.74	0.01
30	482302357.1	9.34	6.68	11.77	9.01	8.20	5.79	10.64	5.16	0.86	0.00
40	342844809.0	15.50	12.04	20.01	15.28	16.86	11.53	15.95	9.64	0.81	0.00
50	256892529.0	27.16	24.57	35.60	30.59	28.66	17.14	29.95	13.50	1.42	0.00
60	199151542.6	35.79	32.53	44.59	33.56	36.21	30.00	35.19	20.50	0.59	0.00
70	159781533.1	44.28	33.08	56.63	47.90	47.39	38.89	43.85	25.59	0.78	0.00
80	130038918.6	53.15	46.64	62.34	50.77	56.69	43.98	51.08	35.07	0.75	0.00
90	111322621.7	56.12	48.94	63.94	51.38	54.40	48.38	44.94	28.17	0.73	0.00
100	97352045.7	60.41	54.74	46.21	46.21	35.95	35.95	42.99	28.00	1.16	0.00
110	86287804.2	60.69	52.78	59.92	49.73	41.44	40.79	43.97	23.76	1.34	0.00
120	76380389.5	62.90	54.00	62.32	52.66	48.96	41.28	38.58	33.56	1.02	0.00
130	68417681.6	65.91	50.73	54.66	38.95	42.34	24.64	38.46	26.30	0.67	0.00
140	61727504.5	62.16	49.82	53.05	45.51	36.00	36.00	30.85	22.04	1.43	0.00
150	56679822.6	66.06	55.05	47.82	40.74	33.43	26.88	25.41	20.05	1.34	0.00
160	52210995.2	59.37	53.16	41.74	34.88	30.85	25.61	25.83	19.32	0.70	0.00
Average error		42.62	35.93	41.40	34.28	32.42	26.71	30.08	19.53	0.90	0.00

In Table 3, two variants of VNS [34] are compared with multi-start versions of K-MEANS, H-MEANS and H+K-MEANS local searches based on equivalent CPU time (150 seconds). Both VNS-1 and VNS-2 are parameter free, i.e., $k_{\max} = m$. VNS-1 uses K-MEANS neighborhoods (for step 2a of the basic VNS from Fig. 1) and H+K-MEANS for step 2b. VNS-2 uses relocation neighborhoods and J+H+K-MEANS local search. The large difference in the quality of the solutions obtained by these two VNS versions can be explained by the effectiveness of the relocation (jump) neighborhood structure. A similar difference in efficiency between reallocation and relocation neighborhoods for the MW problem is reported in [6].

2.6. BBLP

Structured global optimization problems, while having several and often many local optima, possess some particular structure, which may be exploited in heuristics or exact algorithms. One such problem, of considerable generality, is the BBLP. This problem has three sets of variables, x, y and z, with cardinalities n_1, n_2 and n_3, respectively. When all variables of y are fixed, it becomes a linear program in x and z. When all variables of z are fixed, it becomes a linear program in x and y. It can be expressed as follows:

$$\begin{aligned} \min \quad & c_0^{\mathrm{T}}x + d_0^{\mathrm{T}}y + e_0^{\mathrm{T}}z + y^{\mathrm{T}}C_0z + c_0 \\ \text{s.t.} \quad & c_i^{\mathrm{T}}x + d_i^{\mathrm{T}}y + e_i^{\mathrm{T}}z \leqslant b_i, \quad i = 1, \ldots, m_1, \\ & c_i^{\mathrm{T}}x + d_i^{\mathrm{T}}y + e_i^{\mathrm{T}}z + y^{\mathrm{T}}C_iz \leqslant b_i, \\ & \qquad\qquad i = m_1 + 1, \ldots, m, \\ & x, y, z \geqslant 0. \end{aligned}$$

The property recalled above suggests the well-known *alternate* heuristic:

1. *Initialization*: Choose values of variables of z (or y).
2. *LP1*: Solve the linear program in (x, y) (or in (x, z)).
3. *LP2*: For y (or z) found in the previous step, solve the linear program in (x, z) (or in (x, y)).
4. If stability is not reached (with given tolerance) return to 2.

Obviously this algorithm may be used in a multi-start framework. To apply VNS one may observe that neighborhoods $\mathcal{N}_k(x, y, z)$ of a solution (x, y, z) are easy to define. They correspond to

k pivots of the linear program in (x, y) or in (x, z), for $k = 1, 2, \ldots, k_{\max}$. One can then apply the basic VNS of Fig. 1 in a straightforward way. The local search routine is the alternate heuristic described above.

Results on randomly generated test problems are presented in Table 4. Test problems have 30–90, 10–50 and 10 x, y and z variables, respectively, 18 linear and 18 quadratic constraints. Coefficients for linear terms in the objective function, and in the linear terms in the constraints, of type $\leqslant$, $\geqslant$ or $=$ are generated uniformly from $(-20, 20)$, $(-10, 20)$, $(-20, 10)$ and $(-10, +10)$, respectively; right-hand sides are generated uniformly from the interval $(-100, 100)$; coefficients for nonlinear terms (i.e., matrices C_i) are generated in two steps: (i) the number of nonlinear terms is generated from integer interval $[1, 5]$; (ii) nonzero elements are chosen at random and corresponding coefficients generated uniformly from $(-0.5, 1)$.

It appears that VNS improves in almost all cases and sometimes very substantially upon the solution provided by the multi-start alternate heuristic.

3. Extensions

While the basic VNS is clearly useful for approximate solution of many combinatorial and global optimization problems, it remains difficult or long to solve very large instances. As often, size of problems considered is limited in practice by the tools available to solve them more than by the needs of potential users of these tools. Hence, improvements appear to be highly desirable. Moreover, when heuristics are applied to really large instances, their strengths and weaknesses become clearly apparent.

Three improvements of the basic VNS are considered in this section. The first, reduced VNS (RVNS) aims at increasing effectiveness (or speed), at the possible cost of an increase in solution value. The second, VNDS, extends basic VNS into a two-level VNS scheme based upon decomposition of the problem. This is shown, for the PM problem, to improve both effectiveness of the heuristic and quality of the solution obtained. The third extension 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 can then degenerate, to some extent, into the multi-start heuristic (which is known not to be very efficient). So, some compensation for distance from the incumbent must be made and a scheme called SVNS is proposed for that purpose.

Table 4
BBLP: Results for 10 repetitions of ALT (MALT) and VNS; each line reports average results on four random test problems with same parameters

Parameters					CPU time		% Error	
n	m	n_1	n_2	n_3	MALT	VNS	MALT	VNS
50	36	30	10	10	0.7	1.0	1.09	0.00
60	36	30	20	10	1.6	2.4	10.05	0.00
70	36	30	30	10	0.9	5.8	20.38	0.00
80	36	30	40	10	1.8	2.7	51.22	0.00
90	36	30	50	10	1.8	3.7	43.77	0.00
100	36	40	50	10	3.6	11.9	18.45	0.00
110	36	50	50	10	3.7	7.3	15.90	0.00
120	36	60	50	10	10.0	22.4	39.12	0.00
130	36	70	50	10	15.0	30.0	34.56	0.00
140	36	80	50	10	8.8	13.9	21.01	0.00
150	36	90	50	10	6.8	10.0	42.87	0.00
Average					4.35	9.19	23.23	0.00

3.1. RVNS

Usually the most time-consuming ingredient of basic VNS is the local search routine which it uses. A drastic change is proposed in RVNS [36]: this routine is disposed of completely. Thus in RVNS, solutions are drawn at random in increasingly far neighborhoods of the incumbent and replace it if and only if they give a better objective function value. Surprisingly, this simple scheme provides good results, in very moderate time.

Results of experiments with large PM problem (i.e., 3038 user example from the ORLib data base) are reported in Table 5. It appears that RVNS provides solutions of equal average value (0.53% above the average value of the best solution obtained by basic VNS) as FI in 18 times less computing time.

So, RVNS appears to be desirable when one aims at obtaining quickly good solutions, which are not necessary very close to the optimum.

3.2. VNDS

VNS is combined with decomposition in [36] where a VNDS heuristic is proposed. This method follows a basic VNS scheme within a successive approximations decomposition method. For a given solution x, all but k attributes (or variables) are fixed in the local search phase. All such possible fixations define a neighborhood $\mathcal{N}_k(x)$. As in VNS, we start with a random solution x' from $\mathcal{N}_1(x)$. But, instead of performing local search in the whole solution space S with x' as a starting point, we solve a one-dimensional problem in the space of the unfixed variable that has been chosen at random. We then return a new value for this variable into the solution and compute (or update) the objective function value. The other steps are the same as in VNS: if the new solution is not better than the incumbent, then we set $k = k + 1$, i.e., we look for improvements in the subspace where all but two variables are fixed, etc., otherwise a move is made and we set $k \leftarrow 1$ again. The steps of the basic VNDS are presented in Fig. 3:

Note that the only difference between the basic VNS and VNDS is in step 2b: instead of applying some local search method in the whole solution space $\mathcal{S}$ (starting from $x' \in \mathcal{N}_k(x)$), in VNDS we solve at each iteration a subproblem in some subspace $V_k \subseteq \mathcal{N}_k(x)$ with $x' \in V_k$. If a given local search heuristic is used for solving this subproblem (in step 2b of VNDS), then VNDS uses a single parameter, $k_{\max}$. However, we can use some better heuristic, such as VNS, for that purpose. Moreover, an additional problem specific parameter, say b, can be considered. Its aim is to strike a balance between the number of subproblems solved and the desired quality of the solution of each subproblem. Parameter b could represent, e.g., the maximum time allowed for solving each

Table 5
Results for PM problem with 3038 users; methods: VNS – basic VNS; FI; RVNS; VNDS

p	Objective values				CPU time			% Error		
	VNS	FI	RVNS	VNDS	FI	RVNS	VNDS	FI	RVNS	VNDS
50	507809.5	510330.2	510216.4	507655.2	612.9	60.7	311.1	0.50	0.47	−0.03
100	354488.7	356005.1	356666.3	353255.2	1040.7	132.4	885.8	0.43	0.61	−0.35
150	281911.9	284159.0	283024.6	281772.1	1459.2	128.5	1432.4	0.80	0.39	−0.05
200	239086.4	240646.2	241355.6	238623.0	1943.6	107.6	1796.8	0.65	0.95	−0.19
250	209718.0	210612.9	210727.7	209343.3	2395.6	150.3	2189.7	0.43	0.48	−0.18
300	188142.3	189467.5	188709.3	187807.1	2583.4	130.6	1471.7	0.70	0.30	−0.18
350	171726.8	172668.5	172388.5	171009.3	2804.3	153.1	2270.3	0.55	0.39	−0.42
400	157910.1	158549.5	158805.0	157079.7	4083.3	158.7	3670.9	0.40	0.57	−0.53
450	146087.8	146727.2	147062.0	145449.0	4223.8	179.5	1652.7	0.44	0.67	−0.44
500	136081.7	136680.5	136665.0	135468.0	4649.3	209.7	2599.8	0.44	0.43	−0.45
Average					2579.6	141.1	1828.12	0.53	0.53	−0.28

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

Repeat the following until the stopping condition is met:

(1) Set $k \leftarrow 1$; (2) Until $k = k_{max}$, repeat the following steps:

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

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

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

Fig. 3. Steps of the basic VNDS.

subproblem, or the maximum size of the subproblem we are willing to solve. If time or size of the subproblem exceeds its limit, then we set $k \leftarrow 1$, i.e., we continue the decomposition by solving smaller subproblems. In this case, we have in fact a recursive, two level, VNS heuristic. Note that it may be worthwhile to consider neighborhoods built from close smaller ones if there is a significant proximity relation between them. This happens in the examples next discussed.

The PM problem has been defined in Section 2.3. Results for the 3038 user problem taken from TSPLib [57] are given in Table 5. Problems are solved by the following heuristics: VNS; FI; RVNS and VNDS. The CPU time FI spends in the search (t) is multiplied by five to get a stopping condition for basic VNS. We also use t as maximum time allowed for VNDS; the parameter b for VNDS is set to 300. In the last three columns of Table 5, errors relative to the solution obtained by VNS are reported. It appears that (i) VNDS outperforms FI within similar CPU times; (ii) VNDS is 0.28% better on average than basic VNS, using five times less CPU times.

3.3. SVNS

Study of the topology of local optima and the corresponding valleys (or mountains) of the objective function of the problem considered is a rich source of information for understanding how heuristics work. Yet, up to now, such studies appear to have been limited to a few problems only (e.g., graph coloring [37], TSP [43] and graph bisection problem [4]). In [32], such a study is conducted for the weighted maximum satisfiability problem of logic. Given m clauses defined on n logical variables (i.e., unions of literals, or variables in direct or negated form) with positive real weights, the problem is to find a truth assignment to the variables, which maximizes the total weight of the satisfied clauses.

It appears from this study that local optima tend to coalesce at the top of large mountains. However, there may be several of them. This poses a problem for VNS: indeed a few iterations will quite quickly determine the best solution in a large region (the top of the highest mountain in that region). But then no better solution will be found near to it. Moreover, solutions in neighborhoods far from the incumbent will necessarily be rather different from it and possibly of poor value. So they will not provide indications as to where are better solutions (higher mountains, if any). If a single descent (or ascent) is made the local optimum found may not be very good either. In other words, when one must move very far from the incumbent solution, basic VNS tends to degenerate into multi-start, which is not among the best metaheuristics. There are many possible ways to overcome this problem. A simple one is to replace in the test for acceptance of a move the objective function value by an evaluation function taking also into account distance from the incumbent. The resulting SVNS method is presented in

Initialization. Select the set of neighborhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$, that will be used in the search; find an initial solution x and its value $f(x)$; set $x_{opt} \leftarrow x$, $f_{opt} \leftarrow f(x)$: choose a stopping condition and a parameter value α;

Repeat the following until the stopping condition is met:
(1) Set $k \leftarrow 1$; (2) Until $k = k_{max}$, repeat the following steps:
(a) *Shaking.* Generate a point x' at random from the k^{th} neighborhood of x ($x' \in \mathcal{N}_k(x)$);
(b) *Local search.* Apply some local search method with x' as initial solution; denote with x'' the so obtained local optimum;
(c) *Improve or not.* If $f(x'') < f_{opt}$ set $f_{opt} \leftarrow f(x)$ and $x_{opt} \leftarrow x''$;
(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. 4. Steps of the SVNS.

Fig. 4 (in case of minimization, as in other figures).

SVNS makes use of a function $\rho(x, x'')$ to measure distance between the incumbent solution x and the local optimum found x''. The distance used to define the $\mathcal{N}_k$, as in the above examples, could be used for this purpose. The parameter α must be chosen in order to accept exploring valleys far from x when $f(x'')$ is larger than $f(x)$ but not much (otherwise one will always leave x). A good value is to be found experimentally in each case. Moreover, in order to avoid frequent moves from x to a close solution, one may take large value for α when $\rho(x, x'')$ is small. More sophisticated choices for a function $\alpha\rho(x, x'')$ could be made through some learning process.

A first comparison of three heuristics, i.e., GRASP, VNS and TS on the *jnh* set of WMAX-SAT instances from [58] is reported in [32]. These instances all have 100 variables, 800 or 900 clauses and integer weights for clauses uniformly drawn at random in the set $\{1, 2, \ldots, 1000\}$. The stopping rule for all heuristics is 30 seconds computing time per instance. Results are summarized in Table 6.

For this set of test problems, VNS and TS perform better than GRASP (which spends much time to get good first local optima and only moderate time to improve on the solutions so found). TS does slightly better than basic VNS.

Experiments were also done to compare VNS and TS on much larger test problems, with 500 variables and 4000–4500 clauses. A variant where only *low neighborhoods*, i.e., the five directions with smallest increase of $f(x)$ were used did not provide better results than basic VNS. However, when using low neighbourhoods in SVNS, the best results were obtained. They are summarized, for a series of instances with 500 variables, 4500 clauses and 3 variables per clause in Table 7. SVNS gives the best solution known for 23 problems out of 25 vs 5 out of 25 for TS. Consequently, the average error is much smaller for SVNS than for TS.

4. New applications

When a metaheuristic is very efficient and effective, or when its basic principle is sufficiently new and simple, it may suggest not only new

Table 6
Results for *jnh* test problems for WMAXSAT ($n = 100$)

	GRASP	VNS	TS
Number of instances solved exactly	6	42	44
% Average error in 10 trials	0.0678	0.0189	0.0134
% Best error in 10 trials	0.0274	0.0001	0.0000
Total number of instances	44	44	44

Table 7
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

applications but new *types* of applications; this is the case for VNS. In this section, we describe two such types of applications: stabilized column generation, which requires finding quickly near-optimal solution to large instances of combinatorial problems to work well, and the system AutoGraphiX (AGX), which implements application of VNS to graph theory (and not only to solving optimization problems on graphs).

4.1. Stabilized column generation

Consider a feasible and bounded linear program P and its dual D:

$$(\mathrm{P}) \quad \min \quad c^{\mathrm{T}}x \quad \text{s.t.} \quad Ax = b,\ x \geqslant 0,$$

$$(\mathrm{D}) \quad \max \quad b^{\mathrm{T}}\pi \quad \text{s.t.} \quad A^{\mathrm{T}}\pi \leqslant c,\ \pi \geqslant 0.$$

When P has many more variables than D, it is convenient to solve it by column generation. Convergence, however, is often slow. Such undesirable behavior is also observed when one tries to prove optimality of a degenerate solution of P.

In [12], a stabilization scheme is proposed which remains entirely within the linear programming framework. To this effect, it merges a perturbation and an exact penalty method.

Define a primal problem $\tilde{\mathrm{P}}$ and its dual $\tilde{\mathrm{D}}$ as follows:

$$(\tilde{\mathrm{P}}) \quad \min \quad c^{\mathrm{T}}\tilde{x} - \delta_- y_- + \delta_+ y_+ \quad \text{s.t.} \quad A\tilde{x} - y_- + y_+ = b,\ y_- \leqslant \varepsilon_-,\ y_+ \leqslant \varepsilon_+,\ \tilde{x}, y_-, y_+ \geqslant 0,$$

$$(\tilde{\mathrm{D}}) \quad \max \quad b^{\mathrm{T}}\tilde{\pi} - \varepsilon_- w_- - \varepsilon_+ w_+ \quad \text{s.t.} \quad A^{\mathrm{T}}\tilde{\pi} \leqslant c,\ -\tilde{\pi} - w_- \leqslant -\delta_-,\ \tilde{\pi} - w_+ \leqslant \leqslant \delta_+,\ \tilde{\pi}, w_-, w_+ \geqslant 0.$$

In the primal, y_- and y_+ are vectors of slack and surplus variables with upper bounds ε_- and ε_+, respectively. These variables are penalized in the objective function by vectors δ_- and δ_+, respectively. In the dual, this amounts to penalizing dual variables $\tilde{\pi}$ when they lie outside the range $[\delta_-, \delta_+]$. When applying column generation, after finding a column with negative reduced costs, the parameters δ and ε are updated. This is done following problem-specific rules.

For stabilized column generation to work well, it is essential that good estimates of dual variables be obtained at the outset. For the PM, MW and MSSC problems, such estimates may be obtained by solving the primal of large instances with VNS or VNDS and estimating, from sensitivity analysis, lower and upper bounds on the dual variables by the formulae

$$\delta_-^0 = f(C_i) - f(C_i \setminus \{k\})$$

and

$$\delta_+^0 = \min_{j \neq i} f(C_j) \cup \{k\}) f(C_j),$$

where i is the index of the set C_i in the best partition found to which k belongs and $f(C_i)$ the contribution of this set to the objective function value. Combining stabilized column generation with branch-and-bound, MW problems with up to 1060 users could be solved exactly [31,44] (vs 30 for the largest instances solved exactly in the

literature when $p \geqslant 3$). Moreover, the continuous relaxation of PM problems with up to 3038 users could also be solved (work on combining this with branch-and-bound is under way) [64]. The MSSC proved to be more difficult. The bottleneck is the column generation subproblem, a hyperbolic program in 0–1 variables with quadratic numerator and linear denominator, solved by a sequence of unconstrained quadratic 0–1 programs. Nevertheless, several problems from the literature, including the famous 150 iris of Fisher [20], could be solved exactly for the first time [13].

4.2. VNS for extremal graphs

VNS can be helpful in solving approximately numerous optimization problems defined on graphs. But it can also prove useful in the study of graph theory per se.

In [9], conjectures in graph theory are viewed as combinatorial optimization problems on an infinite family of graphs (of which only those of moderate size will be explored). Then VNS can be applied. This leads to an automated system, called AGX whose aim is to suggest, disprove or give hints on how to prove conjectures in graph theory.

Assume an objective function $f(G)$ depending on graph invariants $i_1(G), i_2(G), \ldots, i_l(G)$ is given (recall that invariants are quantities independent of edges or vertices labeling e.g., order, size, radius, diameter, chromatic number, maximum or minimum degree...). Let $\mathscr{G}_n$ denote the family of all graphs G with n vertices (a parameter). The problem considered is then

$$\min_{G \in \mathscr{G}_n} (\max)\ f(G) \tag{4}$$

to which we apply VNS specialized to extremal graphs problems.

AGX comprises a VND component with many neighborhoods $\mathscr{N}'_1(G), \mathscr{N}'_2(G), \ldots, \mathscr{N}'_{k\max'}(G)$. In the present version, there are 10 of them, defined by the following transformations on G: *deletion* of an edge; *addition* of an edge between nonadjacent vertices; *move* of an edge, i.e., deletion followed by addition, but not in the same position (these three neighborhoods are used in most applications); *detour*: removal of an edge and addition of two edges between endpoints of the deleted one and a vertex not adjacent to either of these endpoints; *short cut*, i.e., the reverse operation than detour; *2 opt*: remove two nonadjacent edges and add two different nonadjacent edges connecting the endpoints of the removed ones; *insert pending vertex*: remove the edge of a pending vertex, add edges between it and two adjacent vertices and remove the edge between them; *add pending vertex*: add an edge from one vertex to a new one; *delete vertex* of bounded degree and all edges adjacent to it, *connection*: delete two nonadjacent edges and add two adjacent ones.

It is up to the user to apply all transformations in turn or to select a subset of them, e.g. the less time-consuming ones. VNS itself uses a set of nested neighborhoods $\mathscr{N}_1(G), \mathscr{N}_2(G), \ldots$ defined by the cardinality of the symmetric difference between the edge sets of two graphs $G = (V, E)$ and $G' = (V', E')$

$$\rho(G, G') = \left|(E \setminus E') \cup (E' \setminus E)\right|, \tag{5}$$

i.e., $\rho(G, G') = k$ if there are exactly k pairs of vertices adjacent in G and not in G' or adjacent in G' and not in G. Then

$$G' \in \mathscr{N}_k(G) \Longleftrightarrow \rho(G, G') = k. \tag{6}$$

In addition to these components, which follow the basic schemes of Figs. 1 and 2, AGX comprises a routine for graph analysis which performs the following: (i) recognition of the class to which belongs the extremal graph G obtained (if it is one of the series of well-known classes such as path, star, tree, complete or bipartite graph, ...); (ii) computation of various invariants for G; (iii) visualization of G on the screen using X-Windows, with interactive modifications of position of vertices, (iv) subroutines for interactive modifications of G by deletion or addition of edges or vertices. Moreover, AGX also has a routine for parametric analysis which obtains, stores, analyzes and represents values of invariants, and of functions corresponding to conjectures.

VNS can be used in graph theory for several purposes:

(a) *Find a graph satisfying given constraints.* Let $i_1(G), i_2(G), \ldots, i_l(G)$ denote l invariants of G and $p_1, p_2, \ldots, p_l$ values given to them. Consider the problem

$$\min_{G \in \mathscr{G}_n} f(G) = \sum_{k=1}^{l} |i_k(G) - p_k|. \tag{7}$$

Any graph such that $f(G) = 0$, satisfies these constraints. Note that constraints involving formulae on several invariants can be treated in a similar way. Expression (7) shows constrained problems can be reduced to unconstrained ones.

(b) *Find optimal or near optimal values for an invariant subject to constraints.* Let $i_0(G)$ denote the objective function invariant and assume constraints expressed as above. Consider the problem

$$\min_{G \in \mathscr{G}_n} f(G) = i_0(G) + M \sum_{k=1}^{l} |i_k(G) - p_k|, \tag{8}$$

where M is a constant sufficiently large to ensure that for any pair of graphs G, $G' \in \mathscr{G}_n$ such that $\sum_{k=1}^{l} |i_k(G) - p_k| = 0$ and $\sum_{k=1}^{l} |i_k(G') - p_k| > 0$, $f(G) < f(G')$. Maximum values are tackled in a similar way.

Note that some invariants take integer values and remain constant on large plateaus when G is modified a little at a time, i.e., replaced sequentially by a graph in the neighborhood $\mathscr{N}(G)$. It is then convenient to modify the objective function by adding a secondary objective, with a coefficient sufficiently small not to change the ordering of graphs according to the first objective, and orienting the transformations in the right direction.

(c) *Refute a conjecture.* Consider a conjecture $h(G) \leqslant g(G)$ where $h(G)$ and $g(G)$ are formulae depending on one or more invariants of G. It corresponds to the problem

$$\min_{G \in \mathscr{G}_n} f(G) = g(G) - h(G). \tag{9}$$

If a graph G for which $f(G) < 0$ is found, the conjecture is refuted.

(d) *Suggest a conjecture (or sharpen one).* This can be done in various ways, which usually use parameterization on n or other invariants $i_1(G), i_2(G), i_3(G), \ldots, i_l(G)$. For instance consider

$$\min_{G \in \mathscr{G}_n} f(G) = i_2(G) - i_1(G). \tag{10}$$

If no graph G with $f(G) < 0$ is found, then this suggests $i_1(G) \leqslant i_2(G)$. If the extremal graphs found belong to a recognizable class, then it may be possible to deduce a more precise inequality in $i_1(G)$, $i_2(G)$ and n.

(e) *Suggest a proof.* The way the extremal graphs are obtained, e.g., what transformations of G are used, may suggest strategies to prove conjectures for all graphs or for some classes of them.

The *Graffiti* system of Fajtlowicz [15–19] has been used to generate hundreds of conjectures in graph theory which have been studied by many mathematicians, some of them proved and more disproved. The following conjecture is a reformulation of one of them (i.e., # 834):

$$\delta \bar{l}(G) \leqslant n, \tag{11}$$

where δ denotes the minimum degree and $\bar{l}$ the average distance between pairs of distinct vertices of G. This conjecture has been refuted by AGX [9] in 153 seconds of CPU time using an edge addition, 15 edge deletions, a split, two mergings, six moves, three 2-OPTS, another merging and split after VNS with neighborhoods of size up to seven, and a final edge deletion after another application of VNS with neighborhoods of size up to seven. Two other conjectures of *Graffiti* are also refuted in [9]. Recently five more conjectures of Graffiti could be refuted by AGX in conjunction with a program to generate cubic graphs [7].

The *energy* $E(G)$ of a graph G is the sum of absolute values of the eigenvalues of G's adjacency matrix. It has an important interpretation in chemistry, in connection with Huckel's molecular orbital theory [29]. A systematic study of $E(G)$, using AGX, has been initiated in [8]. Curves of (presumably) minimum and maximum values of $E(G)$ as functions of numbers n of vertices and m of edges of the graph considered were systematically obtained (in several days of computing time). While extremal graph were not always obtained, and sometimes better ones could be found interactively, graphs corresponding to local minima or linear portions of the curves were easily recog-

nized. AGX then led to several conjectures, including the following:

$$2\sqrt{m} \leqslant E \leqslant 2m,$$

which despite its simplicity does not appear to have been proposed before. These bounds could then be proved, and shown to be tight for complete bipartite graphs and for disjoint edges plus additional isolated vertices, respectively.

Moreover, several conjectures related to various other graph invariants (chromatic number, independence number,...) were obtained and some of them proved.

5. Conclusions

When evaluating metaheuristics, it appears that several, possibly conflicting, criteria should be taken into account. Desirable properties of metaheuristics include

1. *Simplicity*: The metaheuristic should be based on a simple and clear principle which should be widely applicable.
2. *Coherence*: The various steps of heuristics for particular problems should follow naturally from the principle of the metaheuristic.
3. *Efficiency*: Heuristics for particular problem should be efficient, i.e., provide optimal or near-optimal solutions for all or at least most realistic instances. Preferably, the heuristics should solve optimally most problems of benchmarks, when available.
4. *Effectiveness*: Heuristics for particular problems should provide good, perhaps optimal or near-optimal, solutions in moderate CPU time.
5. *Robustness*: Heuristics for various problems should be effective and efficient, and for each of these problems should give good solutions for a variety of instances, i.e., not just be fine-tuned to some training sets and less good elsewhere.
6. *User-friendliness*: Heuristics should be well defined, easy to understand and, most important, easy to use. This implies they should have few, ideally no, parameters.
7. *Innovation*: Preferably the principle of the metaheuristic and/or efficiency and effectiveness of heuristics derived from it should lead to new types of applications.

While VNS is still in its infancy, results for a variety of problems described in this paper allow a first assessment. Clearly, VNS is based on a simple principle, i.e., systematic change of neighborhood within the search. This principle, explained in the basic schemes of Figs. 1 and 2 is largely applicable and often in an easy way when a local search heuristic is available. (This simplicity implies that isolated steps of several heuristics or metaheuristics are particular cases or come close to some steps of VNS; their derivation often follows, when it is explicit, from different principles.) While VNS heuristics use several and sometimes many neighborhoods, their various steps stem from a common principle, i.e., hybridation is avoided and VNS is coherent. Experiments with benchmarks of large PM and MW problems show VNS is very efficient: it solves many instances exactly and otherwise gives solutions very close to the optimum. For several other problems VNS appears to perform better than recent heuristics developed by leading researchers. Moreover, VNS is effective: the best solutions are obtained in moderate computing time. This is particularly true when using VNDS. For instance, time to find the best solution of very large instances of PM problem is less than that for a single descent with the fast interchange heuristic. Computational results on simulated and real data show VNS is robust, i.e., about equally efficient and effective for both types of problems. Moreover, VNS is very user-friendly. The basic steps are easy to understand and apply and versions without parameters (except for total CPU time allocated) are not hard to obtain, while remaining efficient and effective. Finally, VNS is innovative: on the one hand, efficiency and effectiveness of VNS heuristics for PM, MW, MSSC, and other problems are an indispensable ingredient in stabilized column generation algorithms for solving those problems exactly; and on the other hand, the basic principle of VNS naturally suggest the approach to graph theory developed in AGX.

Developments of VNS first focussed on applications of its basic principle to combinatorial optimization as well as to a few global optimization problems. More recently, several extensions were

proposed. RVNS aims at solving very rapidly, if approximately, very large instances. VNDS aims at increasing precision and at reducing solution time for decomposable problems. This avenue deserves more research, particularly as there are many ways to decompose a problem. SVNS aims at making efficient the search for better solutions far from the incumbent one. It is a first step towards solution of a problem crucial to obtention of optimal or very close to optimal solution for large instances. Neighborhoods used in VND also strongly influence the efficiency of VNS, as e.g., for the MW problem. Their nature and properties should be investigated in more detail.

Some comparisons of VNS with other metaheuristics have been made and clearly there is room for many more. Hybrid heuristics have not been studied up to now. While they appear to be promising in terms of finding better solutions, particularly for large instances of difficult problems, they would also imply a decrease in simplicity and consequently in the understanding of the heuristics behavior. The same appears to be true for the introduction of additional parameters to guide the search more efficiently either in the descent or in the diversification phase.

A first study of valleys (or mountains) of the objective function and their exploration by VNS has been conducted for the weighted maximum satisfiability problem. It led to the development of an efficient SVNS heuristic. Much more work on the topology of local optima and valleys appears to be desirable. VNS which does not alter the shape of valleys explored could prove to be a very good tool for that purpose. It might thus help to address the important and largely open theoretical question "Why do metaheuristics work as well as they do?"

Acknowledgements

Research of the authors was supported by ONR grant N00014-95-1-0917, NSERC grant OGPOO 39682 and FCAR grant 32EQ 1048. We thank Jack Brimberg, Gilles Caporossi, Michel Gendreau, Fred Glover, Brigitte Jaumard, Anderson Parreira and Dionisio Perez-Brito for discussions of VNS, Dragoš Cvetković, Ivan Gutman, Paul Seymour and Bjarne Toft for discussions of AGX and David Schilling for providing test problems for the PM problem.

References

[1] M.R. Anderberg, Cluster Analysis for Application, Academic Press, New York, 1973.
[2] J.E. Beasley, A note on solving large *p*-median problems, European Journal of Operational Research 21 (1985) 270–273.
[3] J.L. Bentley, Fast algorithms for geometric traveling salesman problem, ORSA Journal on Computing 4 (1992) 387–411.
[4] K.D. Boese, A.B. Kahng, S. Muddu, A new adaptive multi-start technique for combinatorial global optimizations, Operational Research Letters 16 (1994) 101–113.
[5] J. Brimberg, N. Mladenović, A variable neighborhood algorithm for solving the continuous location-allocation problem, Studies in Location Analysis 10 (1996) 1–12.
[6] J. Brimberg, P. Hansen, N. Mladenović, É. Taillard, Improvements and comparison of heuristics for solving the multisource Weber problem, Operations Research 48 (3) (2000).
[7] G. Brinkmann, Fast generation of cubic graphs, Journal of Graph Theory 23 (2) (1996) 139–149.
[8] G. Caporossi, D. Cvetković, I. Gutman, P. Hansen, Variable neighborhood search for chemical graphs, Part 2, Graphs with extremal energy, Journal of Chemical Information and Computer Sciences 39 (1999) 984–996.
[9] G. Caporossi, P. Hansen, Variable neighborhood search for extremal graphs, 1. The AutoGraphix system, Discrete Mathematics 212 (2000) 29–44.
[10] L. Cooper, Location–allocation problems, Operational Research 11 (1963) 331–343.
[11] M. Dorigo, V. Maniezzo, A. Colorni, The ant system: Optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics – Part B 26 (1) (1996) 29–41.
[12] O. du Merle, D. Villeneuve, J. Desrosiers, P. Hansen, Stabilized column generation, Discrete Applied Mathematics 194 (1999) 229–237.
[13] O. du Merle, P. Hansen, B. Jaumard, N. Mladenović, An interior point algorithm for minimum sum-of-squares clustering, SIAM Journal on Scientific Computing 21 (2000) 1485–1505.
[14] T. Feo, M. Resende, Greedy randomized adaptive search, Journal of Global Optimization 6 (1995) 109–133.
[15] S. Fajtlowicz, On conjectures of Graffiti, Discrete Mathematics 72 (1987) 113–118.
[16] S. Fajtlowicz, On conjectures of Graffiti-II, Congressus Numerantium 60 (1987) 187–197.
[17] S. Fajtlowicz, On conjectures of Graffiti-III, Congressus Numerantium 66 (1988) 23–32.

[18] S. Fajtlowicz, On conjectures of Graffiti-IV, Congressus Numerantium 70 (1990) 231–240.
[19] S. Fajtlowicz, On conjectures of Graffiti-V, Seventh International Quadrennial Conference on Graph Theory 1 (1995) 367–376.
[20] R.A. Fisher, The use of multiple measurements in taxonomic problems, in Annual Eugenics VII, Part II, 1936, pp. 179–188.
[21] M. Gendreau, A. Hertz, G. Laporte, New insertion and postoptimization procedures for the traveling salesman problem, Operational Research 40 (1992) 1086–1094 1992.
[22] M. Gendreau, A. Hertz, G. Laporte, The traveling salesman problem with back-hauls, Computers and Operational Research 23 (1996) 501–508.
[23] F. Glover, Tabu search – Part II, ORSA Jounal of Computing 1 (1989) 190–206.
[24] F. Glover, Tabu search – Part II, ORSA Journal of Computing 2 (1990) 4–32.
[25] F. Glover, M. Laguna, Tabu search, in: C. Reeves (Ed.), Modern Heuristic Techniques for Combinatorial Optimization, Blackwell, Oxford, 1993, pp. 70–150.
[26] F. Glover, Tabu thresholding: Improved search by nonmonotonic trajectories, ORSA Journal of Computing 7 (1995) 426–442.
[27] F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, Boston, MA, 1997.
[28] A.D. Gordon, Classification: Methods for the Exploratory Analysis of Multivariate Sata, Chapman & Hall, New York, 1981.
[29] I. Gutman, O.E. Polansky, Mathematical Concepts in Organic Chemistry, Springer, Berlin, 1986.
[30] P. Hansen, B. Jaumard, Algorithms for the maximum satisfiability problem, Computing 44 (1990) 279–303.
[31] P. Hansen, B. Jaumard, S. Krau, O. du Merle, A stabilized column generation algorithm for the multisource Weber problem (in preparation).
[32] P. Hansen, B. Jaumard, N. Mladenović, A. Parreira, Variable neighborhood search for weighted maximum satisfiability (in preparation).
[33] P. Hansen, N. Mladenović, Variable neighborhood search for the *p*-median, Location Science 5 (4) (1998) 207–226.
[34] P. Hansen, N. Mladenović, J-MEANS, a new local search heuristic for minimum sum-of-squares clustering, Les Cahiers du GERAD G-99-14 and Pattern Recognition, forthcoming.
[35] P. Hansen, N. Mladenović, An introduction to variable neighbourhood search. in: S. Voss et al. (Eds.), Metaheuristics, Advances and Trends in Local Search Paradigms for Optimization, Kluwer Academic Publishers, Dordrecht, 1999, pp. 433–458.
[36] P. Hansen, N. Mladenović, D. Perez-Brito, Variable Neighborhood Decomposition Search, Journal of Heuristics, forthcoming.
[37] A. Hertz, B. Jaumard, M. Poggi de Aragao, Local optima topology for the *k*-coloring problem, Discrete Applied Mathematics 49 (1994) 257–280.
[38] J.H. Holland, Adaptation in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, MI, 1975.
[39] R.C. Jancey, Multidimensional Group Analysis, Australian Journal of Botany 14 (1966) 127–130.
[40] D.S. Johnson, L.A. McGeoch, The traveling salesman problem: a case study in local optimization, in: E.H.L. Aarts, J.K. Lenstra (Eds.), Local Search in Combinatorial Optimization, Wiley, London, 1997, pp. 215–310.
[41] O. Kariv, S.L. Hakimi, An algorithmic approach to network location problems; Part 2, the *p*-medians, SIAM Journal on Applied Mathematics 37 (1969) 539–560.
[42] S. Kirkpatrick, C.D. Gelatt, M. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[43] S. Kirkpatrick, G. Toulouse, Configuration space analysis of traveling salesman problems, Journal de Physique 46 (1985) 1277–1292.
[44] S. Krau, Extensions du problème de Weber, Ph D. Thesis, École Polytechnique de Montréal (under direction of P. Hansen and B. Jaumard), 1997.
[45] R.E. Kuenne, R.M. Soland, Exact and approximate solutions to the multisource Weber problem, Mathematical Programming 3 (1972) 193–209.
[46] S. Lin, Computer solutions of the traveling salesman problem, Bell Systems Technical Journal 44 (1965) 2245–2269 1965.
[47] S. Lin, B.W. Kernighan, An effective heuristic algorithm for the traveling salesman problem, Operational Research 21 (1973) 498–516 1973.
[48] J.B. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability 1 (1967) 281–297.
[49] P. Mirchandani, R. Francis (Eds.), Discrete Location Theory, Wiley, New York, 1990.
[50] N. Mladenović, A variable neighborhood algorithm: A new metaheuristic for combinatorial optimization, Abstracts of papers presented at Optimization Days, Montréal, 1995, p. 112.
[51] N. Mladenović, J.P. Moreno, J. Moreno-Vega, A chain-interchange heuristic method, Yugoslav Journal of Operational Research 6 (1) (1996) 41–54.
[52] N. Mladenović, P. Hansen, Variable neighborhood search, Computers and Operations Research 24 (1997) 1097–1100 1997.
[53] I.H. Osman, N. Christofides, Capacitated clustering problems by hybrid simulated annealing and Tabu search, International Transactions on Operational Research 1 (3) (1994) 317–336.
[54] I.H. Osman, G. Laporte, Metaheuristics: A bibliography, Annals of Operational Research 63 (1996) 513–628.
[55] C.H. Papadimitriou, K. Steiglitz, Combinatorial Optimization, Algorithms and Complexity, Prentice-Hall, Englewood Cliffs, NJ, 1982.
[56] C. Reeves (Ed.), Modern Heuristic Techniques for Combinatorial Problems, Blackwell, Oxford, 1993.

[57] G. Reinelt, TSLIB – a traveling salesman library, ORSA Journal of Computing 3 (1991) 376–384.
[58] M.G.C. Resende, L.S. Pitsoulis, P.M. Pardalos, Approximate solution of weighted MAX-SAT problems using GRASP, in: D. Du, J. Gu, P.M. Pardalos, (Eds.), Satisfiability Problem: Theory and Applications, DIMACS Series in Discrete Mathematics and Theoretical Computer Science 35, American Mathematical Society, Providence, RI, 1997.
[59] E. Rolland, D.A. Schilling, J.R. Current, An efficient Tabu search procedure for the *p*-median problem, European Journal of Operational Research 96 (1996) 329–342.
[60] K.E. Rosing, Private communication.
[61] K.E. Rosing, C.S. ReVelle, Heuristic concentration: Two stage solution construction, European Journal of Operational Research 97 (1997) 75–86 1997.
[62] K.E. Rosing, C.S. ReVelle, E. Rolland, D.A. Schilling, J.R. Current, Heuristic concentration and Tabu search: A head to head comparison, European Journal of Operational Research 104 (1998) 93–99.
[63] H. Späth, Cluster Dissection and Analysis (Theory, Fortran Programs, Examples), Ellis Horwood, Chichester, 1985.
[64] N. Thabet, Des algorithmes de génération de colonnes pour le problème de la *p*-mediane, Master thesis, École des HEC (under direction of P. Hansen), 1998.
[65] S. Voss, A reverse elimination approach for the *p*-median problem, Studies in Locational Analysis 8 (1996) 49–58.
[66] R. Whitaker, A fast algorithm for the greedy interchange for large-scale clustering and median location problems, INFOR 21 (1863) 95–108.