Pergamon

PII: S0305-0548(97)00031-2

Computers Ops Res. Vol. 24, No. 11, pp. 1097–1100, 1997
© 1997 Elsevier Science Ltd
All rights reserved. Printed in Great Britain
0305-0548/97 $17.00+0.00

VARIABLE NEIGHBORHOOD SEARCH

N. Mladenović‡ and P. Hansen†§

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

(Received December 1996; in revised form April 1997)

Scope and Purpose—In the last decade much progress was made in the design and application of heuristic algorithms for a large variety of combinatorial and nonconvex continuous optimization problems. General heuristics (or metaheuristics, e.g. simulated annealing, tabu search, genetic search) which avoid being trapped in the first local optimum found have led to much improved results in many practical contexts. However, the sophistication of such heuristics makes it difficult to pinpoint the reasons for their effectiveness. We examine a relatively unexplored reason: change of neighborhood in the search. Using this idea and very little more, i.e., only a local search routine, leads to a new metaheuristic, which is widely applicable. First computational results show this scheme outperforms other heuristics for several combinatorial optimization problems. Its effectiveness is illustrated on the traveling salesman problem without and with backhauls, which have a large number of practical applications such as the drilling of printed circuit boards, automated warehouse routing and operation sequencing on numerically controlled machines.

Abstract—Systematic change of neighborhood within a local search algorithm yields a simple and effective metaheuristic for combinatorial optimization. We present a basic scheme for this purpose which can be implemented easily using any local search algorithm as a subroutine. Its effectiveness is illustrated by improvements in the GENIUS algorithm for the traveling salesman problem [1], without and with backhauls [2]. © 1997 Elsevier Science Ltd

1. INTRODUCTION

Local search methods for combinatorial optimization proceed by performing a sequence of local changes in an initial solution which improve each time the value of the objective function until a local optimum is found. That is, at each iteration an improved solution x' in the neighborhood $\mathcal{N}(x)$ of the current solution x is obtained, until no further improvements are found. In recent years, several general heuristics (or metaheuristics) have been proposed which extend this scheme in various ways and avoid being trapped in local optima with a poor value (see Reeves [3] for a book-length survey and Osman and Laporte [4] for an extensive bibliography). The purpose of this note is to show that a simple and effective metaheuristic may be obtained by proceeding to a systematic change of neighborhood within a local search algorithm. We call this approach Variable Neighborhood Search (VNS). Contrary to most other local search methods VNS does not follow a trajectory, but explores increasingly distant neighborhoods of the current incumbent solution, and jumps from there to a new one if and only if an improvement was made. In this way often favorable characteristics of the incumbent solution, e.g. that most 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.

2. VNS ALGORITHM

Let us denote a finite set of pre-selected neighborhood structures with $\mathcal{N}_k$, $(k=1,\ldots,k_{\max})$, and with $\mathcal{N}_k(x)$ the set of solutions in the k^{th} neighborhood of x. Local search heuristics usually use one neighborhood structure, i.e., $k_{\max}=1$. When using more than one, the following questions have to be answered.

(i) What $\mathcal{N}_k$ should be used and how many of them?;
(ii) What should be their order in the search?;

† To whom all correspondence should be addressed.
‡ Nenad Mladenović received his Ph.D. from the University of Belgrade, Yugoslavia. Currently, he is at GERAD, where he has come for a visit from University of Belgrade. His research interests include nonlinear and combinatorial optimization location, and clustering.
§ Pierre Hansen received the Agrégation de l'Enseignement Supérieur degree from Brussels University. He is currently director of GERAD research center and Professor of Operations Research at Ecole des Hautes Etudes Commerciales, Montreal. His research interests include global and combinatorial optimization, graphs, location, clustering and mathematical chemistry.

(iii) What search strategy should be used in changing neighborhoods?

A straightforward answer to these questions is given by the rules of the following basic VNS algorithm.

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;

Main step. (1) Set $k:=1$. (2) Until $k=k_{\max}$, repeat the following steps: (a) generate a point x' at random from the k^{th} neighborhood of x ($x' \in \mathcal{N}_k(x)>$); (b) apply some local search method with x' as the initial solution; denote with x'' the obtained local optimum; (c) if the solution thus obtained is better than the incumbent, move there ($x:=x''$), and continue the search with $\mathcal{N}_1$ ($k:=1$); otherwise, set $k:=k+1$;

The main step can possibly be iterated until some other stopping condition is met (e.g. maximum number of iterations, maximum CPU time allowed, or maximum number of iterations between two improvements). Often successive neighborhoods $\mathcal{N}_k$ will be nested. Note that point x' is generated at random in Step 2(a) in order to avoid cycling, which might occur if any deterministic rule was used.

It is worth stressing the ease of implementation of both the basic version of VNS (with only one parameter $k_{\max}$) and various extensions inspired by tabu search [5–7], and other metaheuristics. Step 2(a) is easy to program. For example, if $\mathcal{N}_k$ is obtained by k-interchanges of solution attributes, one need only add a few lines (Step 2(a)) in an existing code for a local search method (Step 2(b)). The basic VNS is in fact a descent, first improvement method. Without much additional effort it could be transformed into a descent-ascent method (in Step 2(c) set $x:=x'$ even if improvement was not reached) and/or a best improvement method (make a move to the best neighborhood k^* among all $k_{\max}$ of them). Other variants of the basic VNS could be:

(i) find solution x' in Step 2(a) as the best among b (a parameter) randomly generated solutions from the k^{th} neighborhood;

(ii) introduce k_1 and k_{step}, two parameters that control the change of neighborhood process, i.e., in the previous algorithm instead of $k:=1$ set $k:=k_1$ and instead of $k:=k+1$ set $k:=k+k_{\text{step}}$. Then intensification and diversification of the search is achieved in an easy and natural way. Indeed, if k_1 and/or k_{step} are set to some large integer ($\leq n$), then the search continues in far away regions of the solution space, i.e., it is diversified; if k_1 and $k_{\text{step}} = \left\lfloor \frac{k \bmod a}{a-1} \right\rfloor$ where a is a small integer and $\lfloor b \rfloor$ 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. (Note that values of k_1 and k_{step} could be changed at random in each iteration; for example choose them randomly from $[1, \max 1, k_{\max}/4]$);

(iii) generally speaking, the neighborhood used for local search in Step 2(b) is independent of the neighborhoods $\mathcal{N}_k$, $k=1,\ldots, k_{\max}$, selected in Step 2(a). In the basic VNS, descent local searches (Step 2(b)) always use the same neighborhood structure, which does not necessarily belong to $\mathcal{N}=\{\mathcal{N}_1,\ldots, \mathcal{N}_{k_{\max}}\}$. This is done to point out the simplicity of deriving a VNS algorithm when some local search routine is available. However, an extended version of VNS could contain more than one neighborhood in Step 2(b) as well. In that case, many different search strategies in changing the neighborhood structures in both Step 2(a) and Step 2(b) could be tried out, among which the first improvement one is the simplest.

3. TRAVELING SALESMAN PROBLEM (TSP)

Given n cities with intercity distances, the traveling salesman problem (TSP) seeks a minimum cost tour (i.e., a permutation of the cities which minimizes the sum of the n distances between adjacent cities in the tour). We try our basic VNS with the GENIUS algorithm recently developed by Gendreau *et al.* [1].

GENIUS consists of a tour construction phase (GENI, for generalized insertion), followed by a tour improvement (local minimization) phase (US, for unstringing and stringing). GENI starts with a tour consisting of three cities. At each iteration, a new city is inserted into the partially constructed tour while performing a local reoptimization of the tour. The neighborhood $N_p(v)$ of each city v is defined as the set of the p (a parameter) cities already on the tour that are closest to v (if p is greater than the number of cities on the tour, then all of them define $N_p(v)$). City v is inserted between two vertices v_i and v_j from $N_p(v)$. There are two types of GENI insertions. Let v_{i-1} and v_{i+1} denote the predecessor and the successor of v_i for a given orientation.

Type 1. For both orientations of the tour, insert city v as follows: for all v_i and v_j ($i \neq j$) from $N_p(v)$ and all $v_k \in N_p(v_{i+1})$ ($k \neq i,j$), delete arcs (v_i, v_{i+1}), (v_j, v_{j+1}) and (v_k, v_{k+1}); insert arcs (v_i, v), (v, v_j), (v_{i+1}, v_k) and (v_{j+1}, v_{k+1}); keep the best tour.

Type 2. For both orientations of the tour, insert city v as follows: for all v_i and v_j ($i \neq j$) from $N_p(v)$, all $v_k \in N_p(v_{i+1})$ ($k \neq j, j+1$) and all $v_\ell \in N_p(v_{j+1})$ ($\ell \neq i, i+1$), delete arcs (v_i, v_{i+1}), $(v_{\ell-1}, v_\ell)$, (v_j, v_{j+1}) and (v_{k-1}, v_k); insert arcs (v_i, v), (v, v_j), (v_ℓ, v_{j+1}), $(v_{k-1}, v_{\ell-1})$ and (v_{i+1}, v_k); keep the best tour.

When all n cities are in the tour, GENI stops. In the local search phase the neighborhood solutions are defined with delete/insert (or drop/add) moves. Deletion of a city is done using the same type of moves as in the insertion step, but in reverse order. Extensive computing results in Gendreau *et al.* [1] and Johnson and McGeoch [8] show that the GENIUS algorithm is among the best heuristics for the TSP.

Because the size of the neighborhood depends on p, we immediately get a set of neighborhood structures for VNS by denoting with $\mathcal{N}_p(x)$ all tours obtained by delete/insert (defined with Type 1 and Type 2 deletion and insertion moves of the GENIUS algorithm) with parameter value p. The basic VNS algorithm using US as a local search routine (G+VNS) works as follows:

Initialization. Choose p and find an initial tour r by GENI; set $k_{max} = [p/2+1]$;

Main step. (1) Set $k=2$; (2) Until $k=k_{\max}$, repeat the following steps: (a) Find a new tour by the US procedure with parameter value k; denote it by r; (b) If this is the best solution obtained so far, set $k=2$; otherwise, set $k=k+1$;

The value of $k_{\max}$ was selected in order for GENIUS and G+VNS to take similar computing times.

Results on the same type of test problems (i.e., Euclidean instances) as reported in Gendreau *et al.* [1] are given in Table 1. It appears that 0.75% improvements in value is obtained on average within a similar CPU time (increase of 1.8%). Moreover, improvements are obtained for all problem sizes. The improvements are significant when compared with other studies. For example a 0.5% improvement on average for the 2.5-opt heuristic [9] 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 Johnson and McGeoch [8]. Usually CPU time for G+VNS in Table 1 is larger than CPU time for GENIUS. Exceptions are instances with 700 and 800 cities. That could be explained by the small size of the sample chosen, i.e., for $n \geq 600$, average results on 10, not 100, instances are reported.

4. TRAVELING SALESMAN PROBLEM WITH BACKHAULS (TSPB)

In the traveling salesman problem with backhauls (TSPB) customers (or cities) are divided into three disjoint sets: depot, line-haul L and backhaul B customers. Starting from the depot, a route must be designed such that all linehaul customers are visited contiguously before all backhaul customers. In fact, TSPB is a single-vehicle routing problem which can be reformulated into a TSP by adding large distances (or costs) between customers that belong to different subsets (L or B).

In Gendreau *et al.* [2] six heuristic methods for TSPB are proposed and compared. Their results indicate that, in terms of solution quality, GENIUS (applied to the reformulated problem) is the best algorithm. In Table 2 we compare GENIUS with our GENIUS based VNS on the series of random Euclidean instances, as designed and explained in [2]. It appears that a 0.40% improvement in value is obtained on average with 30% increase in running time. Again, improvement is obtained for all problem sizes.

Preliminary tests of the basic VNS algorithm on the p-median problem, the multisource Weber

Table 1. TSP: average results for random Euclidean problems over 100 trials for $n = 100, \ldots, 500$ and 10 trials for $n = 600, \ldots, 1000$.

			Best value found			% Imp	CPU times	
n	p	k_{max}	GENI	GENIUS	G+VNS		GENIUS	G+VNS
100	6	4	801.35	789.66	786.39	0.42	4.7	5.9
200	6	4	1110.37	1093.65	1087.59	0.56	12.6	12.3
300	6	4	1349.80	1327.46	1318.50	0.68	21.8	21.7
400	6	4	1557.28	1532.15	1521.89	0.67	38.3	44.0
500	6	4	1732.54	1705.44	1691.98	0.80	44.3	56.6
600	6	4	1884.76	1858.50	1837.43	1.15	134.4	160.8
700	6	4	2041.34	2010.96	1989.74	1.07	224.8	215.5
800	6	4	2178.99	2148.37	2135.05	0.62	392.8	284.8
900	6	4	2314.80	2276.25	2264.36	0.53	505.1	595.0
1000	6	4	2439.02	2405.24	2381.40	1.00	354.3	367.3
Average			1741.03	1714.77	1701.43	0.75	173.31	176.39

Table 2. TSPB: average results for random Euclidean problems over 30 trials.

				Best value found			% Imp	CPU times	
n	$\lvert B\rvert/n$	k	k_{max}	GENI	GENIUS	G+VNS		GENIUS	G+VNS
100	0.1	6	4	1012.50	994.12	987.11	0.71	4.7	5.4
	0.2	6	4	1068.70	1047.01	1044.66	0.22	4.8	5.8
	0.3	6	4	1109.66	1088.09	1085.34	0.25	4.8	5.5
	0.4	6	4	1125.63	1106.69	1102.29	0.40	5.1	5.4
	0.5	6	4	1133.87	1114.34	1108.68	0.51	4.4	5.6
200	0.1	6	4	1418.63	1387.22	1378.80	0.61	36.3	31.6
	0.2	6	4	1498.83	1470.95	1464.88	0.41	32.4	35.9
	0.3	6	4	1550.52	1525.26	1519.93	0.35	31.7	30.7
	0.4	6	4	1585.76	1555.26	1548.73	0.42	31.2	38.8
	0.5	6	4	1586.93	1554.13	1546.97	0.46	39.6	43.1
300	0.1	6	4	1720.82	1683.76	1675.82	0.47	106.4	109.3
	0.2	6	4	1824.62	1784.80	1782.62	0.12	105.9	87.2
	0.3	6	4	1886.48	1854.86	1849.05	0.31	70.9	100.1
	0.4	6	4	1903.29	1874.43	1865.75	0.47	69.6	105.6
	0.5	6	4	1927.34	1892.20	1887.35	0.26	72.3	101.1
500	0.1	6	4	2197.16	2158.79	2156.61	0.10	325.6	343.6
	0.2	6	4	2342.99	2297.11	2292.04	0.22	289.5	248.1
	0.3	6	4	2409.80	2370.45	2363.16	0.31	317.7	383.3
	0.4	6	4	2443.12	2399.35	2388.07	0.47	374.0	326.1
	0.5	6	4	2464.11	2418.20	2405.55	0.53	405.9	472.2
1000	0.1	6	4	3099.17	3042.60	3029.76	0.42	1130.3	1417.9
	0.2	6	4	3281.34	3232.65	3213.61	0.59	1211.1	1637.2
	0.3	6	4	3366.07	3314.80	3302.93	0.36	1019.8	1643.1
	0.4	6	4	3451.02	3387.43	3366.23	0.63	1302.8	1898.3
	0.5	6	4	3455.69	3388.16	3379.67	0.25	1324.6	1762.6
	Average			2034.56	1997.71	1989.82	0.39	332.86	433.74

problem (or continuous location-allocation problem), the minimum sum-of-squares clustering problem and the weighted maximum satisfiability problem are equally favorable.

We are well aware that some building blocks of VNS were exploited before by others (for an early example, Erlenkotter's DUALOC algorithm [10] for the simple plant location problem uses two neighborhood structures) and that other metaheuristics incorporate similar strategies, attained by other means (e.g. the candidate list diversification and intensification strategies of Tabu search modify the set of solutions from which one is selected at each iteration through forbidden moves, i.e. Tabu lists, or exploitation of long term memory). However, the proposed basic scheme for VNS appears to be new, to the best of our knowledge, and is very effective. In view of the ease with which it can be applied to a variety of problems we believe it is worthy of further study and use.

Acknowledgements—Research supported by ONR grant N00014-95-1-0917, NSERC (Natural Scientific Research and Engineering Council of Canada) grant GPO105574 and FCAR (Fonds pour la Formation de Chercheurs et l'Aide à la Recherche) grant 95ER1048. We thank Michel Gendreau, Alain Hertz and Gilbert Laporte for making the GENIUS code available to us.

REFERENCES

1. Gendreau, M., Hertz, A. and Laporte, G., New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 1992, **40**, 1086–1094.
2. Gendreau, M., Hertz, A. and Laporte, G., The traveling salesman problem with backhauls. *Computers and Operations Research*, 1996, **23**, 501–508.
3. Reeves, C., ed., *Modern Heuristic Techniques for Combinatorial Problems*. Blackwells, Oxford, 1993.
4. Osman, I. H. and Laporte, G., Metaheuristics. A bibliography . *Annals of Operational Research*, 1996, **63**, 513–628.
5. Glover, F., Tabu Search–Part I. *ORSA Journal of Computing.*, 1989, **1**, 190–206.
6. Glover, F., Tabu Search–Part II. *ORSA Journal of Computing*, 1990, **2**, 4–32.
7. Hansen, P. and Jaumard, B., Algorithms for the maximum satisfiability problem. *Computing*, 1990, **44**, 279–303.
8. Johnson, D. S. and McGeoch, L. A., The traveling salesman problem, a case study in local optimization. In *Local search in combinatorial optimization*, eds E.H.L. Aarts and J.K. Lenstra. Wiley, New York, 1997 (to appear).
9. Bentley, J. L., Fast algorithms for geometric traveling salesman problem. *ORSA Journal of Computing*, 1992, **4**, 387–411.
10. Erlenkotter, D., A dual-based procedure for uncapacitated facility location. *Operations Research*, 1978, **26**, 992–100911.