Int. J. Appl. Math. Comput. Sci., 2026, Vol. 36, No. 3, 395–405
DOI: 10.61822/amcs-2026-0025

THE LESS–IS–MORE APPROACH TO VARIABLE NEIGHBORHOOD SEARCH: A COMPARATIVE STUDY

MARTA KASPRZAK [a]

[a] Institute of Computing Science
Poznan University of Technology
Piotrowo 2, 61-138 Poznań, Poland
e-mail: mkasprzak@cs.put.poznan.pl

The less-is-more approach applied to metaheuristic variable neighborhood search combines simplicity and effectiveness in a unique way. With a minimal volume of source code, one can quickly obtain very good solutions. However, the time spent on algorithm implementation may grow significantly on attempts to fit its elementary components, such as, for example, selection of an initial solution, local search moves, or perturbation moves. This study aims to support such a process through a comparison of several possibilities of element composition, including new ideas for elementary moves. The systematic comparative analysis includes 36 versions of this metaheuristic tested on instances of two problems: the traveling salesman one, since it is a model problem commonly used in real-world applications, and that of DNA sequencing, which is a basic and well-studied problem in bioinformatics. The results give basis for conclusions about the compositions of procedures that perform better than others, in the sense of solution quality and computation time.

Keywords: metaheuristic, variable neighborhood search, local search, less-is-more approach, traveling salesman problem, DNA sequencing.

1. Introduction

Variable neighborhood search (VNS) is a metaheuristic based on local search, employing a series of moves of growing complexity to escape from local optima (Mladenović and Hansen, 1997) (for a review see, e.g., the works of Brimberg *et al.* (2023) or Sleptchenko *et al.* (2023)). Its canonical structure allows one to define quite advanced ideas and can be further enriched by mixing with other heuristic approaches (see, e.g., Hansen and Mladenović, 2009). Here we want to focus, however, on another aspect of this method, as interpreted in the spirit of minimalism. The less-is-more approach to VNS was first proposed by Mladenović *et al.* (2016) as a strategy where only one type of move is assumed, with the same type also serving as a diversification element (see also Mladenović *et al.*, 2022). Surprisingly, such a simple scheme beat much more complex heuristics, which was also demonstrated further in the literature.

Until now, research on the less-is-more approach to VNS (VNS-LIMA for short) has not been widespread, mainly due to novelty of this proposition. (There are obviously many papers related to VNS, even employing some minimalistic concepts, but still more advanced than the approach proposed by Mladenović *et al.* (2016), especially because of more than one type of move.) Mladenović *et al.* solved the minimum differential dispersion problem (2016) or the obnoxious p-median one (2020), while Chagas *et al.* (2024) dealt with some other facility location problems, Costa *et al.* (2017) with a clustering problem, Brimberg *et al.* (2017) with a transportation problem, and Lu *et al.* (2020) with a scheduling one.

These papers report the new algorithms as outperforming other state-of-the-art methods for particular problems, such as a balanced k-means heuristic (Costa *et al.*, 2017), multistart local search (Brimberg *et al.*, 2017), metaheuristics tabu search and GRASP (Mladenović *et al.*, 2020), simulated annealing and tabu search (Lu *et al.*, 2020), a hybrid metaheuristic combining GRASP, tabu search and the strategic oscillation method (Chagas *et al.*, 2024), and a hybrid metaheuristic combining GRASP, path relinking and VNS (Mladenović *et al.*, 2016). All these VNS-LIMA algorithms used swaps as the only move, both in the intensification and

diversification phases, the latter consisting in random selection of swapped elements. Depending on the problem solved, the swaps were realized either within a solution or between a solution and an external set. Most of these algorithms chose a first improving move found during the local search, while Mladenović *et al.* additionally tested a version with the best improving move selected (2016) or with a mixed strategy (2020). The algorithms started with initial solutions generated randomly, except for that by Brimberg *et al.* (2017), who generated the solution in a greedy manner. The algorithm by Chagas *et al.* (2024) was parallelized, so it technically differs from the others: it started with a population of initial solutions processed in different threads and chose the best improving move in the local search. The population was partly composed of random solutions and partly generated by the greedy approach with the first element of a solution randomly selected.

However, less-is-more VNS can offer more, even within its minimalistic strategy. The aim of this study is to propose and compare several compositions of elementary parts of the algorithm, which include generation of an initial solution (three methods), choice of a move (three types), selection of an improving move (two ways), and application of a perturbation move (two ways). These elements, including also new propositions, are defined in Section 3, after the general method is presented in Section 2. Such compositions have not been examined before, and the systematic comparative analysis provided here may support future authors in choosing elements of their VNS-LIMA algorithms. Data for tests and problems are described in Section 4, while the comparison is summarized and discussed in Sections 5 and 6. We conclude the paper in Section 7.

2. General method

The general method of variable neighborhood search explores the solution space of an optimization problem with a series of moves of growing complexity, applied to a single current solution through the local search approach (Mladenović and Hansen, 1997). It begins with an initial solution and performs the moves, starting from the simplest ones, in defined cycles of operations. The more complex moves come when the former do not longer improve a solution. This way, the searched neighborhood changes, and the moves are designed to reach farther and farther, enlarging the possibility of reaching the global optimum. When a solution is improved, we return to the simplest moves, and the procedure is repeated. We return to those moves also when there are no more complex ones to be performed, and we restart in such a way for a given number of iterations. Diversification in this metaheuristic is in random selection of neighbors, when we want to escape from a local optimum, and intensification is realized via the local search employed after that.

Algorithm 1. VNS-LIMA.

Input: instance of a problem, VNS-LIMA parameters
Output: $bestsol$

```
sol ← initialSolution()
localSearch(sol)
bestsol ← sol
best ← quality(sol)
iter ← 1
for i = min..max with step step do
  perturbationMoves(sol,i)
  localSearch(sol)
  b ← quality(sol)
  if b < best then
    best ← b
    bestsol ← sol
    i ← min
  else
    sol ← bestsol
  end if
  if i ≥ max and iter < iterNo then
    i ← min
    iter ← iter + 1
  end if
end for
return
```

The less-is-more approach to VNS differs from the original one in the number of types of moves, which is only one. However, we obtain a greater number of neighborhoods by applying this type of move in series: the more moves performed in a row, the wider the neighborhood (Mladenović *et al.*, 2016). The successive neighborhoods are obtained by increasing the number of moves performed in a row by one or by some other defined step. The random moves used to escape from local optima are called perturbation moves.

Variable neighborhood search, when considered in the less-is-more approach, can be outlined by the pseudocode given as Algorithm 1 (cf. Mladenović *et al.*, 2016; Costa *et al.*, 2017). The minimization of a criterion function is assumed. The parameters of the method are: $iterNo$ (the number of outer iterations), min and max (the range for the number of perturbation moves executed at once), and $step$ (the step for incrementing the number of perturbation moves). The current and best solutions are denoted by sol and $bestsol$, respectively.

The procedure `perturbationMoves(`sol`,`i`)` realizes the diversification strategy and introduces i more or less random moves in a row to the solution sol. The procedure `localSearch(`sol`)` operates on sol and usually works deterministically. Further,

`initialSolution()` returns a solution generated in some other way, and `quality(`sol`)` returns a value of the criterion function for sol.

The instantiations of this pseudocode used in the comparison are described in Section 3.

3. Variants

It is assumed here that the optimization problem to be solved is to minimize a total cost and that a solution can be encoded as a sequence of indices of problem elements, where every pair of neighboring elements has its cost. Actually, the traveling salesman problem (TSP) is used in the comparison as a model one. However, the general ideas presented in the following subsections can be adjusted for many other problems, with that of DNA sequencing (SEQ) serving here as an example. The aim of the following study is to assess several compositions of procedures, which are to some extent independent of the example problem.

3.1. Initial solution. Initial solutions are generated in three ways. The random solution is a random (closed) sequence of indices of the problem elements, being a permutation in the case of the TSP and SEQ. Another way is to use the randomized greedy approach, like in GRASP (Feo and Resende, 1995). For a current element, up to three of its best available neighbors (having the lowest cost of connection) are identified, and one of them is randomly selected as the next element in the solution; such a solution starts with a random element. The third way is to use the regular greedy approach, where only the first element is selected randomly, with the remaining ones being chosen deterministically (from among the ones available) according to the lowest cost. All the procedures offer the possibility of obtaining different solutions when run multiple times.

3.2. Moves. Three types of moves that are applied to a solution are defined. A swap is the exchange of two single elements in the solution. A translocation is the removal of an element from its place and insertion into another one. A shift is the removal of a series of consecutive elements and their insertion in the same order in another place of the solution. It is understood to use within a VNS-LIMA algorithm moves of only one type.

These moves were selected because of their simplicity, as expected in the less-is-more approach. Although the shift seems to be more complex, it may need the same number of elementary operations as the translocation, depending on the data structure used for encoding the solution. When operating with pointers, these moves need three elementary operations, whereas the swap, being the easiest move in static arrays, needs four. However, the shift entails additional complexity connected with an extra parameter, i.e., the number of consecutive elements. In this study the value of this parameter is limited to at most $1/3$ of the size of the solution, as it is enough (in complete search) for moving a subsequence of any length (see Fig. 1). When the solution is just a sequence of indices, this limit can be set to $1/2$ of the solution size. Another restriction on this value is mentioned in the next subsection.

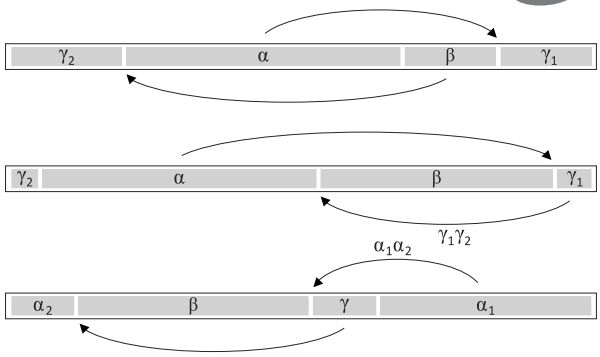


Fig. 1. If the solution is represented by a closed sequence of indices of the problem elements, the operation of shifting a fragment α can be viewed as transforming the solution from the form $\alpha\beta\gamma$ into $\beta\alpha\gamma$. Here it is demonstrated that, regardless of the length of α and β, we can always do that by moving a fragment not longer than $1/3$ of the size of the solution. If α is longer than this limit and β is shorter, we move β. If both α and β are longer, we move γ. In the figure, the intended moves are visualized above the bars and the executed ones below them.

3.3. Best vs. first. During the local search phase, the next improving move can be made by selecting the best one of all current possibilities or by choosing the first found one. Both options were examined, but the latter in the round-robin manner. Namely, moves are considered cyclically to avoid the concentration of changes in the first part of the solution. For swaps, the first element from the pair is taken from a current position in the solution, and the second one is selected from all possibilities. In the lack of improvement, the current position is incremented. This is similar for translocations, where the second choice is related to the destination place of the element taken from a current position. For shifts, a current position indicates the beginning of the moved subsequence, the end is selected from a restricted set of values, and the destination place is selected from all possibilities. The length of the subsequence is set to $1 + 10a$, where a is a nonnegative integer that gives at most $1/3$ of the size of the solution. In this way the shift is a generalization of the translocation, and the results of tests can be compared taking this into account. This further restriction on the length of the subsequence is dictated by the necessity of

bringing the computation time closer to the other move cases, for the purpose of this comparison. However, in a general situation, all lengths up to $1/3$ (or $1/2$) of the solution size can be checked for the benefit of solution quality.

3.4. Random vs. half-random. Random moves consist in selecting randomly elements and their destination. They are used as a diversification element in typical VNS-LIMA algorithms (the procedure `perturbationMoves()` in the pseudocode of the method). Half-random moves are used for the same purpose in the alternative version of the procedure. There, a moved object is randomly selected, and the destination place is chosen to minimize the cost of connection of the object moved with the new neighborhood but only from one side (left or right). Such a choice is a step toward joining elements into well-matched clusters, perhaps not achievable by other means; on the other hand, it is far from the choices applied during the local search phase, thus it still works as a perturbation move. More precisely, for translocations and shifts, the choice is made based on the value cost(ln,lo) – cost(ln,rn) (or cost(ro,rn) – cost(ln,rn)), where ln and rn stand for the left and right neighbors of the object in the new destination, and lo and ro for the left and right sides of the object. For swaps, this is cost(ln,lo) – cost(ln,sw) (or cost(ro,rn) – cost(sw,rn)), where sw stands for the second swapped element. It goes beyond a choice based on the cost matrix only and takes into account also specific connections present in a current solution. This can differentiate the choices made for the same object in different solutions.

3.5. Combinations. The compositions of the procedures described above that are used in the comparison can be expressed as the Cartesian product $A \times B \times C \times D$, where A stands for the three procedures for generating the initial solution, B for the three types of move, C for the two ways of selecting an improving move during the local search, and D for the two versions of perturbation moves. These compositions function within the general scheme of VNS-LIMA presented as the pseudocode in Section 2.

4. Problems and instances

The first problem selected for the comparison is the classical symmetric traveling salesman problem (TSP), which is strongly NP-hard (Garey and Johnson, 1979). It was chosen because it is a model optimization problem commonly used in real-world applications where routes are to be determined. It is employed even in such areas as astronomy to define the movement of a telescope, in bioinformatics to build physical maps of a genome, or in aircraft maneuver planning to identify Dubins paths for multiple locations (Tian *et al.*, 2025; Liu *et al.*, 2022). This problem was also chosen by the authors of VNS in the original paper (Mladenović and Hansen, 1997). An instance of the TSP is originally understood as the matrix of distances between all pairs of points (cities) located in a planar area. The goal is to find the cycle that involves all points exactly once and minimizes the total distance, calculated as the sum of distances between all pairs of neighbors in the cycle.

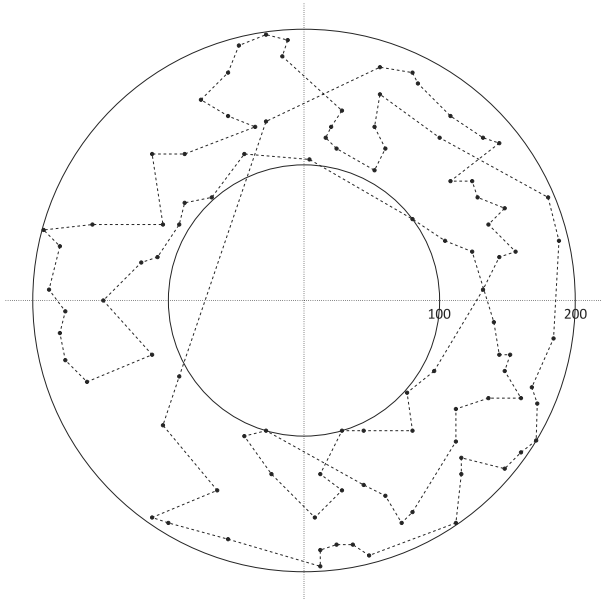


Fig. 2. One of the TSP instances from the experiment, with 100 points randomly located within the annulus of radii 100 and 200. The drawn cycle (of length 3318) is the result of the VNS-LIMA algorithm in this popular setting: the initial solution generated randomly, the move defined as the swap of points, the selection of a best neighbor in the local search phase, and random perturbation moves. The algorithm was run once with the parameters set as in Section 5. As will be seen in further comparison, other combinations of the algorithm components may lead to a significant improvement in the value of the criterion function.

Following this, we generated the instances for tests from points randomly distributed in the Cartesian plane within the annulus of radii 100 and 200. This gives us sets of points naturally allocated around a circle; on the other hand, the range 100 in radii, with the smaller one being 100, is big enough to make the instances hard to solve by heuristics. The points were located to keep the distance at least 10 between all their pairs, and 100 points were generated per instance. After that, the distance matrices were filled with Euclidean distances between the points, rounded to the nearest integer; see Fig. 2 for illustration.

Please note that instances can be simplified by

narrowing the radii of the annulus, and then the optimum value of the criterion function will be easier to estimate. Such tests were also performed; see the next section.

The second examined problem is a DNA sequencing problem (SEQ), whose aim is to reconstruct a DNA sequence based on a set of shorter fragments (oligonucleotides). More precisely, it is the classical version of sequencing by hybridization with a standard oligonucleotide library, where oligonucleotides have the same length and experimental errors are expected (both false negatives and false positives are assumed in the data). Thus, based on a set of subsequences of equal length, the goal is to find a sequence of a length upper bounded by a given value, composed of a maximum number of the input subsequences; this problem is strongly NP-hard (Blazewicz *et al.*, 2004), and Fig. 3 shows a simple example of it.

This problem was intensively studied for years and gained many effective algorithmic solutions, hybrid metaheuristics and hyper-heuristics among them. Its benchmark instances are publicly available on the website `http://bioserver.cs.put.poznan.pl`[1]. They were derived from DNA sequences coding human proteins and published in NCBI's GenBank, contain a significant number of false negatives and false positives, and have known optimum values. These circumstances, as well as the partial similarity to the TSP, make SEQ a good complement to this study. SEQ is somehow similar to the asymmetric traveling salesman problem but differs in a few aspects. SEQ maximizes a criterion function, but locally we can see it as choosing minimal costs between neighbors in a solution, the cost being equal to the offset in the maximum overlap of two subsequences. SEQ needs no cycle as a solution, but we simply can break it in some place. Finally, SEQ does not include all problem elements in a solution, but we still can operate on permutations and count only a continuous part of them.

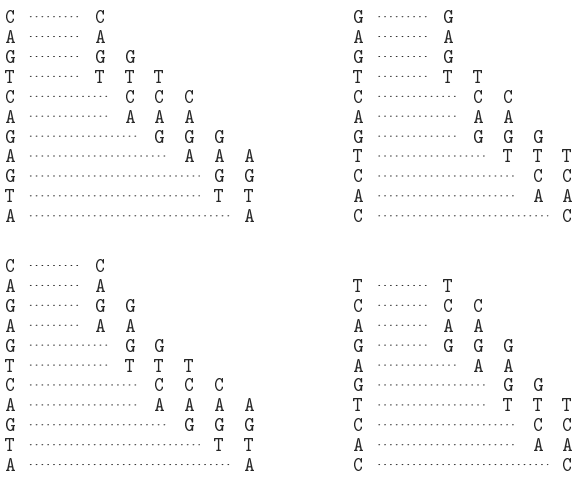


Fig. 3. Four feasible solutions of the DNA sequencing problem (SEQ) with the set of oligonucleotides {AGTA, CAGA, CAGT, GAGT, GTCA, TCAC, TCAG} and the target length of the solution equal to 11. Two optimum solutions are shown on the left. With CAGTCAGAGTA as an example source for these data, we have TCAC as a false positive error and AGAG and AGTC as false negatives. In reality, the lengths of sequences are greater: several hundred nucleotides for sequences to reconstruct and around 10 for oligonucleotides.

5. Comparison for the TSP

In the experiment, 100 TSP instances were generated, each based on 100 points. For each instance, 10 iterations of computations were performed, where one iteration went from generation of the initial solution (via the random, randomized greedy or regular greedy approach) through all tested methods; see Fig. 4. In each iteration all the methods started with the same initial solution; this way the results are better comparable within the same table. Between tables, the TSP instances stay the same (except for Table 4), but the initial solutions differ. A solution in the algorithms is represented by a sequence of indices of

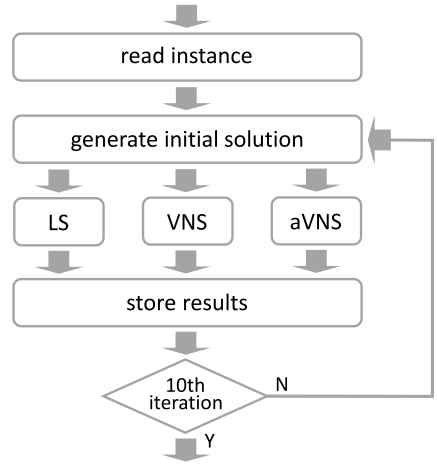


Fig. 4. Each tested TSP/SEQ instance was processed according to this scheme. LS stands for pure local search, VNS for the VNS-LIMA algorithm with random perturbation moves, and aVNS for the alternative version of VNS-LIMA with half-random perturbation moves. Inside each of these three blocks, computations were done thrice with different types of move assumed, but starting with the same initial solution.

[1]For the instances used, see Table 1 available on the website `https://bio.cs.put.poznan.pl/research_fields/5193ee2f9dfb89b38b000001`.

Table 1. Total cost of TSP solutions (minimized) obtained by the analyzed methods with the implemented selection bestLS (the best possible improving move chosen in the local search phase), presented as the average and best obtained values. The initial solutions were generated with the random, randomized greedy or regular greedy approach. LS stands for the local search, VNS for the VNS-LIMA algorithm with random perturbation moves, and aVNS for the alternative version of VNS-LIMA with half-random perturbation moves.

		Random		Randomized greedy		Regular greedy	
		Average	Best	Average	Best	Average	Best
	Initial	19726.4	16787	5252.3	4424	3277.7	2727
LS	Swap	5214.1	3935	3496.6	2936	3100.7	2671
	Transl.	3623.9	2749	2975.1	2623	2890.2	2607
	Shift	2833.8	2541	2813.0	2544	2759.1	2465
VNS	Swap	3429.8	2759	3155.0	2768	2997.7	2600
	Transl.	2758.1	2502	2727.5	2494	2712.3	2506
	Shift	2629.1	2456	2628.4	2458	2628.6	2456
aVNS	Swap	3366.3	2721	3116.0	2660	2980.8	2567
	Transl.	2757.5	2466	2714.3	2482	2698.8	2488
	Shift	2623.9	2459	**2623.7**	2458	2625.1	2456

Table 2. Total cost of TSP solutions obtained by the analyzed methods with the implemented selection firstRR (the first improving move chosen in the round-robin manner in the local search phase), presented as the average and best obtained values.

		Random		Randomized greedy		Regular greedy	
		Average	Best	Average	Best	Average	Best
	Initial	19772.7	16346	5257.3	4283	3278.1	2785
LS	Swap	4952.3	3580	3538.1	2878	3111.2	2716
	Transl.	3181.0	2559	3100.6	2642	2931.9	2582
	Shift	2846.9	2543	2829.2	2566	2789.2	2497
VNS	Swap	3482.8	2890	3249.9	2827	3071.4	2696
	Transl.	2716.7	2487	2719.8	2507	2715.9	2528
	Shift	2650.4	2470	2649.8	2479	2649.0	2470
aVNS	Swap	3369.6	2773	3144.7	2651	3028.2	2582
	Transl.	2695.2	2456	2699.2	2463	2687.0	2485
	Shift	2634.6	2465	2634.6	2463	**2633.6**	2461

the TSP points, processed as a cycle and kept in a static array.

As stated above, the goal of this experiment was to compare a wide range of different compositions of procedures of VNS-LIMA, including those not used before, not to propose a new solution for the TSP. Therefore, the instances were generated for this purpose, to keep a good balance between hardness and possibility to obtain solutions reaching imaginable optima. We are aware that there are recognized databases of benchmark instances for the TSP and plenty of methods, with many of them outranking a simple VNS-LIMA algorithm; on the other hand, as pointed out at the beginning of this paper, published VNS-LIMA algorithms outperformed earlier best methods known from the literature for various problems. This is possible for the DNA sequencing problem as well; see the next section for a comparison including all advanced methods known from the literature and tested on the employed benchmark instances for SEQ.

The VNS-LIMA parameters were set to the following values: 10 VNS outer iterations and the range for the number of perturbation moves executed at once from 5 to 50 with step 5. These values correspond to the ones reported in the literature for VNS-LIMA and seem more or less universal. We especially refer to the work of Costa *et al.* (2017), where the values were discussed based on experimental verification. The authors remarked that a small number of perturbation moves is often not sufficient to escape from local optima, a small step in its incrementing deteriorates neighborhood expansion while a larger one could miss local optima close to another. Obviously the number of iterations can be increased at the cost of computation time but likely with a profit in the solution's quality. In general, metaheuristics need an initial experimental step of tuning values of their parameters used in further tests in order to

Table 3. Average computation time per run, given in milliseconds, for the results from Tables 1 and 2.

		Random		Randomized greedy		Regular greedy	
		BestLS	FirstRR	BestLS	FirstRR	BestLS	FirstRR
VNS	Swap	182	33	157	30	144	29
	Transl.	262	32	232	32	221	31
	Shift	1469	122	1459	121	1442	119
aVNS	Twap	154	32	132	29	120	27
	Sransl.	175	23	151	23	146	22
	Shift	1107	113	1107	114	1087	111

optimize results. Here, however, this was not our goal, as we intended the comparison to be not targeted at a particular problem. Also, the programs compared here are instantiations of the same algorithm, of the same general structure, differing only in highlighted aspects. Thus they compete on equal terms having the same set of values selected as above, giving us a basis for conclusions about their relative performance.

The results presented in this section are either in the form of the average or best values from 1000 tests. The tests were performed in the single-threaded mode, under Windows OS, on a PC with the Intel Core i5 1.6 GHz processor and 16 GB RAM.

Tables 1 and 2 contain the results for all the analyzed versions of the methods. For reference, the initial solutions and the solutions gained by the pure intensification stage (LS) are presented. The best average results in the tables are given in bold. In general, aVNS (the VNS-LIMA version with half-random perturbation moves) gives better quality than VNS (the version with random perturbation moves), and bestLS (the best possible improving move chosen in the local search phase) better than firstRR (the first improving move chosen in the round-robin manner). The regular greedy approach for initial solutions performs better, in general, than randomized greedy, and the latter better than the random variant. The shifts are better than the translocations, and the latter better than the swaps. In particular cases, however, other preferences come to mind, especially for firstRR, which is better than bestLS for the translocations, or for the randomized greedy approach, which is the best option for the shifts in the bestLS mode. As we see, the combination of elementary components of VNS-LIMA used until now in the literature is far from the optimal one. An illustrative example for the best setting is given in Fig. 5. The time spent on these computations is shown in Table 3.

There is a big difference in computation times between the algorithms, and the bestLS versions work many times longer compared to the firstRR ones. The same can be said about the shifts compared to the other moves. This may determine the choice when time is crucial, but in this comparison all the versions stay.

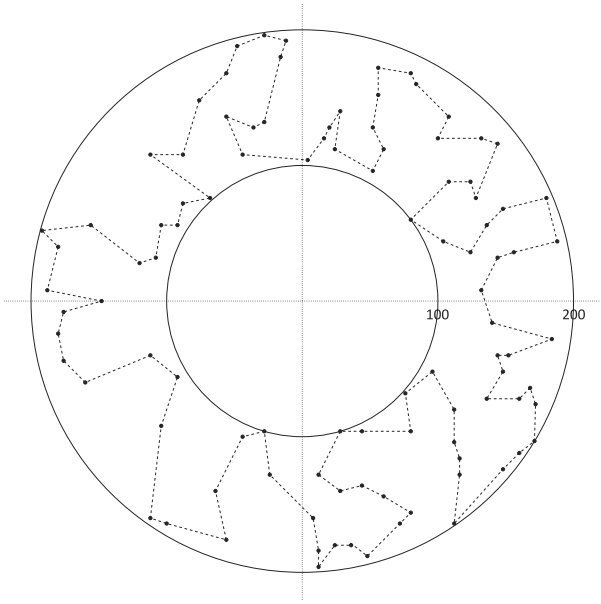


Fig. 5. Solution (of length 2562) generated for the TSP instance from Fig. 2 by the VNS-LIMA algorithm run once (with the parameters set as before) in the following setting: the initial solution generated with the randomized greedy approach, the move defined as the shift, the selection of a best neighbor in the local search phase, and half-random perturbation moves. It was the best combination of the procedures for this instance, as well as, in general, for the whole experiment in the TSP part.

The shifts, although extremely restricted due to their time complexity, appear to be the most effective moves, with a significant gain in the criterion function. One can wonder, however, why there is such a small difference for the shifts when changing initial solutions. Perhaps the global optimum is nearby? We can partially answer this question by narrowing the area defined for the TSP instances; see Table 4.

For these new instances, all runs of VNS and aVNS with the selection firstRR and moves being translocations or shifts finished with the same result for each instance, as well as with the bestLS and shifts. We can thus

Table 4. Total cost of solutions obtained by the analyzed methods starting with the random initial solution, for the TSP area defined as the annulus of radii 180 and 200.

		BestLS		FirstRR	
		Average	Best	Average	Best
	Initial	24320.6	20233	24374.8	20237
LS	Swap	5878.3	3739	5450.3	3300
	Transl.	2275.4	1534	1772.6	1534
	Shift	1994.2	1534	1899.2	1534
VNS	Swap	2737.6	1549	2772.0	1559
	Transl.	1605.3	1534	1604.6	1534
	Shift	1604.6	1534	1604.6	1534
aVNS	Swap	2752.6	1573	2590.2	1534
	Transl.	1606.2	1534	1604.6	1534
	Shift	1604.6	1534	1604.6	1534

suppose that they were global optima for these instances. Indeed, they are close to the length of the (distorted) circumference of the TSP area. The results obtained for the bestLS and translocations are almost of the same quality. For the translocations and shifts, the alone local search in the firstRR mode also found very good solutions; swaps performed on average 2–3 times worse. Here, in general, the bestLS mode did not perform better than firstRR despite slightly better initial solutions and worked 5–14 times longer.

6. Comparison for SEQ

From among benchmark instances of SEQ available on the website `http://bioserver.cs.put.poznan.pl`, the hardest ones were chosen, i.e., including the greatest number of errors, both false negatives and false positives. Every instance was created as the set of all oligonucleotides covering a longer DNA sequence, from which 20% of randomly selected oligonucleotides were removed (as false negatives) and the same number of erroneous oligonucleotides were introduced (as false positives). The oligonucleotides had the length of 10, the longer sequences were of sizes 200, 400 or 600, and there were 40 different sequences resulting in 40 SEQ instances per each size (Blazewicz *et al.*, 2013). For example, an instance derived from a sequence of length 200 contains 191 alphabetically sorted oligonucleotides: 152 correct ones and 39 false positives; the optimum value of the criterion function is 152.

The scheme of the experiment is as in the previous section. For each of the 40 instances of a given size, 10 iterations of computations were performed, where one went from generation of the initial solution (via the random, randomized greedy or regular greedy approach) through all tested methods. Thus, the results presented in Tables 5 and 6 are in the form of either the average or best values from 400 tests. The VNS-LIMA parameters were set to the same values: 10 VNS outer iterations and the range for the number of perturbation moves executed at once from 5 to 50 with step 5.

It was our intention to test SEQ in possibly the same circumstances as the TSP, although they significantly differ, as the current study was intended to be problem-independent to some extent. Following this, we made only minimum necessary adjustments of the algorithms to produce correct solutions of SEQ. Now oligonucleotides are problem elements (a solution is a sequence of their indices), and costs are calculated as the offset in a greatest possible exact overlap of an ordered pair of elements; for example, (CAGA, AGAT) gives us the cost of 1, (AGAT, CAGA) gives us 4. Such costs form a matrix of distances between elements. A solution is still processed within the algorithms as a cycle, but its evaluation is done for its best continuous fragment: a longest subsequence of indices that creates a DNA sequence not longer than a target length. The local search mechanisms did not change, but improvement of the entire solution may result in worsening its evaluated fragment, so the local search procedure returns now the better of these two: the initial solution or the "improved" one. All other parts of the algorithms remained unchanged.

The results presented in Tables 5 and 6 mostly confirm the conclusions drawn for the TSP. The shifts significantly outperform other moves, and the translocations take the second place. The regular greedy approach for initial solutions consistently gives better results than randomized greedy, which works better than the random variant. Half-random perturbation moves (aVNS) give better quality than random perturbation ones (VNS). This time, however, we cannot claim the advantage of bestLS over firstRR; it depends on a case, but now any of these schemes of selecting an improving move can win for any type of move used in the algorithms. The best average results in these tables are given in bold.

For further tests, whose aim is to compare VNS-LIMA algorithms with best meta- and hyper-heuristics known for SEQ, four variants of VNS-LIMA were selected. Both versions, aVNS and VNS, were tested, as well as both bestLS and firstRR, supplemented by the shifts and the regular greedy initial solution as unquestionably best options. See Table 7 for the results.

In this table two methods stand out as the best: the hybrid decomposition metaheuristic by Caserta and Voss (2014) and the hybrid genetic algorithm by Blazewicz *et al.* (2006), since for almost all instances they reached global optima. The VNS-LIMA algorithms performed worse, but generally better than the other best heuristics known for SEQ, including the hyper-heuristic methods

Table 5. Criterion function values (maximized) obtained for SEQ instances by the analyzed methods with the implemented selection bestLS. The instances were derived from DNA sequences of length 200 with the optimum equal to 152.

		Random		Randomized greedy		Regular greedy	
		Average	Best	Average	Best	Average	Best
	Initial	22.1	24	67.6	82	135.3	152
LS	Swap	48.0	57	69.0	82	135.4	152
	Transl.	62.4	79	118.2	144	135.6	152
	Shift	118.9	152	111.9	152	144.5	152
VNS	Swap	67.3	82	74.5	86	135.9	152
	Transl.	89.5	114	129.5	152	137.3	152
	Shift	**152.0**	152	**152.0**	152	**152.0**	152
aVNS	Swap	76.9	94	76.7	92	136.7	152
	Transl.	109.4	142	134.5	152	140.1	152
	Shift	**152.0**	152	**152.0**	152	**152.0**	152

Table 6. Criterion function values obtained for SEQ instances by the analyzed methods with the implemented selection firstRR. The instances were derived from DNA sequences of length 200 with the optimum equal to 152.

		Random		Randomized greedy		Regular greedy	
		Average	Best	Average	Best	Average	Best
	Initial	22.0	24	67.0	81	134.6	152
LS	Swap	55.2	64	68.7	82	134.7	152
	Transl.	69.9	91	112.9	150	134.8	152
	Shift	129.4	152	128.5	152	143.9	152
VNS	Swap	74.2	89	77.6	91	135.4	152
	Transl.	99.0	121	129.6	152	139.7	152
	Shift	**152.0**	152	**152.0**	152	**152.0**	152
aVNS	Swap	76.7	94	78.1	92	135.9	152
	Transl.	114.0	144	132.8	152	140.4	152
	Shift	**152.0**	152	**152.0**	152	**152.0**	152

involving mechanisms of simulated annealing (SimAnn) or roulette choice (RouletteC). We must keep in mind that the strategy in the VNS-LIMA algorithms is not specialized for SEQ, especially in the aspect of the solution length. Here the local search phase can worsen an initial solution for SEQ, as it is constructed for improving the entire vector encoding a solution. We can imagine a better specialized strategy, but it was not the goal of this study, which was to test in different conditions new propositions of procedures and various compositions within the VNS-LIMA framework, and we may draw some common conclusions. The firstRR selection, where the first improving move is chosen in the round-robin manner during the local search, can give better results than the exhaustive examination in bestLS and in a time an order of magnitude shorter. The swap, the elementary move commonly used in published algorithms based on VNS-LIMA, demonstrated effectiveness much worse than the two other types of moves tested in this work. The shift, proposed here in a time-optimized version, appeared to be the best move regardless of a tested configuration. It can perform even better if time is not a limitation and all lengths of a shifted fragment (up to $1/3$ of the solution length) can be checked. Another element commonly used in published algorithms, the random initial solution, also performed worst. And the half-random scheme of perturbation moves proposed here (aVNS) won against the usual full random scheme.

7. Conclusions

The systematic comparative analysis provided here gives a broad view on performance of the variable neighborhood search realized in the less-is-more paradigm. Several variants of the algorithm components were composed into 36 programs and tested on instances of two problems: the traveling salesman one and the DNA sequencing with erroneous data, while not being specialized for any of these problems. The results give basis for conclusions about the compositions that perform better than others, in the sense of solution quality and computation time. The better performance partially results from the new

Table 7. SEQ instances of all sizes (200, 400, 600) solved by best meta- and hyper-heuristics known from the literature and by four VNS-LIMA algorithms proposed here. All the methods were run once per instance (for the known methods, the results are cited from the literature), and each entry is based on 40 instances. Quality and time are given as average values, where the quality of 100% means optimum, and optimum# is the number of instances solved optimally. For the VNS-LIMA algorithms, the number of instances for which an optimum initial solution was generated is 0.

200			400			600		
Quality [%]	Optimum#	Time [s]	Quality [%]	Optimum#	Time [s]	Quality [%]	Optimum#	Time [s]
Hyper-heuristic SimAnn (Blazewicz *et al.*, 2013), Intel Core 2 Duo, 3.16 GHz								
99.64	34/40	$\leq$300	98.93	22/40	$\leq$300	98.20	14/40	$\leq$300
Hyper-heuristic RouletteC (Blazewicz *et al.*, 2013), Intel Core 2 Duo, 3.16 GHz								
100.00	40/40	$\leq$300	99.92	37/40	$\leq$300	97.20	5/40	$\leq$300
Hybrid genetic algorithm (Blazewicz *et al.*, 2006), Pentium 4, 2.0 GHz								
100.00	39/40	6	100.00	40/40	20	99.98	37/40	44
Tabu and scatter search (Blazewicz *et al.*, 2004), Pentium 4, 2.0 GHz								
99.93	38/40	9	99.67	28/40	69	99.36	19/40	186
Hybrid decomposition metaheuristic (Caserta and Voss, 2014), Pentium 4, 2.0 GHz								
–	40/40	2	–	40/40	11	–	38/40	63
VNS-bestLS-shift-regular greedy, Intel Core i5, 1.6 GHz								
100.00	40/40	5	99.49	31/40	46	98.75	22/40	166
VNS-firstRR-shift-regular greedy, Intel Core i5, 1.6 GHz								
100.00	40/40	0.5	99.71	36/40	4	99.11	26/40	16
aVNS-bestLS-shift-regular greedy, Intel Core i5, 1.6 GHz								
100.00	40/40	4	99.78	36/40	35	98.67	20/40	141
aVNS-firstRR-shift-regular greedy, Intel Core i5, 1.6 GHz								
100.00	40/40	0.4	99.94	37/40	4	99.19	28/40	15

propositions defined in this paper. But, even when restricted to standard components of the VNS-LIMA method, the better compositions were not used before in the literature in current implementations of the method. Such an analysis may be useful for authors of new VNS-LIMA algorithms, as it can reduce time spent on their implementation. What is more, this study demonstrated once more that VNS-LIMA algorithms can achieve very good suboptimal results for NP-hard optimization problems in very short time with a minimal volume of source code. They can compete on equal terms with advanced specialized heuristics and could become more popular as a good reference in research.

References

Blazewicz, J., Burke, E., Kendall, G., Mruczkiewicz, W., Oguz, C. and Swiercz, A. (2013). A hyper-heuristic approach to sequencing by hybridization of DNA sequences, *Annals of Operations Research* **207**: 27–41.

Blazewicz, J., Glover, F. and Kasprzak, M. (2004). DNA sequencing—Tabu and scatter search combined, *INFORMS Journal on Computing* **16**(3): 232–240.

Blazewicz, J., Glover, F., Kasprzak, M., Markiewicz, W., Oguz, C., Rebholz-Schuhmann, D. and Swiercz, A. (2006). Dealing with repetitions in sequencing by hybridization, *Computational Biology and Chemistry* **30**(5): 313–320.

Brimberg, J., Mladenović, N., Todosijević, R. and Urošević, D. (2017). A basic variable neighborhood search heuristic for the uncapacitated multiple allocation p-hub center problem, *Optimization Letters* **11**: 313–327.

Brimberg, J., Salhi, S., Todosijević, R. and Urošević, D. (2023). Variable neighborhood search: The power of change and simplicity, *Computers & Operations Research* **155**: 106221.

Caserta, M. and Voss, S. (2014). A hybrid algorithm for the DNA sequencing problem, *Discrete Applied Mathematics* **163**(1): 87–99.

Chagas, G., Lorena, L., dos Santos, R., Renaud, J. and Coelho, L. (2024). A parallel variable neighborhood search for α-neighbor facility location problems, *Computers & Operations Research* **165**: 106589.

Costa, L., Aloise, D. and Mladenović, N. (2017). Less is more: Basic variable neighborhood search heuristic for balanced minimum sum-of-squares clustering, *Information Sciences* **415–416**: 247–253.

Feo, T. and Resende, M. (1995). Greedy randomized adaptive search procedures, *Journal of Global Optimization* **6**: 109–134.

Garey, M. and Johnson, D. (1979). *Computers and Intractability. A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, San Francisco.

Hansen, P. and Mladenović, N. (2009). Variable neighborhood search methods, *in* C. Floudas and P. Pardalos (Eds), *Encyclopedia of Optimization*, Springer, Boston, pp. 3975–3989.

Liu, G., Wu, S., Zhu, L., Wang, J. and Lv, Q. (2022). Fast and smooth trajectory planning for a class of linear systems based on parameter and constraint reduction, *International Journal of Applied Mathematics and Computer Science* **32**(1): 11–21, 10.34768/amcs-2022-0002.

Lu, S., Pei, J., Liu, X., Qian, X., Mladenović, N. and Pardalos, P. (2020). Less is more: Variable neighborhood search for integrated production and assembly in smart manufacturing, *Journal of Scheduling* **23**: 649–664.

Mladenović, N., Alkandari, A., Pei, J., Todosijević, R. and Pardalos, P. (2020). Less is more approach: Basic variable neighborhood search for the obnoxious p-median problem, *International Transactions in Operational Research* **27**(1): 480–493.

Mladenović, N., Drezner, Z., Brimberg, J. and Urošević, D. (2022). Less is more approach in heuristic optimization, *in* S. Salhi and J. Boylan (Eds), *The Palgrave Handbook of Operations Research*, Palgrave Macmillan, Cham, pp. 469–499.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search, *Computers & Operations Research* **24**(11): 1097–1100.

Mladenović, N., Todosijević, R. and Urošević, D. (2016). Less is more: Basic variable neighborhood search for minimum differential dispersion problem, *Information Sciences* **326**: 160–171.

Sleptchenko, A., Sifaleras, A. and Hansen, P. (Eds) (2023). *Variable Neighborhood Search: 9th International Conference, ICVNS 2022, Abu Dhabi*, Springer Nature Switzerland, Cham.

Tian, M., Shi, Y., Li, R. and Tlelo-Cuautle, E. (2025). Rapid generation of the shortest generalized Dubins path in forced landing, *International Journal of Applied Mathematics and Computer Science* **35**(2): 195–207, DOI: 10.61822/amcs-2025-0014.

Marta Kasprzak is a professor at the Institute of Computing Science, Poznan University of Technology. She focuses her scientific research on bioinformatics, especially the theoretical analysis of problems: combinatorial modeling of real-world problems, determining their computational complexity, designing algorithms. She has published 76 articles and monographs, 47 of them indexed in the Web of Science. She is a laureate of, among others, the Prime Minister Award for her doctoral dissertation, the Award of the IV Division of Technical Sciences of the Polish Academy of Sciences in the field of computer science for her habilitation thesis, and the Medal of the National Education Commission.

Received: 8 December 2025
Revised: 17 February 2026
Re-revised: 16 April 2026
Accepted: 24 April 2026