A double-adaptive general variable neighborhood search algorithm for the solution of the traveling salesman problem

Panagiotis Karakostas, Angelo Sifaleras*

Department of Applied Informatics, School of Information Sciences, University of Macedonia, 156 Egnatia Str., Thessaloniki 54636, Greece

Abstract

This work addresses a novel General Variable Neighborhood Search (GVNS) solution method, which integrates intelligent adaptive mechanisms to re-order the search operators during the intensification and diversification phases, in an effort to enhance its overall efficiency. To evaluate the performance of the new GVNS scheme, asymmetric and symmetric instances of the classic Traveling Salesman Problem (TSP) from the TSPLib were solved. The obtained results of the Double-Adaptive GVNS were compared with those achieved by two single-adaptive GVNS, which use an adaptive mechanism either for the intensification or the diversification phase and with a conventional GVNS. For a fair comparison, all GVNS schemes were structured using the same local search and shaking operators. Moreover, the novel GVNS algorithm was compared with some recent solution methods for the TSP, found in the open literature. The comparative studies revealed the high efficiency of the novel VNS scheme and underlined the significant impact of intelligent mechanisms on the performance of classic metaheuristic frameworks.

Keywords: Variable Neighborhood Search, Metaheuristics, Adaptive Search, Intelligent Optimization, Traveling Salesman Problem

*Corresponding author

Email addresses: pankarakostas@uom.edu.gr (Panagiotis Karakostas), sifalera@uom.gr (Angelo Sifaleras)

1. Introduction

The optimal design and scheduling of complex supply chain systems has a significant positive impact in creating and enhancing the competitive advantage of organizations (Bhatnagar & Teo, 2009). The optimal configuration of such complex systems requires advanced decision making processes, that are achieved through the simultaneous utilization of optimization models and tools (Grazia Speranza, 2018). A class of such optimization tools are the metaheuristic algorithms. Metaheuristics are general, high-level, and flexible optimization frameworks, that can be properly modified to generate powerful heuristic algorithms for the solution of complex NP-hard optimization problems, such as supply chain optimization problems (Sevaux et al., 2018). Variable Neighborhood Search (VNS) is a well-known and widely used metaheuristic optimization framework, proposed by Mladenović & Hansen (1997). It is characterized by its simplicity and high efficiency on the solution of hard optimization problems (Hansen et al., 2017). A VNS algorithm consists of three main parts, a neighborhood change step, an intensification phase and a shaking phase (Hansen & Mladenović, 2014). The neighborhood change step is actually a strategy for guiding the search during the exploration of solution space, as it denotes which neighborhood structure will be applied as well as the acceptance criterion of an obtained solution (Hansen et al., 2017). The shaking phase contains a mechanism to avoid local optimum solutions, while the improvement phase constitutes the main mechanism for improving an incumbent solution, by applying, in accordance to the neighborhood change step, a predefined set of local search operators (Hansen et al., 2017). General VNS (GVNS) is a classic variant of VNS in which one of the Variable Neighborhood Descent (VND) procedures is selected as the main improvement phase. A VND consists of a set of predefined local search operators that are executed either in a sequential or a nested fashion (Hansen et al., 2017).

Sevaux et al. (2018) mentioned the critical role of the proper configuration of a metaheuristic, by defining the most efficient structure and tuning of the parameters of each framework, in its performance and efficiency. Thus, the development of an efficient metaheuristic algorithm is quite difficult, as it demands plenty amount of time in designing and

tuning processes. To this end, Sevaux et al. (2018) underlined the necessity of developing adaptive metaheuristic algorithms, as a structured solution of self-configuration during the execution of a metaheuristic. Talbi (2021) stated that metaheuristics generate large amount of data during their execution. According to the authors, such data are divided into static and dynamic. Static data refer to problem- or instance-specific features, while dynamic data are linked to more complex information generated by the iterative application of a metaheuristic. The authors highlighted the important role of utilizing generated data to produce useful knowledge for enhancing the performance of metaheuristics, by considering intelligent machine learning mechanisms. specifically, embedding such data-driven intelligent mechanisms in the core procedures of metaheuristics leads to the construction of improved solution approaches in terms of quality and robustness (Karimi-Mamaghan et al., 2022). , the utilization of generated data leads to useful knowledge and finally to improved decisions (Karimi-Mamaghan et al., 2022).

GVNS is a powerful metaheuristic which has been successfully applied for the solution of hard optimization problems of increased practical interest (Menéndez et al., 2017; Karakostas et al., 2019, 2020a). Some research contributions have attested the beneficial effects integrating low-level machine learning techniques with intensification procedures of GVNS heuristics (Todosijević et al., 2016; Karakostas et al., 2022). Moreover, Karakostas et al. (2020b) have recently mentioned the positive impact of embedding such intelligent mechanisms into diversification procedures of GVNS solution methods. However, the potential performance benefits of the simultaneous integration of intelligent adaptive mechanisms into both critical components of GVNS has not been investigated. Thus, the aim of the present study is to eludicate if the mentioned integrated approach can improve the decision-making ability of a GVNS heuristic. To this end, this work proposes a new Double-Adaptive GVNS (DA-GVNS), by considering intelligent adaptive mechanisms both in intensification and diversification phases of the algorithm. These mechanisms process dynamic data generated in each iteration of the algorithm, to improve its configuration and consequently its performance in every next iteration. More specifically, short adaptive memories are adopted both in improvement and shaking phases, which store the number of

improvements achieved by each local search and shaking operator. In every next iteration, the proposed algorithm uses the previous generated data of improvements to configure its structure, by properly re-ordering its local search and shaking operators. To evaluate the performance of the DA-GVNS, a single local search-adaptive GVNS, a single shaking-adaptive GVNS and a conventional GVNS were also developed for computational comparison purposes. All GVNS schemes contain the same local search and shaking operators, follow the best improvement search strategy and were applied to the classic Traveling Salesman Problem (TSP), a well-known and widely studied combinatorial optimization NP-hard problem, with several applications in supply chain systems (Punnen, 2007). Moreover, the solutions obtained by the novel GVNS method were compared to those produced by other improved metaheuristics found in the recent open literature.

This work is structured as follows. Section 2 presents a literature review on recent TSP contributions and on recent advances and applications of VNS. The proposed solution method is presented in Section 3, while the computational analyses are provided in Section 4. Finally, Section 5 summarizes the concluding remarks and potential future extensions.

2. Literature review

2.1. Recent contributions on TSP

Osaba et al. (2018) developed an imrpoved discrete water cycle algorithm for the solution both of symmetric and asymmetric TSP instances. The authors enhanced the classic water cycle algorithm by considering further features, such as the utilization of the Hamming distance metric for the computation of distances among obtained solutions and the consideration of an adaptive modification of the movement function. To evaluate the performance of their proposed solution method, they proceeded a computational study using six metaheuristics from the open literature, applied on 33 TSPLib instances. Boryczka & Szwarc (2019) proposed an improved Harmony Search (HS) algorithm for the solution of asymmetric TSP, by addressing a weak mechanism to enhance the effectiveness of harmony memory. The authors found that their improved HS lead to average solution error from 13.42% to 5.54%.

Papalitsas et al. (2019) performed an extended computational analysis for the evaluation of the potential impact of different diversification methods on the performance of GVNS during the solution of asymmetric, symmetric and national TSPLib instances. They studied the effects of three diversification methods, the classic intensified shaking, a quantum-inspired diversification method, and a shuffle method. They found that both intensified shaking and quantum-inspired method performed equivalently in case of asymmetric TSP and much better than using the shuffle diversification, while no significant differences observed in case of symmetric TSP.

Zhong et al. (2019) proposed a discrete variant of a nature-inspired metaheuristic, called Pigeon-Inspired Optimization (PIO) for the solution of large-scale symmetric TSP instances. The authors enhanced the main operators of their algorithm by considering learning abilities in order to improve the exploration and the exploitation of their algorithm. Moreover, the Metropolis acceptance criterion was adopted in an effort to avoid premature convergence. The proposed solution method was tested on 33 large symmetric TSP instances and the achieved solutions compared to those obtained by the state-of-the-art solution methods for such large-scale problem cases. Wang et al. (2019) developed a discrete symbiotic organism search solution method for the solution of TSP. They integrated their method with excellence coefficients in an effort to accelerate the exploration of their search. Moreover, they adopted a self-escape strategy in order to avoid the local optimum solutions.

Campuzano et al. (2020) focused on the improvement of the computational performance of the Miller-Tucker-Zelmin model for the assymetric TSP, by proposing relative valid inequalities. Ebadinezhad (2020) proposed an improved Ant Colony Optimization (ACO) algorithm, by considering a self-adaptive mechanism to overcome specific drawbacks of classic ACO metaheuristic, such as low convergence speed and trapping in local optimum solutions. The author performed a computational study between the new method and the classic one, on 10 symmetric TSP instances from TSPLib. The results showed that the new method presents faster convergence speed and higher accuracy of search. A hybrid metaheuristic solution method, integrating the Deer Hunting Optimization Algorithm (DHOA) and the Earhworm Optimization Algorithm (EWA), was proposed by Rajesh Kanna et al. (2021).

The main objective of their work was the development of a powerful heuristic for the solution of large-scale symmetric TSP instances. According to their numerical results, the proposed solution method outperformed several population-based metaheuristics and presented better computational times. Panwar & Deep (2021) integrated the classic Grey Wolf Optimization (GWO) metaheuristic with 2-opt local search operators to develop a new discrete GWO for the solution of symmetric TSP instances. To evaluate the performance of the new solution method, the authors conducted a comparative study on 17 symmetric TSP instances from TSPLib, with other nature-inspired metaheuristics, such as the Bat Algorithm and the Discrete Firefly Algorithm.

2.2. Recent advances and applications of VNS

Li & Tian (2016) developed a self-adaptive VNS algorithm for the solution of the prize-collecting vehicle routing problem. The proposed algorithm performs changes on the sequence of search operators according to the search history. More specifically, the authors implemented a two-level Basic VNS. In the first one, the selection of customers is performed, while in the second one, the sequence of the selected customers is improved. The self-adaptive mechanism is applied in the second level, according to a selection probability, which is computed based on the improvements achieved by its search operator. The computational analysis showed that, the two-level Basic VNS scheme within the adaptive mechanism performed better than the corresponding classic solution method with respect to all solved problem instances. Todosijević et al. (2016) proposed an adaptive GVNS heuristic algorithms for solving the unit commitment problem, by considering both different VND schemes and an adaptive re-ordering mechanism. The adaptive mechanism was designed to change the sequence of local search operators according to their improvements achieved in its previous iteration.

An improved VNS for the solution both of asymmetric and symmetric TSP was proposed by Hore et al. (2018). The improved variant of VNS utilized a clever choice during the construction of neighborhoods and a stochastic approach, such as the one applied in the simulated annealing method, to escape local optimum solutions. The authors underlined

the improved performance of their solution method based on their numerical experiments. Karakostas et al. (2019) proposed a problem-level data-driven GVNS algorithm for the solution of a new variant of the complex Location-Inventory-Routing Problem (LIRP). More specifically, the authors proposed an adaptive search strategy, which incorporates the first and best improvement search strategy according to the size of each problem instance.

Karakostas et al. (2022) developed an efficient hybrid GVNS algorithm for the solution of the Pollution LIRP (PLIRP). The proposed hybrid solution method considers both static and dynamic data to achieve a highly-adaptive performance. More specifically, according to the size of under solution problem instances, different intensity levels of shaking method and VNS variants were applied, while for some of those variants a local search operators' adaptive re-ordering mechanism was also used. The same authors proposed adaptive GVNS heuristic algorithms for the solution of the Fleet size and mix PLIRP (Karakostas et al., 2020b). A major novelty of their solution approach was the development of new adaptive shaking mechanisms. These intelligent components enhanced the performance of adaptive GVNS, which performed much better than both the conventional one and the well-known optimization solver, CPLEX.

Ren et al. (2020a) developed an improved VNS for the solution of the bi-objective mixed-energy fleet vehicle routing problem. The authors embedded a selection mechanism in a BVNS framework for the selection of partial elite solutions to continue the search in each next iteration of the method, in an effort to simultaneously achieve diversity, without being trapped in local optimum solutions. Ren et al. (2020b) developed an improved VNS solution method for the static bike-sharing rebalancing problem, under the consideration of depot inventory. More specifically, the authors addressed a hybridization of BVNS and GVNS, as well as a hybridization on the level of search strategy. The utilized search strategy is similar to the one proposed by Karakostas et al. (2019), as it applies first and best improvement strategy under conditions. The numerical results showed that, the proposed solution method outperformed the corresponding conventional GVNS.

2.3. Knowledge gap and research contribution

The conducted literature review reveals the consistent effort of research community to develop improved solution methods for efficiently tackling hard optimization problems. The proper configuration of the critical components of such solution methods remains a crucial factor, which significantly affects their performance. The adoption of data-driven machine learning procedures constitutes an effective approach for addressing this challenging process. In case of VNS and its main variants, only a limited number of contributions have applied intelligent procedures to improve the performance of the developed heuristic algorithms. The majority of those studies focused on the adoption of data-driven adaptive mechanisms exclusively into the intensification phase of the proposed algorithms. However, Karakostas et al. (2020b) have applied intelligent adaptive mechanisms to configure an effective structural order of the diversification component of their solution method. The authors have concluded that their approach has achieved improved solutions compared to its corresponding conventional method. These findings have led to the research question whether the simultaneous consideration of intelligent adaptive mechanisms into both intensification and diversification components of a GVNS heuristic can lead to further improvements compared to its conventional and single-adaptive corresponding variants.

3. The Double-Adaptive GVNS

This section is divided into two main parts. The first one attempts to provide a general structure of the novel GVNS scheme, while the second part provides a specific modification of this scheme for the solution of symmetric and asymmetric TSP instances. More specifically, the section of the general structure of DA-GVNS consists of information about GVNS, details about improvement and shaking adaptive mechanisms and the overview of the new scheme. The last part of this section presents the utilized local search and shaking operators, the adopted search strategy, and the overview of the solution method, as they are formed to solve the TSP.

3.1. General VNS

VNS is a trajectory-based metaheuristic framework which utilizes a variable number of neighborhoods instead of only one (fixed) neighborhood. The simplest version of this framework is also known as Basic VNS (BVNS). The BVNS scheme interchanges a neighborhood, one after the other, firstly for local search in it (intensification part) and afterward for shaking in it (diversification part) (Hansen et al., 2017). GVNS is another variant of VNS, which is characterized by a stronger intensification phase than BVNS. The key difference between GVNS and BVNS is that, the GVNS scheme -sequentially- performs local search in several neighborhoods one after the other (intensification part) and afterward it utilizes one or more neighborhoods for shaking (diversification part). The intensification phase of the GVNS scheme is also denoted as VND. Due to the simplicity and the efficiency of the GVNS, several recent research contributions have developed GVNS-based solution approaches for hard combinatorial optimization problems (Menéndez et al., 2017; Mikić et al., 2019; Karakostas et al., 2020a,b). A GVNS consists of a shaking phase, a VND variant as its main improvement phase, and a neighborhood change step (Hansen et al., 2017). Although several VND variants have been proposed in the literature, pipe-VND (pVND) has been experimentally proved as one of the most efficient (Karakostas et al., 2019, 2020b). Algorithm 1 provides the algorithmic procedure of pVND.

Algorithm 1 pipe-VND

procedure pVND_1(S, l_{max})
  $l = 1$
  while $l \leq l_{max}$ **do**
    $S' \leftarrow Local\_Search\_Operator\_l(S)$
    if $f(S') < f(S)$ **then**
      $S \leftarrow S'$
    else
      $l = l + 1$
    end if
  end while
  Return S
end procedure

A pVND method receives an incumbent solution and a set of local search operators, which are systematically applied. According to the pVND neighborhood change criterion, the search continues with the same local search operator while this one produces improvements. If no improvement found in an iteration, the search continues with the next local search operator in the set of neighborhoods (Hansen et al., 2017).

An overview of a GVNS algorithm is provided in Algorithm 2.

Algorithm 2 General VNS

procedure GVNS($S, k_{max}, max\_time, l_{max}$)
  while $time \leq max\_time$ **do**
    for $k \leftarrow 1, k_{max}$ **do**
      $S^* = Shake(S, l)$
      $S' = pVND(S^*, l_{max})$
      if $f(S^*) < f(S)$ **then**
        $S \leftarrow S^*$
      end if
    end for
  end while
  return S
end procedure

A GVNS method receives an initial feasible solution S, a value k_{max} which denotes the maximum number of iterations of the shaking phase, the number of local search operators utilized in the improvement phase, l_{max} and a stopping limit, such as a maximum CPU time of execution, max_time (Hansen et al., 2017).

3.2. Adaptive mechanisms

Various research contributions in the open literature have addressed the benefits of adopting adaptive re-ordering mechanisms within the improvement phase of a VNS solution method, in order to provide more execution time in the most promising local search operators in each iteration (Todosijević et al., 2016; Sevaux et al., 2018; Karakostas et al., 2022). Such intelligent learning components, usually, utilize short memories to store historical execution data, such as the improvements of each local search operator in previous iterations and use the stored information to re-order the operators in an effort to enhance the performance of improvement phase (Simeonova et al., 2018). These procedures are known as low-level data-driven machine learning components (Talbi, 2021). A pseudocode which provides the structure of a local search re-ordering adaptive mechanism is given in Algorithm 3.

Algorithm 3 Local_Search_Adaptive_Mechanism

procedure LOCAL_SEARCH_ADAPTIVE_MECHANISM($LSOperators\_Order, LSO\_Initial\_Order, Improvements\_Counter$)
  if no improvement is found in any neighborhood **then**
    $New\_LSOperators\_Order \leftarrow LSO\_Initial\_Order$
  end if
  if an improvement is found **then**
    $New\_LSOperators\_Order \leftarrow Descending\_Order(LSOperators\_Order, Improvements\_Counter)$
  end if
  $LSOperators\_Order \leftarrow New\_LSOperators\_Order$
  return $LSOperators\_Order$
end procedure

Typically, such intelligent mechanisms receive the current order of local search operators and a $1 \times l\_max$ vector, $LS\_Improvements\_Counter$, where l_max is the number of local search operators used in the improvement phase. Each position of this vector is assigned to each one of the available local search operators and stores the number of improvements achieved by using each of them, in previous iteration(s). According to the initial order of local search operators, several rules can be found in the literature. Commonly, the complexity- or random-based initial order are applied (Todosijević et al., 2016; Karakostas et al., 2022).

Recently, Karakostas et al. (2020b) focused on the development of adaptive shaking procedures to enhance the performance of a GVNS solution method for the solution of a complex supply chain network optimization problem. The authors found that the consideration of adaptive mechanisms during the diversification phase of their GVNS heuristic leads to higher quality solution than those obtained by the corresponding conventional method. Algorithm 4 provides the pseudocode of the shaking adaptive mechanism.

Algorithm 4 Shaking_Adaptive_Mechanism

procedure SHAKING_ADAPTIVE_MECHANISM($ShakingOrder, ShakingInitialOrder, Sh_{max}$)
  if no improvement is found in any neighborhood **then**
    $ShakingOrder = ShakingInitialOrder$
  end if
  if an improvement is found **then**
    for $i \leftarrow 1, Sh_{max}$ **do**
      $l =$ Operator with maximum number of improvements
      $ShakingOperatorChecked(l) = .true.$
      $NewShakingOrder(i) = l$
    end for
    $ShakingOrder = NewShakingOrder$
  end if
  return $ShakingOrder$
end procedure

The shaking adaptive mechanism receives the current order of shaking operators, $ShakingOrder$, their initial order, $ShakingInitialOrder$, and the maximum number of shaking operators, Sh_{max}. In case of not achieving any improvement in the previous iteration, the order of shaking operators returns to the initial one. Otherwise, a descending re-ordering of the shaking operators is performed.

3.3. Overview of the proposed solution method

This section provides the procedural overview of the DA-GVNS, using the pVND in the improvement phase, and the intensified shake as its diversification phase. The overall method is summarized in Algorithm 5.

Algorithm 5 Double Adaptive GVNS

procedure DA-GVNS($S, k_{max}, max\_time, l_{max}, LSO\_Initial\_Order, ShakingInitialOrder$)
  while $time \leq max\_time$ **do**
    $ShakingOrder = Shaking\_Adaptive\_Mechanism(ShakingOrder, ShakingInitialOrder, Sh_{max})$
    for $k \leftarrow 1, k_{max}$ **do**
      for $i \leftarrow 1, Sh_{max}$ **do**
        $l = ShakingOrder(i)$
        $S^* = Shake(S, l)$
        $Local\_Search\_Adaptive\_Mechanism(LSOperators\_Order, Improvements\_Counter)$
        $S' = pVND(S^*, l_{max}, LSOperators\_Order)$
        if $f(S') < f(S)$ **then**
          $S \leftarrow S'$
        end if
      end for
    end for
  end while
  return S
end procedure

The input for the DA-GVNS are:

- an initial feasible solution, S
- an integer value of k_{max} parameter, which denotes the maximum number of consecutive applications of a shaking operator,
- the maximum CPU execution time,
- the number of local search operators included in the pVND procedure,
- an initial order of shaking operators, $ShakingInitialOrder$
- an initial order of local search operators, $LSO\_Initial\_Order$

In each iteration a re-ordering of the shaking operators is performed in descending order, based on the improvements achieved by applying each of them in each previous iteration. Either in the first or in each unsuccessful iteration, an initial predefined shaking order (from the most to the least complex operator) is adopted. Next, for each shaking intensity value (line 4) the shaking operators are successively selected according to their order, and the incumbent solution, S is diversified (line 7) to produce a new solution, S^*. Then, the local search operators in pVND are being re-ordered (line 8) based on their previous performance (a predefined initial order is adopted similarly to the shaking process). Considering the new order of local search operators, the pVND is applied on the solution S^* to produce a new, hopefully improved, solution S' (line 9). The new solution is then compared with the currently best one (line 10) to examine whether any improvement was achieved (line 11). This process is iteratively executed until the maximum CPU execution time is reached.

3.4. Modifications on DA-GVNS for the solution of TSP

3.4.1. Initial solution

DA-GVNS receives an initial solution and iteratively attempts to improve it. The classic Nearest Neighbor heuristic (Flood, 1956) is applied to construct a feasible initial solution of TSP instances.

3.4.2. Neighborhood structures

As the main objective of this work is to investigate the potential enhancement of the performance of the GVNS scheme, by considering both local search and shaking adaptive mechanisms, three classic and commonly adopted neighborhood structures are utilized. More specifically, the same neighborhood structures are used in improvement and diversification phases. These neighborhood structures are the $1-0$ *relocate*, which relocates the position of a node in a route, the $1-1$ *exchange*, also known as *swap* operator, which exchanges two nodes in a given route, and the $2-$ *opt* operator, which removes and reconnects two edges in the route. The solution representation adopted in this work is the path representation. Thus, each local search operator modifies an incumbent solution by iteratively applying single moves, which leads to a new solution path. An illustrative example of such changes of the

solution path made by a single application of each local search operator, on a six customers' path representation route, is provided in Figure 1. Despite relocate and exchange constitute classic path-based search operators, 2-opt is typically considered as an edge-based operator. However, the 2-opt local search operator using an edge-based representation is equivalent to the sub-tour reversal algorithm using a path representation Hillier & Lieberman (2021).

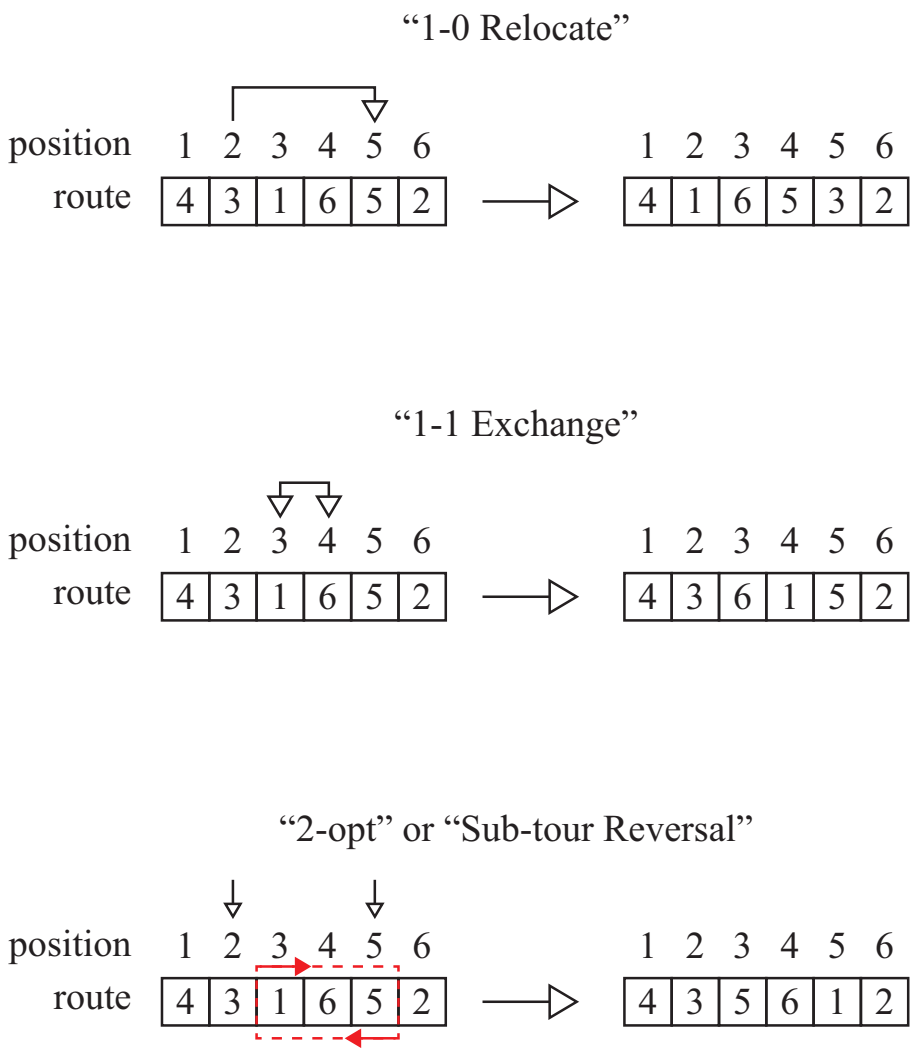

Figure 1: An illustrated example of the changes made in a route after a single application of each local search operator.

In case of the $1-0$ *relocate* search operator, node 3 is removed from its current position and it is inserted exactly after node 5. This move leads to the dropping of the edges $(4, 3)$,

$(3,1)$, and $(5,2)$ and to the addition of the edges $(4,1)$, $(5,3)$, and $(3,2)$. In the example of $1-1$ *exchange*, nodes 1 and 6 are selected and being swapped. This moves modifies the route by dropping the edges $(3,1)$, $(1,6)$, and $(6,5)$ and by adding the edges $(3,6)$, $(6,1)$ and $(1,5)$. It should be mentioned that, no cost change will occurred by the replacement of the edge $(1,6)$ from the edge $(6,1)$ in case of a symmetric TSP instance. Finally, in the example of the $2-opt$ operator or sub-tour reversal algorithm, the edges $(3,1)$ and $(5,2)$ are dropped and the edges $(3,5)$ and $(1,2)$ are added into the solution. In terms of implementation, this move of $2-opt$ is performed by selecting the nodes 3 and 2, and reversing the sub-tour $[1,6,5]$.

4. Computational analysis and results

4.1. Computing environment

The proposed DA-GVNS and the other developed GVNS variants were coded in Fortran, and they were executed by Intel Fortran compiler 18.0 using the optimization option /O3. The execution was performed on a laptop PC running Windows 10 Home 64-bit with an Intel Core i7-9750H CPU at 2.6 Ghz and 16 GB RAM. The computational experiments were conducted on symmetric and asymmetric TSP instances of TSPLib (`http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95`) with a CPU execution time limit of $60s$. It should be clarified that, the following reported results are the average and the best objective values of 10 runs for each problem instance.

4.2. k_{max} computational analysis

The shaking intensity is a critical parameter for the efficiency of a GVNS solution method, which should be tuned. Small values of this parameter cannot guarantee the avoidance of being trapped in local optimum solutions, while greater values can lead to random permutations (Hansen & Mladenović, 2014; Hansen et al., 2017). The selection of the most suitable value of k_{max} parameter, in each developed GVNS method, is made empirically through computational experimentation. This process is known as off-line parameter initialization strategy (Talbi, 2009). This section presents the computational analysis performed

to evaluate the impact of different shaking levels on the performance of the developed GVNS variants. Table 1 summarizes the average objective values achieved by each GVNS variant when solving asymmetric TSP instances.

Table 1: k_{max} analysis on the performance of $GVNS$ methods on 19 asymmetric TSP instances

	GVNS			$LSA-GVNS$			$SHA-GVNS$			$DA-GVNS$		
Instance	$k_{max}=8$	$k_{max}=10$	$k_{max}=12$	$k_{max}=8$	$k_{max}=10$	$k_{max}=12$	$k_{max}=8$	$k_{max}=10$	$k_{max}=12$	$k_{max}=8$	$k_{max}=10$	$k_{max}=12$
br17	39	39	39	39	39	39	39	39	39	39	39	39
ft53	7039	7080	7064	7043	7183	7053	7132	7048	7072	7011	7018	7011
ft70	39590	39791	39545	39453	39654	39769	39963	39811	39756	39585	39650	39874
ftv33	1286	1286	1286	1286	1286	1286	1286	1286	1296	1286	1286	1286
ftv35	1481	1475	1481	1480	1477	1478	1475	1475	1476	1473	1473	1479
ftv38	1539	1540	1542	1535	1542	1540	1538	1539	1535	1535	1534	1538
ftv44	1651	1629	1616	1634	1640	1633	1638	1627	1629	1631	1640	1647
ftv47	1802	1804	1792	1818	1806	1793	1808	1798	1803	1788	1786	1801
ftv55	1663	1635	1648	1636	1626	1668	1653	1648	1651	1636	1648	1657
ftv64	1895	1939	1911	1904	1888	1930	1907	1911	1908	1895	1889	1895
ftv70	2094	2093	2101	2058	2104	2111	2063	2082	2119	2078	2098	2092
ftv170	3399	3407	3504	3363	3364	3415	3456	3410	3511	3307	3339	3418
kro124p	36788	36586	36546	36865	36714	37020	36451	36492	36701	36403	36544	36470
p43	5620	5620	5620	5620	5620	5620	5620	5620	5620	5620	5620	5620
rbg323	1461	1481	1485	1450	1450	1443	1450	1467	1487	1451	1455	1451
rbg358	1298	1304	1290	1288	1278	1284	1295	1277	1298	1276	1290	1280
rbg403	2489	2493	2508	2482	2488	2486	2492	2490	2513	2481	2493	2491
rbg443	2769	2766	2766	2767	2761	2762	2774	2781	2768	2761	2769	2767
ry48p	14468	14453	14445	14451	14468	14441	14451	14478	14473	14465	14451	14436
Average	6756.37	6759	**6746.79**	**6745.89**	6757.26	6777.42	6762.68	**6751.53**	6771.32	**6722.16**	6738	6750.11

It should be clarified that different k_{max} values don't always lead to significant performance fluctuations. However, it is always critical to investigate which value fits better in each problem type. To this end, using $k_{max} = 12$ has led to better results in case of conventional GVNS scheme. The numerical experiments on the effect of different shaking intensity on the performance of LS-Adaptive GVNS showed that $k_{max} = 8$ has led to slightly better solutions compared to those obtained with $k_{max} = 10$ or $k_{max} = 12$. Moreover, the GVNS scheme enhanced by a shaking adaptive mechanism performs better with $k_{max} = 10$. Finally, the off-line parameter testing showed that DA-GVNS performs better under the consideration of a limited shaking intensity, such the $k_{max} = 8$.

4.3. Computational results on asymmetric TSP instances

Tables 2 provides the average objective values found by using all GVNS variants.

Table 2: The average results of asymmetric TSP instances using GVNS variants

Instance	GVNS	LSA-GVNS	SHA-GVNS	DA-GVNS
br17	39	39	39	39
ft53	7064	7043	7048	**7011**
ft70	39545	**39453**	39811	39585
ftv33	1286	1286	1286	1286
ftv35	1481	1480	1475	**1473**
ftv38	1542	1535	1539	1535
ftv44	**1616**	1634	1627	1636
ftv47	1792	1818	1798	**1788**
ftv55	1648	1636	1648	1636
ftv64	1911	1904	1911	**1895**
ftv70	2101	**2058**	2082	2078
ftv170	3504	3363	3410	**3307**
kro124p	36546	36865	36492	**36403**
p43	5620	5620	5620	5620
rbg323	1485	**1450**	1467	1451
rbg358	1290	1288	1277	**1276**
rbg403	2508	2482	2490	**2481**
rbg443	2766	2767	2781	**2761**
ry48p	**14445**	14451	14478	14465
Average	6746.79	6745.89	6751.53	**6722.16**

The DA-GVNS produced better solutions in terms of quality compared to those obtained by other GVNS variants, as it is indicated by the average costs presented in Table 2. It is clear that the consideration of intelligent re-ordering mechanisms both in intensification and diversification phase has a positive impact on the performance of the GVNS scheme. This enhancement may be attributed by the potential creation of better patterns of improvement and shaking operators, which can lead to the exploration of more promising areas in the search space, as well as by providing more execution time to the most successful operators.

To further evaluate the performance of DA-GVNS on the solution of asymmetric TSP instances, numerical comparisons were conducted with recent research contributions. More specifically, the results obtained by the DA-GVNS was compared to those achieved by a discrete water cycle algorithm proposed by Osaba et al. (2018) and an improved Harmony Search algorithm, proposed by Boryczka & Szwarc (2019). Table 3 provides these numerical

comparisons. Herein, it should be clarified that the bold highlighting in the following Tables is applied to mention the best values achieved among the tested methods. The best-known values are given as a reference information.

Table 3: Average results of DA-GVNS and recent literature contributions on 19 asymmetric TSP instances.

Instance	Best-known	DA-GVNS	DWCA	HS
br17	39	39	39	39
ft53	6905	**7011**	7199.4	7148.3
ft70	38673	**39585**	40111.1	39722.03
ftv33	1286	**1286**	1308.7	1320.57
ftv35	1473	**1473**	1485.8	1490.6
ftv38	1530	**1535**	1549	1547.13
ftv44	1613	**1631**	1665	1645.4
ftv47	1778	**1788**	1827.8	1800.43
ftv55	1608	1636	1691.4	**1625.17**
ftv64	1839	1895	1961	**1876.2**
ftv70	1950	2078	2126.2	**2027.83**
ftv170	2755	**3307**	-	3393.07
kro124p	36230	**36403**	39252.8	38348.2
p43	5620	5620	5620	5620.27
rbg323	1326	**1451**	-	1555.4
rbg358	1163	**1276**	-	1424.63
rbg403	2465	**2481**	-	2637.8
rbg443	2720	**2761**	-	2914.33
ry48p	14422	**14465**	14517.8	14513.9
Average	6599.7	6722.2	8596.79*	6876.3

The asterisk symbol, in the average value of the results achieved by the DWCA, denotes that this value refers to the available solutions in the work of Boryczka & Szwarc (2019). The corresponding average value in case of DA-GVNS is 8317.5. To this end, it is clear that the DA-GVNS performed 3.25% better than the DWCA and 2.24% better than the improved HS.

4.4. Computational results on symmetric TSP instances

According to numerical testing, all GVNS variants produced best results, using the same k_{max} values adopted in case of the solution of asymmetric TSP instances. Thus, for the

solution of symmetric TSP instances, the DA-GVNS had a shaking intensity of $k_{max} = 8$, the LSA-GVNS used $k_{max} = 8$, the SHA-GVNS used $k_{max} = 10$, and the GVNS adopted a shaking intensity of $k_{max} = 12$. The average and best performance of all GVNS variants on the solution of symmetric TSP instances are illustrated in Figures 2 and 3 correspondingly.

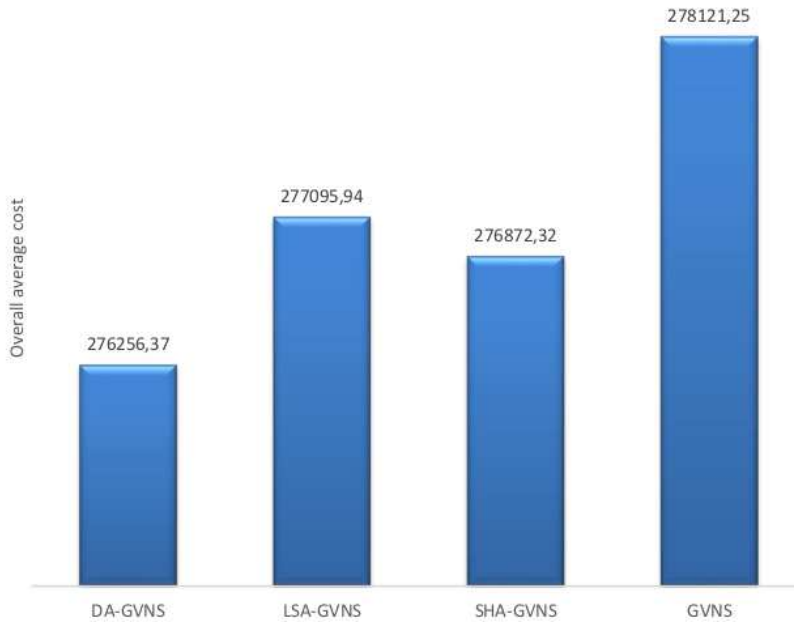

Figure 2: Comparison between GVNS variants on symmetric TSP results in average.

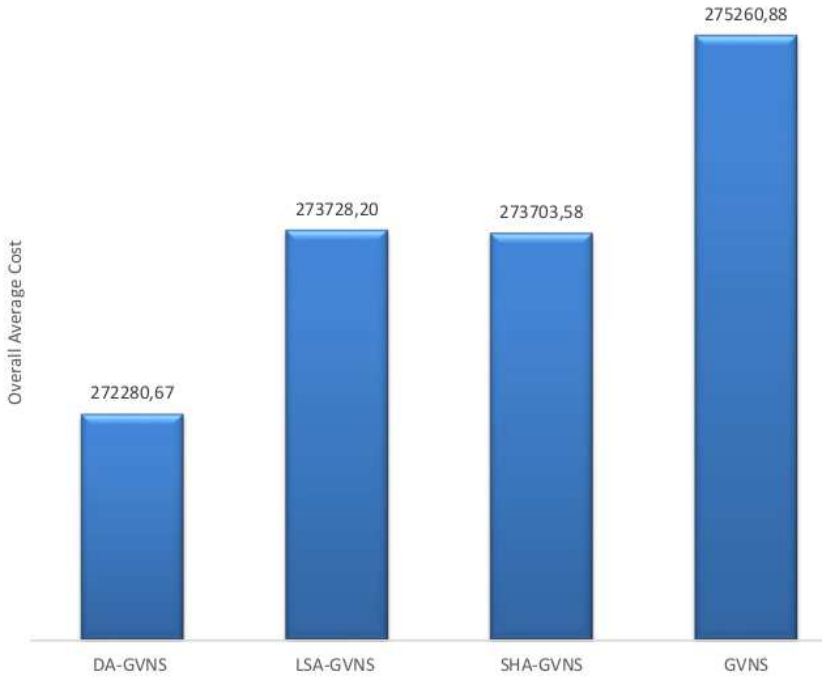

Figure 3: Comparison between GVNS variants on best found symmetric TSP results.

As it has been observed in case of asymmetric TSP, the consideration of the double-adaptive mechanism within the GVNS scheme has, also, a positive impact on its performance in the solution of symmetric problem instances.

To further evaluate the performance of DA-GVNS, it was compared to other recently proposed solution methods in the open literature for the TSP. More specifically, the results achieved by the DA-GVNS were compared to those obtained by an improved VNS (Hore et al., 2018), an improved HS (Osaba et al., 2018), and a discrete GWO (D-GWO) (Panwar & Deep, 2021). Table 4 summarizes these computational comparisons.

Table 4: Comparisons between DA-GVNS and recently proposed methods for the solution of symmetric TSP instances (average values).

Instance	**Best-known**	**DA-GVNS**	**iVNS**	**HS**	**D-GWO**
bays29	2020	**2020**	**2020**	-	-
berlin52	7542	7542	7544.36	7542	-
bier127	118282	119122	**119006.4**	-	-
brazil58	25395	**25395**	25592.72	-	-
ch130	6110	6154	**6153.72**	-	-
ch150	6528	**6595**	6644.95	-	-
d198	15780	**15855**	16079.28	-	-
d1291	50801	**54778**	56095.33	-	-
d1655	62128	**67292**	70337.23	-	-
dantzig42	699	699	699	-	-
eil51	426	**426**	428.98	428.4	-
eil76	538	**538**	552.57	547.9	-
eil101	629	**633**	648.27	645.9	-
fl417	11861	**12019**	12183.14	-	-
fl1400	20127	21858	**21085.98**	-	-
fri26	937	937	937	-	-
gil262	2378	**2451**	2501.86	-	-
gr17	2085	2085	2085	-	-
gr21	2707	2707	2707	-	-
gr24	1272	1272	1272	-	-
gr48	5046	5046	5046	-	-
kroA100	21282	**21282**	21695.79	21348.1	-

Continued on next page

Table 4 – continued from previous page

Instance	Best-known	DA-GVNS	iVNS	HS	D-GWO
kroB100	22141	22165	**22140.2**	22450.7	22444.6
kroC100	20749	**20749**	20809.29	20934.7	21078
kroD100	21294	**21294**	21490.62	21529.6	-
kroE100	22068	**22121**	22193.8	22246.2	22410
kroA150	26524	**26817**	26947.17	-	-
kroB150	26130	**26256**	26537.04	-	26756.2
kroA200	29368	**29807**	30339.67	-	-
kroB200	29437	**30015**	30453.22	-	-
lin105	14379	**14390**	14395.64	-	14520
lin318	42029	**43201**	43964.93	-	-
pcb442	50778	53009	**50800.24**	-	-
pcb1173	56892	**61725**	63435.95	-	-
pcb3038	137694	**152209**	154565.4	-	-
pr76	108159	108159	108159	-	108900
pr107	44303	**44303**	44314.92	44647.1	44685.1
pr124	59030	**59050**	59051.82	59338.9	59390.9
pr136	96772	**97062**	97985.84	98761.4	99310.5
pr144	58537	**58537**	58563.97	58734.6	58600.5
pr152	73682	**73839**	73855.11	74202.6	74230
pr226	80369	80880	**80514.64**	-	81135.7
pr264	49135	49880	51197.14	**49528.6**	-
pr299	48191	**49719**	50373.12	-	-
pr439	107217	112600	**111771.2**	-	112850.3
pr1002	259045	277867	280563.9	-	**267713.2**
rat99	1211	**1211**	1241.26	-	-

Continued on next page

Table 4 – continued from previous page

Instance	Best-known	DA-GVNS	iVNS	HS	D-GWO
rat195	2323	**2364**	2453.81	-	-
rat575	6773	**7179**	7362.51	-	-
rat783	8806	**9445**	9707.36	-	-
rd100	7910	**7910**	7918.36	-	-
rd400	15281	**15915**	16250.21	-	-
rl1323	270199	**292819**	295611.2	-	-
st70	675	**675**	677.11	678.6	-
swiss42	1273	1273	1273	-	-
u159	42080	**42168**	42467.61	-	42563.3
u574	36905	**39583**	39629.11	-	-
u724	41910	**44814**	45729.71	-	-
u2319	234256	**243332**	262595.6	-	-
vm1748	336556	369787	**366757.8**	-	-
Average	47578	49847.25	50423.58	31472.83*	70439.22*

The results of symmetric TSP instances indicate the significant role of adaptive mechanisms on the performance of the GVNS. It is crucial to mention that even considering classic search operators, the data-driven intelligent re-ordering mechanisms can enhance the performance of the GVNS algorithm, in order to perform better than other improved VNS implementations, such as the single-adaptive GVNS variants and the improved VNS of Hore et al. (2018). Also, the DA-GVNS performs better than the improved HS of Osaba et al. (2018) and almost equivalently to the D-GWO of Panwar & Deep (2021). The average values of DA-GVNS in problem instances solved by improved HS and D-GWO are 31256 (improved by 0.69%) and 70676.4 (0.34% worse) correspondingly. Thus, the DA-GVNS provides a good base for further improvements, such as the consideration of improved local search

operators, to construct an even more efficient heuristic algorithm.

5. Conclusions

This work presents a novel GVNS scheme, which considers intelligent, low-level data-driven, mechanisms both in intensification and diversification phases, to efficiently re-order the search operators, and finally to enhance the overall performance of the algorithm. To evaluate the performance of the new GVNS variant, single-adaptive and the conventional GVNS were also implemented and properly modified to be able to solve both symmetric and asymmetric TSP instances. The conducted computational study showed that, the consideration of the double-adaptive mechanism has a significant positive effect on the performance of the GVNS method. Moreover, further computational comparisons were performed between the proposed DA-GVNS and other recently proposed solution methods. These additional comparisons showed that DA-GVNS, even with the adoption of non-improved local search operators, can provide high quality solutions and perform better than other efficient methods in the open literature.

The aforementioned remarks stated the benefits of integrating intelligent learning components in the main process of classic metaheuristic frameworks, such as the VNS. To this end, an interesting future work direction could focus on the development of wider adaptive mechanisms. More specifically, adaptive mechanisms for the selection of VND variants and k_{max} during the execution of a GVNS solution method can potentially enhance the performance of the DA-GVNS. Moreover, the selection and the implementation of improved local search operators in the DA-GVNS can also lead to further improvements. Another interesting future research direction is the further investigation of the performance of the DA-GVNS on the solution of more complex supply chain network optimization problems.

References

Bhatnagar, R., & Teo, C. (2009). Role of logistics in enhancing competitive advantage: A value chain framework for global supply chains. *International Journal of Physical Distribution & Logistics Management*, *39*, 202–226.

Boryczka, U., & Szwarc, K. (2019). The Harmony Search algorithm with additional improvement of harmony memory for Asymmetric Traveling Salesman Problem. *Expert Systems with Applications*, *122*, 45–53.

Campuzano, G., Oberque, C., & Aguayo, M. M. (2020). Accelerating the Miller–Tucker–Zemlin model for the asymmetric traveling salesman problem. *Expert Systems with Applications*, *148*, 113229.

Ebadinezhad, S. (2020). DEACO: Adopting dynamic evaporation strategy to enhance ACO algorithm for the traveling salesman problem. *Engineering Applications of Artificial Intelligence*, *92*, 103649.

Flood, M. (1956). The Traveling-Salesman Problem. *Operations Research*, *4*, 61–75.

Grazia Speranza, M. (2018). Trends in transportation and logistics. *European Journal of Operational Research*, *264*, 830–836.

Hansen, P., & Mladenović, N. (2014). Variable Neighborhood Search. In E. Burke, & G. Kendall (Eds.), *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques* (pp. 313–337). New York: Springer Science+Business Media.

Hansen, P., Mladenović, N., Todosijević, R., & Hanafi, S. (2017). Variable neighborhood search: Basics and variants. *EURO Journal on Computational Optimization*, *5*, 423–454.

Hillier, F. S., & Lieberman, G. J. (2021). *Introduction to Operations Research*. (11th ed.). McGraw Hill.

Hore, S., Chatterjee, A., & Deqanji, A. (2018). Improving variable neighborhood search to solve the traveling salesman problem. *Applied Soft Computing*, *68*, 83–91.

Karakostas, P., Panoskaltsis, N., Mantalaris, A., & Georgiadis, M. (2020a). Optimization of CAR T-cell therapies supply chains. *Computers & Chemical Engineering*, *139*, 106913.

Karakostas, P., Sifaleras, A., & Georgiadis, C. (2019). A general variable neighborhood search-based solution approach for the location-inventory-routing problem with distribution outsourcing. *Computers & Chemical Engineering*, *126*, 263–279.

Karakostas, P., Sifaleras, A., & Georgiadis, C. (2020b). Adaptive variable neighborhood search solution methods for the fleet size and mix pollution location-inventory-routing problem. *Expert Systems with Applications*, *153*, 113444.

Karakostas, P., Sifaleras, A., & Georgiadis, C. (2022). Variable neighborhood search-based solution methods for the pollution location-inventory-routing problem. *Optimization Letters*, *16*, 211–235.

Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A. M., & Talbi, E. (2022). Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. *European Journal of Operational Research*, *296*, 393–422.

Li, K., & Tian, H. (2016). A two-level self-adaptive variable neighborhood search algorithm for the prize-collecting vehicle routing problem. *Applied Soft Computing*, *43*, 469–479.

Menéndez, B., Bustillo, M., Pardo, E. G., & Duarte, A. (2017). General variable neighborhood search for the order batching and sequencing problem. *European Journal of Operational Research*, *263*, 82–93.

Mikić, M., Todosijević, R., & Urošević, D. (2019). Less is more: General variable neighborhood search for the capacitated modular hub location problem. *Computers & Operations Research*, *110*, 101–115.

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

Osaba, E., Del Ser, J., Sadollah, A., Bilbao, M. N., & Camacho, D. (2018). A discrete water cycle algorithm for solving the symmetric and asymmetric traveling salesman problem. *Applied Soft Computing*, *71*, 277–290.

Panwar, K., & Deep, K. (2021). Discrete Grey Wolf Optimizer for symmetric travelling salesman problem. *Applied Soft Computing*, *105*, 107298.

Papalitsas, C., Karakostas, P., & Andronikos, T. (2019). A Performance Study of the Impact of Different Perturbation Methods on the Efficiency of GVNS for Solving TSP. *Applied System Innovation*, *2*, 31.

Punnen, A. P. (2007). The Traveling Salesman Problem: Applications, Formulations and Variations. In G. Gutin, & A. P. Punnen (Eds.), *The Traveling Salesman Problem and Its Variations*. Springer, Boston, MA volume 12.

Rajesh Kanna, S., Sivakumar, K., & Lingaraj, N. (2021). Development of Deer Hunting linked Earthworm Optimization Algorithm for solving large scale Traveling Salesman Problem. *Knowledge-Based Systems*, *227*, 107199.

Ren, X., Huang, H., Feng, S., & Liang, G. (2020a). An improved variable neighborhood search for bi-objective mixed-energy fleet vehicle routing problem. *Journal of Cleaner Production*, *275*, 124155.

Ren, Y., Meng, L., Zhao, F., Zhang, C., Guo, H., Tian, Y., Tong, W., & Sutherland, J. W. (2020b). An improved general variable neighborhood search for a static bike-sharing rebalancing problem considering the depot inventory. *Expert Systems with Applications*, *160*, 113752.

Sevaux, M., Sorensen, K., & Pillay, N. (2018). Adaptive and multilevel metaheuristics. In R. Martí, P. Pardalos, & M. Resende (Eds.), *Handbook of Heuristics* (pp. 1–19). Springer.

Simeonova, L., Wassan, N., Salhi, S., & Nagy, G. (2018). The heterogeneous fleet vehicle routing problem with light loads and overtime: Formulation and population variable neighborhood search with adaptive memory. *Expert Systems with Applications*, *114*, 183–195.

Talbi, E. G. (2009). *Metaheuristics: From Design to Implementation*. Hoboken, New Jersey: John Wiley & Sons, Inc.

Talbi, E.-G. (2021). Machine learning into metaheuristics: A survey and taxonomy. *ACM Computing Surveys*, *54*, 1–32.

Todosijević, R., Mladenović, M., Hanafi, S., Mladenović, N., & Crévits, I. (2016). Adaptive general variable neighborhood search heuristics for solving the unit commitment problem. *International Journal of Electrical Power & Energy Systems*, *78*, 873–883.

Wang, Y., Wu, Y. W., & Xu, N. (2019). Discrete symbiotic organism search with excellence coefficients and self-escape for traveling salesman problem. *Computers & Industrial Engineering*, *131*, 269–281.

Zhong, Y., Wang, L., Lin, M., & Zhang, H. (2019). Discrete pigeon-inspired optimization algorithm with Metropolis acceptance criterion for large-scale traveling salesman problem. *Swarm and Evolutionary Computation*, *48*, 134–144.