Swarm and Evolutionary Computation 94 (2025) 101887

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Learning-assisted improvements in Adaptive Variable Neighborhood Search

Panagiotis Karakostas, Angelo Sifaleras [*]

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

ARTICLE INFO

Keywords:
Learning-assisted intelligent optimization
Adaptive search
Metaheuristics
Traveling Salesman Problem
Quadratic Assignment Problem

ABSTRACT

This study presents the design and integration of novel adaptive components within the Double-Adaptive General Variable Neighborhood Search (DA-GVNS) algorithm, aimed at improving its overall efficiency. These adaptations utilize iteration-based data to refine the search process, with enhancements such as an adaptive reordering mechanism in the refinement phase and a knowledge-guided approach to adjust the search strategy. Additionally, an adaptive mechanism for dynamically controlling the shaking intensity was introduced. The proposed knowledge-guided adaptations demonstrated superior performance over the original DA-GVNS framework, with the most effective scheme selected for further evaluation. Initially, the symmetric Traveling Salesman Problem (TSP) was used as a benchmark to quantify the impact of these mechanisms, showing significant improvements through rigorous statistical analysis. A comparative study was then conducted against six advanced heuristics from the literature. Finally, the most promising knowledge-guided GVNS (KG-GVNS) was tested against the original DA-GVNS on selected instances of the Quadratic Assignment Problem (QAP), where detailed statistical analysis highlighted its competitive advantage and robustness in addressing complex combinatorial optimization problems.

1. Introduction

Metaheuristics, an essential class of optimization techniques, play a pivotal role in informing decision-making processes to address complex and challenging practical optimization problems. These sophisticated algorithms are indispensable when traditional methods prove inadequate due to the combinatorial nature, nonlinearities, or sheer computational complexity of real-world problems encountered in various domains such as logistics, manufacturing, finance, and transportation, among others [1,2]. Metaheuristics excel in navigating large solution spaces, leveraging their ability to explore and exploit the problem landscape efficiently. By cleverly balancing exploration and exploitation strategies, metaheuristics can effectively guide decision-makers towards near-optimal or satisfactory solutions, even when an exact solution is elusive within a reasonable time frame [3]. This adaptability, robustness, and capacity for fine-tuning make metaheuristics indispensable tools for tackling the intricate optimization challenges that underpin critical decision-making processes across various sectors, thereby facilitating informed choices and improving operational outcomes. Their continued advancement and application remain instrumental in addressing the complexities of contemporary real-world optimization problems [4].

The imperative to develop novel, valid, and improved variants of metaheuristics is paramount within the domain of optimization science. It is essential that researchers devote their efforts to genuine advancements in algorithmic design rather than simply repackaging existing methods under different appellations [5]. The primary motivation to explore new variants stems from the continuously evolving landscape of optimization challenges. As real-world problems grow in complexity and scale, and as computational resources continue to advance, there is an inherent need to adapt and innovate. Novel variants of metaheuristics present opportunities to address unique aspects of problem domains, improve efficiency, and deliver more robust, accurate, and reliable solutions. By focusing on substantive improvements rather than cosmetic modifications, scientists can unlock the true potential of metaheuristics to facilitate optimal decision-making across various disciplines, ultimately advancing the frontiers of optimization research and practice.

A particularly promising research avenue in the field of optimization lies in the development of adaptive mechanisms that can be seamlessly integrated into critical components of metaheuristic algorithms [4,6–8]. These mechanisms can operate at both low and high levels, offering the potential to dynamically tailor the algorithm's behavior to suit the specific characteristics and demands of the optimization problem at hand. Low-level adaptive mechanisms, involving the development of tailored mechanisms from the ground up, and high-level adaptive mechanisms, leveraging advanced machine learning techniques to

* Corresponding author.
E-mail addresses: pankarakostas@uom.edu.gr (P. Karakostas), sifalera@uom.gr (A. Sifaleras).

https://doi.org/10.1016/j.swevo.2025.101887
Received 21 June 2024; Received in revised form 15 December 2024; Accepted 7 February 2025

adapt key components of a metaheuristic, represent two promising avenues for enhancing the efficacy of these optimization algorithms [9]. By integrating intelligent, data-driven mechanisms into the core processes of these algorithms, it becomes possible to construct improved solution methodologies that excel in terms of solution quality and robustness [10].

In alignment with this research direction, a recent contribution to the field, known as the Double-Adaptive General Variable Neighborhood Search (DA-GVNS), has been introduced [10]. This heuristic approach strategically incorporates low-level adaptive mechanisms into both the improvement and shaking phases of the renowned Variable Neighborhood Search (VNS) variant, the General Variable Neighborhood Search (GVNS). The empirical investigation carried out in the aforementioned research study elucidated the beneficial influence exerted by these adaptive components on the overall performance of GVNS. In the realm of VNS-based solution methods, a recent survey conducted by Brimberg et al. (2023) has accentuated the imperative to explore advanced adaptive strategies. This includes the exploration of sophisticated approaches for reordering search operators and the development and integration of intelligent mechanisms aimed at selecting critical method parameters, such as the intensity level for the shaking phase.

In alignment with the aforementioned research objectives, the present study embarks on an exploration of innovative adaptive elements, with the primary aim of improving the computational efficiency of DA-GVNS. The effectiveness of these enhancements is evaluated using the Traveling Salesman Problem (TSP) and the Quadratic Assignment Problem (QAP) as benchmark test cases. To this end, this study encompasses the following key research contributions:

- The incorporation of a novel adaptive re-ordering mechanism within the improvement phase of the DA-GVNS.
- Development of a knowledge-based adaptive mechanism aimed at dynamically regulating the level of shaking intensity throughout each iteration of the DA-GVNS.
- The introduction of an innovative knowledge-guided adaptive mechanism, strategically devised to facilitate the judicious application of the adaptive search strategy.
- The efficiency of the newly proposed Knowledge Guided - GVNS (KG-GVNS) has been rigorously substantiated by applying a robust statistical testing methodology.

The paper is structured as follows: Section 2 presents the novel adaptive components proposed for the DA-GVNS, followed by Section 3, which outlines the computational study conducted to assess the performance of the new DA-GVNS method. Finally, concluding remarks and future research directions are provided in Section 4.

2. Novel adaptive mechanisms for the DA-GVNS

2.1. Adaptive strategies in Metaheuristics

Building on the groundwork laid in the introductory sections, the focus now shifts to a critical aspect of modern metaheuristics: the adaptive mechanisms that enhance algorithmic performance by dynamically adjusting to problem-specific conditions and the evolving search landscape. These adaptive strategies, categorized into three primary groups, are central to the robustness and efficiency of metaheuristics.

1. **Adaptive Operators' Management**. This category encompasses mechanisms such as Adaptive Operator Selection [12,13] and Adaptive Neighborhood Management [10,14–17], which are crucial for dynamically selecting and applying the most effective operators based on their performance histories. These adaptations are instrumental in optimizing the search process by continuously refining the tactics used for exploration and exploitation, ensuring that the approach of metaheuristic is ideally suited to the current state of the search.
2. **Adaptive Parameter Tuning**. Crucial algorithmic parameters are dynamically adjusted to enhance search behavior, ensuring the search process is efficient and effective under varying conditions. For instance, the tabu tenure in Tabu Search adjusts the memory length of the search history to prevent cyclic behavior [18], while the temperature parameter in Simulated Annealing crucially modulates the acceptance probability of suboptimal solutions, facilitating escape from local minima [19]. Similarly, the alpha parameter in Reactive Greedy Randomized Adaptive Search Procedures (GRASP) modulates the balance between greediness and randomness during the construction phase, adapting the search strategy to the requirements of the problem [20]. In evolutionary algorithms such as Genetic Algorithms (GA) and Differential Evolution, mutation and crossover parameters are pivotal in managing genetic variance to foster exploration and avoid premature convergence [21]. In Ant Colony Optimization (ACO), the pheromone evaporation rate is a key parameter that influences how quickly the pheromone trails, which guide the search of artificial ants, decay; this prevents the algorithm from overemphasizing previously successful paths and encourages exploration of new solutions [22]. Additionally, the population size in metaheuristics like GA and Particle Swarm Optimization (PSO) is crucial for maintaining genetic diversity and ensuring a robust search process [23,24].
3. **Adaptive Control of Exploration–Exploitation Balance**. This category encompasses adaptive mechanisms that effectively manage the balance between exploring new regions of the search space and exploiting well-understood areas to refine solutions. Techniques such as the Adaptive Search Strategy selectively apply different search methodologies, including first improvement and best improvement strategies, to optimize solution development processes [10,25]. Adaptive Restart Strategies involve reinitializing the search from new starting points to overcome local optima [20]. Additionally, Advanced Diversity Control mechanisms are employed to prevent the algorithm from stagnating due to limited exploration of the solution space or from inefficiently exploring due to excessive diversity [26–28]. These adaptive controls are crucial in avoiding premature convergence on suboptimal solutions and in driving continuous progress towards the global optimum.

2.2. Adaptive approaches within VNS methods

In recent years, a considerable body of research has been dedicated to enhancing the performance of VNS-based solution methodologies through the consideration of adaptive procedures. This research focus has primarily gravitated towards the investigation of two primary categories: low-level adaptations [29–31] and high-level adaptations [17,32]. These adaptive mechanisms have been designed with the specific aim of optimizing the reordering of local search operators within the improvement phase of VNS-based solution approaches. However, it is worth noting that a comparatively limited body of work has addressed distinct avenues, including VNS schemes that adapt solely during the shaking phase [33] or those that encompass adaptive reordering mechanisms for both the improvement and shaking phases [10,34].

2.3. The DA-GVNS method

GVNS represents a formidable iteration of VNS, demonstrating its adaptability and effectiveness in addressing a spectrum of challenging optimization problems, such as routing and assignment problems. [35–39]. For a more comprehensive understanding of the GVNS, the reader is referred to the works of Brimberg et al. [11],Hansen et al. [40], and Karakostas et al. [33].

Algorithm 1 The Double Adaptive GVNS

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

Table 1
Notation and Description.

Symbol	Description
S	Current solution in the search space
k_{max}	Maximum shaking level
max_time	Maximum allowed execution time
l_{max}	Maximum number of local search operators
LS_IO	Initial order of local search operators
SH_IO	Initial shaking order for solution perturbation
Sh_{max}	Maximum number of shaking operators
$f(S)$	Objective function value of solution S
S'	New solution generated from local search or shaking
S^*	Temporary solution used during shaking phase
$ShakingOrder$	Order of application for shaking operators
LS_Order	Order of local search operators
n	Problem instance size or number of cities/nodes
TUI	Cumulative count of iterations with no improvement
$ITER$	Total number of iterations executed so far
PBS	Objective value of the best solution found prior to the current iteration
CBS	Objective value of the best solution found in the current iteration
CPU_ST	CPU time at the start of the algorithm or a specific procedure
CPU_CT	Current CPU time used by the algorithm or procedure
FI	Flag indicating if the first improvement strategy is used
$Global\_BS$	Objective value of the global best solution found so far

A promising evolution of GVNS, incorporating adaptive mechanisms for the reordering of search operators during both the improvement and the shaking phases, has recently emerged [10] and it has already been adopted to solve efficiently complex real-world optimization problems [41,42]. Before presenting the pseudocode of the DA-GVNS, Table 1 provides a concise glossary of the symbols and their respective meanings to facilitate understanding of the notation used throughout this paper.

Algorithm 1 succinctly outlines the procedural steps of the DA-GVNS as detailed in Karakostas and Sifaleras [10].

The DA-GVNS is formulated through the incorporation of three well-established local search operators: the Swap operator, the Relocate operator, and the 2-Opt operator. These operators are utilized within both the improvement and shaking phases of the algorithm. They are seamlessly integrated into a pipe Variable Neighborhood Descent (pVND) method [11], complemented by an adaptive reordering mechanism, thus constituting the core of the algorithm's improvement component. Additionally, the same operators are harmoniously integrated into an intensified shaking approach, accompanied by an adaptive reordering procedure, which comprises the fundamental elements of the shaking component within the DA-GVNS framework. The adaptive mechanisms utilized in this context draw upon an empirical evaluation of the previous performance of the operators. This evaluation is based on the frequency of improvements attained by each operator during previous iterations of the algorithm. Subsequently, these empirical data inform the strategic determination of the most effective execution order for these operators in subsequent iterations.

2.4. Improved adaptive features

This section introduces innovative adaptive enhancements designed to increase the efficiency of the DA-GVNS. The first step in this process involves the identification of critical components, with the exclusion of alternative local search operators from the purview of consideration. This exclusion is made because the primary objective of this study is to explore the potential benefits of advanced adaptive mechanisms in enhancing the performance of the DA-GVNS. Therefore, in alignment with the prevailing research trends delineated in recent scientific contributions on adaptive enhancements within the framework of VNS methods [4,10,11,17,25,38], this study directs its attention to specific components of the DA-GVNS:

- Reordering of local search operators in the improvement phase.
- Search strategy.
- Dynamic adaptation of the shaking intensity level.

2.4.1. New reordering adaptive feature

Regarding the adaptive mechanism for reordering operators during the improvement phase, relying solely on the frequency of improvements achieved by each operator in previous iterations may yield limited benefits in terms of establishing an efficient order for local search operators. This limitation arises from the omission of critical information regarding the magnitude of improvements and the requisite execution time for each operator. Furthermore, adopting a more intricate reordering strategy may potentially consume a substantial amount of computational time without a guaranteed commensurate enhancement in solution quality.

Consequently, this study endeavors to address this challenge by introducing a balanced adaptive reordering approach. Specifically, the Relative Improvement per Execution Second (RIPES) metric is computed in each iteration of the algorithm for each local search operator. To facilitate this calculation, an array of dimensions $1 \times l_{max}$, denoted as *RIPES(:)*, is employed instead of the previously used $Improvements\_Counter$ within the DA-GVNS framework. Similarly, each position of *RIPES(:)* is associated with an individual search operator. The formula for calculating RIPES for each local search operator is expressed as follows: $RIPES(operator's\ number) = \frac{\frac{PBS}{CBS}}{Operators'\ ExecutionTime}$.

This novel approach aims to comprehensively account for both the quality of improvements and the associated execution time, thereby enabling a more refined and effective operator re-ordering strategy within the DA-GVNS algorithm.

Similarly to the approach used in the DA-GVNS framework, when a specific iteration does not produce improvements, the subsequent iteration proceeds by adhering to an initial predefined sequence LS_IO for executing local search operators. Conversely, in instances where improvements have been realized during a given iteration, an adaptive reordering strategy is invoked. This adaptive mechanism involves arranging the local search operators in descending order based on their respective RIPES values. The pseudocode of the modified adaptive mechanism is provided in Algorithm 2.

Algorithm 2 Local_Search_Adaptive_Mechanism

```
procedure LOCAL_SEARCH_ADAPTIVE_MECHANISM(LS_Order, LS_IO, RIPES)
    if no improvement is found in any neighborhood then
        New_LS_Order ← LS_IO
    end if
    if an improvement is found then
        New_LS_Order ← Descending_Order(LS_Order, RIPES)
    end if
    LS_Order ← New_LS_Order
    return LS_Order
end procedure
```

2.5. Knowledge-guided adaptive search strategy

The concept of an adaptive search strategy entails the dynamic selection between two distinct search strategies: the best improvement search strategy and the first improvement search strategy. Karakostas et al. (2019) put forth an adaptive search strategy that makes a choice between the first and best improvement search strategies, primarily contingent on the problem instance's size. On the contrary, a more complex approach has been advanced by Ren et al. (2020). In their work, the authors introduced a probabilistic function that takes into consideration both the current iteration number and the maximum iteration limit. The core premise underlying this approach is the utilization of the best improvement search strategy initially and, as the execution progresses towards completion, transitioning to the first improvement search strategy.

This study seeks to leverage the data generated during the computational process, encompassing metrics such as execution time, unimproved iterations or solution quality rates, and the size of the problem instance (indicated by n). To achieve this goal, the research has formulated two distinct mathematical expressions:

- Percentage of Unimproved Iterations (PUI) - based formula: $PUI\_Factor = \frac{CPU\_CT - CPU\_ST}{n} \cdot \frac{TUI}{ITER}$
- Solution Quality Improvement Rate (SQIR) - based formula: $SQIR\_Factor = \frac{CPU\_CT - CPU\_ST}{n} \cdot \frac{PBS - CBS}{PBS}$

The approach based on the PUI serves as an indicator of how frequently the solution method achieves improvements, rendering it a potentially effective means of detecting stagnation or the absence of progress within the algorithm. On the contrary, the approach founded on SQIR guides the decision to change the search strategy based not solely on the frequency of improvements but also on their magnitude. However, the SQIR-based approach may not encompass a comprehensive assessment of the overall algorithmic behavior, as the PUI-based approach does, and it might refrain from triggering search strategy switches in cases where improvements are consistently small. To make a well-informed selection between these criteria, extensive experimentation is imperative, encompassing a diverse array of problem instances. The mechanism introduced for the knowledge-guided adaptive selection of search strategy is presented in Algorithm 3.

Based on offline testing, two distinct values are taken into consideration for the parameter, denoted as $FactorParameter$. These specific values are 0.1 and 0.5. To clarify, the $FactorParameter$ values were strategically set to 0.1 and 0.5 to regulate the adaptive switching between the first improvement and best improvement strategies. A value of 0.1 enacts as a low threshold and emphasizes exploration by maintaining the algorithm in the first improvement phase for a longer duration. In the PUI-based formula, this value ensures that the algorithm continues exploring even after several unimproved iterations, as long as the stagnation remains relatively low. In the SQIR-based formula, the algorithm avoids switching to best improvement unless the solution quality improvement rate drops significantly, which signals the need for intensification.

Conversely, a value of 0.5 enacts as a mid-range threshold and represents a balanced approach, facilitating a more aggressive transition to the best improvement search strategy. In the PUI-based formula, the algorithm begins intensifying the search when unimproved iterations constitute a larger portion of the total iterations (e.g., half). Similarly, in the SQIR-based formula, the algorithm switches to best improvement when the solution quality improvement rate drops below 50%, ensuring that moderate improvements are sufficient to trigger an intensified search. These thresholds provide a clear and effective balance between exploration and exploitation, allowing the algorithm to adapt dynamically to the evolving search landscape.

2.5.1. Adaptive configuration of shaking intensity level

The last adaptive mechanism introduced pertains to the dynamic adjustment of the parameter k, which signifies the degree of perturbation applied in each iteration of the algorithm. Within each outer iteration (refer to line 2 of Algorithm 1), the parameter k is initialized to a value of one. To facilitate the updating of this pivotal parameter, two formulas have been developed, taking into account pertinent data concerning relative improvements, the frequency of improvements, and the size of the problem instance. The parameter k undergoes a reduction whenever an improvement is achieved, while it experiences an increase under different circumstances. These mathematical expressions are expressed in Eqs. (1) and (2). The usage of the minus sign within the provided formulas signifies the intended reduction of the parameter, while the incorporation of the plus sign in the same equations is indicative of the increase of the parameter k. To clarify, nint() is an elemental intrinsic function in Fortran programming language, which returns the nearest integer to its argument.

$$k = \mathrm{nint}\left(k \cdot \left(1 \mp \frac{|\mathrm{Global\_BS} - \mathrm{CurrentSolution}|}{\mathrm{Global\_BS}}\right)\right) \tag{1}$$

$$k = \mathrm{nint}\left(k \cdot \left(1 \mp e^{-\frac{\mathrm{TUI}}{n}}\right)\right) \tag{2}$$

2.5.2. The improved DA-GVNS

Herein, the DA-GVNS algorithm with the novel proposed adaptive features is provided in Algorithm 4.

3. Computational study

3.1. Computing environment

The proposed DA-GVNS, along with other developed GVNS variants, was implemented using the Fortran programming language and was executed using the Intel Fortran compiler version 18.0, using the /O3 optimization option. These computational procedures were carried out on a PC running the Windows 10 Home 64-bit operating system, equipped with an Intel Core i7-9750H CPU operating at a clock speed of 2.6 GHz and 16 GB of RAM. The computational experiments were carried out on a set of symmetric TSP instances obtained from TSPLib (http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95) as well as on a set of QAP instances from QAPLib (https://qaplib.mgi.polymtl.ca), and a CPU execution time limit of 60 s was imposed on all the developed DA-GVNS-based heuristics. It is essential to clarify that the reported results represent the average and best objective values obtained from 10 independent runs for each problem instance.

Algorithm 3 Adaptive_Search_Strategy_Update

procedure Adaptive_Search_Strategy_Update($CPU\_CT, CPU\_ST, n, TUI, ITER, PBS, CBS, FI$)
  if (PUI is true) **then**
    $Calculate\ PUI\ Factor$
    $Factor = PUI\_Factor$
  else
    $Calculate\ SQIR\ Factor$
    $Factor = SQIR\_Factor$
  end if
  if ($Factor \leq FactorParameter$) or ($Factor \geq 1$) **then**
    $FI = false$
  else
    $FI = true$
  end if
  return FI
end procedure

Algorithm 4 The improved DA-GVNS

procedure improved_DA-GVNS($S, max\_time, l_{max}, LS\_IO, SH\_IO$)
  while $CPU\_CT \leq max\_time$ **do**
    $ShakingOrder = Shaking\_Adaptive\_Mechanism(ShakingOrder, SH\_IO, Sh_{max})$
    $k \leftarrow 1$
    for $i \leftarrow 1, Sh_{max}$ **do**
      $l = ShakingOrder(i)$
      $S^* = Shake(S, l)$
      $Local\_Search\_Adaptive\_Mechanism(LS\_Order, RIPES)$
      $S' = pVND(S^*, l_{max}, LS\_Order, RIPES)$
      if $f(S') < f(S)$ **then**
        $S \leftarrow S'$
        Use Equation (1)
      else
        Use Equation (2) with plus sign
      end if
    end for
    Apply the $Adaptive\_Search\_Strategy\_Update$
  end while
  return S
end procedure

Table 2
Percentage Deviation from Optimal Values Across Methods.

Method	Average	Best
DA-GVNS	8.47	7.35
DA-GVNS_AR	8.44	7.33
DA-GVNS_AR_PUI_01	7.95	6.33
DA-GVNS_AR_PUI_05	**7.67**	**6.27**
DA-GVNS_AR_SQIR_01	8.11	6.8
DA-GVNS_AR_SQIR_05	7.88	6.55

3.2. Computational results on symmetric TSP

This section presents the results of a systematic computational analysis undertaken to discern the most efficient iteration of the DA-GVNS algorithm, enhanced by the novel adaptive features introduced in this study. Herein, it should be clarified that the classic Nearest Neighbor heuristic [43] was employed to initialize each variant of the DA-GVNS method. Consequently, all methods commenced from the same initial solution for each TSP instance, ensuring a consistent starting point across all experimental runs. Table 2 provides a summary of the percentage discrepancy between the averages of the optimal values and the averages, as well as the best-found solutions, obtained through each respective solution method.

To provide clarity, the term DA-GVNS denotes the initial method as introduced by Karakostas and Sifaleras [10], while the method denoted as "DA-GVNS_AR" signifies the incorporation of an alternative adaptive reordering approach within the improvement step of the $DA-GVNS$. Furthermore, the terms DA-GVNS_AR_PUI_01 and DA-GVNS_AR_PUI_05 correspond to the "DA-GVNS_AR" method integrated with the novel adaptive search strategy based on the PUI approach. On the contrary, the terms DA-GVNS_AR_SQIR_01 and DA-GVNS_AR_SQIR_05 denote the amalgamation of the "DA-GVNS_AR" method with the adaptive search strategy approaches founded on SQIR. The notations 01 and 05 correspond to the values "0.1" and "0.5" assigned to the parameter known as *FactorParameter* within the adaptive search strategy mechanism.

The beneficial impact of the novel adaptive features on the performance of the DA-GVNS is evident. Among the newly introduced methods outlined in Table 2, the DA-GVNS_AR_PUI_05 method demonstrates notably superior results. To be precise, it enhances the results achieved by the "DA-GVNS" by approximately 1% in terms of average solutions and approximately 1.1% in terms of the best-found solutions. Although the percentage improvements attained through the incorporation of the proposed knowledge-guided adaptive mechanisms may appear modest, it is essential to underscore that the principal advantage lies in the sustained and consistent enhancements realized across all problem instances.

The computational analysis proceeds with the selection of DA-GVNS_AR_PUI_05 as the primary solution methodology, and additional adaptive features, as proposed, are subjected to testing within this framework. These knowledge-guided adaptive features encompass the approaches devised for the adaptive adjustment of the shaking intensity level of the solution method. More specifically, the following variants of DA-GVNS_AR_PUI_05 are derived:

- The solution method that incorporates Eq. (1) in both cases, whether an improvement is achieved or not, is denoted as DA-GVNS_AR_PUI_05_A.
- The solution method that integrates Eq. (2) in both cases, whether or not an improvement is achieved, is designated as DA-GVNS_AR_PUI_05_B.

Table 3
Percentage Deviation from Optimal Values for DA-GVNS_AR_PUI_05_X Variants (X = A, B, C, D).

Method	Average	Best
DA-GVNS	8.47	7.35
DA-GVNS_AR_PUI_05_A	8.21	7.35
DA-GVNS_AR_PUI_05_B	8.26	7.09
DA-GVNS_AR_PUI_05_C	**6.5**	**6.08**
DA-GVNS_AR_PUI_05_D	8.33	7.19

Table 4
Results of the conducted Shapiro–Wilk Normality Test.

Method	p-Value
DA-GVNS	3.97×10^{-22}
DA-GVNS_AR_PUI_05_C	3.99×10^{-22}

- The solution method that uses Eq. (1) when an improvement is observed and Eq. (2) otherwise is labeled as DA-GVNS_AR_PUI_05_C.
- The solution method that employs Eq. (2) once an improvement is detected, and Eq. (1) otherwise is referred to as DA-GVNS_AR_PUI_05_D.

Table 3 clearly demonstrates that among the variants of DA-GVNS_AR_PUI_05, the one that applies Eq. (1) once an improvement is detected, and Eq. (2) otherwise consistently produces significantly better results compared to the other proposed variants. To be precise, this specific solution methodology produces a significant improvement of approximately 1.8% in terms of average solutions and approximately 1.2% in terms of the best-found solutions compared to the conventional DA-GVNS method. Therefore, DA-GVNS_AR_PUI_05_C represents the newly formulated variant of the solution method, a result of incorporating the knowledge-guided adaptive mechanisms introduced in this study.

An interesting observation related to the adaptive selection of the shaking intensity level is that its upper limit coincides with the predefined parameter k_{max} utilized in the original DA-GVNS ($k_{max} = 8$ [10]). However, the key distinction becomes apparent in the frequency and distribution of specific values of k. This disparity may be due to the adaptive nature of the mechanism, which dynamically adjusts k based on the algorithm's performance and problem characteristics, potentially leading to a more diverse range of values during its execution.

To determine the statistical significance of the difference observed between the proposed solution method and the conventional DA-GVNS, a rigorous statistical analysis was carried out. More specifically, a Shapiro–Wilk normality test was used to examine whether the independent results generated by both methods follow the normal distribution.

As is evident from the p-values presented in Table 4, it is clear that neither dataset follows a normal distribution. Consequently, the application of a non-parametric statistical test becomes necessary. In this context, the Wilcoxon signed-rank test was employed. The computed p-value, amounting to 2.7×10^{-11}, indicates the presence of a statistically significant difference between the two solution methods. Consequently, it becomes evident that the adaptive approaches guided by the proposed knowledge mechanisms have culminated in the development of the improved variant of the DA-GVNS, the KG-GVNS. To clarify, the statistically significant difference remains even after excluding the instance "dsj1000" as an outlier. Specifically, in this case, the reported p-value is 4.29×10^{-11}. Figs. 1 and 2 present the differences in performance between DA-GVNS and KG-GVNS across each sTSP instance. It is important to note that Fig. 1 includes the "dsj1000" instance, whose objective value is significantly larger than those of the other instances. As a result, this figure does not provide a clear view of the relative advantages of KG-GVNS. By excluding the "dsj1000" instance in Fig. 2, the benefits achieved by KG-GVNS become more apparent, allowing for a more accurate comparison of the methods across the remaining problem instances. More specifically, positive values indicate instances where KG-GVNS outperforms DA-GVNS.

3.3. KG-GVNS compared with other heuristic methods

Although the current study primarily concentrates on exploring the potential advantages derived from augmenting the DA-GVNS through the incorporation of knowledge-guided mechanisms, it is imperative to assess the performance of the newly introduced KG-GVNS in comparison to other efficient solution methods. This comparative study incorporates five recently proposed efficient metaheuristic solution approaches, along with one of the latest enhanced variants of the VNS framework, for addressing the TSP. More specifically, an improved VNS [44], noted as "A"; a hybridization of rider optimization and spotted hyena optimization algorithm [45], noted as "B"; a deer hunting linked earthworm optimization algorithm [46], noted as "C"; a discrete sparrow search algorithm [47], noted as "D"; a heuristic smoothing ant colony optimization algorithm with differential information [48], noted as "E", and a discrete komodo algorithm [49], noted as "F". The corresponding average objective values are provided in Table B.8. However, Table 5 presents the performance comparisons between KG-GVNS and the other algorithms. Specifically, the comparisons focus on the **Average Performance**, which indicates the mean solution quality achieved by KG-GVNS and each method across the instances where both reported results. The **Median Performance** provides an additional perspective on central tendency by showing the median solution quality for each method. The **Percentage Performance Deviation** measures how much the other methods deviate from KG-GVNS. Positive values indicate that KG-GVNS outperforms the other methods, while negative values show that the other methods perform better. The **Percentage of Best-Known Solutions** reports the percentage of instances where each method achieved the best-known solution. Finally, the **Average CPU execution time** provides valuable insights into the computational efficiency of each method.

Table 5 provides a comprehensive comparison between KG-GVNS and several state-of-the-art metaheuristic methods (A, B, C, D, E, F). Specifically, the performance metrics evaluated include the average and median solution quality, percentage performance deviation, percentage of best-known solutions achieved, and average CPU execution time for each method. KG-GVNS exhibits a consistent and competitive performance across multiple metrics. The average performance of KG-GVNS is superior to methods B and F and is highly competitive with methods A and C. It achieves an average percentage performance deviation of 1.3% to 2.03% over methods A, B, and F, indicating a significant advantage in solution quality. Positive values in the percentage performance deviation metric confirm that KG-GVNS consistently outperforms the other methods across several instances, highlighting its robustness and efficiency. Additionally, KG-GVNS achieves the best-known solutions in 33.33% to 50% of the instances, demonstrating its capability to consistently reach optimal or near-optimal solutions. In contrast, the competing methods exhibit lower success rates, with methods B, C, and F achieving best-known solutions in only 15% to 25% of instances.

Moreover, the computational efficiency of KG-GVNS is evident in its lower average CPU execution times compared to other methods. For instance, the average CPU time of KG-GVNS is approximately 35.92 s, whereas method A requires significantly more time (3462.48 s on average). Additionally, some methods (B and F) did not report CPU time ("NS"), further highlighting the thoroughness of the KG-GVNS evaluation. Notably, the utilization of double stars (**) in reference to the work by Zhang et al. (2022) signifies that the reported execution time pertains to the average maximum required execution time of the method.

An important note is that the results presented in Table 5 are based on the specific instances where both KG-GVNS and each competing method reported results. Unlike the other algorithms that selectively evaluated only a subset of the TSPLib instances, KG-GVNS was tested on the entire TSPLib dataset. This reinforces the validity and generality of KG-GVNS, suggesting that the other methods may

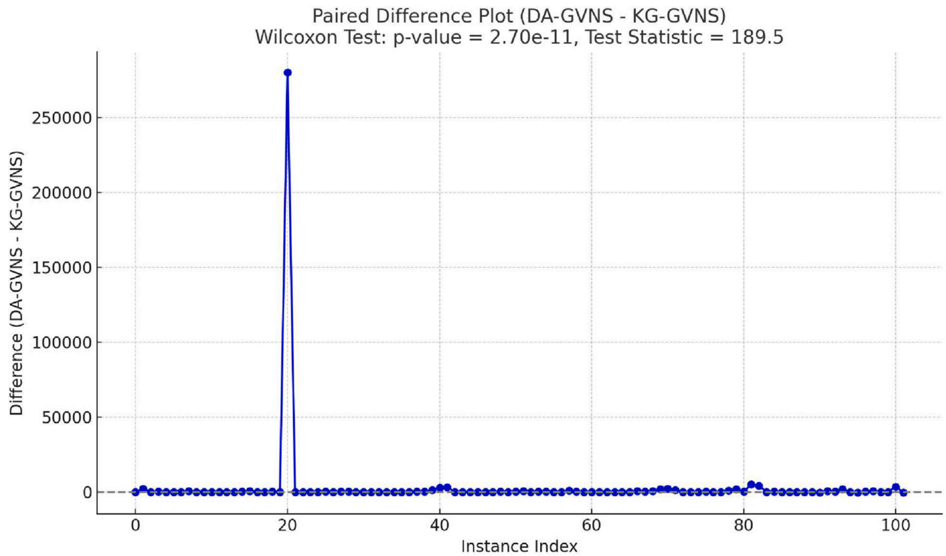

Fig. 1. Differences between DA-GVNS and KG-GVNS.

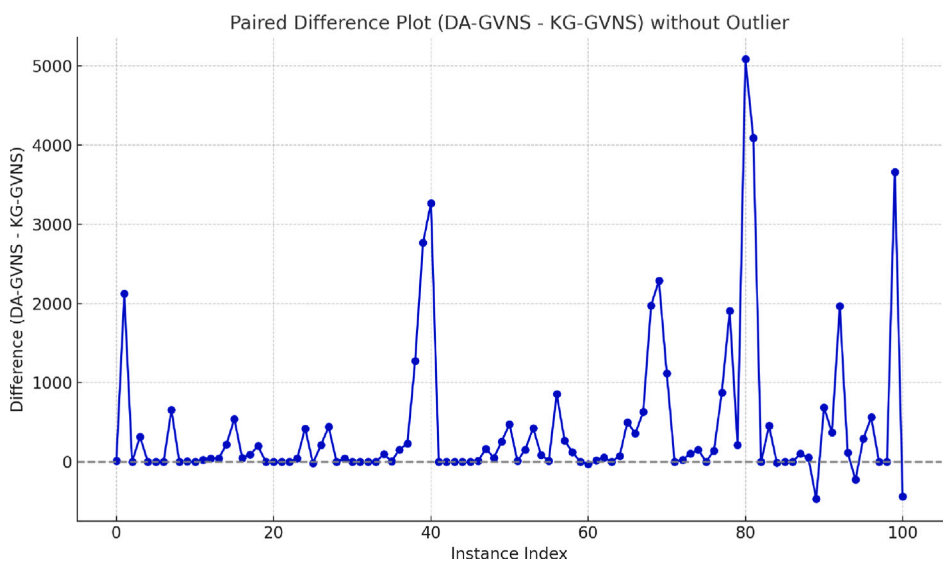

Fig. 2. Differences between DA-GVNS and KG-GVNS by excluding instance "dsj1000".

Table 5
KG-GVNS vs Other Algorithms: Performance Metrics.

Metric	vs A	vs B	vs C	vs D	vs E	vs F
Average (KG-GVNS)	**49 635.12**	57 289.25	47 255.09	43 432.43	518 147.15	**26 291.7**
Average (Other)	50 423.58	**55 513.2**	**45 848.98**	**43 030.01**	**495 693.96**	27 008.7
Median (KG-GVNS)	**22 136.5**	**7230.5**	**21 294**	**26 427**	21 282	**15 075.5**
Median (Other)	22 167	7330.25	21 340	26 460.13	21 282	15 341.35
Percentage Performance Deviation	**1.3**	**2.03**	**0.16**	−0.1	−0.81	**1.79**
% of Best-Known Solutions (KG-GVNS)	**33.33**	**37.5**	**33.33**	**33.33**	34.15	**50**
% of Best-Known Solutions (Other)	15	25	21.21	16.67	70.73	13.64
Average CPU execution time (s) (KG-GVNS)	**35.92**	**37.71**	**35.18**	**34.9**	**33.43**	**26.77**
Average CPU execution time (s) (Other)	3462.48	NS	NS	187.59**	14 022.25	NS

have selected instances that favored their performance, whereas KG-GVNS demonstrated robustness across a wider set of problems. The enhanced performance of KG-GVNS can be attributed to the novel knowledge-guided adaptive mechanisms integrated into the algorithm. These mechanisms enable KG-GVNS to dynamically adjust its search components and strategies, effectively balancing exploration and exploitation. This adaptability allows KG-GVNS to escape local optima and explore the solution space more thoroughly, contributing to its competitive edge, particularly on larger and more complex instances.

In summary, the results substantiate the competitive nature of KG-GVNS, not only in terms of solution quality but also in computational efficiency. Its ability to consistently achieve high-quality solutions across a wide range of TSPLib instances, combined with its lower execution times, highlights its potential as a powerful tool for

Table 6
Performance comparison of DA-GVNS-FI, DA-GVNS-BI, and KG-GVNS.

Metric	DA-GVNS-FI	DA-GVNS-BI	KG-GVNS
Mean Avg Solutions	72880608.28	69994234.88	**68 577 814.02**
Mean Best Solutions	71085883.37	67511773.20	**67 255 390.87**
Mean Worst Solutions	74457955.33	72691669.50	**71 998 220.77**
Median Avg Solutions	264 250.50	263 092.50	**260 994.70**
Median Best Solutions	**259 828.00**	260 562.00	260 080.00
Median Worst Solutions	267 423.00	265 934.00	**261 983.00**
Mean Best CPU Solution (s)	39.77	**37.70**	38.70
Mean CPU (s)	**48.55**	48.75	48.92
Mean SD	1106756.87	1883677.20	1600204.67
Avg Performance Deviation (%)	4.41	4.02	**3.51**

solving challenging instances of the TSP. This competitive performance underscores the significant contributions of its adaptive mechanisms, positioning KG-GVNS as a promising approach in the field of combinatorial optimization. Further refinements of the method could build upon this strong foundation to enhance its efficiency and applicability in solving even more complex problem instances.

3.4. Computational results on QAP instances

This section summarizes the comparative computational analysis between the KG-GVNS and the DA-GVNS on selected instances of the classic QAP. Table 6 provides an overall summary of the results of the conducted computational analysis on the selected QAP instances. The results highlight KG-GVNS as the most efficient method, offering not only the best solutions but also the most stable performance. To mention here, that a random permutation was utilized as the primary initialization approach for the QAP instances. This ensured that even the same method initiated from different solutions across independent runs, introducing variability in the starting conditions for the algorithms. Additionally, it was observed that the KG-GVNS often commenced from lower-quality solutions compared to the DA-GVNS methods. This highlights the robustness and effectiveness of the adaptive components in the KG-GVNS algorithm, which significantly improved upon these initial solutions. Despite KG-GVNS has a slightly higher mean CPU time compared to DA-GVNS-FI and DA-GVNS-BI, the differences are marginal, indicating that while KG-GVNS provides better solution quality and stability, this improvement does not come at a significant computational cost. The slight increase in CPU time for KG-GVNS is a reasonable trade-off considering its superior performance in minimizing the QAP solutions.

To further investigate the performance enhancements achieved in KG-GVNS, a convergence analysis was conducted in two randomly selected QAP instances. More specifically, The convergence performance of the KG-GVNS and DA-GVNS algorithms has been evaluated on two benchmark instances, lipa70b and tai100b, with objective values plotted against CPU time, as illustrated in Figs. 3 and 4 respectively. To clarify, in the subsequent investigations, the most efficient variant of DA-GVNS, referred to as DA-GVNS-BI, is considered as DA-GVNS.

For the lipa70b instance, KG-GVNS starts with a higher initial objective value (5.97 million) compared to DA-GVNS (5.96 million), indicating that KG-GVNS begins from a less favorable solution. However, KG-GVNS shows a more rapid improvement in the objective value over the iterations, reaching the optimal value significantly faster than DA-GVNS. KG-GVNS converges within 22.58 s, whereas DA-GVNS requires more than 37 s to reach the same optimal value. This highlights the superior performance of KG-GVNS, as it converges to the optimal solution using fewer computational resources. Although DA-GVNS eventually reaches the same optimal solution, it does so more slowly, making it less efficient for this particular instance.

In the case of the tai100b instance, the convergence behavior of the two algorithms is even more distinct. DA-GVNS begins with a better initial solution, with an objective value of 1.74 billion, while KG-GVNS

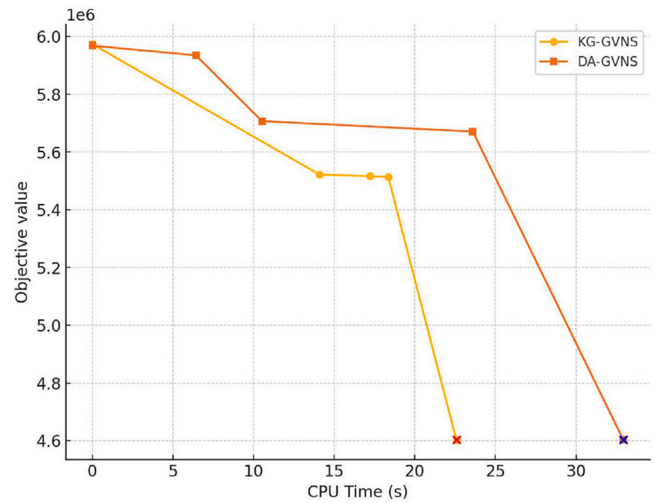

Fig. 3. Convergence analysis between DA-GVNS and KG-GVNS in lipa70b.

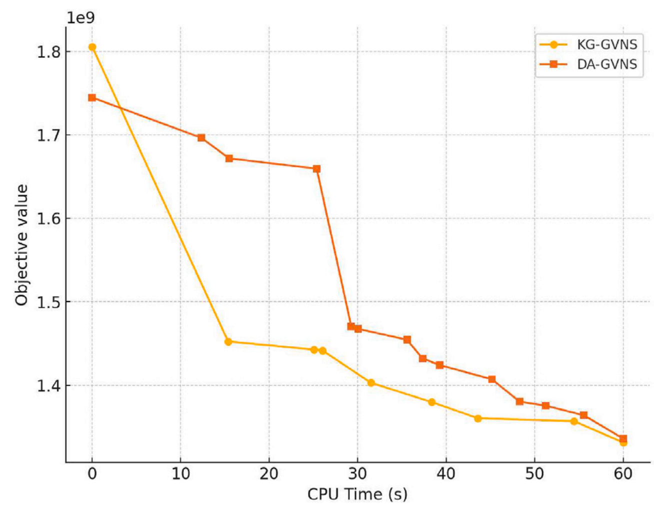

Fig. 4. Convergence analysis between DA-GVNS and KG-GVNS in tai100b.

starts at 1.8 billion, reflecting that KG-GVNS initially explores a worse solution. Despite this, KG-GVNS rapidly improves and surpasses DA-GVNS after approximately 30 s. By the end of the computation time (60 s), KG-GVNS reaches an objective value of 1,331,311,294, better than DA-GVNS, which converges to 1,335,968,289 as well. KG-GVNS demonstrates faster convergence and reaches the best-known solution earlier, making it more efficient in this instance as well.

In both cases, KG-GVNS demonstrates superior performance in terms of convergence speed and final objective value. Notably, KG-GVNS achieves better results with less CPU time in both instances. DA-GVNS, though eventually reaching comparable results to KG-GVNS, takes significantly longer to converge, suggesting it is less suitable for time-sensitive or large-scale problems where computational efficiency is critical. The results indicate that KG-GVNS is more effective at quickly escaping local optima and converging to globally competitive solutions, with a clear performance advantage in larger instances such as tai100b, where computational time becomes more critical.

Thus, KG-GVNS has demonstrated greater efficiency and robustness in solving combinatorial optimization problems, particularly in larger instances where computational efficiency is critical. While DA-GVNS eventually achieves competitive results, its slower convergence and higher CPU time consumption limit its applicability in scenarios with

Table A.7
Comparison of DA-GVNS and KG-GVNS Results (average values).

Instance	Best-known	DA-GVNS	KG-GVNS
a280	2579	2614	**2603**
ali535	202 339	214 790	**212 663**
att48	10 628	10 628	10 628
att532	27 686	28 947	**28 627**
bayg29	1610	1610	1610
bays29	2020	2020	2020
berlin52	7542	7542	7542
bier127	118 282	119 122	**118 465**
brazil58	25 395	25 395	25 395
brg180	1950	1962	**1954**
burma14	3323	3323	3323
ch130	6110	6154	**6127**
ch150	6528	6595	**6551**
d198	15 780	15 855	**15 813**
d493	35 002	36 497	**36 276**
d657	48 912	52 184	**51 643**
d1291	50 801	54 778	**54 727**
d1655	62 128	67 292	**67 201**
d2103	80 450	83 336	**83 137**
dantzig42	699	699	699
dsj1000	18 659 688	20 080 489	**19 800 452**
eil51	426	426	426
eil76	538	**538**	539
eil101	629	633	**632**
fl417	11 861	12 019	**11 974**
fl1400	20 127	21 858	**21 435**
fl1577	22 249	**24 147**	24 165
fl3795	28 772	35 913	**35 702**
fnl4461	182 566	215 919	**215 477**
fri26	937	937	937
gil262	2378	2451	**2408**
gr17	2085	2085	2085
gr21	2707	2707	2707
gr24	1272	1272	1272
gr48	5046	5046	5046
gr96	55 209	55 306	**55 210**
gr120	6942	6981	**6971**
gr137	69 853	70 123	**69 967**
gr202	40 160	41 011	**40 780**
gr229	134 602	137 426	**136 148**
gr431	171 414	180 715	**177 949**
gr666	294 358	312 962	**309 699**
hk48	11 461	11 461	11 461
kroA100	21 282	21 282	21 282
kroB100	22 141	22 165	**22 163**
kroC100	20 749	20 749	20 749
kroD100	21 294	21 294	21 294
kroE100	22 068	22 121	**22 110**
kroA150	26 524	26 817	**26 649**
kroB150	26 130	26 256	**26 205**
kroA200	29 368	29 807	**29 550**
kroB200	29 437	30 015	**29 538**
lin105	14 379	14 390	**14 379**
lin318	42 029	43 201	**43 045**
nrw1379	56 638	60 576	**60 147**
p654	34 643	35 154	**35 065**
pa561	2763	2899	**2886**
pcb442	50 778	53 009	**52 152**
pcb1173	56 892	61 725	**61 454**
pcb3038	137 694	152 209	**152 087**
pr76	108 159	108 159	108 159
pr107	44 303	**44 303**	44 334
pr124	59 030	59 050	**59 030**
pr136	96 772	97 062	**97 004**
pr144	58 537	58 537	58 537
pr152	73 682	73 839	**73 763**
pr226	80 369	80 880	**80 378**
pr264	49 135	49 880	**49 523**
pr299	48 191	49 719	**49 088**
pr439	107 217	112 600	**110 626**
pr1002	259 045	277 867	**275 581**
pr2392	378 032	414 696	**413 580**
rat99	1211	1211	1212
rat195	2323	2364	**2340**
rat575	6773	7179	**7073**
rat783	8806	9445	**9290**
rd100	7910	7910	7910
rd400	15 281	15 915	**15 772**
rl1304	252 948	279 174	**278 298**
rl1323	270 199	292 819	**290 911**
rl1889	316 536	343 218	**343 002**
rl5915	565 530	688 254	**683 171**
rl5934	556 045	651 867	**647 776**
si175	21 407	21 421	21 421
si535	48 450	49 102	**48 648**
si1032	92 650	**92 962**	92 973
st70	675	675	675
swiss42	1273	1273	1273
ts225	126 643	126 746	**126 643**
tsp225	3916	4001	**3946**
u159	42 080	**4216**	42 636
u574	36 905	39 583	**38 896**
u724	41 910	44 814	**44 444**
u1060	224 094	241 524	**239 558**
u1432	152 970	164 611	**164 492**
u1817	57 201	**61 861**	62 084
u2152	64 253	70 577	**70 284**
u2319	234 256	243 332	**242 769**
ulysses16	6859	6859	6859
ulysses22	7013	7013	7013
vm1084	239 297	256 456	**252 798**
vm1748	336 556	**369 787**	370 225
Average	256 414.66	278 121.25	**273090.16**

(*continued on next page*)

strict time constraints or larger problem sizes. These findings suggest that KG-GVNS is better suited for practical applications requiring high-quality solutions within limited timeframes, making it the more appropriate algorithm for real-world optimization tasks.

However, to substantiate the claim of improved performance of KG-GVNS, the reported differences between DA-GVNS and KG-GVNS should be evaluated using a valid statistical test. The statistical analysis conducted to assess the performance differences between KG-GVNS and DA-GVNS revealed important insights. First, the Shapiro–Wilk test was applied to check the normality of the data for both algorithms. The results showed that the data for both KG-GVNS ($W = 0.284$, $p < 0.0001$) and DA-GVNS ($W = 0.283$, $p < 0.0001$) did not follow a normal distribution, as indicated by the extremely low p-values. Given the non-normality of the data, a non-parametric test was deemed appropriate. The Wilcoxon Signed-Rank test was then used to compare the performances of the two algorithms. The test results (Test Statistic = 68.0,p = 0.0036) indicated a statistically significant difference between the two algorithms, with KG-GVNS showing superior performance. This significant p-value (< 0.05) supports the conclusion that the performance difference between KG-GVNS and DA-GVNS is not due to random chance, but reflects a real, measurable improvement in the efficiency of KG-GVNS over DA-GVNS.

4. Conclusions

In conclusion, this study has undertaken the task of introducing innovative adaptive elements into the DA-GVNS algorithm with the overarching goal of enhancing its computational efficiency. Extensive experimentation, utilizing the TSP and the QAP as benchmarks, has revealed several noteworthy findings. The primary contributions of this research encompass the incorporation of a novel adaptive re-ordering mechanism in the improvement phase of DA-GVNS, the development of a knowledge-driven adaptive mechanism for dynamically adjusting shaking intensity, and the introduction of a knowledge-guided adaptive mechanism to enhance the adaptive search strategy. Through rigorous

Table B.8
Comparison between KG-GVNS and other solution approaches (average values).

Instance	Optimal	KG-GVNS	A	B	C	D	E	F
a280	2579	**2603**		2606.8				
ali535	202 339	212 663						
att48	10 628	**10 628**						10 755.8
att532	27 686	28 627					**27 934**	
bayg29	1610	1610						
bays29	2020	2020	2020				2020	
berlin52	7542	**7542**	7544.36		7560	**7542**	**7542**	7646.25
bier127	118 282	118 465	119 006.39		**118 282**		**118 282**	
brazil58	25 395	**25 395**	25 592.72					
brg180	1950	1954						
burma14	3323	3323						3323
ch130	6110	6127	6153.72	6177.7	6113	6153.65	**6110**	6307.9
ch150	6528	6551	6644.95	6660.5		6590.15	**6528**	6653.65
d198	15 780	**15 813**	16 079.28					
d493	35 002	36 276						
d657	48 912	51 643						
d1291	50 801	54 727	56 095.33		**50 842**			
d1655	62 128	67 201	70 337.23		**62 147**			
d2103	80 450	83 137						
dantzig42	699	**699**	**699**				**699**	701.35
dsj1000	18 659 688	19 800 452					**18 897 396**	
eil51	426	**426**	428.98		469.45	426.6	**426**	429.95
eil76	538	539	552.57		618	543.1	**538**	547.8
eil101	629	632	648.27		703.2	641.5	**629**	652.6
fl417	11 861	11 974	12 183.14				**11 861**	
fl1400	20 127	21 435	**21 085.98**					
fl1577	22 249	24 165						
fl3795	28 772	35 702						
fnl4461	182 566	215 477						
fri26	937	**937**	**937**					939.5
gil262	2378	**2408**	2501.86		2430			
gr17	2085	2085	2085				2085	2085
gr21	2707	2707	2707					2707
gr24	1272	1272	1272					1272
gr48	5046	**5046**	**5046**					5095.9
gr96	55 209	**55 210**						56 279.8
gr120	6942	**6971**			6980			
gr137	69 853	69 967						
gr202	40 160	40 780					**40 292.75**	42 687.4
gr229	134 602	136 148						
gr431	171 414	177 949						
gr666	294 358	309 699		**294358**	**294 358**		301 073	
hk48	11 461	**11 461**						11 487.1
kroA100	21 282	**21 282**	21 695.79		21 329	21 290.2	**21 282**	21 436.65
kroB100	22 141	22 163	**22 140.2**			22 173.1	22 141	22 453.65
kroC100	20 749	**20 749**	20 809.29		20 790	20 770.5	**20 749**	21 080.95
kroD100	21 294	**21 294**	21 490.62		21 347	21 319.05	**21 294**	21 750.2
kroE100	22 068	22 110	22 193.8			22 091.9	**22 068**	22 454.7
kroA150	26 524	26 649	26 947.17		**26 566**	26 699.85		27 358.5
kroB150	26 130	**26 205**	26 537.04			26 220.4		26 902.85
kroA200	29 368	29 550	30 339.67		29 410	29 682.15	**29 368**	30 202.5
kroB200	29 437	**29 538**	30 453.22			29 850.55		30 911.8
lin105	14 379	**14 379**	14 395.64	14 736	**14 379**	**14 379**	**14 379**	14 686.65
lin318	42 029	43 045	43 964.93		**42 059**	42 742.7	42 124	44 379.65
nrw1379	56 638	60 147						
p654	34 643	35 065						
pa561	2763	**2763**		3407.6				
pcb442	50 778	52 152		**50800.24**	50 806		51 596	
pcb1173	56 892	**56 892**	63 435.95					
pcb3038	137 694	**137 694**	154 565.4					
pr76	108 159	**108 159**	**108 159**	**108159**	**108 159**	**108 159**		110 135
pr107	44 303	44 334	44 314.92		44 400	44 322	**44 303**	44 748.25
pr124	59 030	**59 030**	59 051.82		**59 030**	**59 030**	**59 030**	59 985.5
pr136	96 772	97 004	97 985.84			97 302.35	**96 781**	102 296.6
pr144	58 537	**58 537**	58 563.97			**58 537**	**58 537**	59 371.9
pr152	73 682	73 763	73 855.11		73 718	73 731.35	**73 682**	75 287.75
pr226	80 369	80 378	80 514.64			80 369.2	**80 369**	83 231.6
pr264	49 135	49 523	51 197.14			49 271.85	**49 135**	52 501.75
pr299	48 191	49 088	50 373.12				**48 205**	
pr439	107 217	110 626	111 771.2			107 844.9	**107 259**	114 640.8
pr1002	259 045	275 581	280 563.9		**259 045**	266 352.35		
rat99	1211	1212	1241.26		1272		**1211**	1224.55
rat195	2323	2340	2453.81		2343.8		**2326.75**	

(*continued on next page*)

Table B.8 (*continued*).

Instance	Optimal	KG-GVNS	A	B	C	D	E	F
rat575	6773	7073	7362.51				**6841**	7120.95
rat783	8806	9290	9707.36				**8893**	
rd100	7910	**7910**	7918.36	8000				
rd400	15 281	15 772	16 250.21		**15 354**			15 996.05
rl1323	270 199	**270 199**	295 611.2					
st70	675	**675**	677.11		729	675.15	**675**	684.1
swiss42	1273	1273	1273					
tsp225	3916	3946				3926.05	**3916**	4096.05
u159	42 080	42 636	42 467.61		**42 131**	42 262.75		
u574	36 905	**38 896**	39 629.11					
u724	41 910	44 444	45 729.71		**41 910**			
u2319	234 256	**242 769**	262 595.6					
ulysses16	6859	**6859**			6920.1		**6859**	**6859**
ulysses22	7013	**7013**			7036.1		**7013**	**7013**
vm1748	336 556	370 225	**366 757.8**					

Table C.9
Comparison of DA-GVNS-FI and DA-GVNS-BI.

Instance	DA-GVNS-FI				DA-GVNS-BI			
	Average	Best	Worst	SD	Average	Best	Worst	SD
bur26a	5 426 670	5 426 670	5 426 670	0.00	5 426 670	5 426 670	5 426 670	0.00
esc32h	438	438	438	0.00	438	438	438	0.00
esc64a	116	116	116	0.00	116	116	116	0.00
kra32	89 720	89 720	89 720	0.00	**89 040**	88 700	89 920	549.59
lipa30a	**13 178**	13 178	13 178	0.00	13 286.3	13 178	13 370	93.33
lipa40a	31 912	31 912	31 912	0.00	**31 888.5**	31 850	31 921	21.25
lipa40b	476 581	476 581	476 581	0.00	476 581	476 581	476 581	0.00
lipa50a	**62 748.1**	62 734	62 758	6.71	62 758.3	62 675	62 820	40.85
lipa50b	1272803.2	1 210 244	1 420 738	100 734.50	**1 252 443.9**	1 210 244	1 422 914	88 968.69
lipa60a	**108 222**	108 162	108 280	47.12	108 244.8	108 196	108 298	41.07
lipa70b	5 153 361	4 603 200	5 533 792	473 540.32	**5 065 972**	4 603 200	5 534 341	487 810.99
lipa80b	**8 753 901**	7 763 962	9 460 258	852 223.12	8756187.8	7 763 962	9 432 991	854 034.04
lipa90a	**363 265.4**	363 092	363 390	109.47	363 389.6	363 215	363 517	82.56
sko56	34 957	34 754	35 186	142.47	**34 928.2**	34 774	35 120	125.99
sko64	**49 042.2**	48 898	49 136	74.85	49 231.8	48 972	49 414	132.47
sko72	**67 309.6**	67 034	67 576	176.74	67 399.2	66 914	67 764	237.99
sko100a	161 595.6	155 920	165 996	3996.60	**160 486.6**	158 410	162 042	1226.53
sko100d	159 576.2	154 646	163 756	3415.71	**157 082**	156 338	157 992	519.31
tai35a	**2 464 729.4**	2 445 450	2 483 508	11 899.88	2473843.8	2 459 728	2 483 104	8283.99
tai50a	**5 087 017.2**	5 073 226	5 113 610	11 632.78	5097784.6	5 069 298	5 113 190	14 643.69
tai64c	**1 855 928**	1 855 928	1 855 928	0.00	1 856 396	1 856 396	1 856 396	0.00
tai80a	**13 998 383.2**	13 928 418	14 047 106	45 305.48	14 010 838	13 949 636	14 049 848	27 230.74
tai100b	1485392850	1440691662	1521073234	25620429.92	**1398197276**	1335968289	1464573773	47692565.88
tai150b	598103222.3	592422010	606751245	4992159.41	**597 985 263.2**	589683420	609369525	5992960.83
tai256c	**47 218 730**	45 528 164	48 833 050	1049758.50	48094708.8	45 819 930	49 847 968	1293869.86
tho30	150 349	149 936	150 578	226.76	**150 341.4**	149 936	150 810	304.87
tho40	243 210	241 734	245 678	1105.71	**243 054.6**	241 484	244 022	786.30
tho150	9344106.8	9 301 856	9 376 924	32 544.26	**9 269 217.2**	9 212 060	9 288 194	42 954.30
wil50	**49 035.8**	48 934	49 150	64.34	49 048.8	48 946	49 180	80.51
wil100	285 291	277 922	289 168	3111.58	**283 130.4**	279 640	287 846	2750.23

statistical analysis, we have demonstrated the statistical significance and positive impact of these knowledge-guided adaptations on the overall performance of the DA-GVNS. As a result, it is evident that the proposed knowledge-guided adaptive approaches have successfully culminated in the development of an improved variant of the DA-GVNS, the KG-GVNS. These findings underscore the potential of such adaptive mechanisms to significantly enhance the efficiency and effectiveness of metaheuristic algorithms, opening up promising avenues for further research in the field of optimization and operational research. Furthermore, through the comparative analysis of KG-GVNS with recently introduced efficient heuristics, it becomes evident that KG-GVNS represents a highly competitive solution approach, characterized by notably low execution time requirements.

In light of the promising results and insights garnered from this study, several avenues for future research emerge in the realm of adaptive metaheuristics. Firstly, further investigation into the fine-tuning and optimization of adaptive mechanisms, including the exploration of alternative machine learning techniques, could lead to even more sophisticated and effective adaptations within metaheuristic algorithms. Additionally, the application of these adaptive mechanisms to a broader spectrum of combinatorial optimization problems beyond the TSP and the QAP could shed light on their versatility and generalizability.

CRediT authorship contribution statement

Panagiotis Karakostas: Writing – original draft, Software, Methodology, Conceptualization. **Angelo Sifaleras:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Results on symmetric TSP instances

Table A.7 presents the optimal objective values alongside the average performance of both DA-GVNS and KG-GVNS on the symmetric instances from TSPLib. The reported averages were computed over ten

Table C.10
Results of KG-GVNS on QAP instances.

Instance	Average	Best	Worst	SD
bur26a	5 429 421	5 426 670	5 431 255	2367.68
esc32h	438	438	438	0.00
esc64a	116	116	116	0.00
kra32	89 558	88 700	90 760	672.85
lipa30a	13 230.8	13 178	13 364	85.18
lipa40a	31 890.6	31 867	31 906	18.88
lipa40b	476 581	476 581	476 581	0.00
lipa50a	**62 736.2**	62 656	62 786	42.68
lipa50b	1272501.2	1 210 244	1 421 892	100 259.23
lipa60a	**108 171.2**	108 129	108 209	27.91
lipa70b	**4 969 906.1**	4 603 200	5 527 301	473 462.80
lipa80b	**8 413 042.3**	7 763 962	9 400 590	837 999.52
lipa90a	**363 255.9**	363 173	363 389	69.88
sko56	**34 717.2**	34 600	34 820	70.55
sko64	**49 013.2**	48 814	49 182	113.90
sko72	**67 189.6**	66 802	67 468	211.49
sko100a	**159 151**	157 196	161 232	1132.55
sko100d	**156 651.6**	154 976	158 192	991.94
tai35a	**2 459 074.6**	2 445 556	2 475 244	9930.60
tai50a	**5 077 152**	5 059 432	5 095 738	14 717.34
tai64c	1 855 928	1 855 928	1 855 928	0.00
tai80a	**13 975 247.6**	13 899 086	14 016 930	33 781.27
tai100b	**1358377042**	1331311294	1453784915	40324308.65
tai150b	**596 218 569**	586977280	599711727	4894721.09
tai256c	47676653.2	45 552 394	49 582 764	1272443.70
tho30	**150 190.2**	149 936	150 604	280.52
tho40	**241 753.6**	241 176	242 604	486.61
tho150	9 276 032	9 230 520	9 300 286	22 999.23
wil50	**48 972.2**	48 838	49 040	13 936.01
wil100	**280 235.8**	278 984	281 362	1008.06

independent runs of each method for each problem instance. To clarify, the use of bold font highlights the best solution obtained among the methods compared, though these solutions do not necessarily correspond to the known optimal values.

Appendix B. Comparisons between KG-GVNS and other algorithms on sTSPs

Table B.8 presents the average values for each method, calculated from ten independent runs on each sTSP instance of the TSPLib. It is important to clarify that values below the optimal threshold were systematically excluded from the comparative analysis. These exclusions were necessary, as such values could result from rounding errors or deviations from the official guidelines provided by TSPLib. To clarify, the use of bold font highlights the best solution obtained among the methods compared, though these solutions do not necessarily correspond to the known optimal values. Additionally, in instances where the compared methods have achieved the same objective values, even if these values are equal to the known optimum, they are not highlighted in bold

Appendix C. KG-GVNS vs DA-GVNS on QAP instances

Tables C.9 and C.10 present the average, best, and worst objective function values, along with the corresponding standard deviations, obtained from 10 independent runs for each of the selected QAP instances. In Table C.9, bold font is used to indicate the lowest average objective values reported by the two DA-GVNS variants for each selected QAP instance. In Table C.10, bold font highlights the instances where KG-GVNS achieved better solutions than both DA-GVNS variants in terms of average objective values.

Data availability

We provide links to widely-used benchmark datasets.

References

[1] E.G. Talbi, Metaheuristics: From Design to Implementation, John Wiley & Sons, Inc., Hoboken, New Jersey, 2009.

[2] S. Faramarzi-Oghani, P. Neghabadi, E.-G. Talbi, R. Tavakkoli-Moghaddam, Metaheuristics for sustainable supply chain management: A reviews, Int. J. Prod. Res. 61 (2023) 1979–2009.

[3] J. Swan, S. Adriaensen, A.E.I. Brownlee, J. Hammond, G.C. Johnson, A. Kheiri, F. Krawiec, J.J. Merelo, L.L. Minku, E. Özcan, G.L. Pappa, P. García-Sánchez, K. Sörensen, S. Voß, M. Wagner, D.R. White, Metaheuristics "in the large", European J. Oper. Res. 297 (2) (2022) 393–406.

[4] M. Sevaux, K. Sorensen, N. Pillay, Adaptive and multilevel metaheuristics, in: R. Martí, P. Pardalos, M. Resende (Eds.), Handbook of Heuristics, Springer, 2018, pp. 1–19.

[5] C. Aranha, C.L. Camacho Villalón, F. Campelo, M. Dorigo, R. Ruiz, M. Sevaux, K. Sörensen, T. Stützle, Metaphor-based metaheuristics: A call for action - the elephant in the room, Swarm Intell. 16 (2022) 1–6.

[6] K.S.N. Ripon, K. Glette, K.N. Khan, M. Hovin, J. Torresen, Adaptive variable neighborhood search for solving multi-objective facility layout problems with unequal area facilities, Swarm Evol. Comput. 8 (2013) 1–12.

[7] S. Cao, R. Li, W. Gong, C. Lu, Inverse model and adaptive neighborhood search based cooperative optimizer for energy-efficient distributed flexible job shop scheduling, Swarm Evol. Comput. 83 (2023) 101419.

[8] J. Li, R. Liu, R. Wang, Handling dynamic capacitated vehicle routing problems based on adaptive genetic algorithm with elastic strategy, Swarm Evol. Comput. 86 (2024) 101529.

[9] M. Karimi-Mamaghan, M. Mohammadi, P. Meyer, A.M. Karimi-Mamaghan, E. Talbi, Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art, European J. Oper. Res. 296 (2) (2022) 393–422.

[10] P. Karakostas, A. Sifaleras, A double-adaptive general variable neighborhood search algorithm for the solution of the traveling salesman problem, Appl. Soft Comput. 121 (2022) 108746.

[11] J. Brimberg, S. Salhi, R. Todosijević, D. Urošević, Variable neighborhood search: The power of change and simplicity, Comput. Oper. Res. 155 (2023) 106221.

[12] J. Xiong, B. Chen, Z. He, W. Guan, Y. Chen, Optimal design of community shuttles with an adaptive-operator-selection-based genetic algorithm, Transp. Res. C: Emerg. Technol. 126 (2021) 103109.

[13] M. Karimi-Mamaghan, M. Mohammadi, B. Pasdeloup, P. Meyer, Learning to select operators in meta-heuristics: An integration of Q-learning into the iterated greedy algorithm for the permutation flowshop scheduling problem, European J. Oper. Res. 304 (3) (2023) 1296–1330.

[14] R. Todosijević, M. Mladenović, S. Hanafi, N. Mladenović, I. Crévits, Adaptive general variable neighborhood search heuristics for solving the unit commitment problem, Int. J. Electr. Power Energy Syst. 78 (2016) 873–883.

[15] R. Turkeš, K. Sörensen, L.M. Hvattum, Meta-analysis of metaheuristics: Quantifying the effect of adaptiveness in adaptive large neighborhood search, European J. Oper. Res. 292 (2) (2021) 423–442.

[16] S.T.W. Mara, R. Norcahyo, P. Jodiawan, L. Lusiantoro, A.P. Rifai, A survey of adaptive large neighborhood search algorithms and applications, Comput. Oper. Res. 146 (2022) 105903.

[17] P. Kalatzantonakis, A. Sifaleras, N. Samaras, A reinforcement learning-variable neighborhood search method for the capacitated vehicle routing problem, Expert Syst. Appl. 213 (2023) 118812.

[18] Y. Wang, Q. Wu, A.P. Punnen, F. Glover, Adaptive tabu search with strategic oscillation for the bipartite boolean quadratic programming problem with partitioned variables, Inform. Sci. 450 (2018) 284–300.

[19] N. Azizi, S. Zolfaghari, Adaptive temperature control for simulated annealing: a comparative study, Comput. Oper. Res. 31 (14) (2004) 2439–2451.

[20] M.G.C. Resende, C.C. Ribeiro, Greedy randomized adaptive search procedures: Advances and extensions, in: M. Gendreau, J.-Y. Potvin (Eds.), Handbook of Metaheuristics, 2019, pp. 169–220.

[21] R.D. Al-Dabbagh, F. Neri, N. Idris, M.S. Baba, Algorithmic design issues in adaptive differential evolution schemes: Review and taxonomy, Swarm Evol. Comput. 43 (2018) 284–311.

[22] P. Stodola, P. Otřísal, K. Hasilová, Adaptive ant colony optimization with node clustering applied to the travelling salesman problem, Swarm Evol. Comput. 70 (2022) 101056.

[23] R. Poláková, J. Tvrdík, P. Bujok, Differential evolution with adaptive mechanism of population size according to current population diversity, Swarm Evol. Comput. 50 (2019) 100519.

[24] B. Wei, X. Xia, F. Yu, Y. Zhang, X. Xu, H. Wu, L. Gui, G. He, Multiple adaptive strategies based particle swarm optimization algorithm, Swarm Evol. Comput. 57 (2020) 100731.

[25] P. Karakostas, A. Sifaleras, C. Georgiadis, A general variable neighborhood search-based solution approach for the location-inventory-routing problem with distribution outsourcing, Comput. Chem. Eng. 126 (2019) 263–279.

[26] T. Vidal, T.G. Crainic, M. Gendreau, C. Prins, A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows, Comput. Oper. Res. 40 (1) (2013) 475–489.

[27] T. Vidal, Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood, Comput. Oper. Res. 140 (2022) 105643.
[28] V.R. Máximo, J.-F. Cordeau, M.C. Nascimento, An adaptive iterated local search heuristic for the heterogeneous fleet vehicle routing problem, Comput. Oper. Res. 148 (2022) 105954.
[29] R. Todosijević, M. Mladenović, S. Hanafi, N. Mladenović, I. Crévits, Adaptive general variable neighborhood search heuristics for solving the unit commitment problem, Int. J. Electr. Power Energy Syst. 78 (2016) 873–883.
[30] K. Li, H. Tian, A two-level self-adaptive variable neighborhood search algorithm for the prize-collecting vehicle routing problem, Appl. Soft Comput. 43 (2016) 469–479.
[31] S.N. Bezerra, M.J.F. Souza, S.R. de Souza, A variable neighborhood search-based algorithm with adaptive local search for the vehicle routing problem with time windows and multi-depots aiming for vehicle fleet reduction, Comput. Oper. Res. 149 (2023) 106016.
[32] J.P.Q. dos Santos, J.D. de Melo, A.D.D. Neto, D. Aloise, Reactive search strategies using reinforcement learning, local search algorithms and variable neighborhood search, Expert Syst. Appl. 41 (10) (2014) 4939–4949.
[33] P. Karakostas, A. Sifaleras, C. Georgiadis, Adaptive variable neighborhood search solution methods for the fleet size and mix pollution location-inventory-routing problem, Expert Syst. Appl. 153 (2020) 113444.
[34] L.D. Pugliese, D. Ferone, P. Festa, F. Guerriero, G. Macrina, Combining variable neighborhood search and machine learning to solve the vehicle routing problem with crowd-shipping, Optim. Lett. (2022) 1–23.
[35] B. Menéndez, M. Bustillo, E.G. Pardo, A. Duarte, General variable neighborhood search for the order batching and sequencing problem, European J. Oper. Res. 263 (1) (2017) 82–93.
[36] R. Todosijević, A. Mjirda, S. Hanafi, B. Gendron, A general variable neighborhood search variants for the travelling salesman problem with draft limits, Optim. Lett. 11 (2017) 1047–1056.
[37] P. Karakostas, N. Panoskaltsis, A. Mantalaris, M. Georgiadis, Optimization of CAR T-cell therapies supply chains, Comput. Chem. Eng. 139 (2020) 106913.
[38] Y. Ren, L. Meng, F. Zhao, C. Zhang, H. Guo, Y. Tian, W. Tong, J.W. Sutherland, An improved general variable neighborhood search for a static bike-sharing rebalancing problem considering the depot inventory, Expert Syst. Appl. 160 (1) (2020) 113752.
[39] I. Krimi, R. Benmansour, Self-adaptive general variable neighborhood search algorithm for parallel machine scheduling with unrelated servers, Comput. Oper. Res. 163 (2024) 106480.
[40] P. Hansen, N. Mladenović, R. Todosijević, S. Hanafi, Variable neighborhood search: basics and variants, EURO J. Comput. Optim. 5 (3) (2017) 423–454.
[41] W. Liu, M. Dridi, J. Ren, A.H. El Hassani, S. Li, A double-adaptive general variable neighborhood search for an unmanned electric vehicle routing and scheduling problem in green manufacturing systems, Eng. Appl. Artif. Intell. 126 (2023) 107113.
[42] P. Karakostas, A. Sifaleras, The pollution traveling salesman problem with refueling, Comput. Oper. Res. 167 (2024) 106661.
[43] M. Flood, The traveling-salesman problem, Oper. Res. 4 (1) (1956) 61–75.
[44] S. Hore, A. Chatterjee, A. Deqanji, Improving variable neighborhood search to solve the traveling salesman problem, Appl. Soft Comput. 68 (2018) 83–91.
[45] M.M. Krishna, N. Panda, S.K. Majhi, Solving traveling salesman problem using hybridization of rider optimization and spotted hyena optimization algorithm, Expert Syst. Appl. 183 (2021) 115353.
[46] S.K. Rajesh Kanna, K. Sivakumar, N. Lingaraj, Development of deer hunting linked earthworm optimization algorithm for solving large scale traveling salesman problem, Knowl.-Based Syst. 227 (2021) 107199.
[47] Z. Zhang, Y. Han, Discrete sparrow search algorithm for symmetric traveling salesman problem, Appl. Soft Comput. 118 (2022) 108469.
[48] W. Li, C. Wang, Y. Huang, Y. Cheung, Heuristic smoothing ant colony optimization with differential information for the traveling salesman problem, Appl. Soft Comput. 133 (2023) 109943.
[49] G.K. Jati, G. Kuwanto, T. Hashmi, H. Widjaja, Discrete komodo algorithm for traveling salesman problem, Appl. Soft Comput. 139 (2023) 110219.
[50] P. Zhang, J. Wang, Z. Tian, S. Sun, J. Li, J. Yang, A genetic algorithm with jumping gene and heuristic operators for traveling salesman problem, Appl. Soft Comput. 127 (2022) 109339.