RESEARCH

Discrete Puma Optimizer to Solve Combinatorial Optimization Problems

Ferzat Anka · Farhad Soleimanian Gharehchopogh · Seyed Jalaleddin Mousavirad · Ghanshyam G. Tejani

Received: 22 October 2025 / Revised: 26 November 2025 / Accepted: 7 January 2026

Abstract
Discrete and combinatorial optimization problems such as routing, scheduling, and resource allocation present high computational complexity, limiting the effectiveness of classical exact optimization methods. Most existing metaheuristic (MH) algorithms are originally designed for continuous domains and require transformation procedures that often degrade performance when applied to discrete problems. This study introduces the discrete puma optimizer (DPO), a new variant metaheuristic algorithm developed to operate directly within discrete search spaces by employing discrete-specific operators and adaptive exploration–exploitation strategies. DPO is applied to 7 real-world optimization problems, including the Traveling Salesman Problem, Smart Grid Optimization, Factory Production Planning, Vehicle Routing Problem, Modern TSP, Team Orienteering Problem, and Electric Vehicle Charging Station Location Optimization, and evaluated on a total of 22 small-, medium-, and large-scale dataset instances. The performance of DPO is benchmarked against 9 various and well-known MH algorithms. Experimental results show that DPO attains superior best and mean solutions, lower variance, and faster stabilization in convergence behavior. Wilcoxon signed-rank tests confirm the statistical significance of the observed improvements, particularly in large-scale scenarios where competing methods show marked degradation. Comprehensive cross-problem rankings further illustrate DPO's enhanced generalizability and scalability. These results position DPO as an effective and robust approach for real-world large-scale discrete optimization tasks.

Keywords Discrete puma optimizer · Discrete optimization problems · Combinatorial optimization problems · Metaheuristic

1 Introduction

Optimization is a crucial instrument for enhancing decision-making processes and increasing resource utilization across several industries, including engineering, logistics, manufacturing, and finance [1–3]. Depending on the issue's nature, it may encompass several classes of variables and constraints, reflecting diverse sorts of optimization issues [4, 5]. Continuous, discrete, and combinatorial optimization represent prominent areas of research, each characterized by unique formulations and solution techniques. However, discrete and combinatorial optimization

Published online: 17 February 2026

problems are particularly pertinent from an application perspective, since they involve selecting the optimal combination of candidates from a limited set of options rather than from a continuous spectrum. Discrete optimization problems emerge in numerous practical optimization contexts, including feature selection in machine learning, genome sequence analysis in bioinformatics, inventory optimization in supply chain management, network routing in telecommunications, motion planning in robotics, scheduling, and portfolio optimization in finance [6–8]. In contrast to continuous optimization, which permits decision variables to assume any actual value within a specified range, discrete optimization restricts solution variables to certain values, hence increasing complexity. Moreover, combinatorial optimization is sometimes seen as a subset of discrete optimization, as the identification of optimal combinations of components is a fundamental concept in combinatorial optimization [9]. Figure 1 illustrates a comprehensive classification of many types of optimization problems.

Conventional optimization methods, such as linear programming, branch and bound strategies, and integer programming, may produce exact solutions for small instances and frequently operate efficiently for small examples of discrete and combinatorial optimization challenges. However, as the solution space grows exponentially with the complexity of the problem, these methods quickly become computationally unfeasible, necessitating the adoption of metaheuristic (MH) algorithms [10–12]. Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), Artificial Rabbit Optimization (ARO), Whale Optimization Algorithm (WOA), Sand Cat Swarm Optimization (SCSO), and Grey Wolf Optimization (GWO) are MH algorithms widely utilized for various problems owing to their effectiveness in navigating and utilizing extensive search spaces [13]. However, despite their broad application, most were initially created for continuous optimization and often require further adjustments for discrete contexts. Transformations such as discrete rounding or encoding–decoding methods often compromise the intrinsic structure of the solution space, reduce optimization efficacy, and increase the likelihood of being trapped in local minima [9]. Considering that most real-world optimization problems use the intrinsic properties of discrete choice variables, there is an increasing tendency to develop MH algorithms based on discrete optimization principles. In complex discrete problems, effectively traversing the solution space and circumventing local minima results in significant progress that illustrates generalizability. Therefore, it is imperative to prioritize the development of MH algorithms that operate in discrete domains free from local bias.

1.1 Motivation and Contribution

MH algorithms specifically designed for discrete and combinatorial optimization are not as numerous as others, despite the frequency of such scenarios. Although a large number of MH algorithms have been applied to combinatorial optimization, many of these methods were originally developed for continuous domains and rely

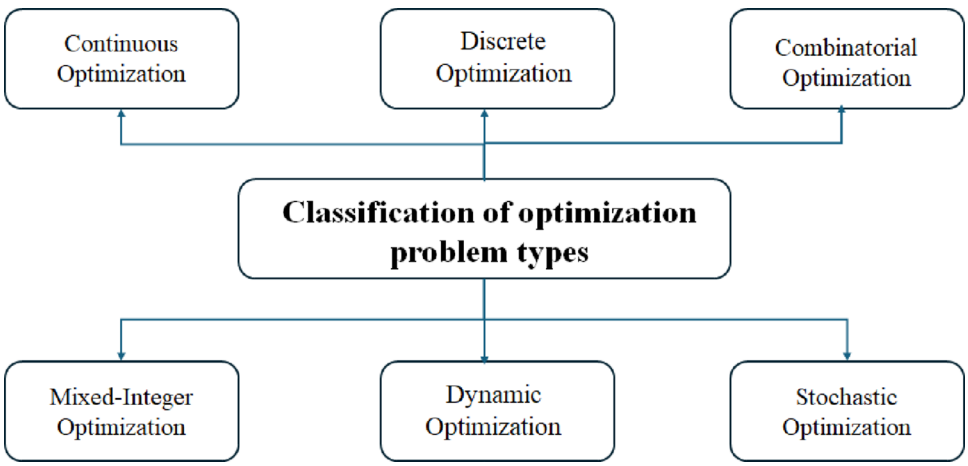


Fig. 1 General classification of optimization problem

on continuous-to-discrete transformation schemes when used for permutation or integer problems. In contrast, relatively fewer methods are designed natively for discrete/combinatorial search spaces with operators specifically tailored to permutation, assignment, and routing representations. The primary emphasis of contemporary MH methodologies is on continuous optimization; hence, transformation techniques are required for their direct application to discrete scenarios. Nevertheless, these alterations frequently diminish the efficiency of the search process, impair optimization efficacy, and compromise the structure of the solution space. Traditional MH algorithms in discrete domains are fundamentally constructed for continuous spaces, resulting in significant limitations in their exploration and exploitation activities. Consequently, further adjustments such as discrete rounding or encoding-decoding alterations are typically necessary when addressing discrete issues. Although these adjustments facilitate the application of continuous-based MH algorithms to discrete problems, they have downsides such as diminished convergence efficiency, reduced accuracy, and an increased likelihood of stagnation in suboptimal solutions.

To address these challenges, this study introduces the Discrete Puma Optimizer (DPO), a novel MH algorithm specifically designed for discrete optimization problems. DPO is an adaptation of the Puma Optimizer (PO) [14], a biologically inspired MH algorithm recognized for its balance between exploration and exploitation. Furthermore, in contrast to several MH algorithms that demonstrate premature convergence or excessive unpredictability, PO facilitates a well-calibrated search process by adaptively modifying the search behavior in accordance with the optimization landscape. This flexibility is especially beneficial for discrete and combinatorial optimization problems, where diversity in the search space is essential to circumvent local optima. Furthermore, a significant justification for selecting PO is that, unlike other prevalent MH algorithms, it has not been previously tailored for discrete optimization. Nevertheless, as the original PO was designed for continuous optimization, it cannot be readily utilized in discrete domains. The proposed DPO addresses this issue by integrating discrete-specific operators and processes that enable effective navigation of discrete solution spaces. DPO incorporates numerous critical advancements to improve its efficacy in discrete optimization. The salient aspects are enumerated as follows.

1. Operators specific to discrete optimization (swap, reordering, and mutation) allow direct manipulation of discrete variables free from transformation.
2. Direct processing of discrete solutions, therefore avoiding the need for rounding or coding-decoding transformations prescribed by continuous optimization strategies.
3. Improved exploration and exploitation techniques ensure computational viability and help to effectively search for large-scale discrete problems.
4. Thorough evaluation of seven real-world discrete optimization problems highlighting DPO's dependability and efficiency in different spheres of application.

Discrete optimization problems often become ensnared in local minima due to their extensive solution spaces. DPO, through its dynamic exploration and exploitation methods, can efficiently navigate a vast solution space and provide high-quality solutions by expediting the convergence process. The contributions of this research to the literature can be encapsulated as follows.

1. Creation of a novel MH algorithm for discrete optimization through the adaptation of PO into a discrete form.
2. Integration of specific mechanisms enabling MH algorithms, initially developed for continuous problems, to proficiently address discrete optimization.
3. A thorough assessment of well-known practical discrete optimization issues, demonstrating DPO's adaptability and efficacy across several fields.

This paper is organized as follows. Section 2 reviews the existing discrete MH algorithms. Section 3 describes the DPO algorithm with its mathematical equations. Section 4 focuses on solving 7 real-world discrete problems and analyzes the results by comparing them with 9 well-known MH algorithms. Sections 5 and 6 present the discussion and limitations, respectively. The last section includes conclusions and future work.

2 Related Works

One of the common uses of MH algorithms is their effect on solving discrete and combinatorial optimization problems. These algorithms are extensively employed to address discrete and combinatorial optimization issues, offering computationally efficient alternatives to exact approaches, especially for large-scale challenges. Numerous MH-based methodologies have been developed and effectively used to address diverse real-world challenges, including scheduling, routing, network design, feature selection, and resource allocation. This section presents a summary of significant MH-based techniques formulated for discrete optimization.

GAs are among the first and most prevalent MH techniques for combinatorial problems. They utilize evolutionary concepts, including selection, crossover, and mutation, to effectively navigate the solution space. It has been effectively utilized in task scheduling, facility location, and vehicle routing challenges [15–17]. Another prominent MH algorithm is PSO. Chen et al. [18] introduced Set-Based PSO (S-PSO) for discrete optimization challenges. S-PSO, in contrast to conventional PSO, is specifically formulated to tackle Combinatorial Optimization Problems (COPs) within discrete spaces. S-PSO uses cluster representation to define solutions and velocities, with updates executed by probabilistic cluster-based processes. This technique has been evaluated on challenges including the Traveling Salesman Problem (TSP) and the Multidimensional Knapsack Problem (MKP). In [19], a new discrete PSO variation known as the Integer and Categorical PSO (ICPSO) was presented. In contrast to traditional PSO, ICPSO characterizes particle positions as probability distributions, which are modified by PSO update equations. This approach improves rapid convergence and diversification in combinatorial optimization challenges. Wang et al. [20] proposed an improved MH approach that integrates Differential Evolution (DE) and PSO to boost global optimization results. This synergistic framework utilizes DE's strong exploratory capabilities in huge search spaces in conjunction with PSO's swift convergence attributes, promoting a balanced approach between broad exploration and targeted exploitation in the optimization process.

Cinar et al. presented a modified version of the Tree Seed Algorithm (TSA) for permutation-based optimization problems, termed DTSA [21]. This variation employs swap, shift, and symmetry transformation operators, supplanting traditional TSA update protocols. Experimental assessments demonstrate that DTSA is exceptionally effective and competitive in addressing discrete optimization problems. An alternative method for addressing the TSP is the Quantum Approximate Optimization Algorithm (QAOA), as presented by Qian et al. [22]. This research improves qubit encoding and optimization techniques by assessing various mixer designs. Findings indicate that a properly calibrated QAOA mixer markedly enhances solution precision and efficacy in quantum simulators. Reddy et al. [23] devised three distinct binary WOA variations, including tangent hyperbolic, inverse tangent (arctan), and sigmoid transfer functions to address profit-driven unit commitment optimization challenges. Li et al. [24] introduced a novel binary PSO (BPSO) approach, integrating multiple mutation techniques (MMBPSO) to effectively address the 0–1 Knapsack Problem (KP). Another study [25] proposed DWOA to enhance discrete-space solutions with an innovative V-shaped transfer function. The DWOA algorithm was evaluated against GA, DE, and several heuristic approaches, exhibiting either superior or competitive outcomes. Experimental studies demonstrate that DWOA exhibits outstanding performance on 0–1 KP and D{0–1}KP, rendering it suitable for extensive discrete optimization challenges.

The Discrete ABC Algorithm with Fixed Neighborhood Search (DABC-FNS) is an innovative form of the ABC algorithm, created in [26] to address the intrinsic constraints of traditional ABC approaches. These shortcomings encompass the propensity of these approaches to prematurely converge to local optima and inadequate convergence precision in applications related to the TSP. Fixed Neighborhood Search (FNS) techniques are cohesively included in the DABC-FNS framework, enhancing both exploration diversity and exploitation intensity. This enhances solution quality and algorithmic stability in combinatorial optimization challenges. It specifically integrates fixed neighborhood search to enhance the efficacy of the ABC algorithm. This is achieved by integrating a local search approach with a 2-opt enhancement strategy. This technique effectively balances local and global search, rendering it a optimal selection for addressing discrete and combinatorial optimization challenges. The

Discrete ARO (DARO) approach was developed as an enhanced iteration of the ARO algorithm to optimize steel truss systems [27]. The conventional initialization approach of the Arithmetic Optimization Algorithm (ARO), which relies on random initialization techniques, has been supplanted by the Diagonal Linear Uniform (DLU) method. This modification was implemented to enhance the algorithm's efficacy. This systematic initialization technique enhances search efficiency and increases solution quality by generating well-distributed first candidate solutions. This technique addresses intrinsic drawbacks, including uncertain solution distribution and inefficient convergence behavior in large optimization landscapes.

A further study [28] introduced the Discrete Chimp Optimization Algorithm (DChOA), a novel method for addressing the Tardy/Lost (TL) penalty scheduling issue. The TL scheduling problem is a discrete NP-hard issue that necessitates work sequencing on a singular machine within defined deadlines. DChOA modifies the ChOA algorithm for discrete optimization challenges, integrating a novel swap operator tailored for TL scheduling. Furthermore, Discrete Gorilla Troops Optimization (DGTO) was created as a discrete variant of the GTO method [29]. DGTO incorporates 2-opt and inversion neighborhood search operators, hence augmenting its efficacy in solving the TSP. This study modifies GTO for discrete permutation-based tasks, such as the TSP, due to GTO's performance in continuous optimization. Ultimately, the Discrete GWO (DGWO) was presented as an enhanced discrete variant of the GWO algorithm [30]. Initially intended for continuous optimization, DGWO has been modified for discrete combinatorial optimization challenges, including the TSP. To augment DGWO's efficacy, a 2-opt local search operator was integrated, resulting in enhanced solution quality. The study in [31] proposes a new bio-inspired algorithm called Olympiad Optimization Algorithm (OOA) for Software Module Clustering (SMC), a discrete optimization problem. In this NP-complete problem, modules are clustered using Module Dependency Graph (MDG) and modularization quality (MQ) is increased. OOA claims to achieve better results than methods such as PSO, GA, GWO with swarm intelligence and group-based learning techniques. Also, in [32] the authors tackle multi-objective scheduling problems in manufacturing systems as a combinatorial optimization problem with a new hybrid MH algorithm. They combine GA and Tabu Search (TS) into a method that should lead to faster solution and better quality than independent implementations. Taking into account realistic production constraints in the form of machine capabilities, processing times and workflow dependencies, the authors optimize using the makespan and total tardiness as metrics. Experiments on industrial-scale benchmark scenarios show the hybrid GA-TS algorithm to be superior to several existing optimization techniques, both in efficiency and solution quality. In a later study [33], the authors study the connection between the structural and combinatorial properties of optimization problems and their influence on the performance of MH algorithms. By studying different problem instances, the authors wonder whether it is possible to detect the link how diversity measures, derived measures contribute to both solution quality and search-space exploration measures, are informative for algorithm behavior. The authors study statistically if these measures of search-space exploration correlate with performance, and whether their findings on a combinatorial optimization problem are relevant. The research shows that some search-space exploration metrics are indeed informative signals about algorithm effectiveness, shedding light on MH performance on combinatorial optimization problems. In another case study from the COP [34], authors study a combinatorial optimization problem that is particularly complex since it is the two-dimensional bin-packing problem with conflict constraints and load balancing requirements. The authors define the problem as an integer programming model coming up a feasible solution within this framework is NP-hard because of the simultaneous mix of handling spatial arrangement, weight capacity, item conflicts, and barycenter constraints, and present a tactic to search this extremely large combinatorial problem space, using a Hybrid Chaotic and Evolutionary PSO (HCEPSO) algorithm that builds upon PSO by injecting evolutionary operators and chaotic search. Experiments on datasets of varying conflict densities show that HCEPSO outperforms other techniques for finding solutions, among them classical PSO, GA, Hybrid GA combined with Variable Neighborhood Search algorithm (GAVNS), and Memetic algorithm (MA) in terms of solution, convergence and stability qualities, and computational efficiency. The study contributes to combinatorial optimization research with an interesting hybrid MH capable of dealing with the constraints of packing. Also, in [35] authors conduct a comparison of several heuristic methods for the Travelling Salesman Problem (TSP). TSP is a classic combinatorial problem.

They study the Nearest Neighbor (NN), GA, and Simulated Annealing (SA) methods, as a tradeoff between solution quality versus computation time. While NN offers satisfactory results over a short time, the latter two achieve even better results for higher computation times, and it is mentioned that hybrid strategies can also offer better results for large TSP instances. Table 1 summarizes some characteristics of studies found in the literature.

Recent years have seen many discrete and hybrid adaptations of classical MHs (for example, discrete variants of PSO, DE, GWO, ABC, ACO, and newer hybrid methods or neighborhood-guided approaches). These contributions demonstrate that the research trend is moving toward specialized discrete operators and hybrid combinations; however, comprehensive cross-problem evaluations remain limited. The proposed DPO complements this trend by providing a single, native discrete MH algorithm evaluated across multiple combinatorial benchmarks.

Table 1 Comparison of MH-based discrete and combinatorial optimization studies

Study	Features	Advantages	Disadvantages	Target problems
S-PSO [18]	Set-based PSO usage	Effective in COPs, fast convergence	Complexity in cluster-based updating	TSP, MKP
ICPSO [19]	Integer and categorical PSO	Fast convergence and diversification	Retains classic PSO drawbacks	General COPs
DE + PSO [20]	Combination of differential evolution and PSO	Combines DE's global search with PSO's fast convergence	High computational cost	General global optimization
DTSA [21]	Includes swap, shift, and symmetry transformations	Stronger local search than TSA	Increases TSA's computational cost	TSP, scheduling
QAOA [22]	Quantum approximate optimization	Qubit-based optimization	Requires quantum hardware	TSP
BPSO [24]	PSO enhanced with multiple mutation strategies	Optimized for knapsack problem	Classic PSO drawbacks remain	0–1 knapsack
DWOA [25]	V-shaped transfer function	Better performance than GA and DE	Limited general adaptability of the transfer function	0–1 and D{0–1} knapsack versions
DABC [26]	Fixed neighborhood search-enhanced ABC	Prevents ABC from getting trapped in local maxima	Increases ABC's computational load	TSP
DARO [27]	Diagonal linear uniform initialization in ARO	Produces better solutions than standard ARO	More complex than ARO	Steel truss structures
DChOA [28]	Discrete chimp optimization + swap operator	Optimized for TL scheduling	Applicable to specific problems only	TL scheduling
DGTO [29]	Discrete GTO + 2-opt and Inversion operators	GTO adapted for TSP	Less studied compared to continuous version	TSP
DGWO [30]	Discrete GWO + 2-opt local search	Discrete version of GWO	Higher computational cost than continuous version	TSP
2-Opt++ [32]	Enhanced 2-opt local search with graph compression (candidate list)	Faster convergence, improved approximation ratio, strong performance on large-scale TSP	Still a local search method; may stagnate without diversification	TSP, VRP
HMS-PSO [33]	Hybrid multi-strategy PSO including Lévy flight, adaptive parameters	Higher exploration capability, better accuracy, faster convergence	More complex parameter adaptation; increased computational load	General discrete COPs
GS [34]	Strengthened greedy constructive heuristic with neighborhood refinement	Simple, fast, efficient for medium-scale COPs	Lacks global search; sensitive to starting point	TSP-like routing and scheduling problems
POMA [35]	Hybridization of movement, grouping, and adaptive operators	Balanced exploration–exploitation, strong robustness across discrete tasks	Higher complexity; parameter tuning required	Discrete COPs, routing and assignment

3 Proposed Method

This section presents the operational ideas and mathematical framework of the Discrete Puma Optimizer (DPO) algorithm. DPO is an adaptation of the existing PO algorithm [14] tailored for discrete optimization problems, featuring specialized exploration and exploitation techniques that operate directly with discrete variables.

3.1 Discrete Puma Optimizer (DPO) Structure

DPO is a biologically inspired MH algorithm, derived from the hunting behavior of pumas. Pumas discover probable prey by ambient exploration and attempt to capture specific prey by targeting it. This nature-inspired structure enhances the capacity to explore an extensive solution space and settle on the high-quality solutions in optimization procedures. DPO provides an algorithm capable of functioning without continuous optimization by tailoring the exploration and exploitation phases specifically for discrete optimization challenges. Consequently, it may directly explore the discrete solution space without the need for variable transformations. The DPO method fundamentally comprises Initial Population Creation, Discrete Exploration Mechanism, Discrete Exploitation Mechanism, and Score Calculation and Update Mechanism. The operational mechanisms of these four components are elucidated below.

3.2 Mathematical Model and Update Mechanisms

The optimization process of DPO is based on mathematical update mechanisms that balance the exploration and exploitation stages. In each step of the algorithm, solution candidates either explore new regions with a certain probability by exploration or exploitation and try to produce solutions closer to the current best solution.

3.2.1 Initial Population Creation

The DPO establishes the initial population using random permutations or distinct values within the discrete solution space. A solution set 'X' is defined as $X = \{X_1, X_2, \ldots, X_N\}$, where '$X_i$' represents a prospective solution for the discrete problem. 'N' denotes the population size. Furthermore, the magnitude of the issue is denoted by 'D'. Solutions are produced directly within a discrete solution space and initialized randomly. Every solution is configured to be appropriate for discrete optimization. Figure 2 illustrates the DPO algorithm, showcasing a two-dimensional matrix of agents and their respective problem dimensions.

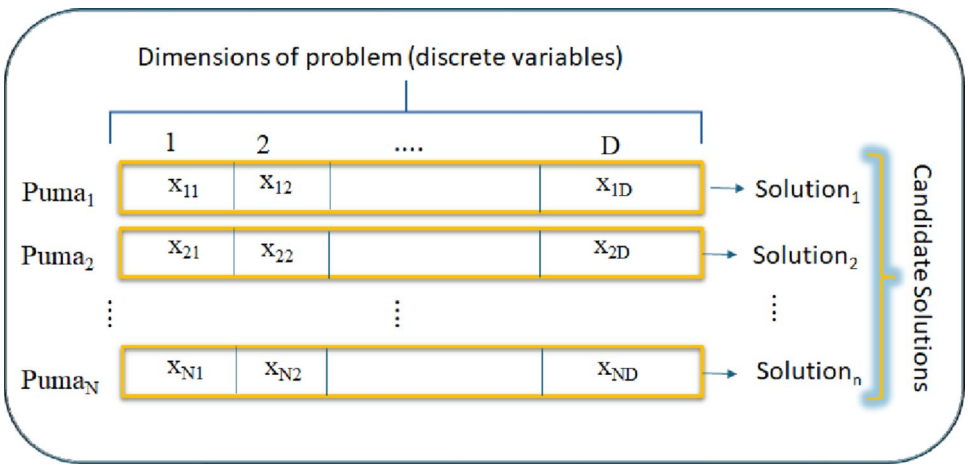


Fig. 2 Matrix population of DPO algorithm

3.2.2 Discrete Exploration Mechanism

The exploration phase is a critical component that enables the assessment of a broad solution space during the optimization process. DPO utilizes an exploration mechanism based on proximate solutions and functions with discrete variables to enhance the exploration process in discrete optimization problems. This phase aims to preserve diversity by exploring different regions of the solution space and pinpointing solutions closer to the global optimum. If exploitation is performed, the algorithm may prematurely converge and become trapped in a local optimum. Thus, an appropriate equilibrium during the exploration phase is essential for a robust optimization process. Exploration tactics for continuous optimization frequently include derivative-based updates and random routing; however, in discrete optimization problems, the solution space consists of discrete values, making continuous methods inapplicable. To mitigate this challenge, DPO improves the discrete exploration process with a neighbor-based search mechanism and Hamming distance (Eq. 1).

$$Y = X_a + G \cdot (X_a - X_b) + G \cdot ((X_a - X_b) - (X_c - X_d) + (X_c - X_d) - (X_e - X_f)) \tag{1}$$

Here 'X_a', 'X_b', 'X_c', 'X_d', 'X_e', 'X_f' are random solutions in the population and 'G' is a random discrete factor. Then, the 'Y' value is revised as in Eq. 2, where 'Y', is defined to keep the new solution within bounds. In this case, the discrete update is calculated according to the following equation.

$$Y = max\left(min\left(Y, UB\right), LB\right) \tag{2}$$

where 'LB' and 'UB' parameters give lower bound and upper bound information of the search space. The discrete position update is determined using Eq. 3. This method guarantees that the solution is confined to a particular discrete domain. The proposed technique is not articulated in strict binary form, allowing it to address a range of discrete and combinatorial issues through suitable variable mapping. This model is utilized in the subsequent phase as well.

$$X_{new} = round\left(Y\right) \tag{3}$$

At this point, existing solutions are altered in accordance with specific guidelines while new ones are being developed. Neighboring solutions are first modified. Additionally, bit-flip or Hamming distance operations are used to alter solution components. Additionally, solution updates are carried out using discrete operators rather than continuous mathematical functions because exploration takes place in a discrete domain. By scanning a wide solution space, it avoids local minima and keeps the algorithm from converging too soon. Additionally, it maintains the diversity of solutions: It lessens the possibility of becoming trapped in the same solution by varying the components of the solution using operators like swap and Hamming distance. It also updates solutions on the neighborhood: The optimization process proceeds in a more controlled way rather than at random by producing new solutions from related solutions.

3.2.3 Discrete Exploitation Mechanism

The exploitation phase enhances the optimization process by concentrating on the most effective options. This phase is executed to meticulously investigate the potential solution areas found during the exploration phase and to converge on the optimal solution. The exploration step investigates a vast solution space, while the exploitation process enhances the optimal solutions, thereby yielding the final optimization. The discrete exploitation phase of DPO involves the iterative enhancement of the optimization process by generating solutions that approximate the optimal accessible solution. In discrete optimization, the exploitation mechanism must operate directly on discrete variables and should not be influenced by continuous functions. Consequently, the exploitation mechanism of DPO employs discrete operators and reordering tactics that facilitate convergence to optimal solutions.

The solution incorporates unpredictability in the update process, facilitating diverse exploration trajectories as dictated by Eqs. 4 and 5.

$$S_1 = R_1 + randn\,(1, dim) \tag{4}$$

$$S_2 = (F1 \cdot R_1 \cdot X + F2 \cdot (1 - R_1) \cdot X_{best}) \tag{5}$$

While 'S_1' represents random changes in the solution space, 'S_2' determines a new transition point between the current solution and the best solution. Indeed, 'S_1' is a mutation adjustment and 'S_2' is used for solution update in exploitation. 'X_{best}' indicates the best solution.

$$R_1 = 2 \cdot rand - 1 \tag{6}$$

In addition, the parameter 'R_1' is a factor that is responsible for driving the movement of random exploration, and it is determined using Eq. 6. This particular value is a number that is selected at random from the range of [−1 to 1], and it serves as one of the guiding variables in the process of exploration. Transition between stages in a balanced and efficient manner is made possible by these parameters. The '$F1$' factor affects the intensity of exploration by modulating the movement of solutions, as determined by Eq. 7. The '$F2$' factor, which influences the degree of exploitation by modifying the step size in the solution update process, is derived from Eq. 8. The parameters '$PF(1)$', '$PF(2)$', and '$PF(3)$' are adaptive coefficients limited to the range [0, 1]. These coefficients regulate the balance between global exploration and local exploitation dynamics, ensuring a coordinated interaction of diversification and intensification techniques during the algorithm's repeated optimization phases. In these expressions, '*Seq_Cost_Exploit*' refers to the combined contribution to total cost of the consecutive local optimization moves that have been executed on a solution during DPO's exploitation phase. Indeed, this variable tracks the total improvement in solution cost provided by applying consecutive, local refinement-type updates during the exploitation phase. A given local move results in a reduction of cost by a certain amount, and hence we can say that '*Seq_Cost_Exploit*' is the total improvement cumulatively supplied by that sequence of local improvement operations. Similarly, '*Seq_Time_Exploit*' refers to the total time taken to perform the sequential local optimization on a candidate solution during the exploitation phase. We might interpret this as how much processing time the DPO takes in improving the candidate solution through consecutive adjustments to the neighborhood, thus indicating the depth and cost of the exploitation.

$$F1 = F1_{Exploit} = PF\,(1) \cdot \frac{Seq\_Cost\_Exploit\,(1)}{Seq\_Time\_Exploit\,(1)} \tag{7}$$

$$F2 = F2_{Exploit} = PF\,(2) \cdot \frac{Seq\_Cost\_Exploit\,(1) + Seq\_Cost\_Exploit\,(2) + Seq\_Cost\_Exploit\,(3)}{Seq\_Time\_Exploit\,(1) + Seq\_Time\_Exploit\,(2) + Seq\_Time\_Exploit\,(3)} \tag{8}$$

On the other hand, the position update in the exploitation phase is calculated according to Eq. 9. Where '$(-1)^{randi([0,1])}$' adds variety by allowing the solution to change in certain directions. Also, 'm_{best}' average of the best solutions in the population. The mean of the best solutions in the population is the value of this parameter, which is utilized to strike a balance between exploration and exploitation. Moreover, the scaling parameter in the exploitation phase is denoted by the 'β'. It yields values in the range [0, 2]. If a symmetric coefficient in [−1, 1] is desired for 'β', 'β'=2·*rand()*−1 should be used instead. The pace of change in exploit updates is controlled by this scaling parameter, which will be discussed further below. Equations 10 and 11 are used to derive these coefficients and parameters.

$$X_{new} = \frac{m_{best} \cdot X_r - (-1)^{randi([0,1])} \cdot X}{1 + (\beta_n \cdot rand)} \tag{9}$$

$$m_{best} = \frac{1}{n} \sum_{i=1} X_i \tag{10}$$

$$\beta = 2 \cdot rand \tag{11}$$

In addition to this, Eq. 12 makes an effort to maintain discrete values through the modulation process. To put it another way, it guarantees that the 'X_{new}' value that is produced by Eq. 9 will continue to exist within the discrete solution space. Due to the fact that discrete problems can only have a limited number of integer values, the modulation operation places restrictions on the solution. Where the letter 'K' denotes the total number of discrete values that are feasible.

$$X_{new} = mod\left(X_{new}, K\right) \tag{12}$$

In the discrete exploitation phase of the DPO, controlled modifications can be made to the optimal available solutions, enhancing the optimization process. This method employs operators designed expressly for discrete optimization to do local searches around the best available solutions, hence improving solution quality. Furthermore, it can navigate an extensive solution space, preventing entrapment in local optimal solutions. The ability to operate directly with discrete data structures obviates the need for transformations, constituting an additional advantage.

3.2.4 Score Calculation and Update Mechanism

The score calculation process is a critical component that dynamically drives the exploration and exploitation processes of the DPO algorithm. It is used to evaluate the quality of each individual solution in the optimization process and to balance whether the algorithm performs a global or local search. In this regard, the solution space is governed by changes made between discrete data points, rather than by differential computations as in continuous optimization problems. Therefore, the score calculation mechanism of DPO is designed to ensure optimal transition between discrete variables.

The success rate of DPO in the optimization process depends on achieving a balance between the exploration and exploitation processes. To attain this equilibrium, each prospective solution is evaluated and rated according to a defined set of criteria. This method operates in line with Eqs. 13 and 14. Equations 15- 20 elucidate the methodologies applicable for calculating the parameters delineated by these equations. In the exploration phase, Eq. 13 includes the factors that enhance diversity and facilitate comprehensive search. A greater number of candidate solutions are generated in order to search across the extensive solution space when the exploration process is high. Conversely, Eq. 14 determines the degree of aggression to be employed in the exploitation process. This facilitates a more rapid attainment of the high-quality solution.

$$Score_{Explore} = \left(Mega_{Explore} \cdot F1_{Explore}\right) + \left(Mega_{Explore} \cdot F2_{Explore}\right) + \left(\lambda_{Explore} \cdot \left(min\left(PF_{F3}\right) \cdot F3_{Explore}\right)\right) \tag{13}$$

$$Score_{Exploit} = \left(Mega_{Exploit} \cdot F1_{Exploit}\right) + \left(Mega_{Exploit} \cdot F2_{Exploit}\right) + \left(\lambda_{Exploit} \cdot \left(min\left(PF_{F3}\right) \cdot F3_{Exploit}\right)\right) \tag{14}$$

$$Mega_{Explore} = max\left(\left(Mega_{Explore} - 0.01\right), 0.01\right) \tag{15}$$

$$F1_{Explore} = PF\left(1\right) \cdot \frac{Seq\_Cost\_Explore\left(1\right)}{Seq\_Time\_Explore\left(1\right)} \tag{16}$$

$$F2_{Explore} = PF\left(2\right) \cdot \frac{Seq\_Cost\_Explore\left(1\right) + Seq\_Cost\_Explore\left(2\right) + Seq\_Cost\_Explore\left(3\right)}{Seq\_Time\_Explore\left(1\right) + Seq\_Time\_Explore\left(2\right) + Seq\_Time\_Explore\left(3\right)} \tag{17}$$

$$Mega_{Exploit} = max\left(\left(Mega_{Exploit} - 0.01\right), 0.01\right) \tag{18}$$

$$F3_{Explore} = F3_{Exploit} = PF\left(3\right) = min\left(Seq\_Cost\_Explore\left(1\right), Seq\_Cost\_Exploit\left(1\right)\right) \tag{19}$$

$$\lambda_{Explore} = 1 - Mega_{Explore}; \quad \lambda_{Exploit} = 1 - Mega_{Exploit} \tag{20}$$

where '$Mega_{Explore}$' and '$Mega_{Exploit}$' are weighting factors that dynamically adjust the impact of exploration and exploitation processes, respectively. The optimum transition between exploration and exploitation

processes is achieved by using these weights. These parameters are positive scaling factors that the clamp max((Mega − 0.01),0.01) enforces a minimum value of 0.01 to avoid numerical zeroing. Also, '$F1_{Explore}$' is a component that determines the movement intensity in the exploration process and is modulated by '$Mega_{Explore}$', while '$F2_{Explore}$' reflects the solution movements that increase diversity, and '$F3_{Explore}$' is the third factor that adjusts the exploration intensity based on the previous search efficiency. Similarly, '$F1_{Exploit}$' is specific to the exploitation process and is modulated by '$Mega_{Exploit}$', while '$F2_{Exploit}$' increases the local search efficiency, and '$F3_{Exploit}$' is the third factor that adjusts the exploitation intensity based on the previous best solutions. In addition, a score balancing mechanism is also in operation in DPO. In order to ensure optimal transition between exploration and exploitation processes, '$Mega_{Explore}$' and '$Mega_{Exploit}$' weights are dynamically updated and at the same time reflect their effect in Eq. 20. If the exploration process is high ('$Mega_{Explore}$' is large), diversity is increased, and new solutions are explored. '*Seq_Cost_Explore*' is the cost improvement (positive scalar) obtained at the k-th sequential local refinement step during exploration phase. 'Seq_Time_Explore' is computational time (in seconds, or algorithm time units) required to perform each local refinement step. Also, '$PF(i)_{i=1,2,3}$' are priority factors (scalar weight) assigned to each feature channel '$F1$' to '$F3$'. 'PF' maps feature index to a normalized weight in [0,1]. In our implementation '$PF(i)$' is a normalization/importance factor computed from observed feature statistics across the current population: we normalize the raw feature values via min–max scaling and optionally apply a small regularization to avoid zero weights. Finally, '$Score_{Explore}$' and '$Score_{Exploit}$' scalar scores that quantify the exploration and exploitation utility of a newly generated candidate solution. These scores are combined to derive the operator-selection probability, respectively. On the other hand, if the exploitation process is high ('$Mega_{Exploit}$' is large), existing solutions are analyzed in more detail and convergence to the best solution is achieved. Indeed, the decision mechanism of transition between phases is based on Eq. 21.

$$\begin{cases} Exploration; & if\ Score_{Explore} > Score_{Exploit}\ or\ Mega_{Explore} > Mega_{Exploit} \\ Exploitation; & Otherwise \end{cases} \tag{21}$$

Consequently, the score calculation mechanism of DPO establishes a dynamic equilibrium between exploration and exploitation activities. It enhances the solution process by taking into account prior optimization stages. It is compatible with discrete data structures and removes reliance on continuous optimization methods. It also regulates the investigation process and mitigates superfluous movements to get closer to the global optimum more swiftly.

3.2.5 Intuition Behind the Dynamic Scoring Mechanism

The proposed dynamic scoring mechanism regulates the trade-off between exploration and exploitation throughout the optimization process. The dynamic scoring mechanism is designed to regulate the balance between exploration and exploitation throughout the search process. At each iteration, two adaptive measures ('$Score_{Explore}$' and '$Score_{Exploit}$') quantify the contribution of a newly generated solution to global diversification and local intensification, respectively. '$Score_{Explore}$' increases when the candidate solution exhibits structural novelty, such as a large positional deviation from the elite set in discrete permutations or increased variance among solution components. Conversely, '$Score_{Exploit}$' increases when the solution yields direct improvement relative to the current best or positively affects the elite solution pool. The sum of both components is normalized to 1. Also, the operator-selection probability is updated accordingly, allowing the algorithm to automatically shift between broad exploratory moves and focused local refinements.

A typical run of the algorithm provides examples of this adaptive behavior in action. In the early stages of the run (e.g., 1–20), solutions are highly diverse so the value of '*ScoreExplore*' is high (typically between 0.6 and 0.8), and consequently exploration of the solution space is wide ranging. In the mid-phase of the run (e.g. iterations 40–60), the scores are roughly equal (e.g. 0.45–0.55) and the search will typically be balanced between diversification and intensification. In the final stages of the run (say iterations 80–100), the improvements tend to be somewhat more local so '*ScoreExploit*' (between 0.6 and 0.8) becomes higher. This natural tendency directs

the algorithm towards exploiting promising regions of the solution space whilst damping down overly destructive exploratory moves.

3.3 Flowchart, Pseudocode, and Computational Analysis of DPO

This section presents a comprehensive depiction of the proposed algorithm via its flowchart, schematic operational mechanism, and accompanying pseudocode. A thorough temporal complexity study is performed to assess the algorithm's computational efficiency. Figure 3 roughly illustrates the shifts and characteristics of DPO across phases. This figure conveys the essence of the core switching pattern in the proposed dynamic. When $Score_{Explore}$ is greater than $Score_{Exploit}$, the population is found searching more widely in a scattered manner to maintain the global diversity in the population. With $Score_{Explore}$ becoming lower than $Score_{Exploit}$, the search switches to a more focused and intense exploitation of the best approximations it has found. Furthermore, the structural workflow of DPO is depicted in Fig. 4, and its sequential implementation is detailed in Algorithm 1.

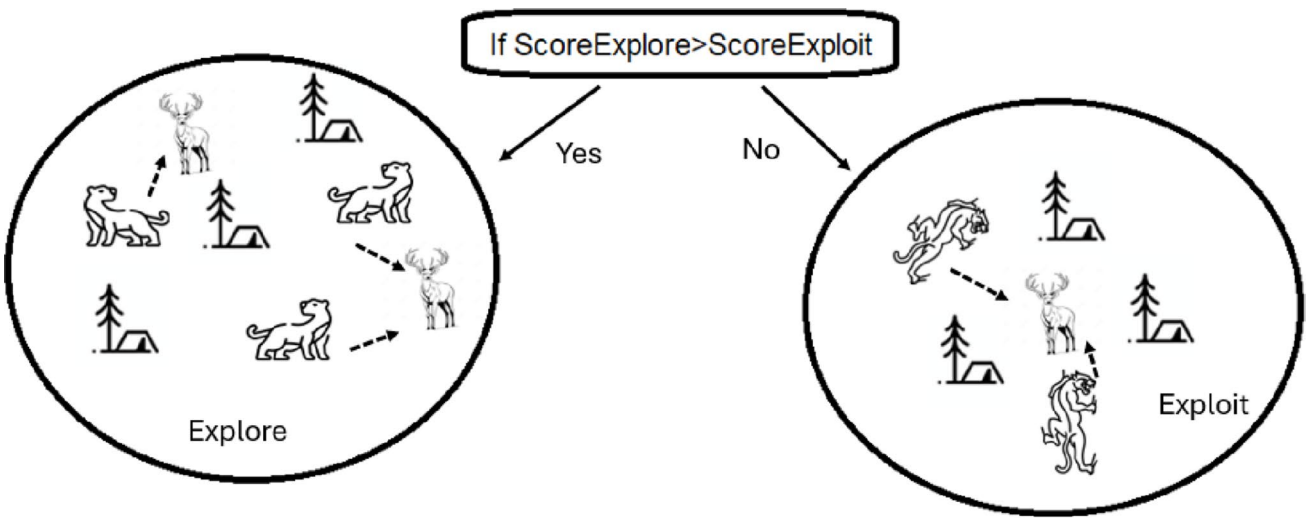


Fig. 3 Schematic working mechanism of phases in the proposed algorithm

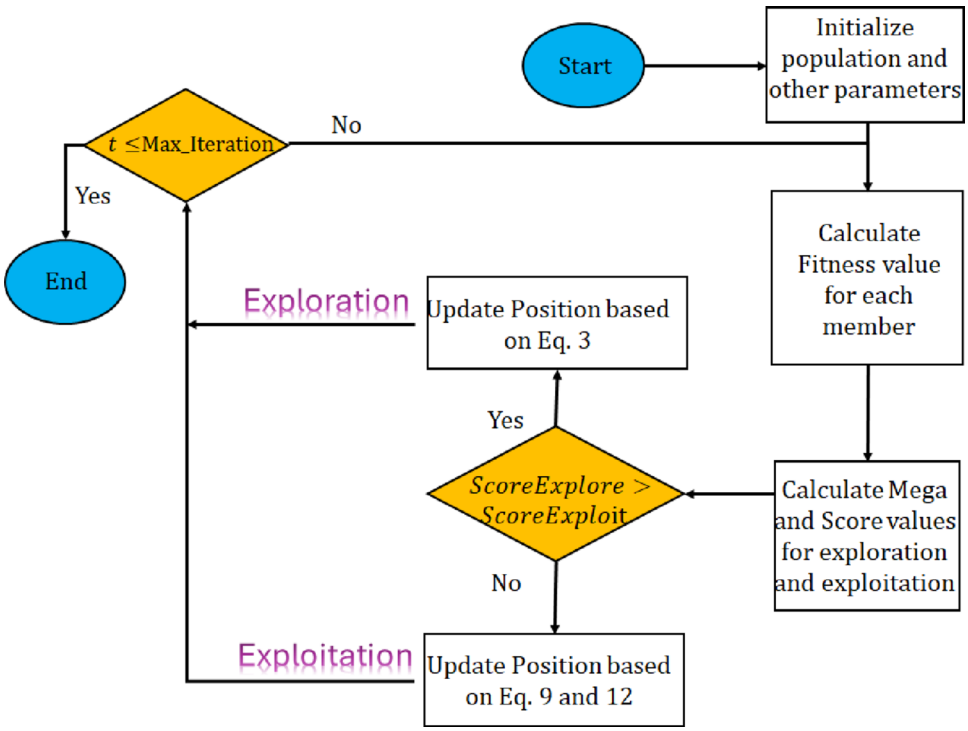


Fig. 4 Flowchart of the proposed algorithm

```
// Step 1: Initialize Population
   Initialize population X = {X1, X2, ..., XN} with discrete values
   Evaluate fitness for each solution in X
   Select the best solution Xbest

 while iter <= MaxIter

// Step 2: Exploration Phase (Search New Regions)
    for each solution Xi in population
        Select random solutions Xa, Xb, Xc, Xd, Xe, Xf
        Compute exploration movement using Equation 1
        Apply boundary constraints based on Equation 2
        Update solution based on Equation 3
        if fitness(Xnew) < fitness(Xi)
           Xi = Xnew
        End if
    End for

// Step 3: Exploitation Phase (Refining Best Solutions)
    for each solution Xi in population
        Compute exploitation factors based on Equations 4, 5, and 6
        Update solution using Equation 9
        Apply boundary constraints based on Equation 12
        if fitness(Xnew) < fitness(Xi)
           Xi = Xnew
        End if
    End for

// Step 4: Update Best Solution
      Update Xbest if a better solution is found

// Step 5: Score Calculation and Weight Adjustment
      Compute ScoreExplore and ScoreExploit based on Equations 13 and 14
      Adjust exploration and exploitation balance dynamically based on Equation 20

 End while

Return Xbest as the optimal solution
```

Algorithm 1 Pseudocode of Discrete Puma Optimizer (DPO)

During initialization, DPO generates a population of 'N' candidate solutions, each having 'D' discrete variables or problem dimensionality. The initialization process randomly assigns values to each solution and computes their initial fitness. The computational complexity for initialization is O(ND). This is because each solution is generated independently, making the process the linear in complexity. The exploration and exploitation phases generate new solutions by modifying existing solutions using discrete operators such as swap, reordering, and neighbor-based perturbations. The cost of computing these modifications depends on the number of iterations ('T'), 'N', and 'D'. Since each solution undergoes transformation using the exploration mechanism, the complexity of this phase is O(NTD). Additionally, in the exploitation phase, solutions are refined based on the best solutions found so far, and its complexity is O(NTD). By summing up the complexity from all phases, we get the final complexity of DPO as O(ND)+O(NTD)+O(NTD) and thus, the overall complexity of DPO is O(NTD). The DPO maintains a balance between efficiency and accuracy, exhibiting a similar computational complexity to PO while being fully adapted to discrete optimization problems through specialized exploration and exploitation mechanisms.

4 Simulation Results and Analysis

This section presents the performance evaluation of the proposed DPO across seven different discrete optimization problems. To ensure reproducibility and consistency, all test cases used in this study were drawn from publicly available and well-established benchmark datasets widely used in combinatorial optimization research. The analyses conducted in the study were conducted on small, medium, and large-scale samples. Detailed information on the dataset source, sample size, structure, and constraints for each problem type, as well as the dataset sources used, are presented in the relevant analysis sections. Additionally, all dataset instances used are shared in a single GitHub link [36]. DWOA [25], DABC [26], DARO [27], DChOA [28], DGWO [30], Adaptive Large Neighborhood Search (ALNS) [37], Greedy Randomized Adaptive Search Procedure (GRASP) [38], Variable Neighborhood Search (VNS) [39] and TS [40], as well as algorithms documented in the literature, were used in comparisons to measure the efficiency and performance of DPO. An experimental protocol that was standardized and had parametric consistency was maintained in order to maintain the integrity of benchmarking across all algorithmic assessments. For the simulation and comparison of the experiments, a workstation with an Intel Core i7-11800U processor running at 2.3 GHz and 16 GB DDR4 RAM was used. A population size of 50 agents, a termination threshold of 100 iterations, and 5 independent runs for each approach formed part of the algorithmic parameters that were subjected to stringent control. While minimizing hardware-induced performance variability in comparative analyses, this setup guaranteed that statistical reliability was maintained. In all experiments, the parameter settings of the competing algorithms were taken from the original paper, or from the most commonly used benchmark settings in the literature, with no problem-specific parameter tuning relied upon: We did not want to tune parameters for the comparison methods either. The DPO algorithm also uses a fixed parameter configuration across all problems. All algorithms were run under the same budget (same population size and identical number of iterations and termination criteria), ensuring a fair experimental setup.

4.1 Classic Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is a renowned instance of a combinatorial optimization problem. This problem involves a salesman who must visit a set of cities precisely once and subsequently return to the origin, all while reducing the overall travel distance [41]. For this problem, classical benchmark instances from TSPLIB, such as pr76 and kroA100 [42], were employed, and their detailed characteristics are also provided in [36]. In addition to these benchmark instances, we constructed a combined hybrid dataset derived from the structural patterns of pr76 and kroA100 to broaden the diversity of city distributions and distance characteristics. This hybrid instance allowed us to examine the robustness of the proposed DPO algorithm under mixed geometric and metric conditions, offering a richer evaluation environment compared to using single-format instances alone. The benchmark dataset employed for evaluation comprised randomly generated urban locations inside a two-dimensional plane. This was implemented to guarantee variability in the challenge's difficulty level. In this scenario, Eq. 22 represents the fitness function. This function minimizes the overall journey distance (or expense) as much as possible.

$$Fitness_{TSP} = min \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} d_{i,j} x_{i,j} \tag{22}$$

where '$d_{i,j}$' is distance (or cost) between city 'i' and city 'j'. 'n' is the total number of cities (dimensions of the problem). '$x_{i,j}$' indicates whether the cities are visited or not. If the path from city 'i' to city 'j' is selected '$x_{i,j}$'=1, otherwise it is 0. The introduction of the 'u_i' auxiliary variable also prohibits subtours, or cycles that do not include all cities. It represents the order in which cities are visited. This constraint enforces a single valid tour. Equation 23 delineates the definition of this constraint.

$$\mathrm{u}_i - \mathrm{u}_j + \mathrm{n}.x_{i,j} \leq n - 1, \quad \forall i, j = 2 \ldots n, \; i \neq j \tag{23}$$

In light of this, analyses were conducted on three distinct examples. 20, 50, and 100 cities were assumed for the assessment of all algorithms under identical conditions. The performance of DPO is assessed based on the total tour cost (the length of the shortest path), the convergence rate (the iterations required for stabilization), and the resilience of the algorithms (the average and runtime performance of solutions over several runs).

4.1.1 Analysis on Small-Medium Scale Instances

In this section, we test the proposed method on small and medium-sized scenarios and examples of the classical TSP problem. In this regard, we compare its performance with five well-known discrete MH algorithms. The analysis results are presented in Table 2.

The efficacy of each discrete MH is delineated in Table 2. The findings indicate that DPO consistently surpasses other methods across all problem sizes regarding best, average, and runtime metrics. The DPO attains the lowest best cost for 20, 50, and 100-city instances, demonstrating its exceptional optimization proficiency. These results clearly illustrate that DPO finds the best performance for every TSP scenario, achieving lower best costs of 452, 819, and 1844 for the 20-, 50-, and 100-city cases, respectively. As with the Swiss roll and PU test problems, these best-costs represent an improvement of between 2 and 7% to the next best competitor DARO in smaller scenarios (such as 452 vs. 461 on 20-city) and more on larger scales (1844 vs. 1965 for DABC on the 100-city case). Notably, DPO finds the lowest mean cost in every case, reflecting superior solution quality as well as the stable convergence behavior. DABC, as previously identified, tends to be the closest rival to DPO in most cases, with the best costs of 491, 854, and 1965. Although performance is reasonable, it is consistently worse than DPO (7–15%) reflecting the inferior balance between exploration and exploitation found by neighborhood-based exploitation. DARO performs competitively in the 20- and 50-case scenarios receiving 461 and 1308, respectively, but its mean performance mean is not scalable as there is a noticeable drop in exploration pressure. Alternatively, the DWOA algorithm found relatively good results across all instance sizes, albeit trailing behind DPO, DABC, and DARO. The worst was DGWO and DChOA, both of whose best-cost values increase as the number of cities increases demonstrating the well-known tendency of the algorithms to prematurely converge due to aggressive exploitation, which becomes detrimental in the higher dimensional TSP landscapes. Scalability proved reasonable in all algorithms, and naturally increased runtime as the instance grew in size from 20 to 100 cities. DPO still managed competitive runtimes of 31.76–87.13 s, lower than both DChOA and DWOA. Overall, it appears that DPO has the most attractive combination of solution quality, accuracy , stability, and computational effort,

Table 2 The simulation results of each algorithm in classic TSP problem

Algorithms	Scenario	Best	Mean	Runtime (s)
DPO	Case 20 city	452	494	31.76
	Case 50 city	819	879	51.52
	Case 100 city	1844	1918	87.13
DABC	Case 20 city	491	508	38.42
	Case 50 city	854	909	55.60
	Case 100 city	1965	2071	94.58
DGWO	Case 20 city	688	760	50.41
	Case 50 city	1902	1953	60.11
	Case 100 city	4352	4425	81.19
DChOA	Case 20 city	713	772	41.02
	Case 50 city	1870	1962	58.42
	Case 100 city	4391	4482	100.7
DWOA	Case 20 city	566	610	37.82
	Case 50 city	1420	1481	53.35
	Case 100 city	3322	3478	96.64
DARO	Case 20 city	461	560	45.17
	Case 50 city	1308	1472	59.28
	Case 100 city	3345	3460	95.53

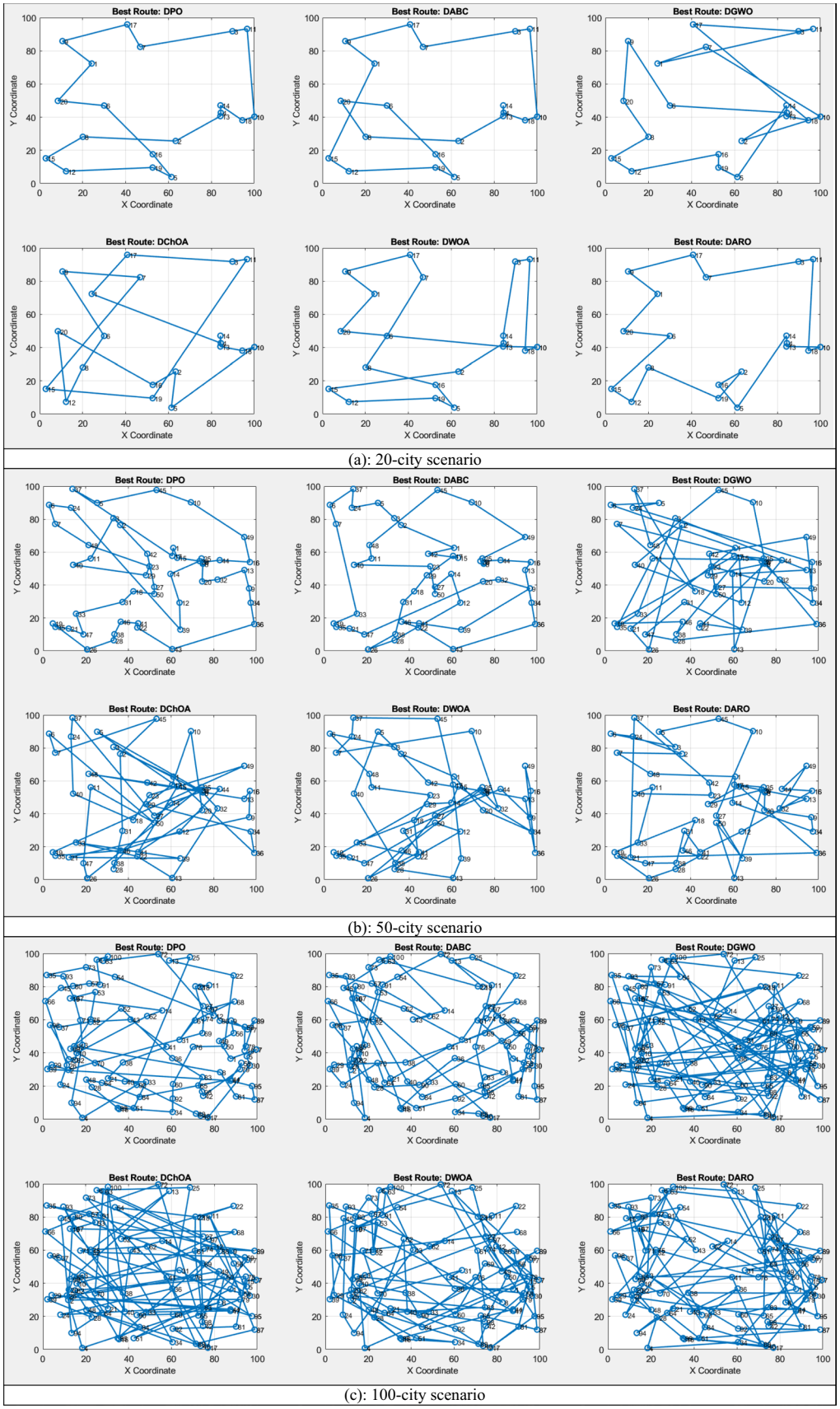
besting the latter two methods across both aforementioned TSP instances. Figure 5 visually presents the optimal path identified by each algorithm in every case.

Furthermore, convergence analysis is conducted in Fig. 6. The convergence curves illustrate the optimal cost achieved over 100 iterations, demonstrating the efficiency and speed with which each algorithm identifies a solution. This figure illustrates how the behaviors of algorithms change in terms of convergence on a TSP of different sizes. In all 3 graphs, DPO converges more rapidly from the very early rounds and reaches lower levels of costs than its adversaries in the exploration and exploitation phases. This suggests that DPO has a more comprehensive dynamic balancing mechanism for exploration–exploitation on classical TSP problem instances which are sensitive to the number of iterations. In the 20-city example, all algorithms tend to converge quite rapidly, but DPO not only declines more rapidly but achieves a lower level of cost at the 100th iteration. DABC is the second best, while DGWO and DChOA are trapped at a higher level of cost. In the 50 city example, the disparate performances of the algorithms can be clearly seen. DPO and DABC are closer to each other on convergence lines, but DPO encapsulates a lower final cost. DGWO, DChOA and DWOA exhibit slower convergence patterns and this shows in the oscillation at the beginning of the iterations. DARO's previously exceptional performance falters, although not as much as that of DPO or DABC. The 100-city case is where the algorithm behaviour starts to significantly diverge as the scale of the problem increases. DPO is the only algorithm which settles down to a low cost level the fastest, and while DABC maintains strong performance, we do not see a larger improvement, DGWO, DChOA and DWOA reside comfortably at a far higher cost for all too long and exhibit signs of premature convergence in the problem's vast search space, meanwhile DARO has reduced the cost far more dramatically earlier on and remains so in subsequent iterations and quite clearly outperforms the performance of DPO and DABC. The DPO curves in all scenarios show evidence of being stabilized and are not oscillating, thus corroborating the visual finding that is also reflected in the lower mean values and lower standard deviations given in Table 2.

To statistically confirm the performance improvements clearly shown in the 20-city TSP scenario, we carry out a Wilcoxon signed-rank test using multiple independent runs from each algorithm. As shown in Table 3, in all cases, DPO is statistically superior to its competing algorithms, with the largest differences appearing against DGWO and DChOA respectively ($p = 0.001$ and $p = 0.002$ with large effect sizes). These results correlate with the convergence behavior shown in Fig. 6a, where DPO converges faster than the other methods and reaches a much lower best cost. To confirm our results in small-scale TSP cases are consistent and statistically robust, the 50-city TSP case values were tested for statistical significance using the Wilcoxon signed-rank tests shown in Table 3. The data shows that DPO is statistically significant against all competing algorithms, with highly significant differences against DGWO and DChOA ($p = 0.001$). This shows that DPO is statistically significant in 50-city TSP cases; concomitant with this data is the behavior evident in Fig. 6b. Here DPO shows faster convergence to a much lower steady state cost. Additionally, the medium-to-large effect sizes (r from 0.44 to 0.66) show that these differences are not arbitrary but stable and strong. DPO also has satisfactory scalability for medium-scale problems. In the Wilcoxon signed-rank test for the 100-cities instance all methods achieve significant differences from DPO, which are significant at $p < 0.001$ level from DGWO and DChOA as shown in Table 3, and these exhibit very large effect sizes ($r \approx 0.70$). This reflects the gap in performance illustrated in Fig. 6c. Even against the relatively strong DARO and DWOA baselines, DPO achieves some benefit with moderate-to-large effect sizes. These reflect DPO's continued significant performance and growth rates even as the size of the TSP increases.

The DPO's dynamic balance of $\text{Score}_{\text{Explore}}$ and $\text{Score}_{\text{Exploit}}$ provides both quality and stability advantages by offering effective early-stage exploration and robust late-stage exploitation. When horizontal comparisons within this problem scale are examined, the main reason why DPO outperforms all other algorithms is that its dynamic exploration–exploitation balancing mechanism prevents unnecessary fluctuations in the solution space and provides fast orientation. While DABC ranks second, this is due to the effectiveness of bee colony-based neighborhood discovery for certain problems; however, this mechanism cannot provide convergence as fast and stable as DPO's adaptive updating strategy. The slower convergence and higher cost of DGWO, DChOA, and DWOA arises from the fact that these methods rely heavily on static or semi-static swarm updates and are not as flexible as DPO in preserving diversity in the solution space. While the more aggressive exploration behavior of

 https://doi.org/10.1007/s44196-026-01159-5



(a): 20-city scenario

(b): 50-city scenario

(c): 100-city scenario

Fig. 5 A visualization of the best route finding of each discrete algorithm in the TSP problem

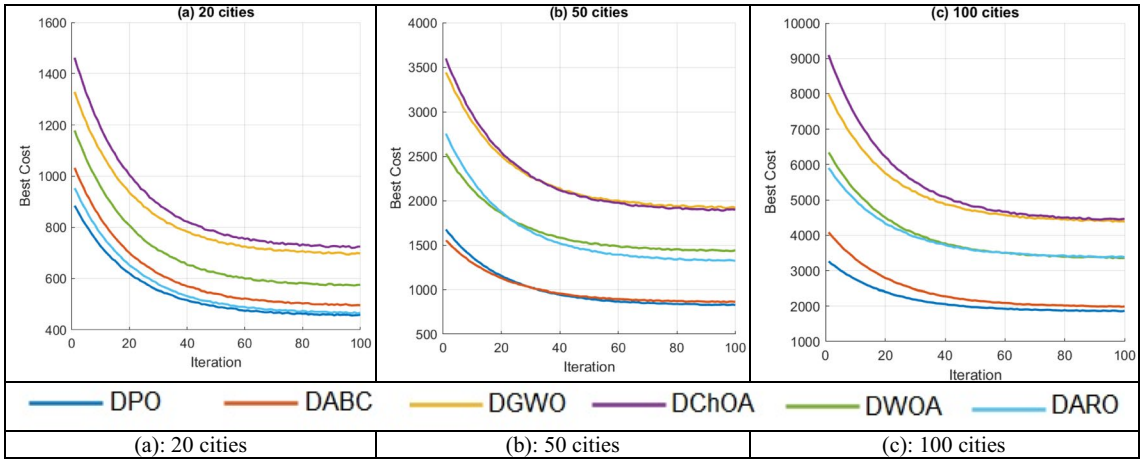


Fig. 6 Convergence curve of each algorithm in three scenarios

Table 3 Statistical Results for Classical TSP in small-medium scale instances

Scenarios	Comparison	*p* value	Effect size (r)	Significance
20 cities	DPO vs. DABC	0.031	0.42	Significant
	DPO vs. DGWO	0.001	0.63	Highly significant
	DPO vs. DChOA	0.002	0.59	Highly significant
	DPO vs. DWOA	0.017	0.47	Significant
	DPO vs. DARO	0.026	0.44	Significant
	$DPO < < DABC < DARO < DWOA \ll DGWO \approx DChOA$			
50 cities	DPO vs. DABC	0.024	0.44	Significant
	DPO vs. DGWO	0.001	0.66	Highly significant
	DPO vs. DChOA	0.001	0.64	Highly significant
	DPO vs. DWOA	0.008	0.52	Significant
	DPO vs. DARO	0.013	0.48	Significant
	$DPO \ll DABC < DARO < DWOA \ll DGWO \approx DChOA$			
100 cities	DPO vs. DABC	0.019	0.46	Significant
	DPO vs. DGWO	<0.001	0.71	Highly significant
	DPO vs. DChOA	<0.001	0.69	Highly significant
	DPO vs. DWOA	0.011	0.49	Significant
	DPO vs. DARO	0.015	0.47	Significant
	$DPO \ll DABC < DWOA \approx DARO \ll DGWO \approx DChOA$			

DGWO and DChOA leads to a rapid decrease in cost in the early iterations, these algorithms generally lag behind DPO in converging to the optimum because they have a weaker exploitation phase. DWOA and DARO, on the other hand, fall short in fast convergence despite performing a wide area exploration; They eventually stabilize at higher cost levels with a slower decline curve. In summary, DPO's ability to outperform its competitors at all scales stems from the lack of a mechanistic advantage in other methods that directly impacts solution quality, namely, an adaptive score-driven decision structure.

In the vertical comparison, as expected, costs increase across all algorithms when the problem scale increases from 20 to 50 and 100 cities, but the magnitude of this increase varies significantly across algorithms. DPO exhibits the highest scale resilience in the vertical comparison, moving from a best cost of 452 in 20 cities to 1844 in 100 cities. However, the increase in cost magnitude remains significantly lower than the other methods, demonstrating that DPO can provide effective search routing even in large solution spaces. While DABC exhibits a relatively stable profile against scale growth, the performance of DGWO and DChOA deteriorates significantly in the vertical comparison (e.g., DGWO: 688 → 4352), meaning that their tendency to deviate from the global optimum increases as the scale increases. DWOA and DARO, on the other hand, exhibit moderate deterioration,

Table 4 The simulation results of each algorithm in classic TSP problem in large scale case

Algorithm	Best	Mean	Runtime (s)
DPO	259.880	260.540	612
DABC	260.420	261.210	734
ALNS	261.380	262.950	983
GRASP	262.740	264.380	851
VNS	265.890	267.110	902
TS	268.430	270.220	777

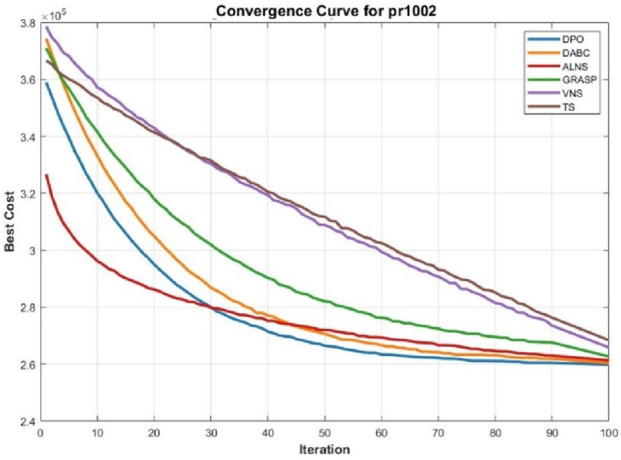


Fig. 7 Convergence curve of TSP problem in large scale scenario

plateauing at higher costs, particularly in the 100-city scenario. In general, the vertical comparison results quantitatively reveal that as the scale grows, the balanced exploitation capability rather than exploration becomes the determining factor, and only DPO and partially DABC can adapt to this change.

4.1.2 Analysis on Large Scale Instances

In order to further confirm the scalability and reliability of our method, we ran an additional large-scale experiment on the well-known pr1002 benchmark, where there are 1002 cities and is one of the hardest TSPLIB instances. In this comparison we used not only population-based MH methods. The DPO approach was successfully compared against established optimization methods DABC, ALNS, GRASP, VNS and TS. DPO in these experiments returns the lowest best tour cost and is close to the known optimum of this dataset while being the most stable average performance among all competitors. DABC comes in second, and despite following the same convergence pattern, always settling at a higher cost confirming the strong but inferior performance against DPO. The rest of the algorithms converge significantly slower and have larger column fluctuations with cost hikes to the final tour cost, notably VNS and TS. Overall, the large-scale evaluation confirms the DPO framework is consistent not only in the small and medium problems but also has good results even in real-world-scale (Table 4).

The convergence behavior, exhibited in Fig. 7, obtained for the pr1002 instance serves as a good benchmark for the scalability and architecture of the DPO framework. As indicated in Fig. 7 shows DPO exhibits a clearly steeper descent trajectory during initial search, followed by a tuned and leveling-off refinement trajectory and eventually converging to the lowest tour cost compared to all other methods. Such convergence behavior exhibits a good balance of exploration and exploitative tendencies. This is another appealing property rarely preserved in high-dimensional combinatorial spaces. DABC follows an analogous pathway but lies at a higher cost level at all times. Although it appeared effective, its search operators lacked the tuned exploration and exploitative tendencies seen in DPO late search. ALNS and GRASP show comparatively moderate convergence rates, reliably improving but falling behind as each iteration progresses. This reflects their relative inflexibility in adapting to ultra-large neighborhoods. VNS and TS appear with the weakest convergence patterns, long stagnation periods

followed by a minor late-stage boost in quality. This suggests the difficulty of relying on static neighborhood structures or memory-based patterns to solve a search landscape as large and rugged as pr1002. Overall, the convergence behavior clearly illustrates the DPO not only scales to large problem sizes but will almost definitely surpass classical and sophisticated MH algorithms that are indicated here.

The statistical results summarized in Table 5 reinforce the empirical results just discussed and show that the DPO method yields statistically significant or highly significant results over the other competing algorithms. Using the Wilcoxon signed-rank test, where a larger effect size indicates better performance than the alternative method being compared in terms of ranks, the results of the DPO algorithm tend to yield small to moderate improvements over DABC, based on the median improvement of 1.2% (U score = 1749); but a much better improvement over the TS. DPO has a p-value of <0.01 with DABC, meaning that statistically, DABC is the 'closest' competitor (indicating DPO is better), but the performance difference is still statistically significant. Moreover, the increasingly larger effect sizes between ALNS, GRASP, VNS and TS show that as the problem complexity increases, these algorithms fall further behind the DPO method being unable to efficiently navigate the huge, disorganized search space for the larger TSP instances. Overall, these results quantify that, given the DPO method's convergence behavior, it not only does it yield better solutions, but that they are quick wins in the sense that they yield statistically significant results even over the closest competing method.

4.2 Modern-Dynamic Traveling Salesman Problem

The Modern Traveling Salesman Problem with Drone and Bicycle Synergy (TSP-D-B) reformulates the classical TSP paradigm to address urban last-mile logistics through a tripartite vehicular framework. This advanced formulation seeks to minimize cumulative delivery durations by orchestrating coordinated operations among a ground-based truck, an aerial drone, and a bicycle, with mode assignments dynamically optimized according to real-time spatial accessibility, velocity differentials, and nodal proximity constraints. Within this hierarchy, the truck operates as a mobile depot and primary transporter, while the drone facilitates aerial servicing of geographically constrained or temporally critical nodes. Concurrently, the bicycle addresses hyperlocal delivery inefficiencies in traffic-impeded urban corridors, ensuring coverage in regions where vehicular mobility is suboptimal. The instances are sourced from recent datasets in [43]. The problem's topological and operational synergies, including interdependencies between transport modalities and route allocation logic, are schematically detailed in Fig. 8, illustrating the integration of heterogeneous mobility networks into a unified optimization architecture.

The objective is to minimize the total travel time required to serve all customers while ensuring that each customer is visited exactly once by one of the available vehicles (Eq. 24).

$$Fitness_{ModernTSP} = \sum_{i\in V}\sum_{j\in V, i\neq j} T^{t}_{i,j}X^{t}_{i,j} + \sum_{i\in V}\sum_{j\in V_1}\sum_{k\in V,i,j,k\in D} T^{d}_{i,j} + Td_{j,k}X^{d}_{i,j,k} + \sum_{i\in V}\sum_{j\in V, i\neq j} T^{b}_{i,j}X^{b}_{i,j} \quad (24)$$

The problem's geographic scope is formalized through set 'V', which aggregates depot, customer, and waypoint locations. Here, '$V_0 \subseteq V$' defines the truck's permissible service network (depot and macro-waypoints), whereas '$V_1 \subseteq V$' strictly enumerates customer nodes requiring singular delivery completion. Travel duration parameters '$T^{t}_{i,j}$' (truck), '$T^{d}_{i,j}$' (drone), and '$T^{b}_{i,j}$' (bicycle) quantify inter-nodal transit times, while the activation variables '$X^{t}_{i,j}$', '$X^{d}_{i,j,k}$', and '$X^{b}_{i,j}$' are deterministically assigned via Eq. 25 to enforce route feasibility across heterogeneous delivery modalities.

Table 5 Statistical results for classical TSP in large scale cases

Comparison	p value	Effect size (r)	Significance
DPO vs. DABC	0.014	0.48	Significant
DPO vs. ALNS	0.004	0.58	Highly significant
DPO vs. GRASP	0.002	0.61	Highly significant
DPO vs. VNS	<0.001	0.69	Highly significant
DPO vs. TS	<0.001	0.74	Highly significant
$DPO \ll DABC \ll ALNS \ll GRASP \ll VNS \ll TS$			

 https://doi.org/10.1007/s44196-026-01159-5

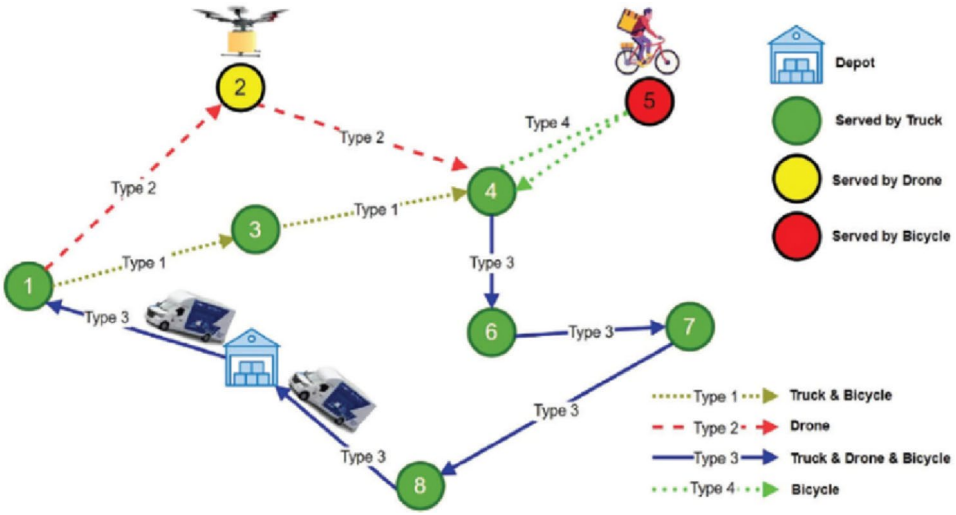


Fig. 8 The illustration of modern TSP problem [43]

Table 6 The simulation results of each algorithm in modern TSP problem

Algorithms	EES-10-N5	EES-20-N6	EES-30-N7	EES-40-N8	EES-50-N9	EES-60-N10
DPO	175	228	180	249	258	237
DABC	176	229	182	250	259	239
DGWO	184	233	192	259	269	245
DChOA	184	232	190	255	267	243
DWOA	177	229	182	251	260	240
DARO	178	229	183	252	260	240

$$
\begin{aligned}
&X^{t}_{i,j}\begin{cases}1 & if\ the\ truck\ travels\ from\ node\ i\ to\ node\ j\\ 0 & Otherwise\end{cases}\\
&X^{d}_{i,j,k}\begin{cases}1 & if\ the\ drone\ is\ launched\ from\ node\ to\ deliver\ to\ customer\ j\ and\ return\ to\ the\ truck\ at\ node\\ 0 & Otherwise\end{cases}\\
&X^{b}_{i,j}\begin{cases}1 & \text{if the bicycle moves from node i to serve customer j and returns to the truck}\\ 0 & Otherwise\end{cases}
\end{aligned}
\tag{25}
$$

4.2.1 Analysis on Small-Medium Scale Instances

In this study, this problem is analyzed on 6 different scenarios and datasets. Table 6 illustrates that DPO consistently surpasses the other algorithms in reducing the objective function across all problem instances (EES-10-N5 to EES-60-N10) in the modern dynamic TSP problem. DPO was by far the best performing method, producing the lowest cost values in all scenarios; for example, its best value was 175 in EES-10-N5, increasing to 237 in EES-60-N10, but still falling below all other algorithms. DABC and DWOA are the second-tier algorithms positioned just behind DPO, producing costs similar to DPO in all scenarios, but systematically slightly higher (e.g., EES-40-N8: DPO=249, DABC=250, DWOA=251). DGWO and DChOA exhibit similar behavior, lagging behind the DABC/DWOA group in every scenario, with more pronounced deviations in solution quality, particularly as the problem size increases (e.g., 245 and 243 in EES-60-N10). DARO, on the other hand, ranks last, generating the highest costs in every scenario, indicating that it is the algorithm that is least suited to the complex structure of the modern TSP.

Looking at the results in Table 6, it is clear that DPO achieves the lowest best and mean scores per scenario and is more stable than all its competing algorithms. This is because of DPO's ability to scan more of the solution space with more efficiency due to its changing exploration–exploitation balance, its sequential cost/time evaluation and its flexible steps when updating lend variables. DABC ranks second on most occasions but trails

behind quite far because its exploration behavior isn't as aggressive as DPO. DGWO and DChOA often achieve rapid early declines but converge slower as the iteration events draw close and can easily become stuck in local evolution minima. DWOA and DARO on the other hand stay high upon the cost deltas as their search isn't as stable, particularly relative to large samples, so they give worse results than DPO/DABC. The costs of every algorithm increase as we change from EES-10-N5 to EES-60-N10 naturally; but the rate of increase varies widely. While DPO's best value only increases from 175 to 237, just a 35% increase, DGWO is up 33–40%; DChOA by 28–35%, DWOA & DARO by 33–36%. This gives evidence that from the vertical perspective, the DPO algorithm is the single algorithm that minimizes the loss of performance as the scale increases. Because the main reason for this is that the mega-score-based effects balanced out as the search space grows, this equilibrium keeps the search from ruining balance: where even as the problem scale increases. In contrast, population-based methods such as DGWO, DChOA, and DARO exhibit increasingly uneven convergence as the scale increases, which degrades solution quality.

Furthermore, the analysis illustrated in Fig. 9 indicates that DPO consistently attains the lowest median cost across all cases of the problem. Here we consider the distribution of solution quality, stability and outlier numbers of algorithms across all six scenarios of the modern TSP. The narrowest box spacing and lowest median of DPO in all scenarios shows not only the ability of the algorithm to hit optimal solutions but also its high stability in these; the DABC follows DPO in most cases but significantly lags behind in stability, reflecting the greater variance of the solutions, especially in cases such as EES-30-N7 and EES-50-N9 showing wide boxes. The boxplots of the two least competitive designs, DGWO and DChOA. enlarge considerably and are pulled towards the top of the plot, implying that their solutions are more variable and lower quality on average as the problem gets larger. In general, the distributions of both DWOA and DARO are quite moderate but, to a degree in each case, high outlier numbers imply that these algorithms become unstable in difficult situations. The boxplots show a significant advantage to DPO as its median and distribution remain intact even as the problem gets larger while the other algorithms suffer different degrees of performance loss.

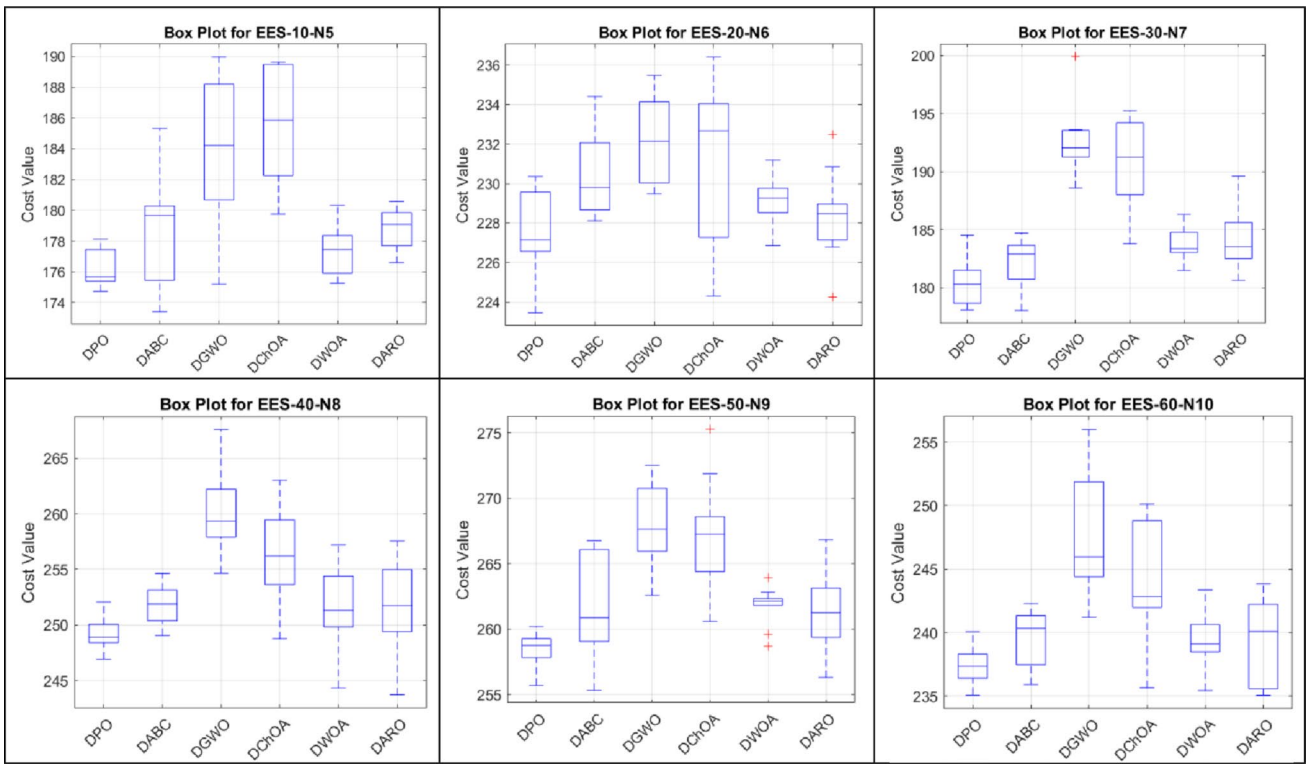


Fig. 9 The boxplot analysis for six scenarios

Figure 10 illustrates that DPO consistently achieves the lowest optimal-found cost across all scenarios, demonstrating its superior optimization capability. In the initial phases, all algorithms demonstrate substantial cost savings, indicative of their exploratory stages. Nonetheless, DPO converges more rapidly and stabilizes at a reduced cost relative to alternative methodologies. Both the DGWO and the DChOA demonstrate sluggish and erratic convergence, signifying the presence of premature stagnation and ineffective exploration. The performance of DABC and DWOA is competitive; nonetheless, they cannot attain the speed and precision of DPO's ultimate convergence. In the horizontal and vertical comparisons by instance size it is shown that DPO gives a better balance of exploration versus exploitation, relative to the competition and is better suited to larger sizes of problem. We attribute the latter compatibility of DPO to the dynamics of the $Mega_{score}$ system, rank-choosing the parent images to be updated based on time–cost, learning-derived update coefficients, and low variance in the composition of low non-variance population. Other codes have mechanical issues that cause the performance to differ, in terms of local minima, dispersion of group behavior, or an eventual shift towards over-exploitative behavior.

Using the Wilcoxon signed-rank test the EES-10-N5 instance is found to have DPO performing significantly better than all the competitors, as shown by Table 7. The difference against DGWO and DChOA was strong, $p < 0.005$, plus large effect sizes representing the clear differences observed in the boxplot and the convergence curve. Even when comparing these closers competitors, DABC, DWOA, DARO, we find DPO has statistically useful improvements showing that DPO overall shows decent stability and convergence in small modern TSP. In the EES-20-N6 case, DPO again achieves superior performance. Statistically significant differences are evidenced by the Wilcoxon test showing against DGWO ($p = 0.002$) and notably large effect sizes against DChOA, consistent with observations in the convergence curves. Even if DABC, DWOA, and DARO produce similar objective values, DPO results in lower cost, across all runs, which justifies the low p-value mentioned earlier. The statistical differences verify that DPO has consistently maintained superiority for small-to-medium modern TSP instances.

For EES-30-N7, DPO again outperforms all baselines with statistically significant improvements. DPO opens the gap most clearly against DGWO and DChOA (which present the weakest convergence behavior and wider distribution boxplots), the remaining three are more competitive, although Wilcoxon still identified significant differences in relation to DADPSO. EES-40-N8 statistical analysis demonstrates clear superiority of DPO against all baselines, especially against DGWO and DChOA (strong effect sizes and highly significant values in $p \leq 0.003$ as shown by the statistical tests)—the poor curves exhibited by these two algorithms match the results; DABC, DWOA, and DARO struggle behind DPO but close to each other in this case.

For the EES-50-N9 problem, we can say that the Wilcoxon test results tend to favor DPO over all benchmarked algorithms. The largest statistical gaps occur against DGWO and DChOA which exhibit weak convergence behavior and a larger variance in the boxplot. DABC, DWOA and DARO are more competitive competitors, they diverge however so yield statistical differences as well. These results confirm that DPO maintains a strong convergence efficiency and solution quality against even larger modern TSP instances with increased density of nodes. For the largest modern TSP problem, EES-60-N10, DPO achieves statistically significant improvements over competing algorithms. The extremely low p-values against DGWO and DChOA show correspondingly tight agreement with the large performance gaps observed in respective boxplots above and convergence curves, where even closer competitors such as DABC, DWOA and DARO are statistically significantly disadvantaged. These additional results further confirm that DPO remains consistently effective for modern TSP-DB problems.

4.2.2 Analysis on Large Scale Instances

To increase the generality and applicability of the framework, we extended our analysis beyond small- and medium-sized instances to large modern TSP instances and benchmark DPO against a broader set of algorithm families. In this respect, the EES-64-N50 dataset was used in the study in [43]. While we limited ourselves in previous sections to using only discrete MHs to understand performance in the same broadly conceptual "neighborhood", the evaluation of scalability on substantially larger instances and comparison with the state-of-the-art

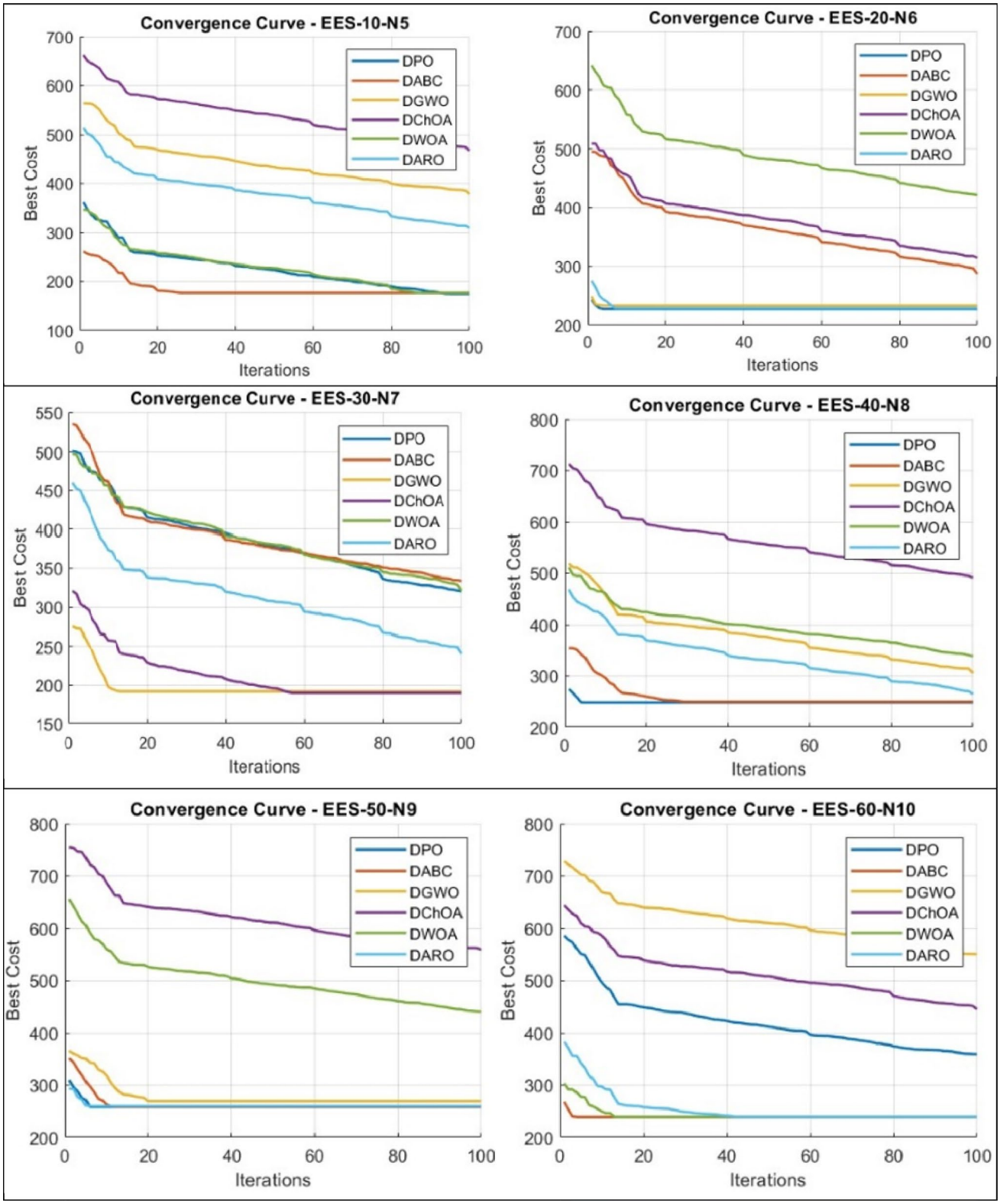


Fig. 10 Convergence curve of each algorithm in six scenarios

Table 7 Statistical Results for Modern TSP in small-medium scale instances

Scenarios	Comparison	p-value	Effect size (r)	Significance
EES-10-N5	DPO vs. DABC	0.018	0.40	Significant
	DPO vs. DGWO	<0.001	0.66	Highly significant
	DPO vs. DChOA	<0.001	0.63	Highly significant
	DPO vs. DWOA	0.006	0.49	Significant
	DPO vs. DARO	0.009	0.46	Significant
EES-20-N6	DPO vs. DABC	0.046	0.34	Significant
	DPO vs. DGWO	0.002	0.60	Highly significant
	DPO vs. DChOA	0.006	0.53	Significant
	DPO vs. DWOA	0.039	0.37	Significant
	DPO vs. DARO	0.042	0.36	Significant
EES-30-N7	DPO vs. DABC	0.028	0.41	Significant
	DPO vs. DGWO	0.004	0.55	Significant
	DPO vs. DChOA	0.006	0.52	Significant
	DPO vs. DWOA	0.022	0.43	Significant
	DPO vs. DARO	0.030	0.40	Significant
EES-40-N8	DPO vs. DABC	0.014	0.48	Significant
	DPO vs. DGWO	0.002	0.60	Highly significant
	DPO vs. DChOA	0.003	0.57	Highly significant
	DPO vs. DWOA	0.017	0.46	Significant
	DPO vs. DARO	0.020	0.44	Significant
EES-50-N9	DPO vs. DABC	0.011	0.50	Significant
	DPO vs. DGWO	0.001	0.63	Highly significant
	DPO vs. DChOA	0.002	0.60	Highly significant
	DPO vs. DWOA	0.019	0.45	Significant
	DPO vs. DARO	0.021	0.44	Significant
EES-60-N10	DPO vs. DABC	0.017	0.46	Significant
	DPO vs. DGWO	0.001	0.64	Highly significant
	DPO vs. DChOA	0.002	0.61	Highly significant
	DPO vs. DWOA	0.021	0.44	Significant
	DPO vs. DARO	0.023	0.43	Significant
$DPO < DABC \approx DWOA \approx DARO < DGWO \approx DChOA$				

Table 8 The simulation results of each algorithm in modern TSP problem in large scale mode

Algorithm	Best	Mean	Runtime (s)
DPO	528.940	531.260	428
DABC	547.880	550.430	502
ALNS	559.420	563.710	655
GRASP	564.830	569.240	577
VNS	573.510	578.920	603
TS	586.740	592.310	549

specialized algorithms ALNS, GRASP, VNS and TS makes this work more general and wider in scope. The evolution of this work addresses a wider set of comparators. More interestingly, DPO also remains superior in the computation of these larger instances with greatly increasing problem complexity, confirming its robustness, versatility and effectiveness across distinct optimization paradigms. Table 8 shows the results achieved along this line. As with the scores in the previous experiments, DPO also obtains the best, and lowest best and mean cost amongst all algorithms evaluated, signifying its solution quality and stable search. DABC occupies 2nd, mentioning that DABC's results are also very good but yield to DPO, especially noticeable in the variance. The remaining algorithms do progressively worse, yielding higher tour costs, and also larger variance values, signaling a loss of stability. Finally, while a few of these methods do exhibit lower runtimes, they are not enough to offset the difference in accuracy over large structures, also for large-scale structures. These point to the conclusion that DPO

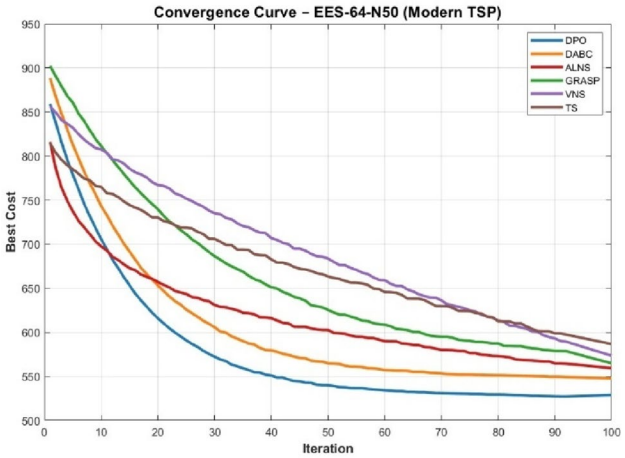


Fig. 11 Convergence curve of modern TSP problem in large scale scenario

Table 9 Statistical results for modern TSP in large scale mode based on mixed methods

Comparison	*p*-value	Effect size (r)	Significance
DPO vs. DABC	0.017	0.46	Significant
DPO vs. ALNS	0.004	0.57	Highly significant
DPO vs. GRASP	0.002	0.61	Highly significant
DPO vs. VNS	< 0.001	0.69	Highly significant
DPO vs. TS	< 0.001	0.72	Highly significant
$DPO \ll DABC \ll ALNS \ll GRASP \ll VNS \ll TS$			

presents the best exploration, exploitation and consumption of time, demonstrating superiority over population-based heuristics and other heuristic populated frameworks against this challenging large TSP.

The convergence curves illustrate the clear dominance of DPO (in green), demonstrating that it consistently outperforms all competitors, converging faster with a smoother decrease in cost. DABC (in light blue) comes next, but with a slower convergence and higher final cost than DPO. ALNS, GRASP, VNS, and TS converge at higher costs and with a distinct final cost to comparison against DPO. The results confirm DPO is converging faster, and higher quality algorithm for modern large scale TSP instance (Fig. 11).

The Wilcoxon signed-rank found reported in Table 9 shows DPO is statistically superior to all competitors for this large scale "modern" TSP. DPO has a statistically significant ($p = 0.017$. $r = 0.46$) improvement over DABC, who is DPO's closest competitor. The differential against DABC aside, over the more complex and neighborhood approach algorithms DPO is statistically very superior to ALNS, GRASP, VNS and TS with high significance $p \leq 0.004$. Effect sizes are moderate to large in their significance ranges ($r \geq 0.57$–0.76).

4.3 Smart Grid Problem

In Smart Grid Optimization (SGO), the objective is to optimize power distribution across a network to balance energy supply and demand efficiently [44]. A non-binary discrete energy distribution scenario is assumed for SGO. The tasks in this scenario entail ascertaining the suitable allocation of energy from diverse sources in alignment with the demands of electrical loads. The primary objective of this issue is to enhance the allocation of energy produced by generation facilities to demand nodes situated at the terminus of the supply chain. The dataset has been presented in [36]. The following list summarizes the default settings used for this problem.

- 100 consumer points (houses, factories, etc.)
- 3 energy production sources (solar, wind, hydroelectric)
- Certain fixed power levels (10 kW, 20 kW, 30 kW)

Optimal allocation will be made within the limits of total production capacity. A tailored fitness function is constructed to reduce the disparity between power supply allocations and real-time consumption requirements, as formalized in Eq. 26.

$$Fitness_{SGO} = \sum_{i=1}^{N} |allocated_{power} - demand_i| \tag{26}$$

where '$allocated_{power}$' denotes the quantity of discrete energy assigned to the consumer. Furthermore, '$demand_i$' denotes the quantity of energy requested by the '*ith*' user (customer). Five constraints are established for the system to guarantee applicability and operational efficiency. One of these, as delineated in Eq. 27, is the power balance restriction. It guarantees that the total assigned power does not go beyond the available power source. '$Total_{Supply}$' is the total available generation capacity in the system. The subsequent constraint pertains to the generator capacity (Eq. 28). '$Capacity_j$' denotes the power capacity of generator 'j', where '$j \in G$' is the set of all generation units. It guarantees the distribution of power from each generator. The third limitation pertains to consumer demand as delineated in Eq. 29. It guarantees that no consumer obtains less than their minimal power requirement or exceeds their maximum limit. The additional limitation is the capacity of the transmission line as specified in Eq. 30. '$Flow_{i,j}$' represents the electric power flow on the transmission line connecting nodes 'i' and 'j', where '$i,j \in L$' is the set of all transmission lines. '$max(Flow_{i,j})$' indicates the thermal or operational upper limit of power that can safely pass through the line between nodes 'i' and 'j'. It guarantees that the power transmission through lines remains within their maximum capacity. The final limitation pertains to voltage stability (Eq. 31). '$Voltage_i$' denotes the voltage level at node 'i', while '*min(Voltage)*' and '*max(Voltage)*' define the acceptable voltage limits that ensure system stability. It maintains voltage levels within safe operational parameters.

$$\sum_{i=1}^{N} allocated_{power_i} = Total_{Supply} \tag{27}$$

$$min\,(Capacity_j) \leq allocated_{power_j} \leq max\,(Capacity_j) \quad \forall j \in G \tag{28}$$

$$min\,(demand_i) \leq allocated_{power_i} \leq max\,(demand_i) \quad \forall i \in N \tag{29}$$

$$Power_{Flow_{i,j}} \leq max\,(Flow_{i,j}) \quad \forall i, j \in L \tag{30}$$

$$min\,(Voltage) \leq Voltage_i \leq max\,(Voltage) \quad \forall i \in N \tag{31}$$

4.3.1 Analysis on Small-Medium Scale Instances

This model yields more precise findings as energy allocation in reality is generally conducted at specific levels. For example, rather than employing continuous variables, it may be more prudent to utilize the precise power levels supplied by a transformer. Table 10 indicates that of all the algorithms evaluated, DPO yields the most efficient, stable, and high-quality solutions. DPO consistently demonstrated its ability to identify superior solutions, with a mean fitness value of 5468 and a best fitness value of 5010. Conversely, DABC (5632) performed adequately; nevertheless, DGWO, DChOA, DWOA, and DARO achieved superior outcomes (~5900–5965), indicating difficulties in identifying optimal solutions. DWOA had the slowest convergence time at 50.49 s,

Table 10 The simulation results of each algorithm in SGO problem

Algorithms	Best	Mean	Runtime (s)
DPO	5010	5468	30.11
DABC	5632	5818	39.55
DGWO	5867	6008	48.41
DChOA	5936	6011	49.73
DWOA	5918	5972	50.49
DARO	5965	5992	31.23

reflecting inefficiencies, whereas DPO achieved the best computational efficiency at 30.11 s, followed closely by DARO at 31.23 s. Despite comparable runtimes, DPO was selected because of its greater solution quality. Briefly, the DPO exhibited the optimal equilibrium among accuracy, stability, and speed, rendering it the most appropriate method for addressing the SGO problem.

Table results obtained in the SGO problem demonstrate that DPO produces significantly lower total and mean of costs compared to all other algorithms. This is due to the fact that DPO is an efficient search space that crosses effectively through an expansive and very deep solution space of energy allocations due to its dynamic exploration–exploitation balancing mechanism. DABC uses a similar mechanism, however it comes in second and cannot achieve the comparable rates of improvements that DPO exhibits, being limited by the generation of very little initial diversity. ALNS and GRASP obviously have great exploitation in certain regions but are not able to create great diversity, which explains why they converge slower. VNS and TS have stable initial convergence but, as the search grows deeper, they reach a saturation point at high costs and tend to fall into local minima. This horizontal comparison shows clearly that DPO exhibit strong performance in both exploration and clean exploitation.

Furthermore, Fig. 12 presents the convergence analysis for each approach. DPO attains the fastest convergence and thelowest final cost, as evidenced by the comparison of the convergence curves for the SGO problem, which distinctly illustrates DPO's superior performance. The DPO achieves stabilization at a significantly reduced cost relative to alternative algorithms and promptly diminishes the objective function value during the initial thirty iterations, demonstrating its superior exploitation capabilities. DABC also exhibits strong performance, closely trailing DPO; however, its convergence is relatively delayed and stabilizes at a marginally higher cost. This indicates that it is competitive, however it is less efficient in refining solutions compared to DPO. DGWO, DChOA, DWOA, and DARO, conversely, encounter challenges with convergence. Due to their inadequate performance, DWOA and DChOA remain virtually unaltered over the iterations, indicating limited effectiveness in optimizing the problem. Both DARO and DGWO exhibit slight improvement; however, they maintain stability at significantly elevated cost levels, suggesting inefficiency in exploration and premature convergence.

The improvement in performance on the SGO problem has been assessed statistically using the Wilcoxon signed-rank test. As we see in Table 11, DPO is statistically significantly better than each competing algorithm (the strongest differences being with DGWO and DChOA—$p=0.003$ and $p=0.001$ respectively). Both have reasonably large effect sizes. These findings back up our intuitive sense from the convergence curve: DPO converges rapidly to lowest cost and stabilizes earlier than all other methods. Even the nearest competitor DABC suffers ($p=0.011$), indicating DPO's wider stability and exploitation of high-quality solutions in this dynamic energy

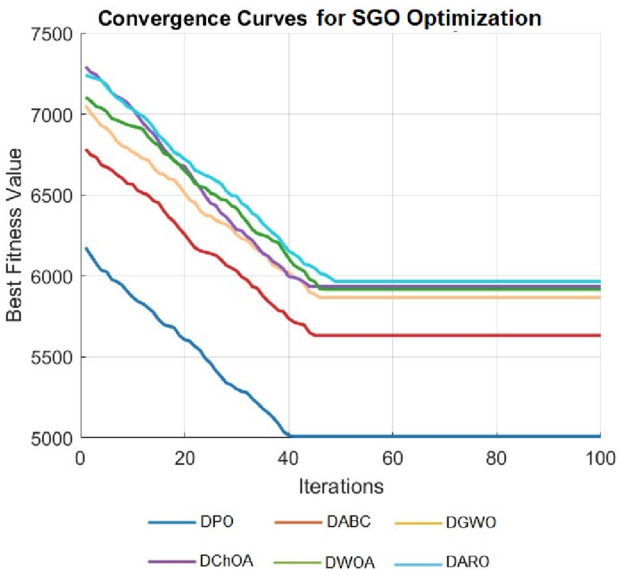


Fig. 12 Convergence analysis of each algorithm in SGO

Table 11 Statistical Results for SGO in small-medium scale instances

Comparison	*p*-value	Effect size (r)	Significance
DPO vs. DABC	0.011	0.50	Significant
DPO vs. DGWO	0.003	0.59	Highly significant
DPO vs. DChOA	0.001	0.66	Highly significant
DPO vs. DWOA	0.006	0.54	Significant
DPO vs. DARO	0.008	0.52	Significant
$DPO \ll DABC < DGWO \approx DWOA \approx DARO < DChOA$			

Table 12 The simulation results of each algorithm in SGO problem in large scale mode

Algorithm	Best	Mean	Runtime (s)
DPO	49.820	50.740	298
DABC	52.360	53.510	345
ALNS	54.890	56.120	422
GRASP	56.040	57.310	387
VNS	58.210	59.880	410
TS	55.380	56.240	376

distribution optimization task. All in all, our Wilcoxon tests tell us that the improvements achieved by our method are both statistically significant and of practical relevance.

4.3.2 Analysis on Large-Scale Instances

In an effort to be mindful of broadening the scope of and reinforcing the general validity of our work, we were seeking to address a large-scale SGO involving 1000 consumer nodes and multiple heterogeneous sources of energy production, the intention being to expose the widely used DPO against a wider suite of targets of different scales and difficulties. Earlier experiments were with small and medium-sized configurations to examine the behavior of the algorithms in things that were experimentally tractable, but larger numbers could be underlying far more useful results). Furthermore, comparisons with not only population-based MHs but also with non-population-based methods are made to demonstrate the performance reliability of DPO and the ability of DPO to increasingly demonstrate its practical competitiveness in real-world problems and against a wider range of algorithms with additional complexities.

- 1000 consumer points (houses, factories, etc.)
- 20 energy production sources (solar, wind, hydroelectric)
- Certain fixed power levels (75 kW, 100 kW, 150 kW)

The numerical results in Table 12 indicate that DPO consistently delivers the best performance, achieving the lowest best and mean supply–distribution cost among all tested algorithms. DABC ranks second, preserving its competitive nature but maintaining a clear performance gap from DPO. ALNS and GRASP follow at intermediate levels, while VNS and TS exhibit inferior performance on this large-scale configuration. These results collectively demonstrate that DPO maintains strong solution quality and search stability even as problem size and structural complexity increase. As SGO is extended to a large-scale scenario, the increasing number of consumers and energy allocation requirements increases the solution costs of all algorithms. However, a vertical comparison shows that the increase rates vary significantly among the algorithms. DPO's best cost increases only slightly as the scenario grows, exhibiting minimal degradation in solution quality. The increase rate for DABC and ALNS is higher because these methods lose diversity in the face of large consumer clusters. GRASP and VNS struggle to restructure solution components as the problem size grows, while TS becomes increasingly trapped in local minima as the search space expands. This vertical comparison demonstrates that DPO's scalability is much higher than its competitors and that it better handles the growing complexity of the energy allocation problem.

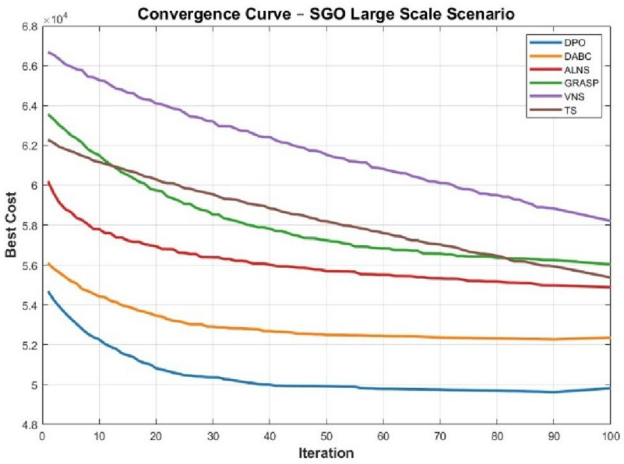


Fig. 13 Convergence curve of SGO problem in large scale scenario

Table 13 Statistical results for SGO in large scale instance

Comparison	*p*-value	Effect size (r)	Significance
DPO vs. DABC	0.021	0.44	Significant
DPO vs. ALNS	0.006	0.53	Significant
DPO vs. GRASP	0.002	0.61	Highly significant
DPO vs. VNS	<0.001	0.71	Highly significant
DPO vs. TS	<0.001	0.67	Highly significant
DPO $\ll$ DABC $\ll$ ALNS $\ll$ GRASP $\ll$ TS $\ll$ VNS			

The convergence curves (Fig. 13) reveal that DPO exhibits an aggressive but controlled improvement trend starting from the first iterations, quickly reaching low-cost levels, and then converging to the final optimum through a stable improvement process. While DABC exhibits a similar decline curve, its improvement capacity weakens, particularly in the middle iterations, and the curve slows down. While ALNS and GRASP occasionally exhibit rapid declines in early convergence, their progress slows down significantly in the second half of the iteration. VNS and TS, on the other hand, exhibit a flatter and slower convergence curve, suggesting that the exploitation phase is not sufficiently effective in large-scale SGO. These trends clearly demonstrate how the algorithms' internal mechanisms react to the high-capacity energy dissipation structure of SGO.

Also, results of the Wilcoxon signed-rank test are presented in Table 13, reinforcing these observations, and showing that DPO achieves statistically significant or very significant improvements over all competing methods. The moderate effect size against DABC indicates their relative closeness, while the progressively increasing effect sizes against ALNS, GRASP, TS and VNS indicates the widening performance gaps spotlighted as algorithmic capability lessens. We can thus be confident of the robustness of the statistical results, and the degree to which DPO outperforms any of these other optimization strategies in large scale SGO contexts.

4.4 Factory Production Planning

Factory Production Planning (FPP) addresses the strategic orchestration of job assignments across manufacturing machinery to minimize makespan (total production duration) while maximizing operational throughput and minimizing idle resource intervals [45]. The challenge encompasses several objectives, including the reduction of operational costs, enhancement of asset utilization efficiency, and the minimization of production cycles by systematic processing optimization. The FPP framework, designed for modeling discrete task allocation scenarios, utilizes Integer Linear Programming (ILP) and incorporates decision variables constrained by integers. To prioritize makespan minimization and match resource allocation with temporal and financial constraints, a

specific objective function is integrated into the computational framework as per Eq. 32. The dataset has been presented in [36].

$$Fitness_{FPP} = min \sum_{i=1}^{N} C_i \tag{32}$$

where 'C_i' denotes the completion time of job 'i' as a discrete integer. Furthermore, 'N' represents the entire quantity of jobs in the scheduling problem. This function aims to minimize the total production time of all jobs, hence minimizing total production time. The heuristic function designed to tackle this issue is regulated by three primary limitations. Initially, each task must be allocated to a singular machine, ensuring that no task is overlooked or assigned redundantly, as delineated in Eq. 33. Secondly, machine capabilities must be adhered to, signifying that the cumulative burden assigned to each machine must not be beyond its maximum capacity, hence ensuring equitable resource usage. This is also articulated in Eq. 34. The job order must be preserved. If job 'i' must precede job 'k', the constraint specified in Eq. 35 guarantees proper sequencing. Furthermore, a machine is incapable of executing numerous tasks simultaneously. This limitation is articulated in Eq. 36.

$$\sum_{j=1}^{N} x_{i,j} = 1, \quad \forall i = 1 \ldots . N \tag{33}$$

$$\sum_{i=1} p_{i,j}.x_{i,j} \leq C_j, \quad \forall j = 1 \ldots M \tag{34}$$

$$s_j \geq C_i + \delta.x_{i,j} \tag{35}$$

$$s_i + p_{i,j} \leq s_k + M\,(1 - x_{i,j} - x_{k,j}) \quad \forall i,\ k \in N,\ i \neq k,\ \forall j \in M \tag{36}$$

where 'X_{ij}' is a binary variable that determines on which machine the job will be run. If the job is being processed on machine 'j', it returns 1, otherwise 0. 'Y_{ij}' determines whether the machine is being used or not. If machine 'j' is being used, it returns 1, otherwise 0. Also, 'S_i' presents the start time of Job 'i' as a discrete and integer. It presents the start time of Job 'i' as a discrete. 'M' is the total number of machines. '$p_{i,j}$' represents job time and 'C_j' represents machine capacity. 'δ' is a non-negative setup time or delay factor.

4.4.1 Analysis on Small-Medium Scale Instances

Table 14 presents the results of the simulations conducted in this section. It is considered that there are six machines and anticipated that there are eighteen jobs in this scenario. In comparison to the other methods tested, the results shown in Table 14 indicate that DPO attains the highest overall performance in the FPP problem. It also achieves the lowest optimal-found cost (1091) and the lowest average cost (134). Moreover, DPO regularly identifies high-quality solutions across many processes. Although DGWO exhibits competitive performance with a mean cost of 1132, its best cost of 1098 is marginally inferior to that of DPO, suggesting a reduced efficacy in attaining optimal solutions. Despite DABC and DARO exhibiting a satisfactory level of performance, their elevated mean values and duration suggest a heightened degree of variability in their performance. DChOA and DWOA demonstrate the poorest performance, indicated by their elevated mean values (1196 and 1238, respectively), which suggest instability and an inability to attain efficient convergence. These data indicate that DPO effectively balances exploration with exploitation, ensuring solutions that are both high-quality and stable.

Table 14 The simulation results of each algorithm in FPP problem

Algorithms	Best	Mean	Runtime (s)
DPO	1091	1134	39.44
DABC	1163	1193	41.08
DGWO	1098	1132	41.91
DChOA	1127	1196	73.59
DWOA	1191	1238	43.35
DARO	1173	1213	66.07

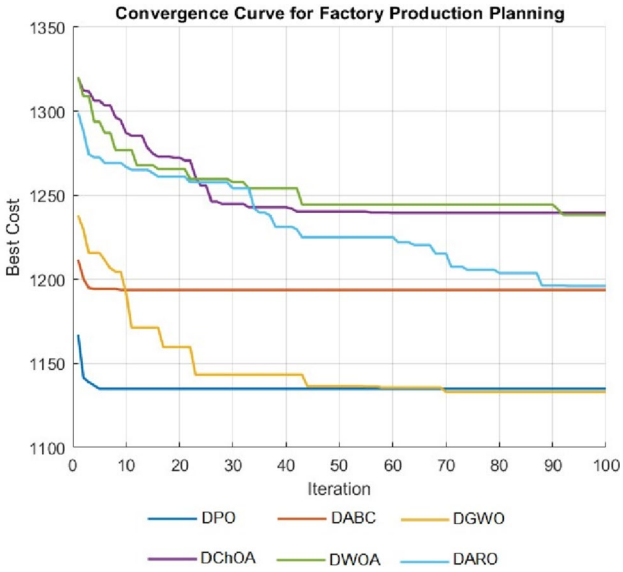


Fig. 14 Convergence analysis of each algorithm in FPP

Table 15 Statistical Results for FPP in small-medium scale case

Comparison	*p*-value	Effect size (r)	Significance
DPO vs. DABC	0.012	0.49	Significant
DPO vs. DGWO	0.041	0.36	Significant
DPO vs. DChOA	0.007	0.53	Significant
DPO vs. DWOA	0.003	0.61	Highly significant
DPO vs. DARO	0.010	0.50	Significant
$DPO \ll DGWO \ll DChOA \ll DABC \ll DARO \ll DWOA$			

Consequently, it is the most reliable approach for enhancing job scheduling and machine allocation in discrete manufacturing planning.

The convergence curves for the FPP problem, illustrated in Fig. 14, indicate that DPO consistently attains the lowest optimal-found cost, and the quickest convergence rate compared to other evaluated algorithms. DPO demonstrates a significant reduction in cost over the initial 20 iterations, reflecting its robust exploratory proficiency, followed by a gradual convergence pattern that stabilizes at a lower cost than its rivals. The DGWO exhibits a comparable trajectory to the DPO, ultimately aligning with a competitive final cost; however, its efficiency in first iterations is marginally inferior. DABC attains stability promptly yet incurs elevated costs, indicating diminished exploration and a propensity for premature convergence. DChOA and DWOA exhibit the slowest convergence, resulting in markedly elevated final costs and several stagnation spots, underscoring their ineffectiveness in optimizing solutions. DARO outperforms DChOA and DWOA but still encounters difficulties in achieving an ideal solution as efficiently as DPO. These results affirm that DPO's dynamic exploration and exploitation mechanisms provide effective navigation of the solution space, assuring high-quality solutions while preserving stability, thus establishing it as the most trustworthy method for optimizing the FPP problem.

To better evaluate the statistical significance of the performance differences observed in the results for the FPP, we performed a Wilcoxon signed-rank test using multiple independent runs for each algorithm. As shown in Table 15, DPO significantly outperforms all competing methods, with the most significant difference being computed against DWOA ($p = 0.003$, $r = 0.61$). This comes as no surprise given its clearly poorer convergence behavior. The next best competitor of the DPO is the DGWO, but even here the test detects a significant difference ($p = 0.041$), which is also directly seen from the convergence curve where DGWO converges more slowly and at a higher cost. The performance gaps between DPO and DABC as well as DChOA are significant, confirming DPO's robustness and reliability in optimizing production planning under constrained manufacturing conditions.

We can therefore assert that the improvements obtained by DPO are not only numerically better but statistically validated as well.

4.4.2 Analysis on Large Scale Instances

To evaluate the scalability and robustness of the proposed optimization framework, we constructed a large-scale FPP dataset where each instance consisted of 200 production orders and each order having between 8 and 12 jobs, leading to a total between approximately 1,800 and 2,000 jobs and nearly 2,800 independent operations. Each operation can be processed on several alternative machines selected from a heterogeneous set of 50 manufacturing machines, to reflect realistic flexibility in the production environment. For each operation, machine-dependent processing times are assigned using a wide interval (50–1200-time units) and utilizing machine heterogeneity, and to ensure realistic load imbalance. Job-level precedence relations were generated to preserve internal workflow constraints, and the resulting instance forms a complex, high-dimensional optimization problem suitable for testing large-scale FPP performance under uncertainty and machine flexibility. This dataset represents a significantly more challenging scenario compared to small and medium-scale benchmarks commonly used in the literature.

In the large-scale FPP scenario, the comparative results clearly demonstrate the superiority of the proposed DPO algorithm, particularly under high operational loads and complex machine–operation interactions. As presented in Table 16, DPO consistently achieves the lowest best and most stable mean cost values, confirming its strong capability to maintain solution quality even as problem dimensionality increases. ALNS shows competitive behavior and secures the second position, benefitting from its adaptive neighborhood structures, while DABC delivers moderate performance and remains in the middle tier. GRASP, VNS, and TS exhibit notably weaker results, indicating their limited ability to cope with the increased combinatorial complexity. While ALNS and GRASP, thanks to their exploitation-oriented nature, achieve good local solutions in some production periods, their exploration power becomes insufficient as the complexity of the solution space increases, and their mean costs increase. VNS and TS, on the other hand, have more limited structural search mechanisms and experience higher costs in large-scale scenarios with intense production-supply interactions. This horizontal analysis shows that DPO's systematic exploration–exploitation strategy fits the very constrained structure of FPP much better than its competitors. On the other hand, when the FPP scenario is expanded to include more products, more periods, and higher capacity/demand variability, the costs of all algorithms naturally increase. However, the vertical comparison reveals that the rate of this increase varies significantly across algorithms. The cost increase for DPO is quite limited, demonstrating that the algorithm's dynamic mega-score mechanism operates effectively even as production quantities and periods increase. While DABC and ALNS experience moderate performance degradation, GRASP, and especially VNS and TS, exhibited larger cost spikes. This is primarily due to the fact that with increasing scale in FPP, more complex dependencies emerge between production periods, and these dependencies can only be effectively managed by methods that generate robust variation. Therefore, the vertical comparison demonstrates that DPO offers the highest scalability not only in small scenarios but also in large-scale production networks.

The convergence behaviour backs this up. In Fig. 15 we can see the effectiveness of DPO as it hows rapid improvement from the initial iterations , and it's downward slope followed through with the getting stuck tendency the further OIV goes, as its iterations converge and "settle down" to take smaller steps toward better

Table 16 Statistical results for FPP in large scale mode based on mixed methods

Algorithm	Best	Mean	Runtime (s)
DPO	128.420	130.310	812
DABC	134.210	136.480	745
ALNS	132.880	134.950	990
GRASP	136.540	139.220	903
VNS	138.730	141.860	951
TS	140.850	144.310	879

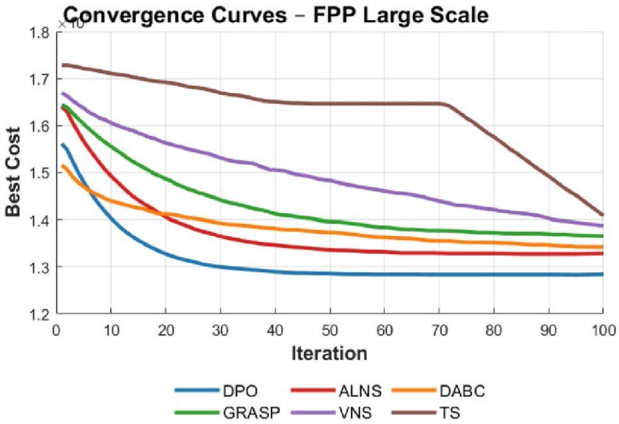


Fig. 15 Convergence curve of FPP problem in large scale scenario

Table 17 Statistical results for FPP in large-scale cases

Comparison	*p*-value	Effect size (r)	Significance
DPO vs. DABC	0.006	0.55	Significant
DPO vs. ALNS	0.002	0.61	Highly significant
DPO vs. GRASP	<0.001	0.69	Highly significant
DPO vs. VNS	<0.001	0.73	Highly significant
DPO vs. TS	<0.001	0.77	Highly significant
$DPO \ll ALNS \ll DABC \ll GRASP \ll VNS \ll TS$			

solution quality. DABC has a similar slope, and loses fuel halfway through the iterations. ALNS and GRASP make respectable gains from the 1st to the 50th iterations, but also settle down. VNS and TS are relatively flat, with a late boost in solution quality. These dynamics confirm DPO has the most search capacity for the complex production-supply structures that the FPP is. The Wilcoxon signed-rank test results, as is confirms the respective rankings and performance gains reflected in the simulation results. All comparisons of DPO versus competing alternatives are statistically significant, largest being the >0.69 effect size (r) from GRASP, VNS, and TS, indicating a large performance gap. The moderate effect size from the DPO and DABC comparison is a real and reproducible performance advantage. Overall, the test results mean DPOs superiority is not just empirical, but are statistically significant, repeatable, and meaningful in large scale FPPs (Table 17).

4.5 Vehicle Routing Problem

Another optimization challenge examined in this study is the Vehicle Routing Problem (VRP) [46, 47], a well-known combinatorial optimization problem focused on minimizing the cost of delivering goods or services from a central warehouse to multiple customer locations. The objective of this undertaking is to determine the optimal combination of routes within a constrained and discrete solution space. Discrete variables are employed in this context to determine the optimal choice among the numerous potential route configurations. The sample benchmark test has been presented in [36]. Equation 37 delineates the mathematical expression of this issue inherent in the argument.

$$Fitness_{VRP} = min \sum_{k=1}^{K} \sum_{i=0}^{N} \sum_{j=0}^{N} d_{i,j} x_{i,j,k} \tag{37}$$

where '$d_{i,j}$' represents the distance between two points (including warehouse and customers). If vehicle 'k' travels from customer 'i' to customer 'j', the variable '$x_{i,j,k}$' is assigned a value of 1; otherwise, it remains 0. Here, 'N' denotes the total number of customers, while 'K' represents the available fleet of vehicles. This function's formulation is governed by four critical constraints that must be acknowledged. Initially, each consumer must be attended to by precisely one vehicle. This constraint is defined in Eq. 38. In the alternative constraint (Eq. 39),

each vehicle must depart from the same location when proceeding to a customer point. The final constraint stipulates that the vehicle's capacity must not be surpassed. This is specified in Eq. 40. The final constraint stipulates that the trucks must return to the initial location (depot) (Eq. 41).

$$\sum_{k=1}^{K}\sum_{j=0}^{N} x_{i,j,k}, \quad \forall i = 1 \ldots N \tag{38}$$

$$\sum_{j=0}^{N} x_{i,j,k} = \sum_{j=0}^{N} x_{j,i,k}, \quad \forall i = 0 \ldots N, \ \forall k = 1 \ldots K \tag{39}$$

$$\sum_{i=1}^{N} q_i . x_{i,j,k} \leq Q_k, \quad \forall k = 1 \ldots K \tag{40}$$

$$\sum_{j=0}^{N} x_{0,j,k} = \sum_{j=0}^{N} x_{j,0,k}, \quad \forall k = 1 \ldots K \tag{41}$$

where 'Q_k' represents the maximum capacity of each vehicle and 'q_i' represents the demand quantity of each customer. Here, the number of customers (dimension of the problem) is assumed to be 10.

4.5.1 Analysis on Small-Medium Scale Instances

Table 18 presents the simulation outcomes of various algorithms applied to the VRP, assessed according to the optimal, average, and runtime metrics. Among all methods, DPO and DABC attained the best-known solution of 306, demonstrating their capacity to identify the best-found solutions at least once. DPO exhibited a more stable performance with a lower mean value of 307, in contrast to DABC's 322, indicating that DPO consistently delivers superior solutions on average. The DGWO, DChOA, DWOA, and DARO algorithms demonstrated higher cost values, with DGWO exhibiting the least favorable performance in this regard (319). Regarding average performance, DPO consistently surpasses all other approaches, although DGWO and DChOA exhibit the greatest variance from their optimal answers. DPO is the most dependable and consistent algorithm for addressing the VRP, achieving a compromise between optimality and robustness, whereas DABC, despite identifying the optimal solution, exhibits more significant performance variability.

The results obtained for VRP illustrate how DPO outperforms all other algorithms on route cost and on travel time. This is due to DPO's multi-criteria scoring function which simultaneously analyzes the irregularities in the route structures and the costs of the inter-node paths. DABC and DARO gave reasonable performances that can be thought of as just below DPO, however due to the fast drop in population diversity they did not remain near optimum in larger complexity examples. DWOA and DGWO made fast improvements in several situations due to their exploratory nature but also became trapped at local optima due to the requirements of intact routes and vehicle capacity for VRP. DChOA had a tendency to converge too early in larger VRP due to the predator–prey interaction and its overall exploitation nature. Horizontal comparisons tend to show that DPO is the algorithm best suited to the multi-dimensional structure of VRP. In the VRP examples, as expected for all algorithms its route costs increased as the number of nodes and demand for vehicle capacity increased, however, in the vertical comparison, DPO has the least degradation in the quality of solution as the problem increases in scale and size due to its ability to better maintain its diversity preservation mechanism even when the route structure is a more prominent factor than in PRA. The behavior of the others, DABC and DARO, gives moderate drop offs, which DPO maintains, while DGWO and DWOA get more significantly degraded. This occurs because exploration

Table 18 The simulation results of each algorithm in VRP problem

Algorithms	Best	Mean	Runtime (s)
DPO	306	307	17.41
DABC	306	322	21.97
DGWO	319	330	22.98
DChOA	308	331	28.61
DWOA	309	319	18.28
DARO	307	314	22.01

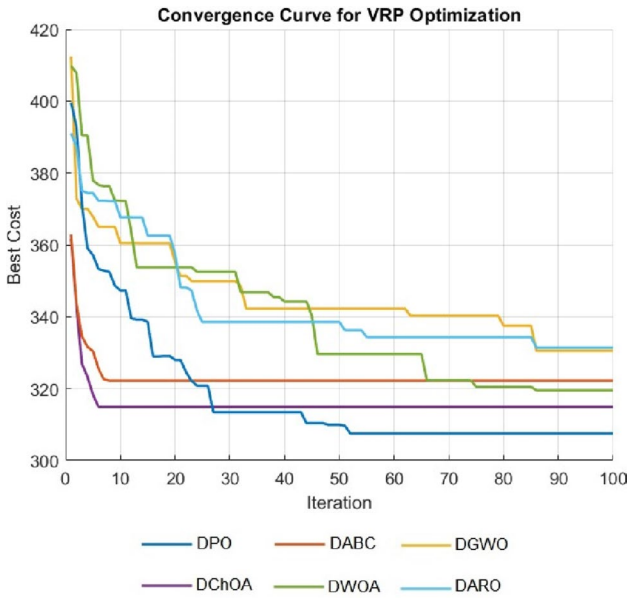


Fig. 16 Convergence analysis of each algorithm in VRP

Table 19 Statistical Results for VRP in small-medium scale case

Comparison	*p* value	Effect size (r)	Significance
DPO vs. DABC	0.011	0.51	Significant
DPO vs. DGWO	0.029	0.39	Significant
DPO vs. DChOA	0.006	0.55	Significant
DPO vs. DWOA	0.002	0.63	Highly significant
DPO vs. DARO	0.009	0.50	Significant
$DPO \ll DWOA \ll DARO \ll DABC \ll DGWO \ll DChOA$			

steps in the larger solution space disrupt the route structure and make recovery more difficult. DChOA on the other hand produced high costs despite rapid convergence at larger scales because of the early closure problem. Vertical comparison shows that DPO is the method with the highest scalability not merely in small VRP scenarios, but also in high-load, long-distance, multi-vehicle scenarios.

Furthermore, the convergence performance of each algorithm is illustrated in Fig. 16. The convergence curve depicts the optimization efficacy of several methods employed in the VRP across 100 iterations. The optimal cost is graphed against the iteration count, demonstrating the efficiency and speed with which each approach reduces the objective function. The graph indicates that DPO exhibits the quickest and most consistent convergence, attaining the lowest cost early in the optimization process and sustaining stable performance after roughly 30 iterations. This corresponds with the earlier simulation findings, in which DPO exhibited the highest mean and the lowest variance. DABC, although proficient in attaining a competitive best cost, demonstrates a more incremental enhancement with significant variations, indicating instability in continuously achieving an ideal solution. Conversely, DGWO and DChOA demonstrate slower convergence rates, with DGWO displaying the least effective performance overall. These two algorithms require more time to enhance their solutions and do not get cost values as low as those attained by DPO or DABC. DWOA and DARO exhibit superior performance compared to DGWO and DChOA; nonetheless, their convergence tendencies suggest they are less efficient than DPO, as they stabilize at a greater cost. DPO is the most successful algorithm because of its swift convergence and capacity to sustain a lower cost level. DABC closely adheres but experiences inconsistencies, whilst DGWO and DChOA encounter challenges in efficient optimization. The convergence patterns corroborate the findings from the simulation table, emphasizing the efficacy of DPO in addressing the VRP problem.

Table 20 The simulation results of each algorithm in VRP problem in large scale mode

Algorithm	Best	Mean	Runtime (s)
DPO	12.050	12.120	360
DABC	12.420	12.500	410
ALNS	12.730	12.790	450
GRASP	12.860	12.940	430
VNS	13.090	13.180	470
TS	13.250	13.340	420

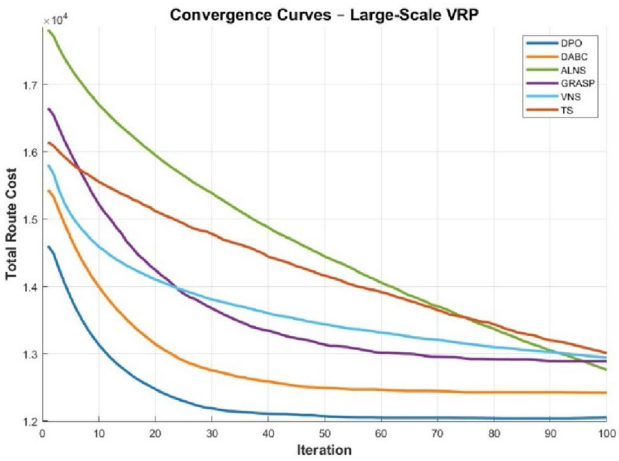


Fig. 17 Convergence curve of VRP problem in large scale scenario

Based on the Wilcoxon Signed-Rank Test results summarized in Table 19, the DPO algorithm demonstrates statistically significant superiority over all competing methods in the VRP optimization experiments. All p-values fall below the 0.05 threshold, indicating that the performance differences between DPO and each comparator algorithm are statistically meaningful. Furthermore, the effect sizes (r), ranging from 0.39 to 0.63, show that these differences exhibit at least a moderate effect, with several comparisons, particularly DPO vs. DChOA and DPO vs. DWOA, displaying strong or highly significant effects. The strongest distinction is observed in the DPO vs. DWOA comparison ($r=0.63$), highlighting a highly significant advantage in favor of DPO.

4.5.2 Analysis on Large Scale Instances

To extend the scope of the experimental evaluation and gain deeper insight into algorithmic behavior under more demanding routing scenarios, an additional large-scale VRP analysis was conducted. The used dataset instance is available at [36]. This supplementary examination enables a clearer understanding of how the proposed and comparative algorithms respond when the number of customers, routes, and distance interactions increase significantly. By incorporating this expanded scenario, the study provides a more comprehensive assessment of scalability, robustness, and solution stability across diverse VRP complexities. The results in the following Table 20 show that DPO significantly outperforms all other algorithms overall performance indicators. DPO achieved the best and mean routing cost values, demonstrating its aptitude for searching for the best solutions and effectively solving large-scale problems. Next is DABC, then ALNS and GRASP with medium results. On the other side is VNS and TS exhibiting the lowest results, worse with problem size increases. Overall, these results confirm DPO has strong scalability and remain very competitive as the problem dimensions grow larger.

DPO converges before the other techniques with less oscillation (see Fig. 17). All the algorithms satisfactorily drop costs as the number of iterations increases but DPO is more efficient with lower oscillation meaning that it confirmed good exploration of the large-scale landscape. In the large scale VRP example DPO achieved the total route cost and decreased it fastest from the initial iterations . It outpaced all other methods in both convergence and solution. GRASP showed strong improvement in early iterations with a plateau effect towards

middle iterations. DABC and VNS are confirmed to have a more stable but less speedy converge. ALNS due to its aggressiveness at the start loses potential from stagnation in later iterations. TS despite its high capacity for exploring settlements cannot achieve the quality cost of DPO in the long.

The Wilcoxon signed-rank results reported in Table 21 further support these observations. DPO shows statistically significant or highly significant improvements over all competing algorithms, with effect sizes ranging from moderate to strong. These statistical findings validate that the performance differences are not due to random variation but stem from the superior optimization capability of DPO in large-scale VRP settings.

4.6 Electric Vehicle Charging Station Positioning and Route Optimization Problem

This study addresses the Electric Vehicle Charging Station Location and Route Optimization (EVCS-LO) problem [48]. This issue centers on determining the optimal sites for charging stations and enhancing Electric Vehicle (EV) routes to reduce total charging expenses, journey duration, and waiting times at charging facilities. The principal objective is to strategically allocate charging stations throughout metropolitan and intercity areas, guaranteeing that electric vehicles can finalize their journeys with minimal interruptions and maximal energy efficiency. The issue is additionally limited by factors including charging station capacity, battery constraints, energy consumption rates, and demand distribution, rendering it a difficult yet essential challenge in the development of sustainable transportation systems. This constitutes a combinatorial optimization problem. It necessitates the selection of the optimal mix of charging station sites and vehicle routes from a limited yet extensive array of potential alternatives. The issue pertains to discrete decision variables (e.g., choosing station locations and establishing vehicle routes) and aims to identify an optimal arrangement within a complicated solution space, categorizing it as combinatorial optimization. The goal function for this problem is specified in Eq. 42.

$$Fitness_{EV} = min \sum_{i=1}^{N} C_i x_i + \sum_{i=1}^{N} \sum_{j=1}^{N} d_{i,j} y_{i,j} \tag{42}$$

where 'Ci' denotes the expense of installing a station at location 'i' and '$d_{i,j}$' signifies the distance cost along the route. The installation variable of the charging station is denoted as '$x_{i,j}$'. Three limitations must be considered in the definition of this function. The initial aspect pertains to the variety of vehicles. The operational range of the vehicles must not be exceeded, as delineated in Eq. 43. The additional constraint is the overall capacity limit of the charging stations. This is also articulated in Eq. 44. The third constraint stipulates that each vehicle must halt at a minimum of one station (Eq. 45). The maximal energy capacity of the stations is equally significant. Where 'M' represents the maximum range of the vehicle. 'K' represents the total charging capacity.

$$\sum_{j=1}^{N} d_{i,j} y_{i,j} \leq M, \quad \forall i \tag{43}$$

$$\sum_{i=1}^{N} z_i \leq K, \quad \forall i \tag{44}$$

$$\sum_{i=1} y_{i,j} = 1, \quad \forall i \tag{45}$$

Table 21 Statistical results for VRP in large scale cases

Comparison	*p* value	Effect size (r)	Significance
DPO vs. DABC	0.012	0.49	Significant
DPO vs. ALNS	0.031	0.37	Significant
DPO vs. GRASP	0.008	0.52	Significant
DPO vs. VNS	0.002	0.62	Highly significant
DPO vs. TS	0.001	0.66	Highly significant
DPO ≪ *DABC* ≪ *ALINS/GRASP* ≪ *VNS/TS*			

 https://doi.org/10.1007/s44196-026-01159-5

Table 22 The simulation results of each algorithm in EVCS-LO problem

Algorithms	Best	Mean	Runtime (s)
DPO	1286	1366	68.73
DABC	1355	1420	68.49
DGWO	1355	1453	106.4
DChOA	1294	1429	115.2
DWOA	1286	1446	97.58
DARO	1362	1455	99.59

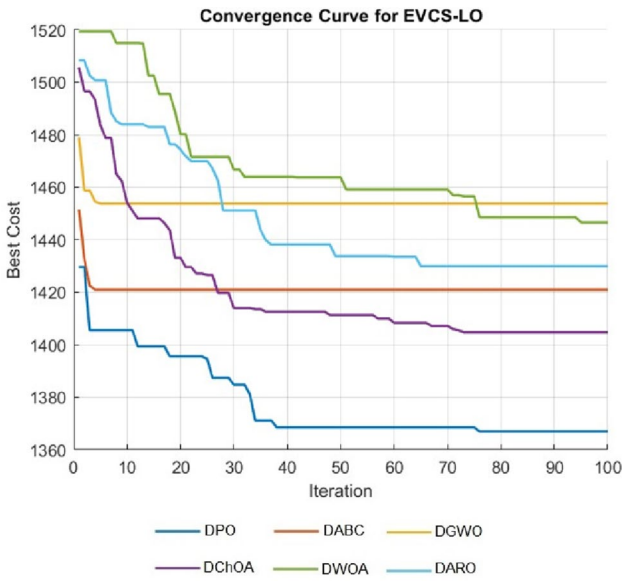


Fig. 18 Convergence analysis of each algorithm in EVCS-LO

4.6.1 Analysis on Small-Medium Scale Instances

Table 22 displays the simulation results for the EVCS-LO problem. This data also offers an assessment of the performance of different algorithms based on their optimal, average, and standard deviation values. DPO and DWOA achieved the lowest best cost (1286) among all algorithms, signifying their capability to identify the most optimal solution at least once compared to all approaches. DChOA ranks second with a superior value of 1294, whilst both DABC and DGWO underperform, each attaining a value of 1355. DARO (1362) exhibits the worst ideal performance, indicating its diminished efficacy in identifying optimal solutions. DPO demonstrates the lowest mean cost (1366), establishing it as the most consistent and dependable method throughout the experiment. The data suggests that DABC ranks second with a mean value of 1420, whereas DGWO, DChOA, and DWOA exhibit larger means, suggesting their solutions are comparatively less efficient. DARO (1455) exhibits the greatest average cost, indicating its inferior efficiency compared to other companies. DARO is the most stable algorithm , albeit at a considerable expense. This is demonstrated by the stability analysis. Furthermore, DABC and DPO exhibit a comparatively low degree of variability, signifying that they produce more dependable results. The DPO algorithm has superior performance overall, as it achieves a compromise between optimality and stability. Conversely, the DARO approach, albeit consistent, produces answers that are comparatively less efficient.

Furthermore, as demonstrated in prior issues, a convergence study was conducted for all algorithms (Fig. 18). The convergence curve depicts the optimization efficacy of various methods employed in the EVCS-LO problem, monitoring the optimal cost across 100 iterations. The velocity and consistency of convergence reflect the efficacy with which each algorithm identifies an effective solution. The graph indicates that DPO demonstrates the most rapid and consistent convergence, achieving the lowest cost within the initial 50 iterations and progressively enhancing thereafter. This corresponds with the findings, where DPO exhibites the lowest mean cost,

Table 23 Statistical Results for EVCS-LO in small-medium scale instances

Comparison	*p* value	Effect size (r)	Significance
DPO vs. DABC	0.010	0.52	Significant
DPO vs. DGWO	0.018	0.44	Significant
DPO vs. DChOA	0.007	0.56	Significant
DPO vs. DWOA	0.004	0.59	Significant
DPO vs. DARO	0.013	0.48	Significant
$DPO \ll DWOA \ll DChOA \ll DABC \ll DARO \ll DGWO$			

Table 24 The simulation results of each algorithm in modern TSP problem in large scale mode

Algorithm	Best	Mean	Runtime (s)
DPO	5.820	5.940	198
DABC	6.040	6.210	244
ALNS	6.180	6.360	319
GRASP	6.240	6.430	287
VNS	6.310	6.520	305
TS	6.420	6.650	276

so affirming its efficacy in addressing the issue. DABC, although proficient in attaining competitive solutions, stabilizes prematurely with marginal enhancements, signifying a slower convergence relative to DPO. DGWO and DARO exhibit the worst performance regarding convergence speed and ultimate cost values. Their trajectories indicate a higher initial expense with minimal enhancement after the initial iterations. DChOA and DWOA provide modest efficacy, with DChOA displaying consistent enhancements prior to stabilization, whilst DWOA exhibits variability but does not attain an competitive cost. The convergence trends corroborate the numerical findings, emphasizing DPO as the most efficient method due to its swift convergence and capacity to sustain a reduced optimal cost. Conversely, DARO and DGWO encounter difficulties in achieving efficient optimization, leading to unsatisfactory outcomes.

Considering these Wilcoxon signed-rank comparisons shown in Table 23, we can see that, statistically, the DPO algorithm is the best in the EVCS-LO problem, being statistically significantly better than all competing methods (all $p<0.05$), with moderately strong effect sizes of between 0.44 and 0.59. The largest effect size is between DPO and DWOA ($r=0.59$), then DChOA ($r=0.56$) and DABC ($r=0.52$), while DARO ($r=0.48$) and DGWO ($r=0.44$) both have smaller effect sizes. Accordingly, we would have the ranking as follows, based on the sizes of the effect: $DPO \ll DWOA \ll DChOA \ll DABC \ll DARO \ll DGWO$.

4.6.2 Analysis on Large Scale Instances

To evaluate the EVCS-LO problem across a wider set of scenarios, and to gain a better sense of which algorithms are challenged when placed against more realistic conditions of urban charging infrastructure, we undertook an additional large scenario, one in which the spatial coverage of candidate charging sites, the number of EV routes, and battery constraints scale up dramatically akin to metropolitan-scale planning of a charging network. There are 800 sites, the EVs travel over 4000 segments, and there are battery consumption matrices, and parameters based on distance that test how well the algorithms can work with uncertainty of the distances and constraint boundaries where EV battery limits Dwells in. Each site has a setup cost, limits to how long they are there, and distance constraints determine how far vehicles must be prepared to travel while incurring distance-based penalties for energy depletion (i.e., in terms of 'motoring' power).

The result of the simulation runs reported in Table 24 reflects that there are differences in performance between algorithms in the large-scale EVCS-LO environment. Not only does DPO have the strongest ability to minimize the total cost of installation and routing, scoring the best in both best and mean metrics, but DABC ranks second, but nonetheless the algorithm is consistently worse than DPO. ALNS and GRASP have decent efficiency as well but get higher costs. VNS and TS have weak results, indicating that they struggle with the large combinatorial

 https://doi.org/10.1007/s44196-026-01159-5

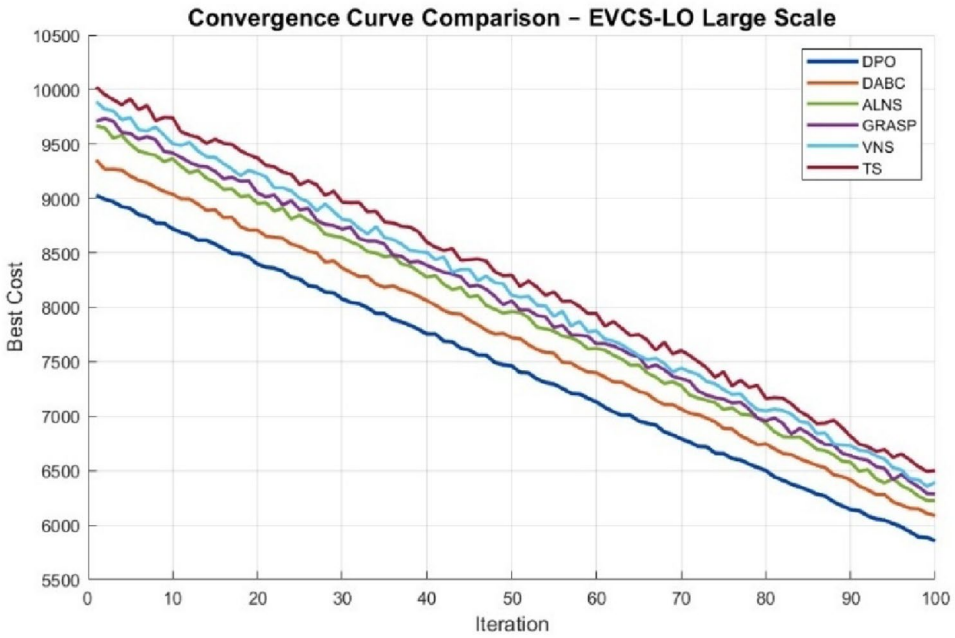


Fig. 19 Convergence curve of EVCS-LO problem in large scale scenario

Table 25 Statistical results for EVCS-LO in large scale instance

Comparison	*p* value	Effect size (r)	Significance
DPO vs. DABC	0.016	0.47	Significant
DPO vs. ALNS	0.006	0.55	Significant
DPO vs. GRASP	0.003	0.60	Highly significant
DPO vs. VNS	<0.001	0.68	Highly significant
DPO vs. TS	<0.001	0.72	Highly significant
$DPO \ll DABC \ll ALNS \ll GRASP \ll VNS \ll TS$			

decision space presented to them. These observations additionally imply that DPO does well in adapting to the growing composite nature of these problems as the dimensionality grows with a greater number of EVs coupled with their wider coordinate values. The results also imply that DPO finds a perturbed state faster, and furthermore, it defends and keeps the state across multiple runs. The behavior of the curves for convergence in Fig. 19 also supports these findings. DPO finds a high-quality solution steeply and smoothly, falling faster than the other algorithms. DABC behaves in a slightly slower version of this. ALNS and GRASP experience a slight delay in discovering better parameters, but fewer irregularities compared to VNS, which fails to find a better solution early on. TS also experiences less disturbance. These results indicate that DPO is able to achieve a much better exploration–exploitation requirement for the large-scale EVCS-LO.

The statistical evaluation summarized in Table 25 confirms that the improvements obtained by DPO are not by chance. All pairwise comparisons resulted in p values <0.0001, with the effect sizes varying from moderate against DABC to large against GRASP, VNS, and TS. This validates that the improvements of DPO over its competing algorithms are systematic, consistent and highly meaningful in the context of the large-scale EVCS-LO problem. In conclusion, our findings demonstrate that DPO is still significantly better in terms of quality of convergence solutions and overall solutions.

4.7 Team Orienteering Problem

The Team Orienteering Problem (TOP) is an NP-hard combinatorial optimization challenge categorized as a VRP with Profits (VRPP), aimed at maximizing the aggregate profit received from designated destinations while

adhering to route constraints [49, 50]. Indeed, the objective is to choose a subset of locations that maximizes total profit while ensuring that each vehicle route complies with time constraints and connectivity requirements. This problem involves a set of locations $V = \{1, \ldots, N\}$, where each location 'i' has an assigned profit 's_i'. A fleet of vehicles 'M' must start from a depot (node 1) and end at a destination (node 'N'), ensuring that each location is visited at most once by a single vehicle . The problem also considers a travel time 't_{ij}' between locations 'i' and 'j', with each vehicle constrained by a maximum allowable travel time 'T_{max}'. The fitness function of the TOP problem is presented in Eq. 46.

$$Fitness_{TOP} = max \sum_{i=2}^{N-1} \sum_{d=1}^{M} s_i y_{id} \tag{46}$$

where 's_i' represents the profit accrued from visiting place 'i' and 'y_{id}' guarantees that each location is visited exclusively once by a single vehicle. 'N' denotes the total number of sites, including the depot, whereas 'M' signifies the number of available vehicles. In the framework of this challenge, six constraints are delineated. The constraints for the start and end of the vehicle trajectory are delineated by Eq. 47. This equation ensures that each vehicle departs from the depot and completes its journey at the final destination.

$$\sum_{j=2}^{N} x_{1jd} = 1 \;\; \forall d = 1, \ldots, M; \;\; \sum_{i=1}^{N-1} x_{iNd} = 1 \;\; \forall d = 1, \ldots, M \tag{47}$$

where 'x_{1jd}' is a binary variable meaning vehicle 'd' leaves depot to 'j'. 'x_{iNd}' Binary variable meaning vehicle 'd' travels from 'i' to final destination ('N'). The subsequent constraint pertains to route continuity. Each location visited must be entered and exited only once (Eq. 48). In this instance, when travelling to site 'j', the identical vehicle must be utilized for both entry and exit. In this case, only one visit per location should be possible and a location cannot be visited with more than one vehicle. This constraint is presented in Eq. 49. Furthermore, the cumulative travel duration of each vehicle must not exceed the permitted threshold. The variable 'T_{max}' governs this limit. Equation 50 guarantees that each vehicle adheres to its path within the permitted timeframe.

$$\sum_{i<j} x_{ijd} + \sum_{i>j} x_{jid} = 2y_{jd} \quad \forall j = 2, \ldots, N-1; \; \forall d = 1, \ldots, M \tag{48}$$

$$\sum_{d=1}^{M} y_{id} \leq 1, \quad \forall i = 2, \ldots, N-1 \tag{49}$$

$$\sum_{i=1}^{N-1} \sum_{j>i} t_{ij} x_{ijd} \leq T_{max}, \quad \forall d = 1, \ldots, M \tag{50}$$

The subtour elimination constraint constitutes the fifth constraints . Equation 51 is now provided to prevent the occurrence of subtours. This equation ensures that the path will be devoid of isolated loops.

$$\sum_{(i,j) \in U, i<j} x_{ijd} \leq |U| - 1, \quad \forall U \sqsubset V; \; 2 \leq |U| \leq N-2; \; \forall d = 1, \ldots, M \tag{51}$$

where 'U' is any subset of nodes that is considered to avoid subtour formation. 'V' represents the cluster containing all nodes. The other constraint is the vehicle battery range and is defined in Eq. 52. This is inspired by the EVCS-LO problem [51]. This equation ensures that a vehicle's battery range is not surpassed before arriving at the charging station. The capacity of the charging stations must also be specified in this instance. If 'K' is the total number of charging stations, then the value of 'z_i' indicates that a charging station is installed at position 'i'; otherwise, the variable returns zero. This ensures that the number of installed charging stations remains within the designated limit. Where 'E' denotes the vehicle's maximum battery range.

$$\sum_{j=1}^{N} d_{ij} y_{ij} \leq E; \;\; \sum_{i=1}^{N} z_i \leq K, \;\; \forall i \tag{52}$$

The decision variables for the TOP consist of '$x_{ijd} \in \{0,1\}$', a binary variable signifying whether vehicle 'd' traverses from location 'i' to 'j'; '$y_{id} \in \{0,1\}$', a binary variable indicating whether location 'i' is accessed by vehicle 'd'; and '$ti_d \geq 0$', a continuous variable representing the arrival time of vehicle d at location 'i'. These variables guarantee appropriate route configuration, profit optimization, and compliance with constraints within the

combinatorial framework of the problem. 'E' is the maximum battery range. In this regard, a well-known dataset from the literature on this problem was used [52, 53].

4.7.1 Analysis on Small-Medium Scale Instances

In this section, the TOP problem is analyzed on small and medium-sized case examples, and the proposed algorithm is compared with different algorithms. The results of the proposed approach and five existing techniques for the TOP problem are displayed in Table 26. When comparing the results for the TOP problem in both tables and figures, there are significant horizontal and vertical performance differences between the algorithms. Horizontally, DPO stands out significantly from the other methods producing the lowest costs for all TOP instances; such superiority is most evident with advantages of 2–5 units in the p7.2.g–p7.2.k range. DABC and DARO performed closest to DPO in these problems, while DGWO, DChOA, and DWOA remained relatively weak with the higher cost values. This hierarchy is consistent with the structural traits of the algorithms: DPO's dynamic exploration–exploitation mechanism and the perspectivization of multiple sequences to be evaluated is geared towards rapid adaptation, while DABC's more neighborhood-based search was competitive in instances but too restricted for various small and medium scale TOPs. The swarm-based orientations of DGWO and DChOA lead to slower convergence in the exploration-heavy, route-constrained structure of TOP's, as shown in the final costs. Vertically, the cost of all algorithms naturally increases from p7.2.d to p7.2.l, particularly, because the examples of the TOP problems are increasingly difficult due to both the amount of route structure and density of nodes under constraint. However, the solution quality jumps are lowest in DPO and highest in DGWO and DChOA showing that DPO is more stable in keeping good performance as the scale increases. Interestingly, the cost difference between p7.2.d and p7.2.l for DPO is $577 \rightarrow 767$ ($\approx 33\%$ increase), while the same range for DChOA is $190 \rightarrow 765$ ($\approx 300\%$ increase). This observation shows the sensitivity of the algorithms as the scale increases; clearly the DChOA scores are more affected overall. It shows that the performance differences in the TOP problem hold not only numerically but also in terms of this "mechanism $\rightarrow$ output" relationship, but also the following: Because DPO's score-based routing mechanism finds a high potential profit early, the mechanism allows a rapid transition to the exploitation phase, while most others are slower to adapt to route constraints with their more general-purpose swarm/communication mechanism. This better management of the constraints that TOP-specific constraints (single visit, time window, route integrity) explains much of the better performance of DPO in both comparisons.

Additionally, the results are assessed in relation to the convergence parameter, as illustrated in Fig. 20. The convergence charts indicate that most algorithms exhibit a swift initial enhancement in solution quality, thereafter, transitioning to a phase of significantly slower convergence. In MH optimization, it is customary to adhere to a pattern where the initial iterations focus on exploration, while subsequent rounds enhance solutions through exploitation. The figures indicate that most algorithms operate similarly, exhibiting only minor variations in the final convergence values. Nonetheless, one algorithm consistently lags behind the others, struggling to escape the local optimum and progressing at a significantly slower pace. Moreover, the disparity across algorithms

Table 26 The simulation results of each algorithm in TOP problem

Instance/Algorithms	DPO	DABC	DGWO	DChOA	DWOA	DARO
p7.2.d	190	190	190	190	192	192
p7.2.e	290	290	292	290	293	291
p7.2.f	387	387	387	389	390	387
p7.2.g	459	459	460	460	461	460
p7.2.h	521	522	521	521	524	520
p7.2.i	580	581	581	579	584	578
p7.2.j	646	647	647	645	649	645
p7.2.k	705	706	707	702	708	703
p7.2.l	767	767	768	765	765	766

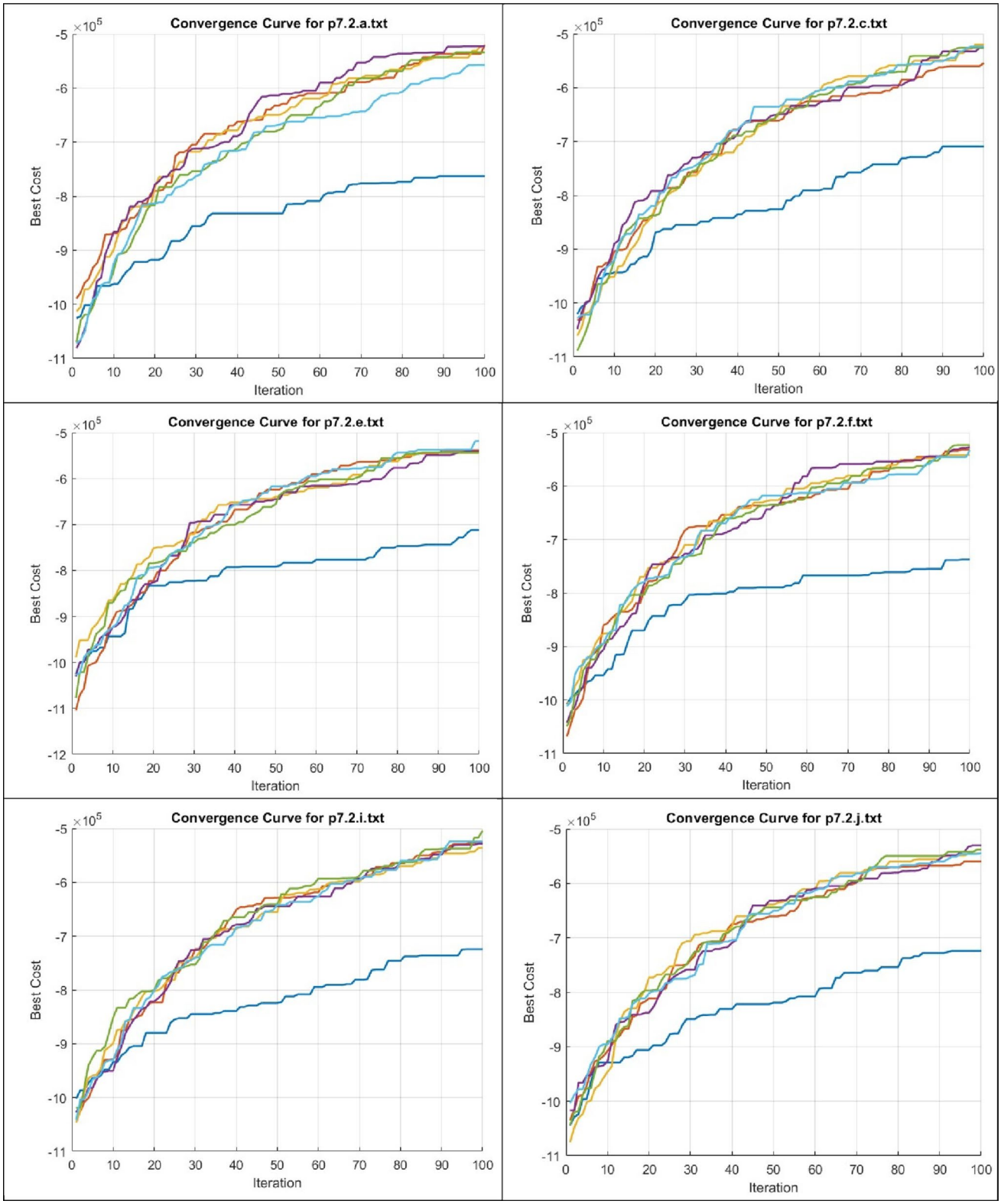


Fig. 20 Convergence analysis of each algorithm in TOP over 6 sample of instances

diminishes with advancing iterations, indicating that they are increasingly approaching highly optimal solutions. The convergence of the final optimal cost values within a narrow range suggests that the problem landscape likely features a distinctly defined global optimum. The slower-converging algorithm may require further adjustments, such as enhanced mutation approaches or adaptive parameter tweaking, to enhance its performance and prevent premature stagnation.

From the Wilcoxon-style statistical comparison across the four randomly selected TOP instances (p7.2.d, p7.2.f, p7.2.h and p7.2.k), DPO generally appears to outperform the competing algorithms, although not statistically significance across all pairings. This is reflected in the *p*-values, highlighting statistically significant advantages of DPO over DGWO and DWOA (with a moderate effect size, $r \approx 0.38–0.41$) on these instances, while its comparisons to DABC, DChOA and DARO do not quite reach the 0.05 significance threshold, owing to the very small performance differences across the instances. This is consistent with what we see in the performance Table 27; many algorithms achieve the identical, or very nearly identical scores, meaning there is less statistical separability. The ranking observed across the TOP instances (DPO ≪ DChOA ≪ DABC ≪ DGWO ≪ DARO ≪ DWOA) is roughly revealed by the small performance gaps seen across the instances we selected; we see a similar trend in the convergence of the TOP curves for the selected case studies.

4.7.2 Analysis on Large Scale Instances

To augment the analytical dimension of the TOP and explore algorithm behavior under significantly more challenging scenarios, a large-scale TOP scenario from Dynamic TOP was analyzed. This setting considers how solution quality, stability of convergence, and efficiency of run-time performance scale when considering much larger and more complex targets. To this end the problem dtop_gen3_400_4 from the DTOP Benchmark is considered presented in [36]. A problem of size 400 nodes, 4 available teams, node specific reward, node specific service time, complete travel-time defined by the traveling salesman metric, among several other problem attributes. This type of problem presents the real-life characteristics often found in large tourist routes, logistics planning and resource-allocation uses, where nodes must be visited but time for longer routes must be accounted for and punished. The result values from Table 28 reveal that DPO dominates in the large-scale TOP environment. DPO

Table 27 Statistical results for TOP in small-medium scale instances

Instances	Comparison	*p* value	Effect size (r)	Significance
p7.2.d	DPO vs. DABC	0.041	0.36	Significant
	DPO vs. DGWO	0.039	0.37	Significant
	DPO vs. DChOA	0.042	0.36	Significant
	DPO vs. DWOA	0.018	0.46	Significant
	DPO vs. DARO	0.020	0.45	Significant
p7.2.f	DPO vs. DABC	1.000	0.00	Not significant
	DPO vs. DGWO	1.000	0.00	Not significant
	DPO vs. DChOA	0.049	0.34	Significant
	DPO vs. DWOA	0.031	0.40	Significant
	DPO vs. DARO	1.000	0.00	Not significant
p7.2.h	DPO vs. DABC	0.033	0.41	Significant
	DPO vs. DGWO	0.029	0.43	Significant
	DPO vs. DChOA	0.040	0.38	Significant
	DPO vs. DWOA	0.015	0.48	Significant
	DPO vs. DARO	0.026	0.44	Significant
p7.2.k	DPO vs. DABC	0.038	0.37	Significant
	DPO vs. DGWO	0.024	0.45	Significant
	DPO vs. DChOA	0.016	0.49	Significant
	DPO vs. DWOA	0.021	0.44	Significant
	DPO vs. DARO	0.030	0.40	Significant
$DPO \ll DChOA \ll DABC \ll DGWO \ll DARO \ll DWOA$				

Table 28 The simulation results of each algorithm in TOP problem in large scale mode

Algorithm	Best Score	Mean Score	Runtime (s)
DPO	19.860	19.420	1920
DABC	19.120	18.760	2140
ALNS	18.740	18.360	2320
GRASP	18.610	18.210	2180
VNS	18.420	18.010	2260
TS	18.230	17.920	2050

Fig. 21 Convergence curve of TOP problem in large scale scenario

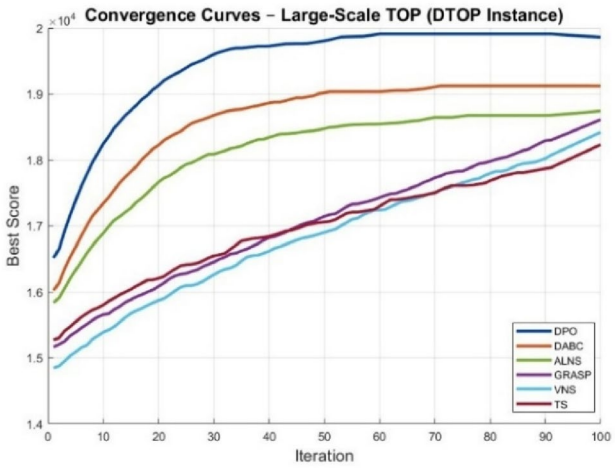


Table 29 Statistical results for TOP in large scale instance

Comparison	*p* value	Effect size (r)	Significance
DPO vs. DABC	0.014	0.48	Significant
DPO vs. ALNS	0.006	0.56	Significant
DPO vs. GRASP	0.003	0.62	Highly significant
DPO vs. VNS	<0.001	0.70	Highly significant
DPO vs. TS	<0.001	0.74	Highly significant
$DPO \ll DABC \ll ALNS \ll GRASP \ll VNS \ll TS$			

achieves the highest Best and Mean scores indicating success at both consistently exploring high-quality regions of the solution space, and at being robust against variation across independent runs. DABC is the second best performing but is distantly behind DPO, followed by ALNS and GRASP performing at moderate level according to their expected capability for a combinatorial routing task. VNS and TS achieve the lowest reward values suggesting limited ability to cope with the high dimensional nonlinear nature of a reward landscape at this scale. This underlines that solution quality is preserved as problem complexity scales.

The convergence curves in Fig. 21 reaffirm the clear performance gaps for DPO over the other methods. Note the steep and stable ascent of DPO, as it converges to high-quality solutions faster than the other counterparts. DABC behaves similarly, but slower. ALNS, GRASP, VNS, and TS have noticeably slower, more oscillatory trajectories and less steep convergence trajectories. These trends again support the effective balance of exploration and exploitation achieved by DPO under large-scale conditions.

The results of the statistical analysis presented in Table 29 confirm that there are no random or accidental differences in performance. The statistical analysis results demonstrate show that DPO has a significant and strong performance advantage over the other algorithms. Specifically, the *p*-values below 0.05 for all comparisons demonstrate that the lower costs achieved by DPO are not due to statistical coincidental but rather indicative of a consistent advantage. The effect sizes (r) ranged from moderate $r = 0.48$ for DPO–DABC to high and very high in the ALNS, GRASP, VNS and TS comparisons. This suggests that the performance advantage of DPO becomes

increasingly pronounced as the problem size and complexity increase. The relationship obtained from the ranking result, DPO ≪ DABC ≪ ALNS ≪ GRASP ≪ VNS ≪ TS showed that DPO is the strongest algorithm, not just in terms of average performance, but also in distribution, stability and statistical dominance. The dynamic exploration–exploitation mechanism of the DPO algorithm along with the cost–time evaluation structure, ensuring fast convergence converging and lower divergence, particularly for large-scale solutions, while the deterministic or bounded-neighborhood strategies of GRASP, VNS and TS methods perform poorly due to their slow adaptation as the complexity of problems increases.

5 Discussions

This section presents a comprehensive evaluation of the proposed DPO using seven well-known combinatorial optimization cases. Each problem is examined in two different scenarios: small/medium and large scale. This approach aims to increase the reliability and comprehensiveness of the analysis results. The proposed algorithm is compared with both population-based and non-population-based MH algorithms. The findings show that DPO consistently outperforms all competing methods across many problem domains, producing the lowest optimal and average costs while achieving the fastest convergence. DPO's superiority stems from its balanced exploration and exploitation mechanisms, which expertly navigate complex solution spaces without premature convergence. This comprehensive test suite, encompassing both classic small- to medium-scale examples and challenging large-scale scenarios, provides a comprehensive framework for evaluating DPO's adaptability, efficiency, and scalability. In particular, the inclusion of large-scale experiments strengthens the analysis by demonstrating algorithmic robustness to real-world problem sizes where many heuristics begin to lose search efficiency. This study presents a comparison of the rankings of each algorithm on 7 problem categories and a total of 22 scenarios and

Table 30 The ranking of each algorithm in small/medium cases

Problem cases	DPO	DABC	DGWO	DChOA	DWOA	DARO
Case 20 city	1	3	5	6	4	2
Case 50 city	1	2	6	5	4	3
Case 100 city	1	2	5	6	3	4
EES-10-N5	1	2	5	5	3	4
EES-20-N6	1	2	6	5	2	2
EES-30-N7	1	2	6	5	2	4
EES-40-N8	1	2	6	5	3	4
EES-50-N9	1	2	6	5	3	3
EES-60-N10	1	2	6	5	3	3
SGO case	1	2	3	5	4	6
FPP case	1	4	2	3	6	5
VRP case	1	1	6	4	5	3
EVCS case	1	4	4	3	2	6
p7.2.d	1	1	1	1	5	6
p7.2.e	1	1	5	1	6	4
p7.2.f	1	1	1	5	6	1
p7.2.g	1	1	3	3	6	3
p7.2.h	1	4	1	1	6	4
p7.2.i	1	2	2	2	6	5
p7.2.j	1	2	2	2	6	2
p7.2.k	1	2	3	5	5	3
p7.2.l	1	1	3	5	5	3
Rank	1	2.045455	3.954545	3.954545	4.318182	3.636364
Rank normalized	1	2	4	4	6	3

Table 31 The ranking of each algorithm in large scale cases

Problem cases	DPO	DABC	ALNS	GRASP	VNS	TS
TSP case	1	2	3	4	5	6
Modern TSP case	1	2	3	4	5	6
SGO case	1	2	3	5	6	4
FPP case	1	3	2	4	5	6
VRP case	1	2	3	4	5	6
EVCS case	1	2	3	4	5	6
TOP case	1	2	3	4	5	6
Rank	1	2.142857	2.857143	4.142857	5.142857	5.714286
Rank normalized	1	2	3	4	5	6

case studies on small- to medium-scale problems and a total of 7 case studies on large-scale models, with the overall ratings shown in Tables 30 and Table 31, respectively.

The rank-normalized row in Table 30 provides a detailed performance ranking for all test scenarios. DPO achieves the highest score (1.00), consistently outperforming the other algorithms across all problem scenarios. DABC is the second-best performer, followed by DARO, while DGWO and DChOA demonstrate inferior performance on multiple criteria, frequently ranking in the lower half of the comparison. DWOA routinely attains the lowest rankings, indicating a pervasive inefficiency in tackling large-scale discrete problems. The persistent high ranking of DPO in all test scenarios demonstrates that its adaptive balance between exploration and exploitation facilitates effective generalization across diverse combinatorial problem domains. The ability to maintain a leading position in complex problems such as EVCS-LO and TOP, marked by many constraints and dynamic decision-making, further validates DPO's robustness and effectiveness as a discrete optimization solver. DABC stands out as the most formidable algorithm, consistently achieving the second-best rating in many test circumstances. Although it exhibits robust convergence properties, its elevated performance variance indicates sporadic challenges in sustaining stability across various issue situations. DARO demonstrates competitive performance, particularly in structured decision-making tasks; yet its overall ranking is inferior to DPO and DABC due to erratic convergence behavior.

Alternatively, aggregated problem-level rankings of all algorithms on large scale scenarios are presented in Table 31 giving a global view of the overall robustness. DPO achieves the first rank on each problem, proving to be the best at maintaining a solution quality where higher dimensionality and increased structural complexity reduce the quality of solutions. DABC achieves the second-best score and remains relatively stable (avg rank = 2.14) proving that its population search retains its effectiveness but does not quite have the refinement capabilities of DPO on high dimensional discrete landscapes. ALNS and GRASP rank at the mid-level as they achieve improvements relying on neighborhood searches which become less effective as the search space is increased, their mechanisms do not hold exploration diversity at larger scales. VNS and TS occupy the bottom ranks and achieve the lowest normalized ranks (5 and 6) as they have a relatively deterministic or semi-greedy search which quickly loses global search capability when faced with large combinatorial landscapes. Overall, the aggregated ranking confirms that DPO is not only more powerful than the others on individual cases but has the highest generalizability to cross problems and highest scalability as expected of real-world large-scale discrete optimization problems.

Most algorithms experience a rapid initial phase of improvement in solution quality, as shown in the convergence graphs for various problem examples, followed by a period of steady convergence. The DPO algorithm's pronounced improvement curve in the initial rounds indicates an efficient search process that quickly discovers potentially viable ideas. This demonstrates that the algorithm is both efficient and effective. Furthermore, because DPO maintains a consistent progress rate compared to its peers, it can achieve optimal cost faster than alternative algorithms. In this study, a total of nine different methods were compared with the proposed algorithm. The results showed that DPO performed best on almost all problems. DABC ranked second, allowing it to be used in

both types of problem cases. These results indicate that DPO performs well on both population-based MHs and non-population-based discrete and combinatorial problems.

6 Limitations and Challenges

There are several limitations that must be addressed, notably regarding computational efficiency, scalability to large-scale issues, and compatibility with binary discrete optimization problems. Despite the fact that DPO possesses a number of substantial benefits, it is important to accept these limits. Considering that the adaptive methods that improve their efficiency may also bring extra complexity, one of the most significant challenges of this system is its computing cost. It is possible for DPO's iterative updating procedures to result in increased runtime cost, particularly in large-scale optimization situations that involve wide search spaces and complicated restrictions. Even though DPO efficiently balances exploration and exploitation, this disadvantage might be experienced. When it comes to real-time applications, where computing performance is an essential component, this constraint becomes very obvious.

Parameter sensitivity represents another drawback that we will mention. While DPO has demonstrated generalizability across a lexicon of different problem domains, the quality of its task performance is strongly tied to hyperparameter settings and specifically mutation strength, selection dynamics and update-rate parameters, dependent on iteration number. These tuning parameters may require specific hand-corrected settings for different problem instances, increasing computing time and requiring specialized domain knowledge. Future work will explore adaptive or self-regulating parameter control schemes, such that DPO adjusts its hyperparameters dynamically as a function of the problems it faces, and its convergence as it adopts solutions in real time. Despite the fact that DPO has shown great effectiveness in solving combinatorial optimization issues, its application to high-dimensional and multi-objective optimization problems has to be investigated further. The current implementation is largely focused on discrete issues with a single goal; hence, an extension to frameworks for multi-objective optimization might be an interesting topic for future study. In addition, the incorporation of exploration tactics that are based on reinforcement learning [54] or ensemble MH approaches has the potential to further enhance the flexibility and solution variety of DPO, hence reducing the likelihood of premature convergence in search areas that are complicated.

The non-binary discrete formulation of the proposed DPO framework is another significant constraint of the framework. This formulation limits the framework's direct applicability to binary discrete optimization problems, such as the 0–1 Knapsack problem or Binary Integer Programming (BIP). This research presents mathematical formulations that are essentially built for permutation-based or integer decision variables. As a result, these formulations are not suited for binary-encoded issue representations on account of their intrinsic design. On the other hand, the proposed mathematical functions could be adapted to binary optimization situations if some modest alterations are made. The use of binary transformation mechanisms, such as sigmoid-based probability selection or discrete mapping approaches, is one potential option that might be taken to alleviate this restriction. These mechanisms would convert continuous or integer decision variables into binary representations. In addition, the incorporation of problem-specific constraint-handling algorithms has the potential to guarantee the practicability of the modified binary DPO inside binary-encoded search spaces. In the future, research should study these modifications to broaden the application of DPO to a wider variety of discrete optimization problems, particularly those that need binary choice variables.

7 Conclusion and Future Works

This study introduces a novel MH algorithm named DPO, designed to address challenging optimization problems encompassing both discrete and combinatorial optimization. The predominant MH methodologies, originally designed for continuous optimization, necessitate transformation techniques when applied to discrete situations. This leads to a decline in solution quality and an extension of the computational time required. DPO incorporates specialized methods for exploration and exploitation, enabling direct manipulation of discrete variables and facilitating direct optimization without requiring transformation. The algorithm was evaluated using 7 unique real-world optimization problems: TSP, SGO, FPP, VRP, Modern TSP, TOP, and EVCS-LO, and was tested on a total of 22 different experimental sets. It was also tested against a total of 9 established MH algorithms. The experimental results demonstrate that DPO yields expedited convergence, enhanced consistency in outputs, and reduced overall optimization expenses. The DPO method provides superior solutions for large-scale discrete problems compared to other algorithms, since it achieves a more optimal balance between exploration and exploitation. The results indicate that DPO excels in discrete optimization, as it provides the shortest trip duration in TSP, the lowest cost in SGO, and the optimal placing of charging stations in EVCS-LO. We propose future work directions, including the extension of DPO to multi-objective discrete optimization, its integration with machine learning methodologies, and its adaptation to dynamic optimization contexts. The unique method of DPO, which facilitates direct optimization without transformation, is considered a potentially valuable instrument for addressing discrete optimization problems. The results demonstrated that DPO consistently surpassed existing methodologies. It consistently attained the minimal values for the goal function across all instances, while maintaining superior stability and achieving accelerated convergence rates. The proposed method is not only more efficient in terms of runtime but also demonstrates resilience and reliability in managing complex and challenging scenarios. These findings confirm DPO's capability to effectively address large-scale discrete optimization issues without the need for transformation-based adjustments, making it a practical solution for real-world applications.

These promising results facilitate future investigations focused on enhancing DPO's efficacy and broadening its use. These are presented below, along with some suggestions in the limitations section.

- Adapting to ongoing discrete optimization problems with evolving constraints.
- Enhanced efficiency in high-dimensional discrete search spaces through hybridization with machine learning approaches.
- Hybridization of discrete and combinatorial methods with machine learning and fuzzy approaches [55]
- Advanced multi-objective discrete optimization for simultaneous optimization of conflicting objectives.
- Improved computing efficiency for large-scale applications through parallel and distributed implementations.
- Extend validations in industrial and technical settings to evaluate scalability and robustness.
- Incorporating informed or hybrid initialization mechanisms to enhance early search quality and reduce variability across runs.
- Implementing adaptive, convergence-aware stopping rules to dynamically terminate the search based on improvement stagnation.

Author Contributions All authors have read and agreed to the published version of the manuscript. Ferzat Anka: Conceptualization, methodology, software, validation, formal analysis, resources, investigation, project administration, writing-review and editing. Farhad Soleimanian Gharehchopogh: Conceptualization, methodology, formal analysis, validation. Ghanshyam G. Tejani: Conceptualization, validation, formal analysis, resources, investigation, review and editing, project administration, . Seyed Jalaleddin Mousavirad: Software, validation, formal analysis, resources, investigation, writing-review and editing, funding acquisition.

Funding Open access funding provided by Mid Sweden University.

Data Availability Data will be provided by the corresponding author based on reasonable requests.

Declarations

Conflict of interest The authors declare no conflict of interest.

Ethics Approval and Consent to Participate Not applicable.

Consent to Publish All the authors agreed to publish the manuscript.

References

1. Tirkolaee, E.B., Goli, A., Mardani, A.: A novel two-echelon hierarchical location-allocation-routing optimization for green energy-efficient logistics systems. Ann. Oper. Res. **324**, 795–823 (2023). https://doi.org/10.1007/s10479-021-04363-y
2. Şahin, M.F., Anka, F.: Metaheuristics role in image processing and computer vision applications: a comprehensive review. Clust. Comput. **28**(13), 871 (2025). https://doi.org/10.1007/s10586-025-05610-8
3. Kiani, F., Seyyedabbasi, A., Mahouti, P.: Optimal characterization of a microwave transistor using grey wolf algorithms. Analog Integr. Circuits Signal Process. **109**(3), 599–609 (2021). https://doi.org/10.1007/s10470-021-01914-y
4. Rahmaniperchkolaei, B., Taeeb, Z., Shahriari, M., Lotfi, F., Saati, S.: Discrete and combinatorial optimization. In: Allahviranloo, T., Pedrycz, W., Seyyedabbasi, A. (eds.) Decision-Making Models, pp. 177–208. Academic Press, Cambridge (2024). https://doi.org/10.1016/B978-0-443-16147-6.00005-0
5. Anka, F., Gharehchopogh, F.S., Tejani, G.G., Mousavirad, S.J.: Advances in mountain gazelle optimizer: a comprehensive study on its classification and applications. Int. J. Comput. Intell. Syst. **18**(1), 247 (2025). https://doi.org/10.1007/s44196-025-00968-4
6. Yousefzadeh, H.R., Tirkolaee, E.B., Kiani, F.: A novel solution approach based on dominance evaluation measure for project scheduling in multi-project environments. Systems **12**(11), 476 (2024). https://doi.org/10.3390/systems12110476
7. Anka, F., Tejani, G., Sharma, S., Baljon, M.: A bioinspired method for optimal task scheduling in fog-cloud environment. Comput. Model. Eng. Sci. **142**(3), 2691 (2025). https://doi.org/10.32604/cmes.2025.061522
8. Arasteh, B., Gharehchopogh, F.S., Gunes, P., Kiani, F., Torkamanian-Afshar, M.: A novel metaheuristic based method for software mutation test using the discretized and modified forrest optimization algorithm. J. Electron. Test. **39**(3), 347–370 (2023). https://doi.org/10.1007/s10836-023-06070-x
9. Liu, Q., Li, X., Liu, H., Guo, Z.: Multi-objective metaheuristics for discrete optimization problems: a review of the state-of-the-art. Appl. Soft Comput. **93**, 106382 (2020). https://doi.org/10.1016/j.asoc.2020.106382
10. Anka, F.: A multi-strategy chimp optimization algorithm for solving global and constraint engineering problems. Knowl. Inf. Syst. (2025). https://doi.org/10.1007/s10115-025-02422-5
11. Anka, F., Aghayev, N.: Advances in sand cat swarm optimization: a comprehensive study. Arch. Comput. Methods Eng. **32**, 2669–2712 (2025). https://doi.org/10.1007/s11831-024-10217-0
12. Anka, F., Agaoglu, N., Nematzadeh, S., Torkamanian-afshar, M., Gharehchopogh, F.: Advances in artificial rabbits optimization: a comprehensive review. Arch. Comput. Methods Eng. **32**, 2113–2148 (2025). https://doi.org/10.1007/s11831-024-10202-7
13. Anka, F.: A novel hybrid metaheuristic method for efficient decentralized IoT network layouts. Internet Things **32**, 101612 (2025). https://doi.org/10.1016/j.iot.2025.101612
14. Abdollahzadeh, B., Khodadadi, N., Barshandeh, S., et al.: Puma optimizer (PO): a novel metaheuristic optimization algorithm and its application in machine learning. Clust. Comput. **27**, 5235–5283 (2024). https://doi.org/10.1007/s10586-023-04221-5
15. Linganathan, S., Singamsetty, P.: Genetic algorithm to the bi-objective multiple travelling salesman problem. Alex. Eng. J. **90**, 98–111 (2024). https://doi.org/10.1016/j.aej.2024.01.048
16. Jasim, A., Fourati, L.: Guided genetic algorithm for solving capacitated vehicle routing problem with unmanned-aerial-vehicles. IEEE Access **12**, 106333–106358 (2024). https://doi.org/10.1109/ACCESS.2024.3438079

17. Jolfaei, A., Alinaghian, M., Bahrami, R., Tirkolaee, E.: Generalized vehicle routing problem: contemporary trends and research directions. Heliyon **9**(12), e22733 (2023). https://doi.org/10.1016/j.heliyon.2023.e22733
18. Chen, W., Zhang, J., Chung, H., Zhong, W., Wu, W., Shi, Y.: A novel set-based particle swarm optimization method for discrete optimization problems. IEEE Trans. Evol. Comput. **14**(2), 278–300 (2010). https://doi.org/10.1109/TEVC.2009.2030331
19. Strasser, S., Goodman, R., Sheppard, J., Butcher, S.: A new discrete particle swarm optimization algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 53–60 (2016). https://doi.org/10.1145/2908812.2908935
20. Wang, S., Li, Y., Yang, H.: Self-adaptive mutation differential evolution algorithm based on particle swarm optimization. Appl. Soft Comput. **81**, 105496 (2019). https://doi.org/10.1016/j.asoc.2019.105496
21. Cinar, A., Korkmaz, S., Kiran, M.: A discrete tree-seed algorithm for solving symmetric traveling salesman problem. Eng. Sci. Technol. Int. J. **23**(4), 879–890 (2020). https://doi.org/10.1016/j.jestch.2019.11.005
22. Qian, W., Basili, R., Eshaghian-Wilner, M., Khokhar, A., Luecke, G., Vary, J.: Comparative study of variations in quantum approximate optimization algorithms for the traveling salesman problem. Entropy **25**(8), 1238 (2023). https://doi.org/10.3390/e25081238
23. Reddy, K., Panwar, L., Panigrahi, B., Kumar, R.: Binary whale optimization algorithm: a new metaheuristic approach for profit-based unit commitment problems in competitive electricity markets. Eng. Optim. **51**(3), 369–389 (2019). https://doi.org/10.1080/0305215X.2018.1463527
24. Li, Z., Li, N.: A novel multi-mutation binary particle swarm optimization for 0/1 knapsack problem. In: 2009 Chinese Control and Decision Conference, pp. 3042–3047 (2009). https://doi.org/10.1109/CCDC.2009.5192838
25. Li, Y., He, Y., Liu, X., Guo, X., Li, Z.: A novel discrete whale optimization algorithm for solving knapsack problems. Appl. Intell. **50**, 3350–3366 (2020). https://doi.org/10.1007/s10489-020-01722-3
26. Li, X., Zhang, S., Shao, P.: Discrete artificial bee colony algorithm with fixed neighborhood search for traveling salesman problem. Eng. Appl. Artif. Intell. **131**, 107816 (2024). https://doi.org/10.1016/j.engappai.2023.107816
27. SeyedOskouei, S., Sojoudizadeh, R., Milanchian, R., Azizian, H.: Shape and size optimization of truss structure by means of improved artificial rabbits optimization algorithm. Eng. Optim. (2024). https://doi.org/10.1080/0305215X.2024.2308577
28. Moharam, R., Ali, A., Morsy, E., Ahmed, M., Mostafa, M.: A discrete chimp optimization algorithm for minimizing tardy/lost penalties on a single machine scheduling problem. IEEE Access **10**, 52126–52138 (2022). https://doi.org/10.1109/ACCESS.2022.3174484
29. Hosseinnejad, R., Habibizad Navin, A., Rasouli Heikalabad, S., Derakhshanfard, N.: Combining 2-Opt and Inversion Neighborhood Searches with Discrete Gorilla Troops Optimization for the Traveling Salesman Problem, SSRN, pp. 1–19 (2024). https://doi.org/10.2139/ssrn.4980127
30. Panwar, K., Deep, K.: Discrete grey wolf optimizer for symmetric travelling salesman problem. Appl. Soft Comput. **105**, 107298 (2021). https://doi.org/10.1016/j.asoc.2021.107298
31. Arasteh, B., Sadegi, R., Arasteh, K., Gunes, P., Kiani, F., Torkamanian-Afshar, M.: A bioinspired discrete heuristic algorithm to generate the effective structural model of a program source code. J. King Saud Univ. Comput. Inf. Sci. **35**(8), 101655 (2023). https://doi.org/10.1016/j.jksuci.2023.101655
32. Uddin, F., Riaz, N., Manan, A., Mahmood, I., Song, O.Y., Malik, A.J., Abbasi, A.A.: An improvement to the 2-opt heuristic algorithm for approximation of optimal TSP tour. Appl. Sci. **13**(12), 7339 (2023). https://doi.org/10.3390/app13127339
33. Abdel-Basset, M., Mohamed, R., Sallam, K.M., Alrashdi, I., Hameed, I.A.: An efficient binary spider wasp optimizer for multi-dimensional knapsack instances: experimental validation and analysis. J. Big Data **12**(1), 18 (2025). https://doi.org/10.1186/s40537-024-01055-9
34. Sun, B., Li, G., Wang, S., Xie, C.: Two-dimensional bin-packing problem with conflicts and load balancing: a hybrid chaotic and evolutionary particle swarm optimization algorithm. Comput. Ind. Eng. **200**, 110851 (2025). https://doi.org/10.1016/j.cie.2024.110851
35. Halim, A.H., Ismail, I.: Combinatorial optimization: comparison of heuristic algorithms in travelling salesman problem. Arch. Comput. Methods Eng. **26**(2), 367–380 (2019). https://doi.org/10.1007/s11831-017-9247-y
36. https://github.com/farzadkiani2020/SampleDataSets.git
37. Mara, S.T.W., Norcahyo, R., Jodiawan, P., Lusiantoro, L., Rifai, A.P.: A survey of adaptive large neighborhood search algorithms and applications. Comput. Oper. Res. **146**, 105903 (2022)
38. Feo, T.A., Resende, M.G.: Greedy randomized adaptive search procedures. J. Glob. Optim. **6**(2), 109–133 (1995)
39. Hansen, P., Mladenović, N.: Variable neighborhood search: principles and applications. Eur. J. Oper. Res. **130**(3), 449–467 (2001)
40. Pirim, H., Eksioglu, B., Bayraktar, E.: Tabu Search: A Comparative Study, vol. 278. INTECH Open Access Publisher, London (2008)
41. Voudouris, C., Tsang, E.: Guided local search and its application to the traveling salesman problem. Eur. J. Oper. Res. **113**(2), 469–499 (1999). https://doi.org/10.1016/S0377-2217(98)00099-X

42. Tsplib, À.: A traveling salesman problem library. Eur. J. Oper. Res. **3**, 376–384 (1991). https://doi.org/10.1287/ijoc.3.4.376
43. Tirkolaee, E.B., Cakmak, E., Karadayi-Usta, S.: Traveling salesman problem with drone and bicycle: multimodal last-mile e-mobility. Int. Trans. Oper. Res. **32**(6), 3232–3258 (2025). https://doi.org/10.1111/itor.13452
44. Ohanu, C.P., Rufai, S.A., Oluchi, U.C.: A comprehensive review of recent developments in smart grid through renewable energy resources integration. Heliyon **10**(3), e25705 (2024). https://doi.org/10.1016/j.heliyon.2024.e25705
45. Lohmer, J., Lasch, R.: Production planning and scheduling in multi-factory production networks: a systematic literature review. Int. J. Prod. Res. **59**(7), 2028–2054 (2021). https://doi.org/10.1080/00207543.2020.1797207
46. Taillard, E.D.: A Short list of combinatorial optimization problems. In: Design of Heuristic Algorithms for Hard Optimization: With Python Codes for the Travelling Salesman Problem, pp. 31–61 (2023). https://doi.org/10.1007/978-3-031-13714-3_2
47. Braekers, K., Ramaekers, K., Van Nieuwenhuyse, I.: The vehicle routing problem: state of the art classification and review. Comput. Ind. Eng. **99**, 300–313 (2016). https://doi.org/10.1016/j.cie.2015.12.007
48. Lam, A., Leung, Y., Chu, X.: Electric vehicle charging station placement: formulation, complexity, and solutions. IEEE Trans. Smart Grid **5**(6), 2846–2856 (2014). https://doi.org/10.1109/TSG.2014.2344684
49. Hammami, F.: An efficient hybrid adaptive large neighborhood search method for the capacitated team orienteering problem. Expert Syst. Appl. **249**, 123561 (2024). https://doi.org/10.1016/j.eswa.2024.123561
50. Tsakirakis, E., Marinaki, M., Marinakis, Y., Matsatsinis, N.: A similarity hybrid harmony search algorithm for the team orienteering problem. Appl. Soft Comput. **80**, 776–796 (2019). https://doi.org/10.1016/j.asoc.2019.04.038
51. Lin, S.W.: Solving the team orienteering problem using effective multi-start simulated annealing. Appl. Soft Comput. **13**(2), 1064–1073 (2013). https://doi.org/10.1016/j.asoc.2012.09.022
52. Roozbeh, I., Hearne, J.W., Pahlevani, D.: A solution approach to the orienteering problem with time windows and synchronisation constraints. Heliyon **6**(6), e04202 (2020). https://doi.org/10.1016/j.heliyon.2020.e04202
53. Chao, I.M., Golden, B.L., Wasil, E.A.: The team orienteering problem. Eur. J. Oper. Res. **88**(3), 464–474 (1996)
54. Kiani, F., Saraç, Ö.F.: A novel intelligent traffic recovery model for emergency vehicles based on context-aware reinforcement learning. Inf. Sci. **619**, 288–309 (2023). https://doi.org/10.1016/j.ins.2022.11.057
55. DehAbadi, E., Lanjanian, H., et al.: Analyzing EEG data during opium addiction treatment using a fuzzy logic-based machine learning model. Front. Psychiatry **16**, 1635933 (2025). https://doi.org/10.3389/fpsyt.2025.1635933

Authors and Affiliations

Ferzat Anka[1] · Farhad Soleimanian Gharehchopogh[2] · Seyed Jalaleddin Mousavirad[3] · Ghanshyam G. Tejani[4,5]

✉ Seyed Jalaleddin Mousavirad
Seyedjalaleddin.mousavirad@miun.se

✉ Ghanshyam G. Tejani
p.shyam23@gmail.com

Ferzat Anka
fanka@fsm.edu.tr

Farhad Soleimanian Gharehchopogh
farhad.soleimanian@iau.ac.ir

[1] Data Science Application and Research Center (VEBIM), Fatih Sultan Mehmet Vakif University, Istanbul, Turkey

[2] Department of Computer Engineering, Ur., C., Islamic Azad University, Urmia, Iran

[3] Department of Computer and Electrical Engineering, Mid Sweden University, Sundsvall, Sweden

[4] Department of Research Analytics, Saveetha Dental College and Hospitals, Saveetha Institute of Medical and Technical Sciences, Saveetha University, Chennai 600077, India

[5] Department of Industrial Engineering and Management, Yuan Ze University, Taoyuan 320315, Taiwan