Secretary Bird Optimization Algorithm for the Vehicle Routing Problem with Time Windows

Anmar Abuhamdah, Ahmad Althunibat, Malek Alzaqebah, and Ghaith Jaradat

Abstract— **The Vehicle Routing Problem with Time Windows (VRPTW) is a commonly encountered problem in logistics and transportation that involves combinatorial optimization. In this study, we apply the Secretary Bird Optimization Algorithm (SBOA) to solve instances of the VRPTW by utilizing its hunting and evasion mechanisms to improve solution quality. The proposed algorithm was evaluated on Solomon's benchmark instances, which include small-, medium-, and large-scale problem instances. Experimental results confirmed the effectiveness of SBOA, obtaining 27 optimal solutions for different instances with 25 customers, 26 optimal solutions for the datasets with 50 customers and 39 optimal solutions for the datasets with 100 customers. The results demonstrate the algorithm's exceptional ability to reduce total travel distance and vehicle usage while satisfying time window constraints. It demonstrated competitive convergence behavior and a high success rate in finding optimal solutions in compared to other existing metaheuristic approaches. The results demonstrate that SBOA is an effective optimization algorithm for large-scale VRPTW under strict constraints.**

Index Terms— **Secretary Bird Algorithm, Vehicle Routing Problem, Metaheuristic, Optimization, Time Constraints.**

I. Introduction

The Vehicle Routing Problem with Time Windows (VRPTW) has been widely studied in the area of logistics and transportation. It generalizes the traditional Vehicle Routing Problem (VRP) with time window constraint, so that it becomes more suitable to the time constraint in practical delivery [1], [2], and [3]. However, solving the VRPTW is NP-hard, and exact methods become infeasible for large-sized problems, which leads to the application of metaheuristic algorithms such as Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Genetic Algorithms (GA), Simulated Annealing (SA), and Tabu Search (TS) [3], [4], [5], [6], [7], and [8]. In this context, SBOA is a new metaheuristic inspired by the hunting behavior of the secretary bird, which forms the basis of the algorithm design. SBOA was introduced by Huohuo and mimics the secretary bird's behavior to navigate through a complex search space using its strategic hunting techniques [9]. More encouraging results thus far include its ability to solve a variety of global optimization problems significantly better than existing approaches [10] and [11].

In recent years, there are many biologically based algorithms have been proposed such as Whale Optimization Algorithms (WOA), Salp Swarm Algorithm (SSA), and Secretary Bird Optimization Algorithms (SBOA), the excellent performance between exploration and exploitation in complex search space has been obtained [7], [8], [9], and [10]. Due to the requirement of efficient routing with strict operational constraints, more and more attention has been focused on adapting new metaheuristics to solve VRPTW efficiently, such as SBOA [1], [2], [3], [9], and [11].

VRPTW solving at scale involves several challenges such as scalability of the solution, sensitivity to constraint, and convergence behavior [3], [8]. For large-scale cases, they can generate the rising computational cost and traditional metaheuristics are vulnerable to promote premature convergence or inefficient in exploring the feasible regions [11] and [12]. Besides, to keep solution feasible, the balancing of demand within a rigid time window is also one of key challenges, particularly when customers are geographically dispersed over time [1], [2], and [3]. The specific contributions of this paper are also to overcome these weaknesses by improving the global search ability and constraint-handling mechanisms of the SBOA algorithm.

The main aim of this paper is to investigate how the effectiveness of SBOA can be measured based on different sizes of VRPTW, the total travel distance and the vehicle usage for VRPTW cases where time window constraints must be considered.

In order to analyze the effectiveness of the proposed SBOA, we test it on the Solomon's benchmark instances adopted in VRPTW literature, that are well-accepted sets of instances, and that encompasses small-, medium-, and large-scale instances [1]. This allows a thorough evaluation of the scalability, effectiveness, and performance at providing closest to the best or best solution under different conditions of the algorithm while comparing it to existing well-established metaheuristic methods.

This paper pertains to the area of combinatorial optimization and presents an adapted SBOA for solving the VRPTW. Key contributions consist in introducing an error penalization scheme when constraint boundaries are

Manuscript received April 14, 2025; revised August 28, 2025.

Anmar Abuhamdah is an associate professor of College of Business Administration, Taibah University, Al Madinah Al Munawwarah, 42353, Saudi Arabia (corresponding author to provide phone: +966-5544573703; e-mail: aabuhamdah@taibahu.edu.sa).

Ahmad Althunibat is an associate professor of Faculty of Science and Information Technology, Al-Zaytoonah University of Jordan, Amman, 11183, Jordan (e-mail: a.thunibat@zuj.edu.jo).

Malek Alzaqebah is an assistant professor of College of Science, Imam Abdulrahman Bin Faisal University, Dammam, 31451, Saudi Arabia (e-mail: maafehaid@iau.edu.sa).

Ghaith Jaradat is an associate professor of College of Computer Sciences and Informatics, Amman Arab University, Amman, 19111, Jordan (e-mail: g.jaradat@aau.edu.jo).

exceeded, and in better local search strategies, together with extensive testing on small, medium and large scale VRPTW instances of Solomon's benchmarks. Moreover, algorithm achieves competitive results with existing metaheuristics and shows good convergence rate and improves quality of solution, mainly in terms of reducing travel distance and vehicle count while maintaining feasibility.

The rest is organized as follows: Section 2 lays out the related work, followed by Section 3 which explains the approach we propose for the methodology, Section 4 presents results, Section 5 discussion of the result, and finally section 6 concludes the paper with the future work.

II. Related work

The VRPTW is a classic combinatorial optimization problem that has received tremendous interest in the logistics and transportation literature. The VRPTW adds a dimension to the classical VRP: customers must be served by vehicles within time intervals. These added constraints increase the complexity of the problem, simulating more realistic situations and localized deliveries that are time-sensitive relative to customer experience and operational efficiency [1]. Since then, many optimization techniques have been proposed, for instance different exact algorithms or heuristics and metaheuristics to solve the VRPTW.

Several optimization techniques for VRPTW (e.g., branch-and-bound and branch-and-cut), have been developed and used to solve small-scale VRPTW instances. While these approaches ensure an optimal solution, the high computational intensity of MIP for large problems makes them practically infeasible [6]. To solve this limitation, researchers have gradually shifted towards heuristic and metaheuristic methods which yield near-optimal solutions fast. Some popular meta-heuristic algorithms are PSO, ACO, GA, SA and TS. The VRPTW problem formulation has shown strong performance using these methods for balancing exploration and exploitation in the search space of solutions [7]. These methods laid the foundation for solving VRPTW but they face limitations in scalability, constraint handling, and parameter sensitivity when applied to large-scale or tightly constrained scenarios.

Over the past years, the bio inspired algorithms have become popular in solving complex optimization problems including the VRPTW. For example, natural phenomena, such as swarm behavior and foraging strategies, have been adapted to solve VRPTW by using an improved SSA [12] or PSO [13]. Similarly, metaheuristic algorithms that perform at a higher-level search space such as Grey Wolf Optimizer (GWO) demonstrated promising results in minimizing total travel distance and the number of vehicles while satisfying time window constraints [14]. In the last years several recursive non-hypolixelic improvements impelled to bio-inspired algorithms have demonstrated promising solutions to the VRPTW in different application contexts. Hybrid and enhanced metaheuristic models have been investigated by few researchers to solve VRPTW, for enhancing the solution quality, convergence rate and tackling the constraints.

PSO was used by Gong et al. in [13] to VRPTW, with remarkable performance in the minimization of total traveling distance. However, the algorithm prematurely converged while it was tested in complex cases, and its performance was unsatisfactory for real problems.

In [14], GWO was proposed by Mirjalili et al. for solving VRPTW, GWO showed more promising exploration over PSO and GA, especially on medium dimensional problems. It did not perform as well as for large freight as it did in the constrained problem with freight of that size, and its performance also has a slight downhill tendency under the tight constraints in time windows, which means that constraints handling in further research could be improved.

Yodgangjai and Malampong suggested an improved WOA for VRPTW in [15]. The modified solution handled the convergence speed of output facility better, but optimal exploration-exploitation selection was still not reached, especially in dynamic environments with fluctuation on customer demands and time window.

An improved ACO algorithm that includes mutation and local search operations was presented to optimize routing in a recent study [16]. It was shown that this hybridization led to considerable reductions in travel distance while still maintaining feasibility movements under time pressure. Besides, the parameter tuning was tedious to optimize the performance and not easily used in practice.

Jung and Moon in [17] applied GA to solve VRPTW and obtained competitive solutions on Solomon's datasets—mostly for small sized instances. Although GA was good at producing solutions that were likely to be possible routes, it suffered from slow convergence as the problem size grew, rendering it less viable in real time scenarios.

A TS with Density Peak Clustering (DPC/TS) for mitigating scalability problems of classic heuristics for VRPTW was proposed by study [18]. By using DPC to cluster customer locations, the original problem was divided into smaller size sub-problems which were effectively solved using TS. Numerical tests demonstrate that the new MLP method performs much better in quality of solutions and efficiency than the classical algorithms.

Another recent study in [19] introduced a SA method for minimizing operational costs for the fuel, driver wage and delivery time, where also time window and capacity were considered. The method was able to solve large-scale instances with up to 350 nodes, obtaining zero constraint violations in less than ten minutes, and outperforming current methods.

In [20], Ahmed et al. proposed the Modified Football Game Algorithm (MFGA) which had good performance with the benchmark example set of Solomon. MFGA performed well in handling large time window constraints and scalability. More experiments, however, are needed under a variety of problem sets to ascertain the robustness and generality of the method.

The hybrid Salp Swarm Algorithm (SFSSA) in [21] performed better than many single metaheuristics in solution quality and convergence speed. The implementation complexity of SFSSA is high, and few practical logistics can be applied to the real-time and large-scale scheme.

A more recent approach was developed in [22], by combining the Large Neighborhood Search with a Modified

Rat Swarm Optimization (MRSO) to solve the VRPTW with the well-known benchmark Solomon instances. The existing best-known solutions were improved by the MRSO by 5.1% for the R2 category and by 8.8% for the RC2 category. However, the strength of the algorithm depends largely on having a good tradeoff between global exploration and local exploitation.

Lastly, in [12], an enhanced SSA was proposed by Cai and Chen. The advanced SSA had superior global search properties, especially in the early stages. Yet, it did not include efficient ways to deal with constraint violations, and so its applicability for very tight CVRPD instances is limited.

These new algorithms are more adaptable, possess better global search performance and resist local convergence for the traditional algorithms but their overall performance on parameter tuning, constraint handling and dynamic response have not been yet solved.

In view of the complexity and dynamic of VRPTW, particularly in large-scale problems, traditional metaheuristics often fail to balance exploration and exploitation well. Hence, bio-inspired techniques, which escape the limitations of previously traditional methods by emulating nature's natural phenomena for organic search space solutions, are even more desired. A good promising nature inspired approach is the SBOA, which shows good global search and adaptability. In the next section, we will introduce SBOA and its possible use as a VRPTW solver.

New metaheuristic algorithms have trailed enhancing this capability to solve difficult VRPTW cases. Hybrid methods such as MISBOA and SFSSA achieve better convergence and results. Yet, many methods require improvement in terms of scalability, handling of the constraints and sensitivity to the parameters. Although classical algorithms such as Tabu Search and Genetic Algorithms formed the building blocks, the more recent bio-inspired algorithms such as GWO, WOA, and SBOA offer better exploration ability. However, despite these advances there is a great demand for adaptive constraint handling and dynamic response strategies to deal with sudden changes in customer demand or traffic conditions.

However, there remains a need for advanced algorithms capable of solving large-scale VRPTW instances with tighter time window constraints. The SBOA is a recently introduced algorithm inspired by the predatory and escapable behaviors of secretary birds. SBOA has shown promise in addressing complex optimization challenges, especially in cases where dynamic modifications and adaptive strategies are necessary [9], [10], and [11]. Although it is a crucial optimization algorithms in practice, few studies in the literature have applied it to VRPTW, so this study makes a significant contribution to the existing literature.

For performance evaluation of SBOA, this study uses Solomon's benchmark instances, which are known as the benchmark instances for VRPTW algorithms. These instances include a wide range of sizes and complexities, allowing for an in-depth evaluation of the algorithm's performance [8]. By contrasting the performance of SBOA with the state of the art in the field of metaheuristics, the purpose of this work is to illustrate its capability to find optimal or near-optimal solutions with low computational costs.

The presented SBOA in this work extends these results by adding a penalty function and new search rules, and it is envisioned to overcome some of the weaknesses found in other methods.

In brief, prior work demonstrates success of different metaheuristics for VRPTW. Exact techniques are bounded to small instances by computational complexity, and classical metaheuristics usually fail to achieve a tradeoff between exploration and exploitation. Bio-inspired approaches have quickly become effective alternatives, with increased flexibility and better global search abilities. But this type of algorithms exists with little effective constraints and have difficulty in dealing with large scale and highly constraints.

It should be mentioned that although SBOA has been successfully used to different combinatorial optimization problems including function optimization, engineering design, scheduling related problems. This work tries to fill this gap by proposing an improved variant of SBOA exclusively designed to solve VRPTW. The tuned algorithm includes the penalty-type constraints management as well as enhanced local search operations with respect to time windows and vehicle routing characteristics.

III. Method

The SBOA is a new metaheuristic algorithm that is inspired by the hunting and evasion behaviors of the secretary bird. Huohuo introduces this algorithm aims to ameliorate global optimization problems by incorporating strategies derived from both natural predation and evasive strategies of the birds. SBOA consists of three main components that are described as follows: the initialization, which generates initial solution population randomly, the hunting strategy, which is an imitation of the chasing style of secretary bird for prey and makes the algorithm good at exploring solution space, and the evasion mechanism, which simulates the escape behavior of the bird and can improve the local exploitation. Moreover, a penalty scheme is incorporated to treat constraint violations by penalizing the infeasible solutions while searching [9]. In the initialization phase, randomly generate an initial population of solutions, which are the hiding spots of the birds. In the hunting phase, the angle of movement in this stage of birds is updated using strategies from the economy of a secretary bird concerning the hunting of prey, which enables an efficient searching of the search space. In the evasion phase, the birds apply evade strategies based on the principle of their natural defenses, which enhances the exploitation capability of the proposed algorithm to find the best available solution [10].

The general pseudo-code of the SBOA is presented in Fig. 1, where the three main phases, namely initialization, hunting strategy, and evasion mechanism, are described. A detailed description of each phase and their integration to solve the VRPTW are given in the next sections. For the exact nature of the algorithm and the update mechanisms,

the reader is referred to the original SBOA paper [9], [11].

To make the proposed SBOA more transparent and reproducible, we present the mathematical equations of the position update and the constraint penalty we consider in the present study.

If the activity time is not satisfied at iteration t, then the location of each secretary bird *i* is renewed at iteration *t* using its present location and inspired search modes that are motivated by hunting and avoidance behaviors. The update equation during the hunting mechanism is given in Equation 1:

$$X_i^{t+1} = X_i^t + \alpha \times r1 \times (X_{best}^t - X_i^t) \quad (1)$$

Where X_i^t is the current position of bird *i* , X_{best}^t is the best solution found so far, α is a scaling factor (set to 0.5 in our experiments), and r1 is a random number uniformly distributed in [0,1].

In evasion, birds move in response to opponents to avoid local optima according to the diversity preservation mechanism represented by Equation 2:

$$X_i^{t+1} = X_i^t + \beta \times (r2 \times X_{rand}^t - r3 \times X_i^t) \quad (2)$$

Where X_{rand}^t is a randomly selected solution from the population, β is an evasive step size coefficient (set to 0.3), and *r2*, *r3* are random numbers in [0,1].

In terms of constraint management, a time window violation may be penalized using a dynamic penalty function. Equation 3 describes the fitness function:

$$Fitness(S) = TotalDis(S) + \lambda \times \sum_{j=1}^{n} \max(0, Aj - Lj) \quad (3)$$

Where S is a solution, *Aj* is the arrival time at customer *j* , *Lj* is the latest allowable service time, *n* is the total number of customers, and λ is a dynamically adjusted penalty coefficient that increases with the number of violated constraints. This ensures that infeasible solutions are gradually pushed toward feasibility while maintaining search diversity.

Fig. 1 presents the pseudo-code of the SBOA along with a detailed description of its process. The SBOA is an evolutionary algorithm, which creatively works to arrive at the end, iteratively optimizing towards the solution of a given task, which can be broken down into three stages - initialization, optimization loop and termination.

As shown in Fig.1, in the initialization phase, a random population of N secretary birds are spread throughout the search space. Birds in this analogy in fact represent the potential solutions of the optimization problem, where its position is analogous to a candidate solution. Each bird's fitness function is then evaluated to measure the quality of its present position. The evaluation acts as a benchmark to help gauge the effectiveness of the solution, and provides a standard against which future improvements can be measured.

```
Procedure Secretary Bird Optimization Algorithm
Initialize population of N secretary birds randomly within search
    space
Evaluate fitness of each bird
Repeat
 FOR each bird i in the population DO:
   Update position using hunting strategy
      Evaluate new position fitness
      IF fitness improved THEN
        Accept new position
      ELSE
        Apply evasion strategy to escape local optima
        Evaluate new position fitness
        IF fitness improved THEN
          Accept new position
        END IF
      END IF
 END FOR
 Update the best solution found so far
Until the termination criteria is met
  Return the best solution found;
```

Fig. 1. SBOA Optimization pseudo-code

The process of iteratively improving a population of solutions is called the optimization loop, which constitutes the core of the algorithm. This continues looping until a defined termination condition is reached, for example, a maximum number of iterations or an adequately optimal solution. In each iteration, each bird modifies its location based on the hunting strategy (finding others to hunt) and the evasion strategy (finding ways to escape). The hunting strategy is inspired by wild hunting behavior of secretaries, which directs secretary birds towards likely regions in the search space. This mechanism reinforces exploration by allowing the algorithm to explore different areas and avoid converging too early. Otherwise, the evasion strategy is used when a bird does not gain fitness from hunting. This approach provides a way to escape from local optima, which ultimately improves the algorithm's search capability for other states. The best solution found so far is updated during the whole process to ensure that the global optimum is tracked.

While the basic SBOA has been found to be effective for solving difficult optimization problems, a number of improved versions have been proposed to enhance its efficiency and convergence property. There are a few other modified versions of this algorithm as well, such as Multi-Strategy Improvement Secretary Bird Optimization Algorithm (MISBOA), which incorporates several other advanced mechanisms such as the incremental PID feedback control, golden sine based guidance during foraging, collaborative camouflage, and cosmic similarity based position updates during evasion. Simulation results on the Congress on Evolutionary Computation (CEC2022) benchmark problems have demonstrated that MISBOA exhibits superior search performance compared to other metaheuristic algorithms and is competitive particularly when used for higher-dimensional optimization problems [10]. One other approach that yielding promising results is the Quantum-Based Secretary Bird Optimization Algorithm (QSBOA) [11], where quantum-inspired search techniques

are coupled with adaptive dynamic and hybrid swarm intelligence properties. Some research has shown that QSBOA converges more quickly and is more robust than standard metaheuristic approaches, which makes it useful for solving very complex optimization problems. [11].

Despite those improvements, the basic SBOA is still a powerful and effective algorithm especially for real-world problems such as the VRPTW. While improved versions such as MISBOA and QSBOA exhibit better convergence characteristics and accuracy, they tends to increase algorithmic complexity, higher computational load, and reliance on parameter dependence, respectively. Consequently, although the improved variants (e.g., MISBOA and QSBOA) potentially provide better performance in terms of convergence speed and quality convergence (which is expected), the original SBOA was adopted in this study mainly because of the simplicity, the ease of implementation, the good trade-off between the exploration and the exploitation, the proved effectiveness on large-scale VRPTW instances, and the lower computational complexity than the crafter variants.

The VRPTW, based on Solomon's benchmark, involves finding the optimal routes for a fleet of vehicles under time constraints. In this study, the existing SBOA is modified for this problem with hunting based on distance and evasion based on time window. Fig. 2 provides pseudo-code designed specifically to solve the VRP with the SBOA framework. This pseudo-code describes how the SBOA is modified to solve VRP, a canonical combinatorial optimization problem that consists of finding the best routes for a fleet of vehicles to deliver to a set of customers. VRP-specific constraints like vehicle capacity and route distance constraints are inherently integrated into the design of the algorithm. The algorithm iteratively polishes solutions, mapping the secretary birds' positions to potential routes and minimizing total travel costs while respecting operational constraints. Problem specific fitness evaluations, trained on problem specific data, allow the algorithm to efficiently discover high-quality solutions for real-world logistics problems. Through this adaptation, Fig. 2 takes advantage of the flexibility of the SBOA to tackle complex optimization approaches such as VRPTW with accuracy and efficiency.

Fig. 2 presents the pseudo-code of the SBOA for VRPTW, along with a full description of the corresponding procedural steps. In the case of VRPTW, each bird (agent) in the algorithm denotes a potential solution comprised of a set of vehicle routes. Such routes are conditioned by operational constraints (e.g., time window limits, vehicle capacity limits) and the need to minimize the total travel distance. This representation allows the algorithm to both stay within the confines of the problem and search for optimal or sub-optimal solutions. The quality of each solution is evaluated using a fitness function. Each of the solutions is evaluated on 3 key criteria:

1) Total distance traveled, the algorithm tends to minimize the overall distance traveled, and considering that directly affects operational effectiveness.
2) Number of vehicles used, the solutions with a smaller fleet are preferred; less fleet means smarter savings.

```
Procedure Secretary Bird Algorithm for VRPTW
Initialize a population of N secretary birds (each representing a set of vehicle
    routes)
Evaluate the fitness of each solution based on:
  - Total traveled distance
  - Number of vehicles used
  - Time window constraints Evaluate fitness of each bird
Repeat
 FOR each bird i in the population DO:
  # Hunting Strategy: Optimize route based on distance minimization
      Select a leader bird (best solution so far)
      Move towards leader using a guided perturbation mechanism
      Repair route if time windows are violated
      Evaluate new solution fitness
      IF fitness improved THEN
        Accept new route configuration
      ELSE
        # Evasion Strategy: Escape local optima by diversifying search
        Apply time-window-based reordering of customer visits
        Perform randomized customer swaps between routes
        Evaluate new solution fitness
        IF fitness improved THEN
          Accept new configuration
        END IF
      END IF
  END FOR
  Update the best solution found so far
Until the termination criteria is met
  Return the best solution found;
```

Fig. 2. SBOA pseudo-code for VRPTW

3) Penalties for constraint violations, where the routes that do not comply with the time window constraints, are penalized to motivate meeting any customer-specific requirements. The penalty mechanism promotes feasible solutions and is very effective in guiding the search to feasible and high-quality solutions.

While in the exploration step, each bird uses a controlled perturbation mechanism that propels it towards the leader (which is the best solution found so far). Such motion emulates hunting techniques used by secretary birds that allows the algorithm exploring the landscape of the/optimal search space. Routes are then refined in a series of iterations to minimize travel distance without affecting feasibility. A repair mechanism is applied if the adjusted solution violates any time window constraints to ensure the solution remains valid.

If the exploration phase fails to show any improvements, the algorithm shifts to the exploitation phase by diversifying the search. That is done via two primary mechanisms:

1) Time-Window-Based Reordering: The order of customer visits is changed to be more suitable for time window constraints and to improve solution feasibility.
2) Randomizing Customers: We swap customers randomly so that we can try out alternate configurations to discover potentially better solutions. These mechanisms allow the algorithm to escape from local optima and stop when further improvements can no longer be made.

The behavior of metaheuristic solvers such as SBOA depends to a great extent on the choice of hyperparameters. The next procedure was to identify the important parameters of SBOA parameters for solving VRPTW, which was considered in this study.

- The number of individuals was defined as 50 after experimenting with part of Solomon's reference set (C101,

R101, and RC101). This value resulted in a fine compromise between exploration ability and computational cost.

- A dynamic adaptive approach was used, where searching began with a greater emphasis on hunting (exploration), then progressively shifts toward evasion-based exploitation, where the algorithm improves promising solutions to avoid premature convergence. This balance was adjustable based on the number of iterations with a factor and is formulated as an equation below.

To this effect the settings were checked on a subset of small-scale and medium scale instances and then applied to large problems. The fine-tuning of parameters is performed to make the algorithm keep a tradeoff between exploration and exploitation and computation efficiency.

During the optimization process, the algorithm keeps a record of the best VRPTW solution that is found at that moment. Such solutions are dynamically updated when a better candidate is found, the algorithm always actually has the best quality result.

The algorithm is executed until at least one of two criteria is met achieving the maximum number of iterations, or there are no improvements noticed over a previously defined number of iterations. This allows the algorithm to function correctly without introducing additional compute time from poor hypotheses in the last few iterations. These termination conditions support the algorithm to explore the solution space comprehensively without causing computational overload.

SBOA can be improved through a variety of advanced strategies to better address the VRPTW to overcome the limitations of the algorithm. The detailed features proposed are:

- First, SBOA can be integrated with local search methods to substantially enhance the quality of the routes generated. Neighborhood structures specific to 2-opt, 3-opt, or for example, TS can be included in the framework to refine the solutions discovered and useful for getting rid of local inefficiencies. Thus, hybridization is created, and it acts as a powerful combination of global optimization and local search that performs excellent results in creating optimized modified routing plans.
- Second, adaptive escape mechanisms based on the diversity of solutions can be introduced to avoid premature convergence. If it notices that the population of solutions has become homogeneous, it can direct the search to explore new areas of solution space, while if diversity is too high, and it can exploit these well-performing solutions by focusing on them. It makes sure that the Algorithm continues to search over all the regions of the solution space, thus limiting the chance of stopping on sub-optimal solutions.
- Finally, dynamic penalty adjustments for time window violations help to balance exploration versus exploitation of the algorithm. The algorithm reach a balance between quality and feasibility by adaptively adjusting the penalty weights attached to constraint violations during the optimization process. When the solution is good enough, the focus move toward feasibility than further optimization. Such an adaptive strategy allows the algorithm to adjust to the changing nature of the solution landscape over time.

This results in a SBOA framework that is more efficient and effective, allowing it to address the complexities of the VRPTW with increased accuracy and reliability. With these three techniques, the algorithm can perform better, providing solutions that are able to run in practice and cost-efficient.

To assess SBOA performance, we utilized Solomon's benchmark instances, a well-established and widely accepted evaluation methodology for the VRPTW [1]. These instances comprise a wide range of problems from small, to medium, to large, allowing for the evaluation of the algorithm in terms of scalability and robustness. In each instance, a set of key parameters are provided including the number of customers, customer locations (x, y coordinates), time windows (earliest time service can start, latest time service can start), customer demands, vehicle maximum capacities and depot locations. The dataset was represented by C-type, R-type and RC-type customers. Using these benchmark instances enables this study to provide a fair comparison against existing metaheuristic approaches and show SBOA's capability in dealing with complex VRPTW constraints effectively. The Fig. 3 depicts the classification of Solomon's benchmark instances for the Vehicle VRPTW. The datasets are classified into three major types (C-type, R-type, and RC-type) and several sub-types based on distribution patterns and the level of complexity of problems.

Dataset Type	Sub-Type	Number of Instances	Customer Distribution	Distribution Characteristics
C-type (Clustered)	**C1-type**	9	Customers are grouped in specific regions	Homogeneous distribution within defined areas
	C2-type	8	Customers are grouped in specific regions	Larger time windows and higher vehicle capacity
R-type (Random)	**R1-type**	12	Customers are randomly distributed	No specific pattern in distribution
	R2-type	11	Customers are randomly distributed	Larger time windows and higher vehicle capacity
RC-type (Random-Clustered)	**RC1-type**	8	A mix of clustered and randomly distributed customers	Combination of structured and random distribution
	RC2-type	8	A mix of clustered and randomly distributed customers	Larger time windows and higher vehicle capacity

Fig. 3. Solomon's benchmark for VRPTW

Fig. 3 shows the classification of Solomon's test instances for VRPTW. Three primary types exist in the dataset: C-type (Clustered), R-type (Random), and RC-type (Random-Clustered); each of which is further divided according to spatial distribution characteristics and the complexity of the problem. The following instances were generated of different size 25 customers, 50 customers and 100 customers which is related to number of delivery points or demand location considered for the instance. The 25-customer and 50-customer instances represent small-scale and medium-scale problems respectively, while the 100-customer instances represent the large-scale of more complex routing problems, respectively, providing an

overall evaluation of the performance of the algorithms in our testing over a variety of routes.

For the reproducibility and transparency of experimental results, we present the following implementations and pre-processing of our datasets:

- All experiments were performed with the same random seed to guarantee reproducibility among different runs. This permits a direct comparison between different trials of the same plant extract and it guarantees an exact reproducibility of results.
- The Solomon instances used in this paper are in the standard VRPTW instance format, which indicates x-y coordinates, time window [earliest arrival, latest departure] and the start node and end node of each customer. All minor preprocessing was done and the raw input files are parsed to arrays of structured data, which are input to the algorithm. Note that customer location coordinates and TWs were employed unchanged without transformation.
- Both vehicle capacity and customer demand were used in their raw form without normalization or rescaling to retain the real-world nature of the problem. During the search, all constraint violations such as over capacity and out of the time windows were penalized via a penalty term added in to the fitness function in order to direct the search towards feasible solutions only.

These are included to increase the clarity, reproducibility and the scientific rigor of the method development.

IV. Experiments

The SBOA was first tested on 56 Solomon's benchmark instances to solve the VRP with Time Windows (VRPTW) in this study. These examples include 25, 50, and 100 customer problems, which are standard in the assessment of routing algorithms [1]. To highlight its stability, the algorithm was run 10 times for each dataset employed, with stopping criteria fixed at 1,000, 10,000 and 30,000 iterations for small, medium, and large problems, respectively. These parameters are consistent with recommendations from the literature and prior studies [15]. The experiments were performed on a computer with i5-4570 CPU @ 3.20 GHz and 8 GB RAM, while the algorithms were implemented and executed using Python.

The results of the SBOA using the VRPTW datasets are shown in Table I, Table II, and Table III. In particular, Table I refers to the results of instances with 25 customers, Table II to those with 50 customers and Table III to those with 100 customers. The Best Known solution (BKS) in all tables are also provided with respective references. The best results obtained by the SBOA algorithm are marked in boldface showing results that are either equal or closer to the optimal or BKS solutions. The reason for this arrangement is to make a simple and direct comparison between the SBOA results and the benchmark solutions possible without too much effort on your part to determine the efficacy of the algorithm.

The computational results of the SBOA applying it to the VRPTW for instances with 25 customers are shown in Table I. The table reports the solutions of the SBOA algorithm against BKS in the literature. Results of the SBOA algorithm that are either equal to or very close to the BKS are in boldface. Currently, of the 56 results in Table I, the SBOA algorithm equal or outperformed the BKS 27 times (around 48.2% of the cases). That is, the SBOA algorithm is capable of providing high-quality solutions for small-scale problems, as able to generate the high quality competitive solutions compared with the BKS results in the literature.

TABLE I
Computational Result Of The SBOA Using Solomon's Benchmark For 25 Customers

instances	SBOA	BKS
C101	**191.3**	191.3 [M1]
C102	**190.3**	190.3 [M1]
C103	**190.3**	190.3 [M1]
C104	187.45	186.9 [M1]
C105	**191.3**	191.3 [M1]
C106	**191.3**	191.3 [M1]
C107	**191.3**	191.3 [M1]
C108	**191.3**	191.3 [M1]
C109	**191.3**	191.3 [M1]
C201	**214.7**	214.7 [M2, M6]
C202	**214.7**	214.7 [M2, M6]
C203	**214.7**	214.7 [M2, M6]
C204	**213.1**	213.1 [M2, M3]
C205	**214.7**	214.7 [M2, M6]
C206	**214.7**	214.7 [M2, M6]
C207	**214.5**	214.5 [M2, M6]
C208	**214.5**	214.5 [M2, M6]
R101	618.33	617.1 [M1]
R102	548.11	547.1 [M1]
R103	464.83	454.6 [M1]
R104	**416.9**	416.9 [M1]
R105	**530.5**	530.5 [M1]
R106	467.85	465.4 [M1]
R107	**424.3**	424.3 [M1]
R108	398.29	397.3 [M1]
R109	450.26	441.3 [M1]
R110	**444.1**	444.1 [M1]
R111	431.12	428.8 [M1]
R112	**393**	393 [M1]
R201	405.48	404.6 [M2, M3]
R202	352.80	350.9 [M3]
R203	476.96	461.1 [M1]
R204	401.79	351.8 [M1]
R205	**332.8**	332.8 [M1]
R206	307.14	306.6 [M1]
R207	418.52	411.3 [M1]
R208	347.31	345.5 [M1]
R209	298.95	298.3 [M1]
R210	294.99	294.5 [M1]
R211	**360.2**	360.2 [M2, M6]
RC101	338.82	338.0 [M2, M3]
RC102	327.69	326.9 [M4, M5]
RC103	300.23	299.7 [M5]
RC104	**338.0**	338.0 [M3, M5]
RC105	325.10	324.0 [M3]
RC106	298.95	298.3 [M3]
RC107	269.57	269.1 [M5]
RC108	**598.1**	598.1 [M1]
RC201	**684.8**	684.8 [M2, M5]
RC202	622.84	613.6 [M4, M5]
RC203	635.42	555.3 [M4, M5]
RC204	**444.2**	444.2 [M7]
RC205	683.79	630.2 [M4, M5]
RC206	612.65	610.0 [M4, M5]
RC207	570.54	558.6 [M5]
RC208	496.95	-

Note: 2-path cuts algorithm [23] is denoted as *M1*, A parallel cutting-plane algorithm [24] is *M2*, Lagrangean duality applied method [25] is *M3*, The shortest-path problem with resource constraints and k-cycle elimination method [26] is *M4*, shortest path based column generation [27] is *M5*, Parallelization of the vehicle routing problem [28] is *M6*, and Accelerating branch-and-price with local search [29] is *M7*.

TABLE II
COMPUTATIONAL RESULT OF THE SBOA USING SOLOMON'S BENCHMARK FOR 50 CUSTOMERS

instances	SBOA	BKS
C101	**362.4**	362.4 [M1]
C102	**361.4**	361.4 [M1]
C103	**361.4**	361.4 [M1]
C104	**358.0**	358.0 [M1]
C105	**362.4**	362.4 [M1]
C106	**362.4**	362.4 [M1]
C107	**362.4**	362.4 [M1]
C108	**362.4**	362.4 [M1]
C109	**362.4**	362.4 [M1]
C201	**360.2**	360.2 [M2, M6]
C202	**360.2**	360.2 [M2, M3]
C203	**359.8**	359.8 [M2, M3]
C204	**350.1**	350.1 [M3]
C205	**359.8**	359.8 [M2, M3]
C206	**359.8**	359.8 [M2, M3]
C207	**359.4**	359.4 [M2, M3]
C208	**350.5**	350.5 [M2, M3]
R101	1054.88	1044.0 [M1]
R102	946.67	909.0 [M1]
R103	813.94	772.9 [M1]
R104	**625.4**	625.4 [M1]
R105	943.95	899.3 [M1]
R106	848.63	793 [M1]
R107	763.97	711.1 [M1]
R108	665.13	617.7 [M2, M3]
R109	821.64	786.8 [M1]
R110	**697.0**	697.0 [M1]
R111	756.21	707.2 [M2, M3]
R112	677.33	630.2 [M2, M3]
R201	823.38	791.9 [M2, 27]
R202	739.86	698.5 [M2, M3]
R203	645.19	605.3 [M4, M5]
R204	**506.4**	506.4 [M4]
R205	**690.1**	690.1 [M4, M5]
R206	673.14	632.4 [M4, M5]
R207	621.07	Not reported
R208	509.70	Not reported
R209	631.76	600.6 [M4, M5]
R210	680.59	645.4 [M4, M5]
R211	566.69	535.3 [M2, M7]
RC101	966.12	944 [M1]
RC102	893.56	822.5 [M1]
RC103	766.70	710.9 [M1]
RC104	**545.8**	545.8 [M1]
RC105	**855.3**	855.3 [M1]
RC106	820.47	723.2 [M1]
RC107	738.68	642.7 [M1]
RC108	**598.1**	598.1 [M1]
RC201	**684.8**	684.8 [M2, M5]
RC202	622.84	613.6 [M4, M5]
RC203	635.42	555.3 [M4, M5]
RC204	**444.2**	444.2 [M7]
RC205	683.79	630.2 [M4, M5]
RC206	612.65	610.0 [M4, M5]
RC207	570.54	558.6 [M5]
RC208	496.95	Not reported

Note: 2-path cuts algorithm [23] is denoted as *M1*, A parallel cutting-plane algorithm [24] is *M2*, Lagrangean duality applied method [25] is *M3*, The shortest-path problem with resource constraints and k-cycle elimination method [26] is *M4*, shortest path based column generation [27] is *M5*, Parallelization of the vehicle routing problem [28] is *M6*, and Accelerating branch-and-price with local search [29] is *M7*.

As in Table I, the results are compared with the BKS and those achieved with the SBOA are emphasized in boldface. Among the 56 cases listed in Table II, the outcomes of the SBOA algorithm were equal or even similar to the BKS in 26 cases (about 46.4% of the cases). This demonstrates that the SBOA algorithm achieves consistent performance on medium-scale problems, with approximately 46.4% optimal or near-optimal solutions.

TABLE III
COMPUTATIONAL RESULT OF THE SBOA USING SOLOMON'S BENCHMARK FOR 100 CUSTOMERS

instances	SBOA	BKS
C101	**828.94**	828.94 [R1]
C102	**828.94**	828.94 [R1]
C103	**828.06**	828.06 [R1]
C104	**824.78**	824.78 [R1]
C105	**828.94**	828.94 [R1]
C106	**828.94**	828.94 [R1]
C107	**828.94**	828.94 [R1]
C108	**828.94**	828.94 [R1]
C109	**828.94**	828.94 [R1]
C201	**591.56**	591.56 [R1]
C202	**591.56**	591.56 [R1]
C203	**591.17**	591.17 [R1]
C204	**590.60**	590.60 [R1]
C205	**588.88**	588.88 [R1]
C206	**588.49**	588.49 [R1]
C207	**588.29**	588.29 [R1]
C208	**588.32**	588.32 [R1]
R101	1642.88	1584 [R2]
R102	1472.62	1374.2 [R2]
R103	1213.62	1158.9 [R2]
R104	**976.61**	976.61 [R1]
R105	1360.78	1355.3 [R2]
R106	1240.47	1212.1 [R2]
R107	**1073.34**	1073.34 [R4]
R108	958.66	947.55 [R4]
R109	**1151.84**	1151.84 [R4]
R110	**1072.41**	1072.41 [R1]
R111	1159.32	1053.50 [R1]
R112	**953.63**	953.63 [R1]
R201	1149.68	1147.80 [R5]
R202	**1034.35**	1034.35 [R1]
R203	**874.87**	874.87 [R1]
R204	736.52	735.80 [R5]
R205	955.82	954.16 [R5]
R206	**879.89**	879.89 [R1]
R207	**799.86**	799.86 [R4]
R208	**705.45**	705.45 [R6]
R209	**859.39**	859.39 [R1]
R210	**910.70**	910.70 [R6]
R211	**755.96**	755.96 [R4]
RC101	1643.41	1595.9 [R2]
RC102	1461.23	1460.9 [R2]
RC103	1277.54	1261.67 [R7]
RC104	1136.81	1135 [R2]
RC105	1518.58	1510.1 [R2]
RC106	1381.23	1367.2 [R2]
RC107	**1212.83**	1212.83 [R4]
RC108	1197.13	1117.53 [R3]
RC201	**1265.56**	1265.56 [R1]
RC202	**1095.64**	1095.64 [R1]
RC203	**928.51**	928.51 [R4]
RC204	**786.38**	786.38 [R5]
RC205	**1157.55**	1157.55 [R5]
RC206	**1054.61**	1054.61 [R1]
RC207	**966.08**	966.08 [R1]
RC208	**779.31**	779.31 [R4]

Note: A hybrid genetic algorithm [17] is denoted as *R1*, A modified football game algorithm [20] is *R2*, A genetic and set partitioning two-phase approach [30] is *R3*, A hybrid genetic algorithm [31] is *R4*, A hybrid search method [32] is *R5*, Time-window relaxations [33] is *R6*, and Using constraint programming and local search methods [34] is *R7*.

The computational results of SBOA for cases with 100 customers are displayed in Table III. The results are compared to the BKS., and the best results obtained by the SBOA algorithm are shown in bold characters. From a total of 56 cases in Table III, the SBOA algorithm produced the same results or better compared to the BKS in 39 cases (about 69.6% of the cases). The results demonstrated that the SBOA algorithm is scalable, as it is able to achieve competitive solutions for larger scale problems compared to

the BKS results in the literature.

To further investigate the performance of the SBOA the experimental values of results, Table IV show the results obtained by SBOA using solomon's benchmark for 100 customers.

TABLE IV
THE RESULTS OBTAINED BY SBOA USING SOLOMON'S BENCHMARK FOR 100 CUSTOMERS

instances	SBOA	Average	Std	Gap	CVar
C101	828.94	828.94	0.00	0	0
C102	828.94	828.94	0.00	0	0
C103	828.06	828.06	0.00	0	0
C104	824.78	824.78	0.00	0	0
C105	828.94	828.94	0.00	0	0
C106	828.94	828.94	0.00	0	0
C107	828.94	828.94	0.00	0	0
C108	828.94	828.94	0.00	0	0
C109	828.94	828.94	0.00	0	0
C201	591.56	591.56	0.00	0	0
C202	591.56	591.56	0.00	0	0
C203	591.17	591.17	0.00	0	0
C204	590.60	612.11	8.01	0	1.31
C205	588.88	588.88	0.00	0	0
C206	588.49	588.49	0.00	0	0
C207	588.29	588.29	0.00	0	0
C208	588.32	588.32	0.00	0	0
R101	1642.88	1643.53	3.45	3.72	0.21
R102	1472.62	1479.19	7.29	7.17	0.5
R103	1213.62	1222.29	5.79	4.73	0.48
R104	976.61	1001.44	7.87	0	0.79
R105	1360.78	1365.70	3.21	0.41	0.24
R106	1240.47	1242.44	2.52	2.35	0.21
R107	1073.34	1083.10	5.45	0	0.51
R108	958.66	968.45	4.24	1.18	0.44
R109	1151.84	1157.27	3.51	0	0.31
R110	1072.41	1082.72	4.52	0	0.42
R111	1159.32	1066.80	8.84	10.05	0.83
R112	953.63	971.89	5.56	0	0.58
R201	1149.68	1153.04	2.69	0.17	0.24
R202	1034.35	1038.40	4.14	0	0.4
R203	874.87	875.87	1.54	0	0.18
R204	736.52	741.41	6.36	0.1	0.86
R205	955.82	964.69	6.06	0.18	0.63
R206	879.89	892.55	6.36	0	0.72
R207	799.86	814.05	6.3	0	0.78
R208	705.45	714.37	5.86	0	0.83
R209	859.39	867.52	5.99	0	0.7
R210	910.70	918.37	12.17	0	1.33
R211	755.96	765.64	3.45	0	0.46
RC101	1643.41	1658.34	6.52	2.98	0.4
RC102	1461.23	1480.82	6.34	0.03	0.43
RC103	1277.54	1313.73	8.45	1.26	0.65
RC104	1136.81	1155.47	10.57	0.16	0.92
RC105	1518.58	1526.80	10.25	0.57	0.68
RC106	1381.23	1397.45	12.25	1.03	0.88
RC107	1212.83	1217.90	3.65	0	0.3
RC108	1197.13	1222.29	7.34	7.13	0.61
RC201	1265.56	1269.94	4.86	0	0.39
RC202	1095.64	1101.03	5.99	0	0.55
RC203	928.51	943.81	12.19	0	1.3
RC204	786.38	799.19	13.04	0	1.64
RC205	1157.55	1164.43	3.5	0	0.31
RC206	1054.61	1067.49	11.25	0	1.06
RC207	966.08	975.24	9.35	0	0.96
RC208	779.31	791.35	18.74	0	2.37

As shown in Table IV, the table is organized as follows: the first column provides the name of the instances, the second column gives the best solution obtained by SBOA for 100 customers, the third column is the average solution (Average), the fourth column is the standard deviation (Std), the fifth column reports the percentage deviation (Gap) between the SBOA solution and the BKS obtained by using Equation 4, and the sixth column shows the coefficient of variation (CVar), calculated using Equation 5, which gives the relative variability of the solutions as a percentage to the mean. Here, the best solutions obtained by the SBOA and BKS for each instance are denoted by *S1* and *S2*, respectively.

$$GAP = \frac{S1 - S2}{S2} \times 100 \tag{4}$$

$$CVar = \frac{Std}{Average} \times 100 \tag{5}$$

The results of SBOA that are presented in Table IV, shows that, Average gives the average solution value obtained in 10 independent runs, and shows the algorithm's stability. A standard deviation (Std) of zero (as in C101–C109) means perfect reproducibility, and a higher Std (e.g., 8.01 for C204) indicates variability in the quality of the solution across runs. Gap is calculated as percentage of SBOA's best solution compared to BKS, and zero gap (e.g., C101–C109) is the optimistic situation and positive gap (e.g., 10.05% for R111) shows the compromise inaccuracy, where a high gap values in R111 and RC108 (10.05% and 7.13%) indicate that SBOA struggles with tightly clustered and time-constrained instances. This suggests a need for enhanced local search in such scenarios. Lastly, the CVar measures the stability of the solution: a lower value (e.g., 0% for C101) implies a high stability, whilst a higher value (e.g., 2.37% for RC208) indicates high spreading. It is interesting to observe that the SBOA achieves optimality (0 Gap) for 29 instances out of 56, and thus illustrate robustness, while variation in Std and CVar for some instances (e.g., RC series) indicates sensitivity to the complexity of the instance.

As the recent study focuses on more customers (which is 100 customers' datasets in this case), Fig. 4 shows a comparison between the performances of the SBOA and other algorithms: PSO [13], WOA [15], MFGA [20], SFSSA [21], and MRSO [22] and Table V shows a comparison between SBOA results and other algorithms results.

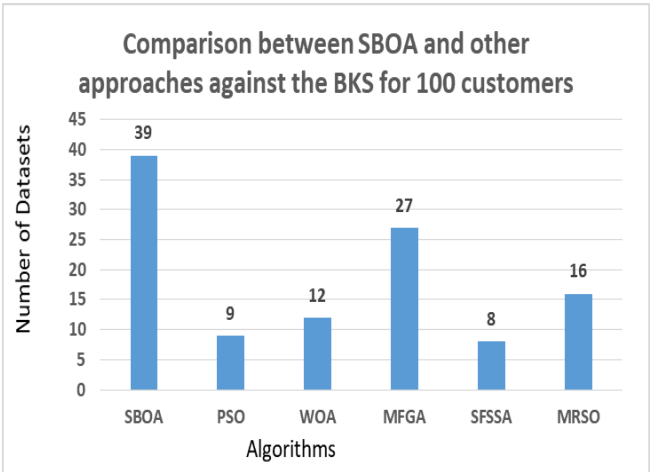

Fig. 4. Comparison between SBOA and other approaches against the BKS for 100 customers

As shown in Fig. 4, SBOA algorithm finds 39 of 56 BKS solutions for 100 customers, WOA managed to get 12 BKS results, MFGA managed to get 27 BKS results, PSO also managed to get 9 BKS results, SFSSA got 8 BKS results, and MRSO got 16 BKS results.

TABLE V
A COMPARISON BETWEEN SBOA AND OTHER ALGORITHMS RESULTS FOR 100 CUSTOMERS

instances	SBOA	PSO	WOA	MFGA	SFSSA	MRSO
C101	828.94	828.94	828.94	828.94	967.51	828.94
C102	828.94	829.72	828.94	828.94	949.59	828.94
C103	828.06	851.38	844.94	828.06	887.47	828.06
C104	824.78	868.53	842.16	824.78	902.52	847.15
C105	828.94	828.94	828.94	828.94	983.1	828.94
C106	828.94	828.94	828.94	828.94	878.29	828.94
C107	828.94	828.94	828.94	828.94	944.17	828.94
C108	828.94	828.94	828.94	828.94	922.94	828.94
C109	828.94	828.94	828.94	828.94	985.08	828.94
C201	591.56	591.56	591.56	591.56	591.56	591.56
C202	591.56	591.56	609.21	591.56	591.56	591.56
C203	591.17	591.18	628.08	591.17	591.17	591.17
C204	590.60	615.43	603.4	590.6	590.6	590.6
C205	588.88	588.88	588.88	588.88	588.88	588.88
C206	588.49	588.88	588.49	588.49	588.49	588.49
C207	588.29	591.35	588.29	588.29	588.29	588.29
C208	588.32	588.5	588.32	588.32	588.32	588.32
R101	1642.88	1652.01	1678.92	1584	1650.79	1659.34
R102	1472.62	1500.81	1552.26	1374.2	1485.85	1476.5
R103	1213.62	1242.65	1315.28	1158.9	1231.34	1240.97
R104	976.61	1042.22	1051.54	996.95	1005.17	1033.52
R105	1360.78	1385.09	1475.5	1355.3	1375.94	1405.18
R106	1240.47	1294.87	1342.53	1212.1	1241.27	1270.87
R107	1073.34	1123.99	1168.01	1075.5	1081.9	1122.96
R108	958.66	1011.69	1041.27	959.88	965.58	986.2
R109	1151.84	1211.63	1245.09	1155.8	1166.95	1207.18
R110	1072.41	1190.37	1153.24	1092.4	1120.88	1128.21
R111	1159.32	1102.99	1159.32	1059.2	1079.61	1062.26
R112	953.63	1029.13	1034.34	979.05	991.76	984.84
R201	1149.68	1274.97	1230.86	1168.7	1223.38	1183.9
R202	1034.35	1247.04	1134.82	1042.4	1086.86	1044.45
R203	874.87	1052.72	948.29	893.97	922.86	900.2
R204	736.52	844.17	807.6	744.02	775.84	775.59
R205	955.82	1061.46	1036.18	969.42	1016.54	962.02
R206	879.89	1016.35	944.13	880.6	902.11	916.3
R207	799.86	946.78	869.62	822.84	847.93	832.16
R208	705.45	834.73	763.69	736.55	729.81	721.04
R209	859.39	1003.19	930.16	905.11	909.7	875.95
R210	910.70	1040.55	957.24	937.06	935.68	925.47
R211	755.96	861.33	815.74	815.09	806.04	783.68
RC101	1643.41	1641.21	1732.3	1595.9	1654.84	1663.36
RC102	1461.23	1510.96	1598.21	1460.9	1503.05	1498.21
RC103	1277.54	1294.74	1395.83	1292.6	1273.11	1346
RC104	1136.81	1190.55	1239.84	1135	1189.84	1186.5
RC105	1518.58	1603.71	1651.24	1510.1	1578.78	1596.7
RC106	1381.23	1410.94	1479.76	1367.2	1374.38	1408.83
RC107	1212.83	1249.8	1297.24	1215.9	1226.79	1335.05
RC108	1197.13	1181.87	1239.92	1120.1	1169.84	1227.24
RC201	1265.56	1423.52	1326.28	1274.8	1483.96	1285.08
RC202	1095.64	1193.6	1184.17	1115.7	1130.01	1106.84
RC203	928.51	1123.42	993.68	945.9	987.15	931.45
RC204	786.38	894.12	830.08	803.91	845.55	824.79
RC205	1157.55	1321.43	1258.78	1209.5	1291.2	1180.12
RC206	1054.61	1307.9	1146.5	1098	1124.5	1072.29
RC207	966.08	1130.37	1060.55	1010.4	1039.11	977.06
RC208	779.31	958.24	818.93	810.04	844.58	805.21

For more understanding on the results of Table V, Fig. 5 breaks down the results of the SBOA algorithm and other algorithms performance for VRPTW across the large (100 customers) VRPTW datasets. In this figure, green color means better, dark red means worse and light blue means similar.

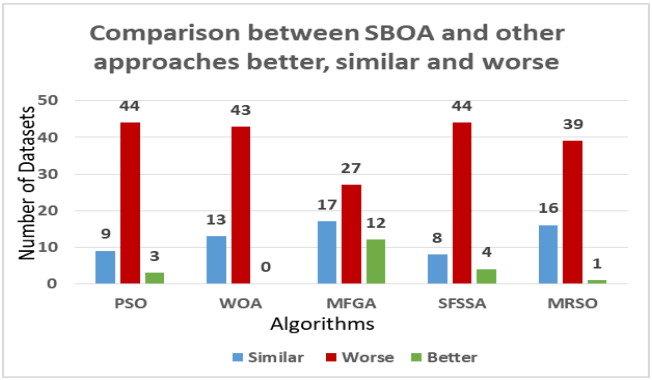

Fig. 5. Comparison between SBOA and other approaches better, similar and worse results for 100 customers

According to the results in Fig. 5, the proposed SBOA was superior to PSO, WOA and SFSSA in most test functions. More specifically, the SBOA variant outperformed WOA on 13 from 56 datasets, achieved a tie on 43, and fared worse in 0. SBOA was superior to MFGA in 27, equivalent in 17 and inferior in 12. These results suggest that although MFGA is relatively a bit more competitive, SBOA is still very effective and robust for all problems, in particular for C-type and R-type problems. SBOA had better results than PSO, WOA, and MFGA SFSSA, as well as MRSO algorithms in some data sets according to BKS solutions, regardless of dimensions.

For further explanation of SBOA against PSO, WOA, MFGA SFSSA, MRSO, and BKS for VRPTW C-type datasets 100 customers, Fig. 6 illustrates the results for C-type datasets for 100 customers, Fig. 7 show the R-type datasets 100 customers, and Fig. 8 for RC-type datasets 100 customers. In all the figures, the horizontal coordinates are the data corresponding to 100 customers while the vertical coordinates are the performance data.

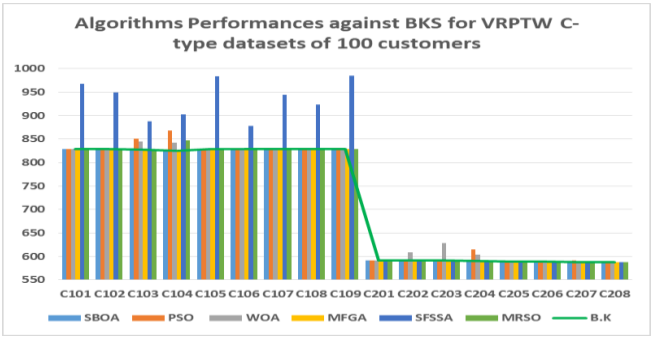

Fig. 6. Comparison between algorithms performance in terms of BKS for C-type datasets of 100 customers

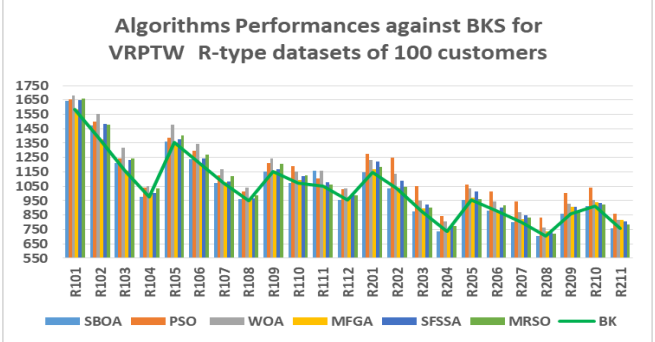

Fig. 7. Comparison between algorithms performance in terms of the BKS for R-type datasets of 100 customers

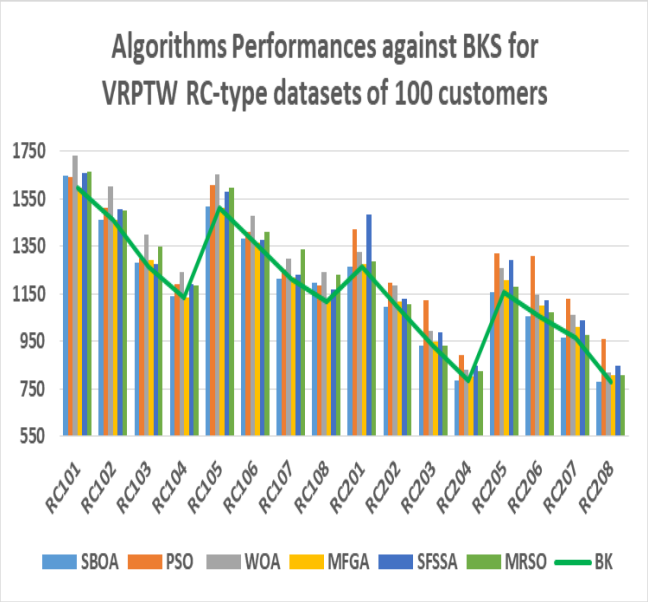

Fig. 8. Comparison between algorithms performance in terms of the BKS for RC-type datasets of 100 customers

In Fig. 6, Fig. 7 and Fig. 8, the BKS results is shown as green line and each algorithm results indicated by the column referring to the color in the horizontal legend. For instance, RC107 dataset in Fig. 8, the column with light blue color shows that the proposed algorithm (the SBOA) is closer to the BKS (the green line).

Despite the competitive performance of the SBOA on multiple classes of VRPTW instances, there are some limitations worth noting:

- While the SBOA avoids trapping in local optima after using evasion strategies, premature convergence may still occur, especially in the extensive or tightly constrained instances of the VRPTW where feasible areas are scarce.
- SBOA exhibit high sensitivity on algorithmic parameters (population size, perturbation range, hunting/evasion balance). It can result in sub-optimal results if not been well calibrated, as there are no self-adaptive mechanisms.
- Although SBOA yielded excellent results for instances with up to 100 customers, its time complexity and space complexity for particularly large instances (e.g., 500+ customers) have not yet been investigated and may become performance bottlenecks.
- As it stands, the implementation is quite appropriate for static problem settings. It cannot accommodate dynamic changes, such as last-minute customer requests or real-time traffic delays, as frequently encountered in practical VRPTW applications.
- While it outperformed some of the common metaheuristics (PSO, WOA and SFSSA), it was not compared to more advanced or hybrid metaheuristics such as Large Neighborhood Search (LNS), Adaptive Large Neighborhood Search (ALNS), or Reinforcement Learning-based methods, which have been tested in similar scenarios and had much better performance.
- Although penalty-based approaches were our methods of choice to deal with time window violations, these techniques do not always direct the search in a helpful way. Further refinement needed, for example, by adding advanced repair strategies or operators that preserve constraints, would enhance feasibility and convergence.

The following Table VI shows a comparison between SBOA results and other algorithms results using Gap metric, and Table VII presents a comparison of the performance of various algorithms applied to three types of datasets: C-type, R-type, and RC-type, each involving 100 customers. The evaluation metric used is the results for each datasets type and the Average gap. This comparison aims to highlight the effectiveness of each algorithm across different dataset types.

TABLE VI
A COMPARISON BETWEEN SBOA AND OTHER ALGORITHMS GAP FOR 100 CUSTOMERS

instances	SBOA	PSO	WOA	MFGA	SFSSA	MRSO
C101	0	0	0	0	16.72	0
C102	0	0.1	0	0	14.56	0
C103	0	2.82	2.04	0	7.18	0
C104	0	5.31	2.11	0	9.43	2.72
C105	0	0	0	0	18.6	0
C106	0	0	0	0	5.96	0
C107	0	0	0	0	13.91	0
C108	0	0	0	0	11.34	0
C109	0	0	0	0	18.84	0
C201	0	0	0	0	0	0
C202	0	0	2.99	0	0	0
C203	0	0.01	6.25	0	0	0
C204	0	4.21	2.17	0	0	0
C205	0	0	0	0	0	0
C206	0	0.07	0	0	0	0
C207	0	0.53	0	0	0	0
C208	0	0.04	0	0	0	0
R101	3.72	4.3	6	0	4.22	4.76
R102	7.17	9.22	12.96	0	8.13	7.45
R103	4.73	7.23	13.5	0	6.26	7.09
R104	0	6.72	7.68	2.09	2.93	5.83
R105	0.41	2.2	8.87	0	1.53	3.69
R106	2.35	6.83	10.77	0	2.41	4.85
R107	0	4.72	8.83	0.21	0.8	4.63
R108	1.18	6.77	9.9	1.31	1.91	4.08
R109	0	5.2	8.1	0.35	1.32	4.81
R110	0	11	7.54	1.87	4.52	5.21
R111	10.05	4.7	10.05	0.55	2.48	0.84
R112	0	7.92	8.47	2.67	4	3.28
R201	0.17	11.08	7.24	1.83	6.59	3.15
R202	0	20.57	9.72	0.78	5.08	0.98
R203	0	20.33	8.4	2.19	5.49	2.9
R204	0.1	14.73	9.76	1.12	5.45	5.41
R205	0.18	11.25	8.6	1.6	6.54	0.83
R206	0	15.51	7.31	0.09	2.53	4.14
R207	0	18.37	8.73	2.88	6.01	4.04
R208	0	18.33	8.26	4.41	3.46	2.21
R209	0	16.74	8.24	5.33	5.86	1.93
R210	0	14.26	5.12	2.9	2.75	1.63
R211	0	13.94	7.91	7.83	6.63	3.67
RC101	2.98	2.84	8.55	0	3.7	4.23
RC102	0.03	3.43	9.4	0	2.89	2.56
RC103	1.26	2.63	10.64	2.46	0.91	6.69
RC104	0.16	4.9	9.24	0	4.84	4.54
RC105	0.57	6.2	9.35	0	4.55	5.74
RC106	1.03	3.2	8.24	0	0.53	3.05
RC107	0	3.05	6.96	0.26	1.16	10.08
RC108	7.13	5.76	10.96	0.23	4.69	9.82
RC201	0	12.49	4.8	0.74	17.26	1.55
RC202	0	8.95	8.09	1.84	3.14	1.03
RC203	0	21	7.02	1.88	6.32	0.32
RC204	0	13.71	5.56	2.23	7.53	4.89
RC205	0	14.16	8.75	4.49	11.55	1.95
RC206	0	24.02	8.72	4.12	6.63	1.68
RC207	0	17.01	9.78	4.59	7.56	1.14
RC208	0	22.97	5.09	3.95	8.38	3.33

TABLE VII
A COMPARISON BETWEEN SBOA AND OTHER ALGORITHMS PERFORMANCE APPLIED TO THREE TYPES OF DATASETS FOR 100 CUSTOMERS

Algorithms	C-type	R-type	RC-type	Average gap
SBOA	17	13	9	0.771786
PSO	9	0	0	7.702321
WOA	12	0	0	6.22625
MFGA	17	5	5	1.192857
SFSSA	8	0	0	5.447857
MRSO	16	0	0	2.727321

As shown in Table VII, the Average gap for SBOA outperformed all compared algorithms (PSO, WOA, MFGA, SFSSA, and MRSO). For instance, SBOA Average gap (0.771786) outperformed MFGA Average gap (1.192857).

V. CONCLUSION

The VRPTW is a NP-hard combinatorial optimization problem close to logistics and transportation tasks. This paper focuses on applying the SBOA to VRPTW instances, emphasizing its unique hunting and evasion strategies to improve solution quality. The algorithm was evaluated using Solomon benchmark instances, which include small-, medium-, and large-scale problems. The experimental results demonstrate the effectiveness of SBOA with BKS solution found for different instances. That is, in 27 (48.2%) of the cases with 25 customers, 26 (46.4%) of the cases with 50 customers and 39 (69.6%) of the cases with 100 customers, the results generated by the SBOA algorithm are equal or close to the BKS. These results demonstrate the algorithm's effectiveness in minimizing the total travel distance and vehicle usage while adhering to time window constraints. SBOA performed competitively regarding convergence behavior and succeeded in locating the optimal solutions when compared to state-of-the-art metaheuristics. The experimental results confirm the effectiveness and efficiency of the approach, indicating that SBOA is a promising optimization algorithm for large-scale and strict constrained VRPTW problems.

While this study sheds light on the promise of the SBOA algorithm for tackling VRPTW challenges, several potential directions for expansion, including but not limited to the incorporation of SBOA with additional heuristic methods, such as GA or SA algorithms, to enhance both exploitation exploration ability; extending the algorithm to accommodate dynamic VRPTW problems, where customer requests and temporal constraints fluctuate in real-time, would enhance its practical significance; and customizing SBOA to investigate multi-objective VRPTW problems, factoring in elements like fuel utilization, driver workload, and ecological effect, would diversify its applicability.

REFERENCES

[1] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," Oper. Res., vol. 35, no. 2, pp. 254–265, 1987. https://doi.org/10.1287/opre.35.2.254

[2] E. Boumpa, V. Tsoukas, V. Chioktour, M. Kalafati, G. Spathoulas, A. Kakarountas, P. Trivellas, P. Reklitis, and G. Malindretos, "A review of the vehicle routing problem and the current routing services in smart cities," Analytics, vol. 2, no. 1, pp. 1–16, 2023. https://doi.org/10.3390/analytics2010001

[3] X. Liu, Y. L. Chen, L. Y. Por, and C. S. Ku, "A systematic literature review of vehicle routing problems with time windows," Sustainability, vol. 15, no. 15, p. 12004, 2023. https://doi.org/10.3390/su151512004

[4] Y. Hou, L. Dang, H. Ma, and C. Zhang, "A selection hyper-heuristic for the multi-compartment vehicle routing problem considering carbon emission," Eng. Lett., vol. 32, no. 10, pp. 2002–2011, 2024.

[5] C. Truden, K. Maier, and P. Armbrust, "Decomposition of the vehicle routing problem with time windows on the time dimension," Transp. Res. Procedia, vol. 62, pp. 131–138, 2022. https://doi.org/10.1016/j.trpro.2022.02.017

[6] J. C. Chu, C. S. Shui, and Y. T. Chuang, "Vehicle routing problem with en-route delivery," Transportmetrica B: Transp. Dyn., vol. 13, no. 1, 2025. https://doi.org/10.1080/21680566.2025.2490509

[7] H. Ma, Y. Hou, and H. Xu, "An iterated local search algorithm for the heterogeneous fixed multi-compartment vehicle routing problem," IAENG Int. J. Appl. Math., vol. 55, no. 5, pp. 1084–1091, 2025.

[8] F. Arnold, M. Gendreau, and K. Sörensen, "Efficiently solving very large-scale routing problems," Comput. Oper. Res., vol. 107, pp. 32–40, 2019. https://doi.org/10.1016/j.cor.2019.03.006

[9] Huohuo, Secretary Bird Optimization Algorithm (SBOA), MATLAB Central File Exchange, Apr. 25, 2024.

[10] C. Wang, H. Ma, D. Zhu, and Y. Hou, "A hybrid genetic algorithm for multi-compartment open vehicle routing problem with time window in fresh products distribution," Eng. Lett., vol. 32, no. 6, pp. 1201–1209, 2024.

[11] Y. Fu, D. Liu, J. Chen, and L. He, "Secretary bird optimization algorithm: A new metaheuristic for solving global optimization problems," Artif. Intell. Rev., vol. 57, no. 123, 2024. https://doi.org/10.1007/s10462-024-10729-y

[12] Y. Cai and H. Chen, "An improved salp swarm algorithm for permutation flow shop vehicle routing problem," Sci. Rep., vol. 15, p. 6704, 2025. https://doi.org/10.1038/s41598-025-86054-3

[13] Y. J. Gong, J. Zhang, O. Liu, R. Z. Huang, H. S. H. Chung, and Y. H. Shi, "Optimizing the vehicle routing problem with time windows: A discrete particle swarm optimization approach," IEEE Trans. Syst. Man Cybern. C Appl. Rev., vol. 42, no. 2, pp. 254–267, 2011.

[14] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," Adv. Eng. Softw., vol. 69, pp. 46–61, 2014. https://doi.org/10.1016/j.advengsoft.2013.12.007

[15] S. Yodwangjai and K. Malampong, "An improved whale optimization algorithm for vehicle routing problem with time windows," J. Ind. Technol., vol. 18, no. 1, pp. 1–12, 2022. https://doi.org/10.14416/j.ind.tech.2022.04.001

[16] M. F. Ibrahim, M. I. Mustofa, P. Meilanitasari, and S. U. Wijaya, "An improved ant colony optimization algorithm for the vehicle routing problem with time windows," J. Tek. Ind., vol. 23, no. 2, pp. 105–120, 2022. https://doi.org/10.22219/JTIUMM.Vol23.No2.105-120

[17] S. Jung and B. R. Moon, "A hybrid genetic algorithm for the vehicle routing problem with time windows," in Proc. 4th Annu. Conf. Genetic and Evolutionary Computation (GECCO), 2002, pp. 1309–1316.

[18] Z. Zhang, Y. Dou, W. Cheng, X. Xu, J. Jiang, and Y. Tan, "A tabu search algorithm based on density peak clustering to solve VRPTW," in Proc. 8th Int. Conf. Big Data Inf. Anal., 2022, pp. 472–478. https://doi.org/10.1109/BigDIA56350.2022.9874007.

[19] M. Mohammadi, N. Mahmoodian, and H. Mohammadi, "A simulated annealing approach to vehicle routing problem with time windows," in Proc. 8th Int. Conf. Control Instrum. Autom., 2022, pp. 1–6. https://doi.org/10.1109/ICCIA54998.2022.9737187

[20] Z. H. Ahmed, F. Maleki, M. Yousefikhoshbakht, and H. Haron, "Solving the vehicle routing problem with time windows using modified football game algorithm," Egypt. Inform. J., vol. 24, no. 4, p. 100403, Dec. 2023. https://doi.org/10.1016/j.eij.2023.100403

[21] Z. Liu et al., "A new hybrid algorithm for vehicle routing optimization," Sustainability, vol. 15, no. 14, p. 10982, 2023. https://doi.org/10.3390/su151410982

[22] X. Wei, Z. Xiao, and Y. Wang, "Solving the vehicle routing problem with time windows using modified rat swarm optimization algorithm based on large neighborhood search," Mathematics, vol. 12, no. 11, p. 1702, 2024. https://doi.org/10.3390/math12111702

[23] N. Kohl, J. Desrosiers, O. B. Madsen, M. M. Solomon, and F. Soumis, "2-path cuts for the vehicle routing problem with time windows," Transp. Sci., vol. 33, no. 1, pp. 101–116, 1999. https://doi.org/10.1287/trsc.33.1.101

[24] W. Cook and J. L. Rich, "A parallel cutting-plane algorithm for the vehicle routing problem with time windows," Dept. Comput. Appl. Math., Rice Univ., Tech. Rep. TR99-04, 1999.
[25] B. Kallehauge, J. Larsen, and O. B. Madsen, "Lagrangean duality applied on vehicle routing with time windows: Experimental results," Internal Rep. IMM-REP-2000-8, Tech. Univ. of Denmark, 2001.
[26] S. Irnich and D. Villeneuve, "The shortest-path problem with resource constraints and k-cycle elimination for k ≥ 3," INFORMS J. Comput., vol. 18, no. 3, pp. 391–406, 2006. https://doi.org/10.1287/ijoc.1040.0124
[27] A. Chabrier, "Vehicle routing problem with elementary shortest path based column generation," Comput. Oper. Res., vol. 33, no. 10, pp. 2972–2990, 2006. https://doi.org/10.1016/j.cor.2005.01.018
[28] J. Larsen, "Parallelization of the vehicle routing problem with time windows," M.S. thesis, Tech. Univ. of Denmark, 1999.
[29] E. Danna and C. L. Pape, "Branch-and-Price Heuristics: A Case Study on the Vehicle Routing Problem with Time Windows," in Desaulniers, G., Desrosiers, J., Solomon, M.M. (eds) Column Generation, Springer, Boston, MA, 2005, pp. 87–112. https://doi.org/10.1007/0-387-25486-2_4
[30] G. B. Alvarenga, G. R. Mateus, and G. D. Tomi, "A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows," Comput. Oper. Res., vol. 34, no. 6, pp. 1561–1584, 2007. https://doi.org/10.1016/j.cor.2005.07.027
[31] J. Berger, M. Salois, and R. Begin, "A hybrid genetic algorithm for the vehicle routing problem with time windows," in Mercer, R.E., Neufeld, E. (eds) Advances in Artificial Intelligence. Canadian AI 1998. Lecture Notes in Computer Science, vol. 1418, Springer, Berlin, Heidelberg, 1998, pp. 114–125. https://doi.org/10.1007/3-540-64575-6_44
[32] H. C. B. de Oliveira and G. C. Vasconcelos, "A hybrid search method for the vehicle routing problem with time windows," Ann. Oper. Res., vol. 180, no. 1, pp. 125–144, 2010. https://doi.org/10.1007/s10479-008-0481-4
[33] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, "Time-window relaxations in vehicle routing heuristics," J. Heuristics, vol. 21, no. 3, pp. 329–358, 2015. https://doi.org/10.1007/s10732-014-9273-y
[34] P. Shaw, "Using constraint programming and local search methods to solve vehicle routing problems," in Maher, M., Puget, J.F. (eds) Principles and Practice of Constraint Programming — CP98. Lecture Notes in Computer Science, vol. 1520, Springer, Berlin, Heidelberg, 1998, pp. 417–431. https://doi.org/10.1007/3-540-49481-2_30