STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **16**, July 2026, pp 684–697.
Published online in International Academic Press (www.IAPress.org)

Q-Learning Driven Artificial Bee Colony Algorithm for Solving the Weapon-Target Assignment Problem

Kadir Yıldız, Emrullah Sonuç*

Department of Computer Engineering, Karabuk University, Karabük, Türkiye

Abstract The Weapon-Target Assignment (WTA) problem is a critical, NP-complete combinatorial optimization problem in military operations research. It aims to allocate defense weapons to incoming targets to minimize the total expected damage to protected assets. This study proposes a Q-learning-driven artificial bee colony (QABC) algorithm that incorporates an adaptive operator selection mechanism based on Q-learning into the standard artificial bee colony (ABC) algorithm. Five permutation-based neighborhood operators are employed to balance exploration and exploitation during the search. We evaluated the proposed algorithm on twelve benchmark WTA instances with problem sizes ranging from five to 200 weapons and targets across 30 independent runs. The experimental results demonstrate that the QABC algorithm consistently outperforms a recent competitive method on nine of the twelve instances, achieving substantially lower cost values and standard deviations, especially for large-scale problems.

Keywords Adaptive operator selection, Artificial bee colony, Combinatorial optimization problem, Q-learning, Weapon-target assignment problem.

AMS 2010 subject classifications 90C27, 90C59, 68T20

DOI: 10.19139/soic-2310-5070-3934

1. Introduction

Weapon-Target Assignment (WTA) is one of the critical optimization problems in military operations research and was introduced to the literature by Manne [21]. The problem aims to allocate defense resources (weapons) most effectively against incoming threats (targets). From a defense perspective, the objective is to assign weapons in the most appropriate manner to minimize the probability of an incoming missile destroying a protected asset. The nature of the problem requires consideration of each weapon's probability of destroying a specific target (kill probability) and each target's strategic value [13]. In the literature, the WTA problem is fundamentally divided into two variations based on its temporal characteristics: Static WTA (SWTA) and Dynamic WTA (DWTA). In the static model, a fixed number of targets and defense weapons are evaluated for a single engagement cycle. In this scenario, time is not considered a parameter, and all assignments are made at once [18]. On the other hand, Dynamic WTA incorporates time as a dimension into the process and allows for multi-stage engagement scenarios enabling the reallocation of weapons based on the survival status of targets [2].

The objective function of the static WTA is typically formulated as a nonlinear minimization problem aimed at reducing the expected damage directed by the offense. Mathematically, the problem is defined as minimizing the sum of the expected values of surviving targets:

$$f(\pi) = \min \sum_{j=1}^{m} V_j \prod_{i=1}^{n} (1 - p_{ij})^{x_{ij}}, \tag{1}$$

*Correspondence to: Emrullah Sonuç (Email: esonuc@karabuk.edu.tr). Department of Computer Engineering, Karabuk University, Karabük, Türkiye.

ISSN 2310-5070 (online) ISSN 2311-004X (print)

$$\text{s.t.} \quad \sum_{j=1}^{m} x_{ij} \leq 1, \quad i = 1, 2, \ldots, n, \tag{2}$$

$$x_{ij} \in \{0, 1\}. \tag{3}$$

In this formulation, V_j represents the strategic value of target j, p_{ij} is the probability that weapon i destroys target j, and x_{ij} is a decision variable indicating whether weapon i is assigned to target j. This minimization is subject to inventory constraints, ensuring that the total number of weapons assigned does not exceed the available supply. Despite its formulation, the WTA problem has been proven to be NP-complete, meaning its computational complexity grows exponentially as the number of weapons and targets increases [1]. For large-scale problems, the number of potential permutations (n^m) becomes so vast that searching for an optimal solution using exact algorithms is often computationally intractable within the real-time constraints required for military decision-making.

The need for scalable, efficient decision support systems is widely recognized in complex operational domains, such as resource allocation and defense planning, where optimal assignment under uncertainty is critical [25]. To address this challenge, intelligent optimization and automated decision systems have been widely studied to support human operators in large-scale, time-critical environments [20, 4].

Recent advances in artificial intelligence and optimization algorithms have demonstrated considerable promise in solving complex assignment and scheduling problems. Swarm intelligence, evolutionary computation, and large-scale neighborhood search strategies have shown strong performance on nonlinear and combinatorial formulations such as the WTA problem [31, 16, 8]. Hybrid frameworks combining heuristics, metaheuristics, and mathematical programming have further improved solution robustness and scalability [23, 24, 34]. Exact optimization techniques, such as branch-and-bound, branch-and-price, and column generation-based formulations, are still valuable benchmarks for moderate-scale problem instances [15, 19, 5]. However, computational demands increase rapidly with problem size, motivating the adoption of heuristics and adaptive metaheuristics for large-scale or real-time scenarios [14, 26].

Among nature-inspired approaches, Genetic Algorithms, Particle Swarm Optimization, Ant Colony Optimization, Bees Algorithm variants, and fuzzy evolutionary models have been explored extensively for static, dynamic, and constrained WTA formulations [10, 7, 28, 9, 6]. Adaptive mechanisms within swarm-based frameworks have been shown to improve convergence stability and exploration capacity under heterogeneous threat and weapon conditions [22, 32, 12, 27]. Parallel and modified metaheuristic designs further enhance computational efficiency for large-scale WTA. Parallel simulated annealing implementations accelerate convergence on high-dimensional instances [29], while nature-inspired algorithm variants such as modified crow search improve diversification in the solution space [30]. A multi-start late acceptance hill climbing algorithm has demonstrated competitive performance for static WTA formulations, achieving strong solution quality with reduced sensitivity to initial solutions [3]. Reinforcement learning has also begun to be combined with metaheuristics for the WTA problem: a deep Q-network has been integrated with an improved multi-objective artificial bee colony algorithm for unmanned ground weapon-target assignment [33], and a deep Q-network-based adaptive mutation operator has been embedded within a multi-objective evolutionary algorithm to balance exploration and exploitation [35]. A recent comprehensive survey further provides a broad overview of WTA models, algorithms, and applications [17]. Overall, the literature indicates that adaptive swarm-based metaheuristics, particularly enhanced variants of metaheuristics, are well-suited for complex assignment problems with nonlinear objective functions and large combinatorial search spaces. Building upon this body of work, this study employs an adaptive artificial bee colony framework tailored for the static WTA problem. The goal is to achieve a better balance between intensification and diversification while maintaining computational efficiency for large-scale instances.

The motivation of this work is to obtain a metaheuristic that adapts its search behavior online, rather than committing to a single fixed neighborhood structure, so that solution quality is preserved as the WTA problem scales to large instances. The main contributions of this study can be summarized as follows: (i) a Q-learning-style adaptive operator selection mechanism, driven by a sliding-window reward signal, is embedded in the employed- and onlooker-bee phases of the ABC algorithm to govern the choice among five permutation-based neighborhood operators, (ii) a per-iteration computational-complexity analysis is provided, showing that the adaptive selection

overhead is asymptotically dominated by the objective-function evaluation, so that the mechanism does not increase the order of the algorithm, and (iii) the resulting QABC algorithm is evaluated on twelve static WTA benchmark instances ranging from five to 200 weapons and targets over 30 independent runs, with the empirical study supported by full nonparametric statistical testing and a convergence-behavior analysis that together characterize both the final solution quality and the search dynamics of the proposed method.

2. Q-Learning Driven Artificial Bee Colony Algorithm

The Artificial Bee Colony (ABC) algorithm is a population-based, swarm-intelligence metaheuristic inspired by the foraging behavior of honeybee colonies [11]. In the standard ABC framework, candidate solutions are represented as food sources, and the search process involves three types of agents: employed bees, which explore their assigned food sources, onlooker bees, which probabilistically select promising solutions based on fitness values, and scout bees, which introduce random solutions to maintain diversity in the population. This study enhances the classical ABC algorithm with a Q-learning-based adaptive operator selection (AOS) mechanism that dynamically guides the selection of neighborhood operators during the search process. This enhanced algorithm is referred to as the Q-Learning Driven Artificial Bee Colony (QABC) algorithm.

2.1. The ABC Framework

The optimization process begins with the random generation of an initial population of food sources, each of which represents a feasible solution to the WTA problem. The population size is set to 40 food sources, and the algorithm runs for 100,000 iterations. During the employed bee and onlooker bee phases, new candidate solutions are generated through neighborhood operators. If a candidate solution improves the objective function value, then it replaces the current solution according to a greedy selection rule.

To prevent premature convergence and maintain population diversity, a limit parameter determines when to abandon a food source. If no improvement is achieved within a predefined number of trials, the employed bee is converted to a scout bee. The scout bee then generates a new random solution. The *limit* parameter is defined as follows:

$$limit = P \times n, \tag{4}$$

where P denotes the population size and n denotes the dimension of the problem instance, that is, the number of weapons (or targets).

2.2. Neighborhood Operator Pool

Rather than depending on a single neighborhood structure, the proposed approach utilizes a pool of five permutation-based operators that are designed for combinatorial optimization problems. These operators modify the current solution in different ways, enabling both exploration and exploitation within the search space. They are as follows:

- *swap_adjacent*: Swaps two adjacent positions in the solution representation.
- *swap_random*: Randomly exchanges the assignments of two targets.
- *insert*: Removes an element from its current position and inserts it into another position.
- *block_move*: Relocates a consecutive block of assignments to another location in the sequence.
- *block_reverse*: Reverses the order of a selected consecutive block of assignments in the solution.

These operators create different neighborhood structures and allow the algorithm to explore the search space with varying intensities, striking a balance between local refinement and global diversification. Figure 1 illustrates the operators.

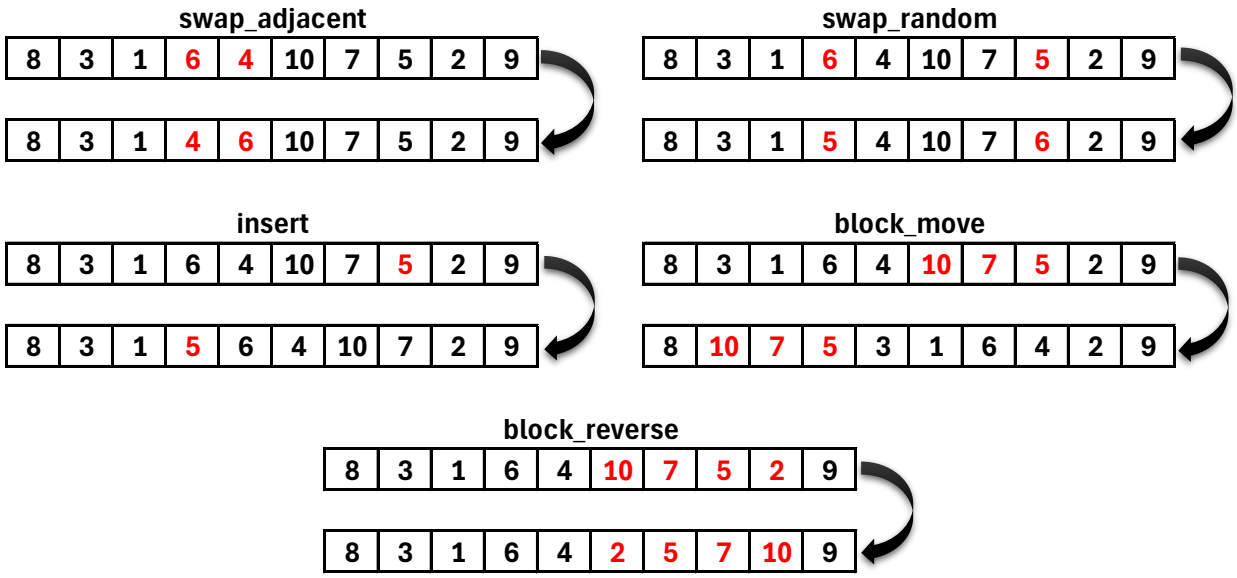

Figure 1. Graphical illustrations of the five operators.

2.3. Adaptive Operator Selection

The efficiency of each neighborhood operator may vary depending on the current stage of the search. The AOS mechanism integrated into the ABC algorithm dynamically identifies and prioritizes the most effective operators. This framework consists of two interacting components.

- Operator selection determines which operator will be applied in the current iteration, based on the current credit values.
- Credit assignment evaluates the effectiveness of the selected operator and updates its credit value accordingly.

This framework allows the algorithm to adaptively prioritize operators that generate better improvements while still permitting less frequently rewarded operators to be explored. This maintains a healthy balance of intensification and diversification throughout the search process.

2.4. Reward and Credit Assignment

Each operator receives a reward based on the improvement it produces in the objective function value. When operator i generates a candidate solution v from the current solution x at iteration t, the reward is computed as follows:

$$r_i(t) = f(x) - f(v), \tag{5}$$

where $r_i(t)$ is the reward assigned to operator i at iteration t, $f(x)$ is the objective value of the current solution, and $f(v)$ is the objective value of the newly generated candidate solution. To reduce the effect of stochastic fluctuations, reward values are not used directly. Instead, a sliding window mechanism is used to calculate the average reward over the last W iterations of each operator, enabling the algorithm to consider recent performance history rather than instantaneous improvements. The credit level of each operator is then updated using the following adaptive rule:

$$q_i(t+1) = (1-\alpha)\, q_i(t) + \alpha\, r_i(t), \tag{6}$$

where $q_i(t)$ and $q_i(t+1)$ denote the current and updated credit values of operator i, $r_i(t)$ is the calculated reward value, and α is the learning rate controlling the influence of newly obtained rewards. This update mechanism gradually favors operators that produce consistent improvements during the search.

It should be noted that the proposed mechanism is an adaptive operator selection scheme based on Q-learning, rather than a full Markov decision process formulation. The operator pool is treated as a set of actions whose value estimates q_i are updated through the incremental rule of Eq. (6), which has the same form as a one-step Q-value update with learning rate α, the immediate reward being the objective improvement produced by the operator. In contrast to a complete reinforcement-learning agent, the present formulation does not maintain an explicit state representation or an ε-greedy/Boltzmann exploration policy, exploration is instead preserved by selecting operators with a probability proportional to their current credit, which keeps lower-rewarded operators available throughout the search. The method can therefore be regarded as a value-based credit-assignment scheme situated between simple adaptive operator selection and a fully state-aware Q-learning controller, and this positioning clarifies the scope of the "Q-learning" terminology adopted in this paper.

2.5. Integration with the ABC Search Process

The adaptive operator mechanism is integrated into both the employed bee and onlooker bee phases of the ABC algorithm. During each iteration, the following occurs: (i) an operator is selected from the pool proportionally to its current credit value, (ii) the selected operator generates a candidate solution from the current food source, (iii) the candidate solution is evaluated using the WTA objective function, (iv) if the candidate solution improves the objective value, it replaces the original solution, and (v) the reward is computed and the operator credit is updated. As the search progresses, operators that contribute significantly to solution improvement are selected more frequently, while less effective operators remain available with lower selection probabilities. This dynamic adjustment improves both convergence speed and robustness when solving large-scale WTA instances. The notations used throughout the proposed QABC algorithm are summarized in Table 1 for reference. The overall procedure of the proposed method is summarized in Algorithm 1, while the common operator-application and credit-update routine invoked in both bee phases is detailed in Algorithm 2.

Table 1. Notations used in the proposed QABC algorithm.

Notation	Description
P	Population size (number of food sources)
X_i	i-th food source (solution)
V_i	Candidate solution generated from X_i
$f(X_i)$	Objective function value of X_i
q_i	Credit (quality) value of operator i
α	Learning rate used in operator credit update
$r_i(t)$	Reward obtained by operator i at iteration t
$\bar{r}_i(t)$	Average reward using a sliding window
W	Sliding window size
$W_{\min}$	Minimum window size
β	Scaling factor for adaptive window size
n	Problem size (number of weapons/targets)
O_i	Selected operator
$MaxIter$	Maximum number of iterations
t	Current iteration index
$limit$	Maximum trial limit for abandonment (scout condition)
X^*	Global best solution

Figure 2 illustrates the overall flowchart of the proposed QABC algorithm. The procedure begins with the random generation of P food sources, each representing a WTA solution, followed by the evaluation of the objective function $f(X_i)$ for each food source and the initialization of operator credits $q_i = 0$ for all operators in the pool. The main loop then iterates from $t = 1$ to $MaxIter$, executing three sequential phases in each iteration. In the employed bee phase, each employed bee X_i applies *Procedure A*, which selects an operator O_i from the pool

Algorithm 1 Q-Learning Driven Artificial Bee Colony (QABC)

Require: Population size P, maximum iterations $MaxIter$, abandonment limit $limit$, operator pool $O = \{o_1, \ldots, o_K\}$, learning rate α
Ensure: Best solution X^*

Randomly initialize P food sources X_i and evaluate $f(X_i)$
Initialize operator credits $q_j \leftarrow 0$ for $j = 1, \ldots, K$; set $trial_i \leftarrow 0$
$X^* \leftarrow \arg\min_i f(X_i)$
for $t = 1$ to $MaxIter$ **do**
  for each employed bee X_i **do** ▷ Employed bee phase
    PROCEDUREA(X_i)
  end for
  Compute onlooker selection probabilities from food-source fitness
  for each onlooker bee **do** ▷ Onlooker bee phase
    Select a food source X_i proportionally to its fitness
    PROCEDUREA(X_i)
  end for
  for each food source X_i **do** ▷ Scout bee phase
    if $trial_i > limit$ **then**
      Replace X_i with a new random solution; evaluate $f(X_i)$; $trial_i \leftarrow 0$
    end if
  end for
  Update X^* if a better solution has been found
end for
return X^*

Algorithm 2 PROCEDUREA(X_i): adaptive operator application and credit update

Select operator O_j from the pool with probability proportional to its credit q_j
Generate candidate solution V_i by applying O_j to X_i
Evaluate $f(V_i)$
if $f(V_i) < f(X_i)$ **then**
  $X_i \leftarrow V_i$; $trial_i \leftarrow 0$
else
  $trial_i \leftarrow trial_i + 1$
end if
Compute reward $r_j \leftarrow f(X_i) - f(V_i)$ ▷ Eq. (5)
Update the sliding-window average reward $\bar{r}_j$ over the last W applications of O_j
$q_j \leftarrow (1 - \alpha)\, q_j + \alpha\, \bar{r}_j$ ▷ credit update, cf. Eq. (6)

proportionally to its current credit q_i, generates a candidate solution V_i, evaluates $f(V_i)$, accepts the candidate if it improves the current solution, computes the reward (Eq. (5)), and updates the operator credit according to Eq. (6). Selection probabilities for onlooker bees are then computed based on food source fitness, after which each onlooker bee selects a food source probabilistically and applies the same *Procedure A*. Following both bee phases, each food source that has failed to improve for a predefined number of *limit* trials triggers the conversion of its corresponding employed bee into a scout bee, which generates a new random solution to restore diversity. The global best solution X^* is updated if any improvement is recorded, and the loop continues until the termination condition $MaxIter$ is reached, at which point the best solution found is returned.

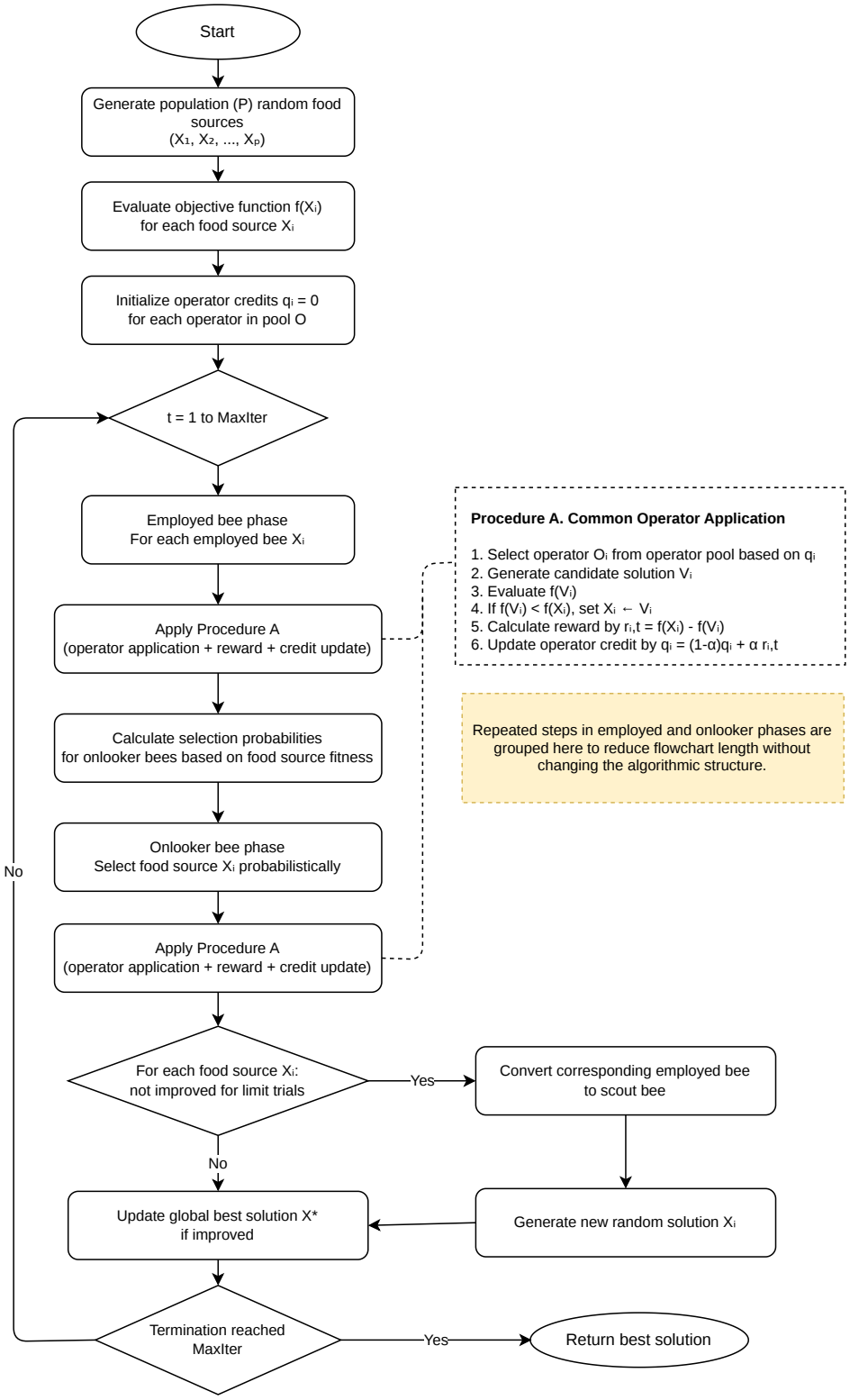

Figure 2. Overall flowchart of the proposed QABC algorithm.

2.6. Computational Complexity

Let n denote the problem size (number of weapons, which equals the number of targets), P the population size, K the number of neighborhood operators in the pool ($K = 5$), and W the sliding-window length. A solution is represented as a permutation of length n, and evaluating the WTA objective function in Eq. (1) requires forming, for each of the n targets, a product over the assigned weapons, giving a cost of $O(n^2)$ per evaluation.

Within a single iteration, each food source processed in the employed-bee phase incurs the following costs: selecting an operator proportionally to its credit, $O(K)$, applying the selected permutation operator, at most $O(n)$, evaluating the generated candidate solution, $O(n^2)$, computing the reward and updating the sliding-window average, $O(1)$ when the window sum is maintained incrementally, and updating the operator credit via Eq. (6), $O(1)$. The per-candidate cost is therefore dominated by the $O(n^2)$ objective evaluation, while the additional adaptive-operator-selection bookkeeping contributes only $O(K + n + W)$. Because the window length is bounded by $W = \max(W_{\min}, \beta n) = O(n)$, this overhead is $O(n)$ and is asymptotically dominated by the evaluation cost. Processing all P employed bees and, subsequently, all P onlooker bees yields a per-iteration cost of $O(P\,n^2)$, scout-bee replacements add at most a further $O(P\,n^2)$ in the worst case. Over $MaxIter$ iterations, the overall time complexity of QABC is $O(MaxIter \cdot P \cdot n^2)$.

3. Experimental Results

The QABC algorithm was tested on twelve WTA problem instances [3]. The results are presented using the following metrics: best, average, worst, and standard deviation. Each problem instance is denoted as *wta_n*, where n indicates the number of weapons and targets. The number of weapons and targets is equal and varies from 5 to 200 across all instances. Results of QABC on each benchmark instance are presented and compared with the Modified Crow Search Algorithm (MCSA), a recent and competitive metaheuristic which is an enhanced version of the original CSA. Similar to the ABC, it adds a trial mechanism with a limit parameter. After a predefined number of iterations, if the solution shows no improvement, the MCSA generates a new random position in the search space. This prevents premature convergence and enhances exploration. MCSA is adopted as the primary method because it was designed specifically for the static WTA problem and has been evaluated on the same benchmark instances under comparable experimental conditions, which enables a direct and fair comparison without confounding differences in problem encoding or evaluation budget. To make a fair comparison, the population size (P) is set to 40, the number of iterations ($MaxIter$) is set to 100,000 and the *limit* is set to $P \times n$. Since both algorithms perform the same number of objective-function evaluations, the comparison is conducted under an identical evaluation budget, which ensures fairness.

The sliding window size (W) determines how many recent reward observations are used to calculate the smoothed reward value ($\bar{r}_i(t)$) for each operator. This approach reduces the effect of stochastic fluctuations rather than relying on instantaneous reward signals. Instead of using a fixed window size, W adjusts dynamically according to the problem dimension (n). Specifically, W is equal to the maximum of $W_{\min}$ and $\beta \times n$, where β controls how quickly the window grows with respect to the problem size. This enables proportional adjustment, balancing responsiveness in small-scale instances with stability in large-scale problems. Additionally, $W_{\min}$ ensures that a minimum window size is always maintained. In this study, β is set to 0.1 and $W_{\min}$ is set to 5. These settings enable the window to expand proportionally to the size of the problem, while ensuring a consistent level of smoothing across all instances.

The learning rate α plays a central role in governing the exploration–exploitation balance of the algorithm. Lower values encourage long-term learning and stable operator selection, while higher values promote rapid adaptation to promising new search directions. Based on empirical sensitivity analysis, the value $\alpha = 0.3$ was selected because experimental results demonstrated that this setting provides a consistent balance between convergence speed and solution stability across different problem instances. This setting also allows the algorithm to gradually shift from exploration in early iterations to exploitation in later stages of the search.

The remaining control parameters were fixed rather than tuned per instance, in order to keep the comparison with MCSA fair and to avoid overfitting the configuration to individual instances. The adaptive-selection parameters $\alpha = 0.3$, $\beta = 0.1$, and $W_{\min} = 5$ were selected on the basis of preliminary trials on a subset of instances. We acknowledge that a systematic parameter-sensitivity study on representative instance sizes (e.g., *wta_40*, *wta_100*,

and *wta_200*) would provide a more thorough assessment of robustness, and we leave this for future work. The complete set of parameter values used in the experiments is summarized in Table 2.

Table 2. Parameter settings used in the QABC algorithm.

Parameter	Symbol	Value
Population size	P	40
Maximum iterations	$MaxIter$	100,000
Abandonment limit	$limit$	$P \times n$
Number of operators	K	5
Learning rate	α	0.3
Window scaling factor	β	0.1
Minimum window size	$W_{\min}$	5
Sliding window size	W	$\max(W_{\min}, \beta n)$

Table 3 presents the computational results of the QABC algorithm across twelve WTA benchmark instances over 30 independent runs. For the five smallest instances (*wta_5* through *wta_40*), QABC consistently yields identical best, average, and worst cost values with a standard deviation of zero, demonstrating that the algorithm reliably converges to the same solution on every run for relatively small-scale problems. This behavior suggests that the Q-learning-guided selection mechanism consistently directs the colony to the same high-quality solution without exhibiting variability at low problem dimensionalities. We note that, because exact optimal values or lower bounds are not available for all of these instances, the zero standard deviation should be interpreted as the algorithm reliably converging to the same solution on every run rather than as a proof that this solution is globally optimal. As the problem scale increases beyond *wta_50*, a gradual increase in both variance and the gap between the best and worst solutions is observed. For *wta_50*, the standard deviation is 0.0959, which remains modest. However, it rises to 1.0600 for *wta_80*, 1.6352 for *wta_90*, and 1.1375 for *wta_100*, before reaching 1.4597 for the largest instance, *wta_200*. Despite this growing variability, the absolute differences between the best and worst solutions remain comparatively small relative to the magnitude of the objective function values. For example, the worst-to-best gap for the *wta_200* instance is only 4.761, suggesting that QABC maintains stable and competitive solution quality even under large-scale conditions. As shown in Table 3, QABC consistently converges to the same zero-variance solution for small instances and consistently delivers high-quality, low-dispersion results as the number of weapons and targets increases to 200.

Table 3. Computational results of the QABC algorithm on WTA instances over 30 runs.

Instance	Best	Average	Worst	Std.
wta_5	48.3640	48.3640	48.3640	0.0000
wta_10	96.3123	96.3123	96.3123	0.0000
wta_20	142.1070	142.1070	142.1070	0.0000
wta_30	248.0285	248.0285	248.0285	0.0000
wta_40	305.5016	305.5016	305.5016	0.0000
wta_50	353.0102	353.1114	353.2426	0.0959
wta_60	414.2222	414.5997	414.8914	0.2507
wta_70	495.7751	495.9938	496.0474	0.0793
wta_80	530.5739	532.7111	534.4484	1.0600
wta_90	588.7246	592.2146	594.5036	1.6352
wta_100	693.2872	695.7513	697.9232	1.1375
wta_200	1281.4460	1283.4421	1286.2074	1.4597

Table 4 shows the comparative results of QABC against MCSA which is a robust optimization algorithm on WTA instances. As in the previous experiment, each metric is calculated over 30 runs. The performance of QABC was further assessed using the Wilcoxon signed-rank test, a standard nonparametric procedure. The following notation is employed: "$>$" (or "$<$") indicates that QABC performs significantly better (or worse) than MCSA at the 95% confidence level, "$\geq$" (or "$\leq$") denotes a marginal but statistically insignificant advantage (or disadvantage) of QABC over MCSA, and "$\approx$" indicates that the two approaches have comparable average performance. For the three smallest instances (*wta_5*, *wta_10*, and *wta_20*), both algorithms achieve identical average costs and zero standard deviations. This reflects the trivial nature of these instances rather than any algorithmic equivalence. However, starting with *wta_30*, QABC outperforms MCSA consistently and significantly in the remaining nine instances. As reported in the last column of Table 4, the Wilcoxon signed-rank test yields p-values below the 0.05 significance threshold for all nine of these instances, with $p < 10^{-8}$ from *wta_50* onwards, indicating that QABC produced the lower-cost solution on essentially every paired run. The test is not applicable to *wta_5*, *wta_10*, and *wta_20*, for which all paired differences are exactly zero.

Table 4. A comparison of the proposed QABC algorithm to the MCSA.

	QABC		vs.	MCSA		
Instance	Average	Std.		Average	Std.	p-value
wta_5	48.3640	0.0000	$\approx$	48.3640	0.0000	–
wta_10	96.3123	0.0000	$\approx$	96.3123	0.0000	–
wta_20	142.1070	0.0000	$\approx$	142.1070	0.0000	–
wta_30	248.0285	0.0000	$>$	248.3055	0.2954	7.45×10^{-5}
wta_40	305.5016	0.0000	$>$	306.3826	0.7199	1.73×10^{-6}
wta_50	353.1114	0.0959	$>$	354.4785	0.9920	1.86×10^{-9}
wta_60	414.5997	0.2507	$>$	418.0557	1.6457	3.73×10^{-9}
wta_70	495.9938	0.0793	$>$	501.3469	1.8573	1.86×10^{-9}
wta_80	532.7111	1.0600	$>$	538.6588	1.9237	1.86×10^{-9}
wta_90	592.2146	1.6352	$>$	598.4644	2.0272	1.86×10^{-9}
wta_100	695.7513	1.1375	$>$	705.3456	2.7312	1.86×10^{-9}
wta_200	1283.4421	1.4597	$>$	1314.7862	3.7881	1.86×10^{-9}

The performance gap widens markedly as the problem size increases. For *wta_30* and *wta_40*, the differences in average cost are modest yet statistically significant at 0.277 and 0.881, respectively. As dimensionality increases, QABC's advantage becomes substantially more pronounced. The gap reaches 3.456 for *wta_60*, 5.353 for *wta_70*, 5.948 for *wta_80*, and 6.250 for *wta_90*. The difference is most striking for the two largest instances: 9.594 for *wta_100* and 31.344 for *wta_200*. These results suggest that the scalability advantage of QABC over MCSA increases with problem dimensionality. This is consistent with the hypothesis that the Q-learning component of QABC enables more efficient navigation of the exponentially growing search space characteristic of large-scale WTA problems. QABC also exhibits considerably lower standard deviations than MCSA in all instances where a meaningful comparison is possible, reflecting greater robustness and consistency across independent runs. For example, at *wta_200*, the standard deviation of QABC (1.4597) is less than half that of MCSA (3.7881), which further underscores QABC's superior stability under large-scale conditions.

The box plots in Figure 3 visually corroborate the statistical findings in Table 4. For the three smaller instances included in the figure (*wta_30*, *wta_40*, and *wta_50*), the QABC boxes are compressed compared to the MCSA boxes. The QABC interquartile range for *wta_30* and *wta_40* appears as a nearly flat line, whereas the MCSA distributions span a wider range and have visible outliers. This reflects the greater solution variability of the benchmark competitor. From *wta_60* onwards, the separation between the two distributions is clear. The MCSA median and mean consistently lie well above the entire QABC distribution. This means that even the worst QABC solutions are frequently superior to or comparable with the median MCSA solution. This pattern of non-overlap becomes increasingly evident for *wta_70*, *wta_80*, *wta_90*, *wta_100*, and *wta_200*, where the MCSA boxes are

positioned entirely above the QABC boxes, with no overlap in their interquartile ranges. The MCSA distributions also show wider spreads and occasional high-cost outliers, indicating inconsistent performance. Overall, Figure 3 supports the conclusion that QABC achieves lower-cost solutions on average with markedly greater reliability across all medium- and large-scale WTA instances.

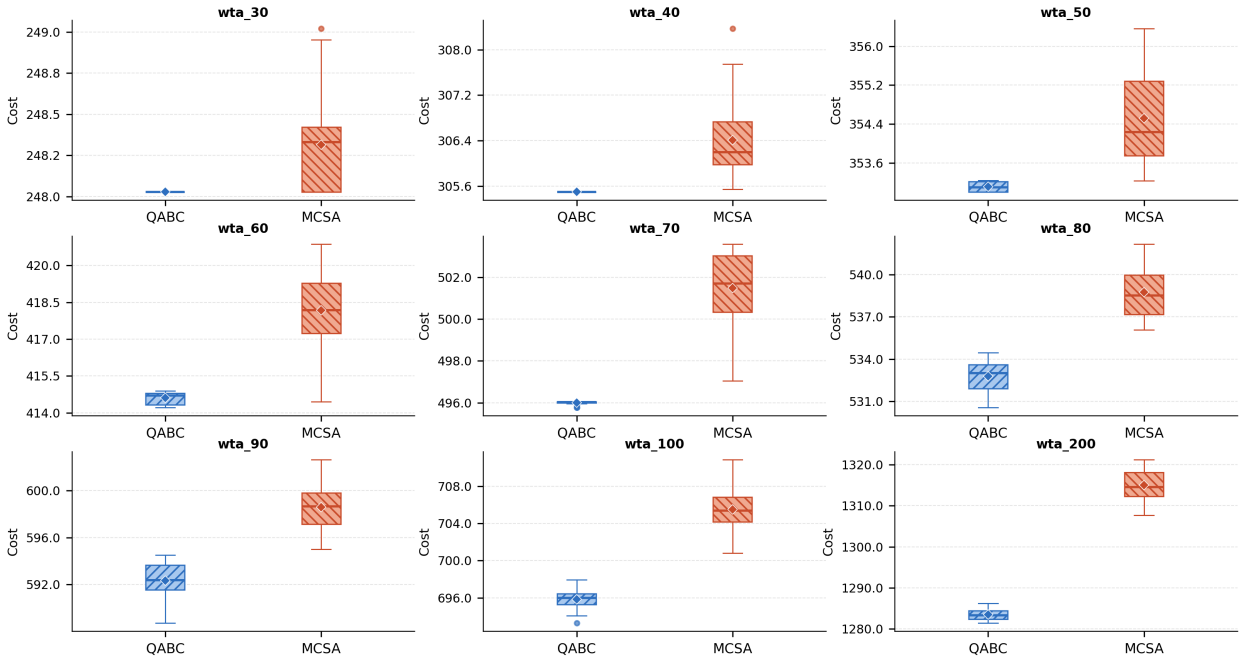

Figure 3. Box plots comparing the cost values obtained by QABC and MCSA over 30 independent runs across nine WTA benchmark instances.

Figure 4 shows the convergence graphs for the proposed QABC and MCSA algorithms on the *wta_40*, *wta_70*, *wta_100*, and *wta_200* instances over 100,000 iterations with a population size of 40. As can be seen in the figure, the optimization behaviors corroborate the statistical superiority of QABC reported in Tables 3 and 4. MCSA exhibits a markedly faster initial descent. It reaches near-plateau cost values within the first few thousand iterations across all configurations, after which it exhibits negligible further improvement. QABC, on the other hand, follows a slower, more gradual trajectory. This trend is characterized by a steady, continuous decline punctuated by stepwise reductions. This is most visible around 45,000 iterations for *wta_40*. These reductions enable continued improvements even in the later stages of the search. Although MCSA can quickly find good solutions at the start, QABC ultimately has lower costs by the end of the trial. For example, QABC obtains approximately 305.5016 for *wta_40*, whereas MCSA obtains around 305.6929. Similarly, QABC obtains approximately 1281.4460 for *wta_200*, while MCSA obtains around 1308.7077. This demonstrates that QABC is more effective at identifying optimal solutions and leveraging these solutions to discover even better ones. Although the initial convergence rate is slower, the Q-learning component provides a better long-term solution. Overall, these curves demonstrate a consistent and interpretable trend. MCSA prioritizes speed over quality by exploiting a narrow neighborhood. In contrast, Q-learning in QABC adapts its strategy based on feedback. This property becomes more advantageous as dimensionality and the search space increase. This explains the growing performance gap across instances in Table 4. The comparison is limited to a single method, MCSA, because it is the most recent best metaheuristic tested on the same twelve static WTA instances. This allows a fair and direct comparison under the same problem settings and evaluation conditions.

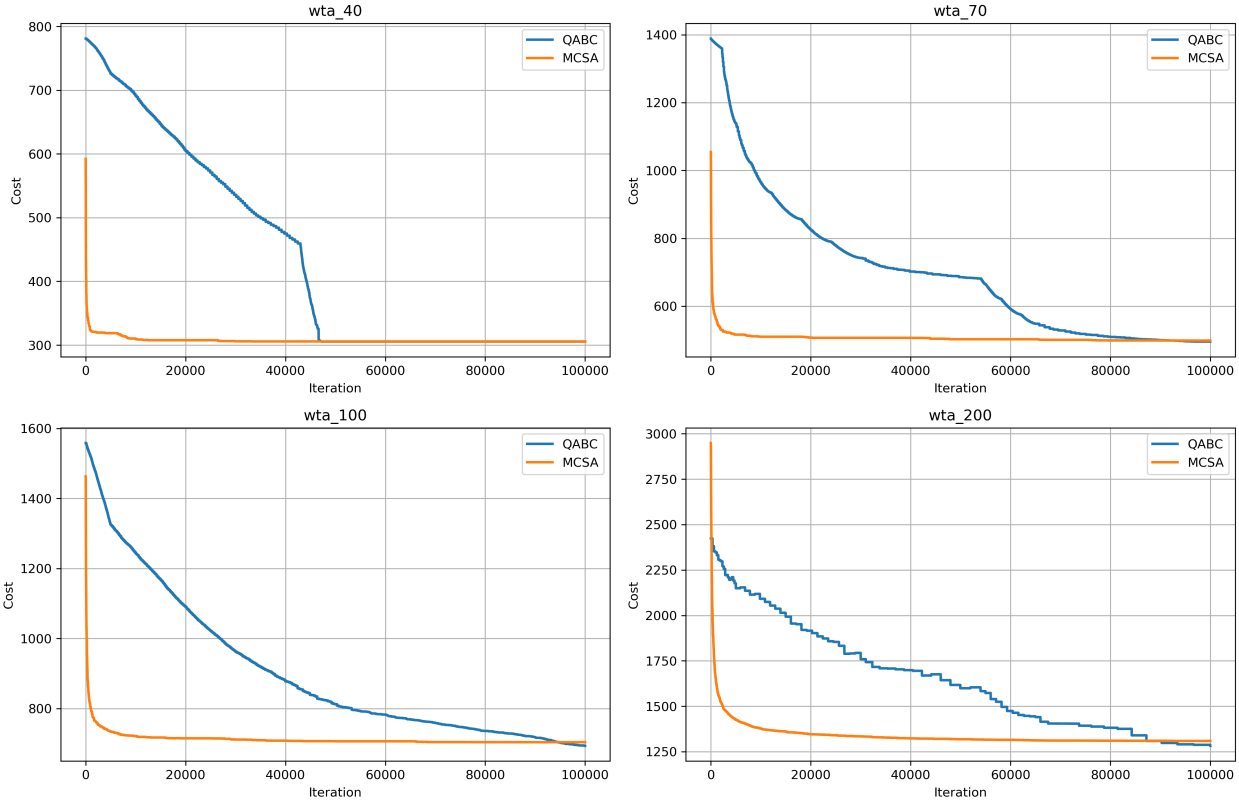

Figure 4. Convergence curves of QABC and MCSA on *wta_40*, *wta_70*, *wta_100*, and *wta_200* instances.

4. Conclusion

This study introduced the Q-learning-driven artificial bee colony (QABC) algorithm, an enhanced variant of the standard artificial bee colony (ABC) algorithm augmented with an adaptive operator selection (AOS) mechanism based on Q-learning. The QABC algorithm was then applied to the weapon-target assignment (WTA) problem. The AOS mechanism adjusts the selection probabilities of five permutation-based neighborhood operators based on their recent performance history. This enables the algorithm to balance exploration and exploitation more effectively throughout the search process.

Twelve benchmark WTA instances ranging from five to 200 weapons and targets were used for experimental evaluations, with each instance evaluated over 30 independent runs. For the five smallest instances, QABC achieved zero-variance convergence, yielding identical best, average, and worst solutions across all runs. As the dimensionality of the problem increased, QABC maintained competitive and stable solution quality. The performance advantage over the state-of-the-art Modified Crow Search Algorithm (MCSA) became increasingly pronounced. On the largest *wta_200* instance, the average cost gap reached 31.344, and the standard deviation of QABC remained less than half that of MCSA. This confirms QABC's superior robustness under large-scale conditions. A Wilcoxon signed-rank test confirmed statistically significant improvements in nine of the twelve instances at the 95% confidence level.

These results suggest that incorporating Q-learning into swarm-based metaheuristics is a promising approach for solving combinatorial optimization problems. The AOS driven by Q-learning enables the algorithm to transition from broad exploration in initial iterations to targeted exploitation in subsequent stages. This proves to be especially beneficial as the search space exponentially expands with the growth of the problem size. Future work may extend

the QABC framework to dynamic WTA formulations, investigate alternative reinforcement learning strategies, such as deep Q-networks, or apply the proposed mechanism to other NP-hard combinatorial problems.

REFERENCES

1. R. K. Ahuja, A. Kumar, K. C. Jha, and J. B. Orlin, *Exact and heuristic algorithms for the weapon–target assignment problem*, Operations Research, vol. 55, no. 6, pp. 1136–1146, 2007.
2. O. Akbel, and A. Kalaycıoğlu, *A solution to dynamic weapon assignment problem based on game theory for naval platforms*, Games, vol. 15, no. 5, p. 33, 2024.
3. S. Alparslan, and E. Sonuç, *Solving static weapon-target assignment problem using multi-start late acceptance hill climbing*, Current Trends in Computing, vol. 2, no. 1, pp. 23–35, 2024.
4. G. Basheer, L. Mohammed, and Z. Algamal, *Solving 0–1 knapsack problem by an improved binary monarch butterfly algorithm*, Statistics, Optimization & Information Computing, vol. 15, no. 4, pp. 2382–2396, 2026.
5. D. Bertsimas, C. Pawlowski, and Y. D. Zhuo, *Solving large-scale weapon–target assignment problems in seconds using branch-price-and-cut*, Naval Research Logistics, 2023.
6. A. Ghanbari, S. Kazemi, and F. Mehmanpazir, *A survey on weapon target allocation models and applications*, IntechOpen, 2021.
7. Z. Guangsheng, L. Jing, and L. Chao, *A survey of intelligent optimization algorithms for weapon target assignment problem*, in Proceedings of the International Conference on Modeling, Simulation, Identification and Control, pp. 62–67, 2020.
8. K. C. Jha, *Very large-scale neighborhood search heuristics for combinatorial optimization problems*, Doctoral dissertation, University of Florida, 2004.
9. Z. Jinshuai, Y. Yongjie, and Y. Ruiying, *Optimization of weapon-target assignment problem by intuitionistic fuzzy genetic algorithm*, MATEC Web of Conferences, vol. 128, 02004, 2017.
10. B. A. Julstrom, *String- and permutation-coded genetic algorithms for the static weapon–target assignment problem*, in Proceedings of the Genetic and Evolutionary Computation Conference, pp. 2553–2558, 2009.
11. D. Karaboga, and B. Basturk, *A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm*, Journal of Global Optimization, vol. 39, no. 3, pp. 459–471, 2007.
12. K. Kim, D. Lee, and I. Hwang, *A Study on Weapon–target assignment considering heading error*, International Journal of Aeronautical and Space Sciences, vol. 25, pp. 880–893, 2024.
13. A. Kline, D. Ahner, and R. Hill, *The weapon-target assignment problem*, Computers & Operations Research, vol. 105, pp. 226–236, 2019.
14. A. G. Kline, D. Ahner, and B. J. Lunday, *A heuristic and metaheuristic approach to the static weapon-target assignment problem*, Journal of Global Optimization, vol. 78, no. 4, pp. 791–812, 2020.
15. O. Kwon, K. Lee, and S. Park, *A branch-and-price algorithm for a targeting problem*, Naval Research Logistics, vol. 54, no. 7, pp. 732–741, 2007.
16. Z.-J. Lee, *Constrained weapon–target assignment: enhanced very large-scale neighborhood search algorithm*, IEEE Transactions on Systems, Man, and Cybernetics – Part A, vol. 40, no. 1, pp. 198–204, 2010.
17. J. Li, G. Wu, and L. Wang, *A comprehensive survey of weapon target assignment problem: model, algorithm, and application*, Engineering Applications of Artificial Intelligence, vol. 137, part B, 109212, 2024.
18. S. P. Lloyd, and H. S. Witsenhausen, *Weapons allocation is NP-complete*, in 1986 Summer Computer Simulation Conference, pp. 1054–1058, 1986.
19. J. Lu, J. Wei, X. Yu, D. Song, and Y. Bei, *A new exact algorithm for the weapon–target assignment problem*, Omega, vol. 98, 102138, 2021.
20. A. M. Madni, C. C. Lin, and C. D. Madni, *Efficient heuristic approach to the weapon-target assignment problem*, Journal of Aerospace Computing, Information, and Communication, vol. 6, no. 3, pp. 291–307, 2009.
21. A. S. Manne, *A target-assignment problem*, Operations Research, vol. 6, no. 3, pp. 346–351, 1958.
22. M. Mekawey, A. Elsayed, M. Darwish, and H. Mostafa, *Novel goal-based weapon target assignment doctrine*, Journal of Aerospace Computing, Information, and Communication, vol. 6, no. 3, pp. 225–238, 2009.
23. N. Nader, M. Shafiei, and M. Nematbakhsh, *A hybrid BA–VNS algorithm for solving the weapon-target assignment considering mobility of resources*, Economic Computation and Economic Cybernetics Studies and Research, vol. 57, no. 3, pp. 53–68, 2023.
24. M. Ni, Z. Yu, F. Ma, and X. Wu, *A Lagrange relaxation method for solving weapon–target assignment problem*, Mathematical Problems in Engineering, vol. 2011, 873292, 2011.
25. C. R. Parson, *Approximate dynamic programming for military resource allocation*, Doctoral dissertation, Air Force Institute of Technology, 2014.
26. J. M. Rosenberger, E. L. Johnson, and G. L. Nemhauser, *The generalized weapon target assignment problem*, Working paper, 2005.
27. Z. Shamami, and M. Dehghan, *War game problem considering the mobility of weapons and targets*, Journal of Engineering Research, 2023.
28. H. D. Soneji, *A comparison of agent-based optimization approaches applied to the weapons to targets assignment planning problem*, Master's thesis, Wright State University, 2006.
29. E. Sonuc, B. Sen, and S. Bayir, *A parallel simulated annealing algorithm for weapon-target assignment problem*, International Journal of Advanced Computer Science and Applications, vol. 8, no. 4, pp. 87–92, 2017.
30. E. Sonuç, *A modified crow search algorithm for the weapon-target assignment problem*, An International Journal of Optimization and Control: Theories & Applications, vol. 10, no. 2, pp. 188–197, 2020.
31. A. Tokgöz, and S. Bulkan, *Weapon target assignment with combinatorial optimization techniques*, International Journal of Advanced Research in Artificial Intelligence, vol. 2, no. 7, pp. 39–50, 2013.

32. K. Volle, D. Kress, and R. Pesch, *Weapon–target assignment algorithm for simultaneous and sequenced arrival*, Journal of Guidance, Control, and Dynamics, vol. 41, no. 10, pp. 2361–2370, 2018.
33. T. Wang, L. Fu, Z. Wei, Y. Zhou, and S. Gao, *Unmanned ground weapon target assignment based on deep Q-learning network with an improved multi-objective artificial bee colony algorithm*, Engineering Applications of Artificial Intelligence, vol. 117, 2023.
34. C. Zhihua, L. Mingfa, and Z. Wenhua, *A heuristic genetic algorithm for solving constrained weapon-target assignment problem*, in Proceedings of the International Conference on Intelligent Computing and Intelligent Systems, pp. 336–340, 2009.
35. S. Zou, X. Shi, and S. Song, *MOEA with adaptive operator based on reinforcement learning for weapon target assignment*, Electronic Research Archive, vol. 32, no. 3, pp. 1498–1532, 2024.