

## REINFORCEMENT LEARNING-BASED LOCAL SEARCH FOR SOLVING THE QUADRATIC 3-DIMENSIONAL ASSIGNMENT PROBLEM

WALID BENZINEB\*<sup></sup>, LAKHDAR LOUKIL<sup></sup>, BAKHTA AMRANE AND ABOU EL HASSAN BENYAMINA<sup></sup>

**Abstract.** Local search methods (LSM) are powerful metaheuristics known for efficiently exploring complex optimization landscapes. However, a key limitation is their “amnesiac” nature – they fail to utilize valuable data from past search iterations. This untapped data contains crucial information that can enhance the efficiency and outcomes of the search process. In this work, we propose a novel reinforcement learning (RL)-based move scoring mechanism to augment LSM, aiming to leverage historical data for improved adaptiveness and intelligence in the search process. We apply this hybrid method to the quadratic 3-dimensional assignment problem (Q3AP), a benchmark in combinatorial optimization that generalizes the quadratic assignment problem (QAP). Notably, few studies have focused on Q3AP, and to our knowledge, no prior work has integrated RL into its solution. Our experimental results demonstrate substantial improvements in both solution quality and execution time, particularly for larger and more complex problem instances. This performance gain is attributed to our unique iterative learning and scoring approach, which effectively guides the search process, balancing exploration and exploitation to escape local optima and find superior solutions.

**Mathematics Subject Classification.** 90C27, 68T05, 90C59.

Received May 20, 2024. Accepted April 23, 2025.

### 1. INTRODUCTION

The increasing prevalence of large datasets across various domains has amplified the significance of efficient optimization techniques. Many real-world problems, ranging from logistics and scheduling to resource allocation and facility layout, can be formulated as complex optimization problems. These problems often demand sophisticated computational approaches to find near-optimal solutions within reasonable timeframes. Local search methods (LSMs), such as iterated local search (ILS) [1], simulated annealing [2], tabu search [3], and variable neighborhood search [4], remain indispensable tools for navigating search spaces. While these methods share a common framework of iterative neighborhood exploration (Algorithm 1), their effectiveness largely depends on move selection strategies that balance exploration and exploitation.

A critical limitation of conventional LSMs lies in treating multiple search runs as independent trials, discarding valuable historical data that could inform future iterations. This observation motivates the integration

---

*Keywords.* Metaheuristics, local search, reinforcement learning, tabu search, simulated annealing, variable neighborhood search, quadratic 3-dimensional assignment problem (Q3AP).

Faculty of Exact and Applied Sciences, Department of Computer Science, University of Oran 1, Oran, Algeria.

\*Corresponding author: [benzineb.walid@edu.univ-oran1.dz](mailto:benzineb.walid@edu.univ-oran1.dz); [benzinebwal@gmail.com](mailto:benzinebwal@gmail.com)

of reinforcement learning (RL) to transform LSMs into adaptive systems that improve through experience [5]. RL's trial-and-error paradigm aligns naturally with LSM mechanics, enabling agents to develop enhanced move selection policies by maximizing cumulative rewards from environment interactions.

We demonstrate this through the quadratic three-dimensional assignment problem (Q3AP), an NP-hard challenge where traditional LSMs show limitations [6, 7]. While RL has succeeded in others optimization problems, its application to Q3AP remains unexplored despite the problem's relevance to wireless communications and scheduling [7, 8]. This article fill this gap by proposing a hybrid methodology that integrates reinforcement learning into traditional local search algorithms. The structure of the paper is as follows: Section 2 provides a comprehensive review of related work, concentrating on the integration of machine learning techniques with optimization methods to enhance their performance. Section 3 delves into the fundamental concepts of the reinforcement learning paradigm and elucidates its significance within our proposed methodology. The specifics of our RL-enhanced local search approach are then detailed. Section 4 presents a thorough evaluation of this approach. We discuss the implementation of our hybrid algorithm, which combines reinforcement learning with two prominent LSMs: simulated annealing (SA) and variable neighborhood search (VNS). Both SA and VNS have demonstrated their effectiveness across a wide range of optimization problems [9–12], with SA employing a stochastic approach inspired by physical annealing processes to discover global optima, and VNS leveraging multi-level neighborhood structures for efficient exploration of the search space. Finally, in Section 4.3, we present a comparative study between standalone SA and VNS algorithms *versus* their reinforcement learning-enhanced counterparts: SA+RL and VNS+RL, respectively.

---

**Algorithm 1.** Template of a local search procedure.

---

```

1: procedure LS
2:   Input: Initial solution  $s_0$ 
3:   Output:  $s_t$ 
4:    $t = 0$ ;
5:   while (stopping criterion not met) do
6:      $\text{move}_t = \text{SELECTCANDIDATEMOVE}(s_t, N(s_t))$ ;
7:      $s_{t+1} = \text{MAKEMOVE}(s_t, \text{move}_t)$ ;
8:      $t = t + 1$ ;
9:   end while
10: end procedure

```

---

## 2. RELATED WORK

The integration of machine learning (ML) techniques with heuristic algorithms has emerged as a powerful and increasingly prevalent paradigm for tackling complex optimization problems. This synergy is particularly pronounced within the realm of metaheuristics, where ML offers the potential to significantly enhance both adaptability and efficiency. Karimi-Mamaghan *et al.* [13] provide a comprehensive review and insightful taxonomy of these advancements, highlighting the strengths of ML-enhanced approaches in scenarios where traditional heuristics often falter, such as navigating intricate local optima landscapes or requiring extensive manual parameter tuning. Supporting this trend, recent surveys [14, 15] further underscore the growing significance of this integration, with researchers actively exploring the use of ML to guide and augment metaheuristic search processes, leading to the development of more robust and efficient optimization algorithms. These studies effectively demonstrate how various ML techniques, with a particular emphasis on reinforcement learning (RL), can dynamically adapt the parameters of metaheuristics during the search, resulting in improved performance across a diverse range of problem instances.

Within the specific context of optimization problems, reinforcement learning (RL) has distinguished itself as a particularly promising and effective approach. Through iterative interactions with a defined environment, RL agents leverage feedback mechanisms to refine their strategies for tackling complex optimization challenges

[16, 17]. The inherent strength of RL lies in its capacity to learn and adapt through experience, making it exceptionally valuable for addressing complex combinatorial optimization (CO) problems characterized by vast solution spaces and non-intuitive relationships between decisions and their outcomes. Current research efforts are actively addressing key challenges associated with applying RL to combinatorial problems, focusing on the development of novel training methodologies, enhanced exploration strategies, and innovative architectures that effectively leverage graph-based representations of problem structures [18, 19].

In the domain of combinatorial optimization (CO), RL has demonstrated remarkable potential for automating and optimizing problem-solving strategies [20]. Both value-based and policy-based RL approaches, notably including policy gradient methods and Proximal Policy Optimization (PPO) [21], have exhibited significant promise. Mazyavkina *et al.* [20] offer a comprehensive review of RL applications in CO, showcasing successful implementations across various domains, including routing, assignment, and scheduling. A noteworthy achievement in this area is the work by Bello *et al.* [22], whose RL-based approach achieved impressive results, attaining solutions within 1–2% of the optimal for Traveling Salesperson Problem (TSP) instances with up to 100 nodes, while also demonstrating greater adaptability compared to traditional heuristics. Continuing this trend, recent studies are actively advancing the application of RL to a diverse array of CO problems, including further explorations of the Traveling Salesperson Problem (TSP) and the Vehicle Routing Problem (VRP) [23–25]. Furthermore, the field is witnessing a growing interest in multi-agent reinforcement learning (MARL) as a means of addressing complex problems involving multiple interacting agents, such as resource allocation and task assignment scenarios [26, 27].

Focusing specifically on the Quadratic 3-Dimensional Assignment Problem (Q3AP), previous research has primarily concentrated on exact solution methods and metaheuristic approaches. Mittelman and Salvagnin [6] investigated mixed-integer programming techniques, cutting planes, and symmetry-handling strategies for tackling challenging Q3AP instances, ultimately highlighting the inherent computational limitations of these exact approaches. Hahn *et al.* [7] made significant contributions through a branch-and-bound algorithm employing Reformulation-Linearization Technique (RLT) bounds, demonstrating effectiveness for instances up to size 13 but facing exponential growth in computation time for larger problems. Their work also adapted stochastic local search (SLS) methods from Quadratic Assignment Problem (QAP) research, including simulated annealing, tabu search, ant colony optimization, and iterated local search (ILS), with ILS emerging as the most effective in finding optimal solutions 1–2 orders of magnitude faster than exact methods for small instances. Notably, their study emphasized practical applications in wireless communication systems, where Q3AP solutions optimize symbol mapping diversity for hybrid ARQ protocols to combat channel fading effects [7]. In the realm of metaheuristics, Loukil *et al.* [8] introduced a parallel hybrid genetic algorithm with simulated annealing to solve Q3AP on computational grids, demonstrating promising results on benchmark instances. More recently, Ait Abderrahim *et al.* [28] proposed a hybrid approach that combines particle swarm optimization (PSO) with a local search algorithm, achieving competitive results compared to PSO and genetic algorithms alone. In a related area, reinforcement learning applications have shown increasing promise in addressing the Quadratic Assignment Problem (QAP), problem closely related to Q3AP. Researchers have demonstrated that novel RL architectures, such as double pointer networks [29], can effectively match or even surpass the performance of high-quality local search baselines for the QAP, thereby establishing the potential of RL for tackling computationally complex assignment problems.

Despite these advancements in both RL-enhanced heuristics and traditional solution methods for the QAP, along with the progress in metaheuristic approaches for the Q3AP, a significant gap remains in the direct application of RL to the Q3AP itself. Given the inherent computational complexity of Q3AP and the limitations of traditional methods to effectively solve large instances, the demonstrated success of RL in related CO problems strongly suggests its potential value for Q3AP. The ability of RL to learn effective strategies within high-dimensional solution spaces and to manage complex interdependencies between decisions aligns particularly well with the challenges posed by the Q3AP. Our work directly addresses this gap by exploring the application of RL techniques to Q3AP, aiming to combine the robustness of local search methods with the adaptive learning capabilities of RL.

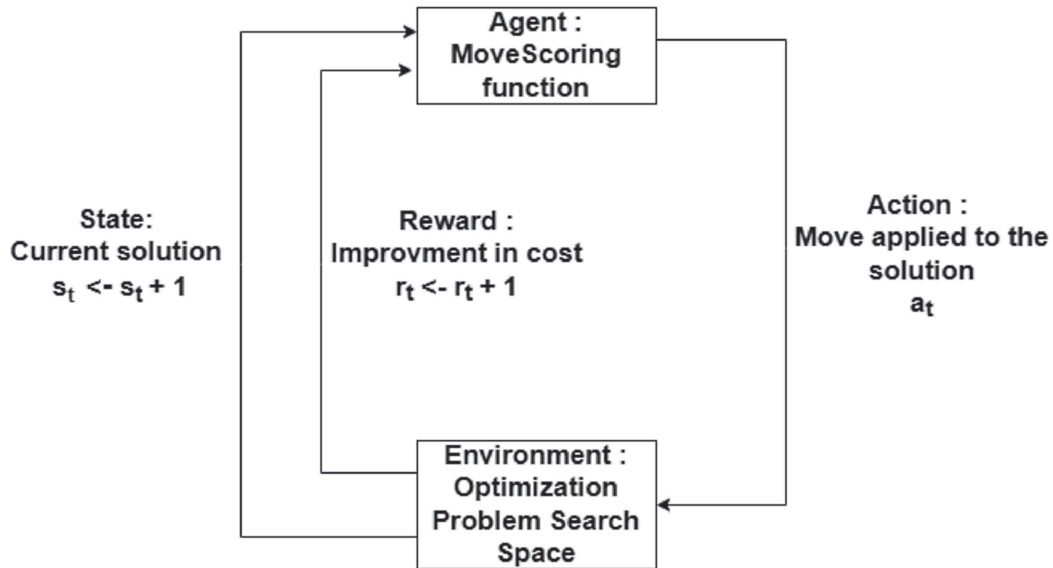


FIGURE 1. Reinforcement learning score-based scheme.

### 3. REINFORCEMENT LEARNING-BASED LOCAL SEARCH METHOD

#### 3.1. Reinforcement learning

Reinforcement learning (RL) is a subfield of machine learning that emphasizes training agents to make optimal decisions by interacting with an environment to achieve a certain goal [30]. In our approach, the local search process is viewed as an agent operating within the search space environment. The agent's actions are the possible moves within the neighborhood of the current solution, and the reward is the change in the objective function value resulting from a move. Formally, an RL framework is modeled as a Markov Decision Process (MDP), defined by:

- **States** ( $s \in \mathbf{S}$ ): Each feasible solution in the search space represents a state.
- **Actions** ( $a \in \mathbf{A}$ ): Possible moves that can be applied to a solution to reach a neighboring solution. The specific actions depend on the neighborhood structure of the local search method.
- **Reward Function** ( $\mathbf{R} : \mathbf{S} \times \mathbf{A} \rightarrow \mathbb{R}$ ): This function assigns a reward after taking an action from a given state. In our case, the reward is implicitly linked to the change in the objective function value after a move.
- **Policy** ( $\pi : \mathbf{S} \rightarrow \mathbf{A}$ ): The policy determines the action to take in each state.

An essential aspect of RL is the trade-off between exploration and exploitation. Exploration refers to the agent taking actions that it is uncertain about, potentially leading to discovering new optimal strategies. Exploitation, in contrast, involves taking actions that the agent currently considers optimal based on its acquired knowledge [30]. A well-designed RL algorithm will strike a balance between exploration and exploitation to achieve superior performance through an adaptive exploration mechanism (Sect. 3.2).

In this work, a model-free and value-based reinforcement learning agent has been designed to improve local search methods. The agent learns an action-value function by interacting with the environment and acquiring knowledge from previous experiences without explicitly modeling the environment's dynamics. The advantage of using such an agent lies in its ability to accumulate knowledge from prior search instances and leverage this knowledge to guide the exploration of the search space more effectively (Fig. 1).

### 3.2. Reinforcement learning-based local search method

Our approach implements a free value-based reinforcement learning framework to enhance local search performance. Instead of relying on heuristic or random selection, we develop a learning agent that directly estimates the value of potential moves through a scoring mechanism that adapts based on observed improvements in solution quality. This model-free approach allows the agent to learn directly from experience without maintaining explicit state-action mappings.

The scoring mechanism (Algorithm 3) implements three key components of value-based learning:

**Direct Value Estimation:** Move scores in `moveScoreList` directly represent the estimated value of each move based on its historical effectiveness. Unlike Q-learning approaches, these scores are independent of the state space and are updated based purely on the observed improvements they generate. This makes the approach particularly efficient for large search spaces where maintaining state-action pairs would be impractical.

**Value Updates:** When a move is executed, its score is updated based on its immediate contribution to improving the solution:

$$\text{moveScore}_{\text{new}} = \text{moveScore}_{\text{current}} + \text{solScore} \times \delta,$$

where  $\delta$  represents the normalized improvement:

$$\delta = \frac{f(s_{t+1}) - f(s_t)}{f(s_{\text{finalSol}}) - f(s_{\text{initSol}})} \times 100.$$

This update rule allows the agent to learn directly from experience without requiring a model of the environment.

**Adaptive Exploration:** The scoring system naturally implements an exploration-exploitation balance through the interaction between move scores and solution quality. The multiplicative relationship between `solScore` and the improvement contribution  $\delta$  allows for adaptive exploration – moves receive stronger updates when they contribute to high-quality solutions.

Algorithm 2 implements this free value-based learning through three interconnected procedures: **Scoring** is the main procedure that manages the learning process. It determines when solutions represent improvements (lines 4–6) and coordinates the value estimation through solution and move scoring (lines 8, 11). Rather than maintaining state-dependent values, it focuses on tracking overall solution quality to guide the learning process.

**MovesScoring** implements the core learning mechanism of our value-based approach. It:

- Initializes new moves with a baseline value (`DEFAULTSCORE`) to encourage initial exploration;
- Computes the normalized improvement contribution ( $\delta$ ) to measure move effectiveness;
- Updates move scores using a direct value update rule that combines immediate improvements with solution quality;
- Maintains a memory of move effectiveness through the persistent `moveScoreList`.

**SolScoring** provides a context for value updates by evaluating solution quality relative to the best known solution. This scaling helps prioritize learning from moves that contribute to high-quality solutions while maintaining sufficient exploration of the search space.

The learned values guide the move selection process, where moves are selected probabilistically based on their accumulated scores. This creates an adaptive local search that becomes increasingly efficient by learning directly from search experience, without requiring explicit modeling of the state-action space.

### 3.3. Integration of value-based agent with local search

The integration of our free value-based learning approach with traditional local search creates a hybrid method that leverages both learned knowledge and heuristic search strategies. Algorithm 3 presents this integration, which centers around two key mechanisms: an adaptive move selection and a fallback strategy.

**Adaptive Move Selection:** The `SelectPossibleMoves` procedure implements the core decision-making mechanism of our value-based agent. It:

---

**Algorithm 2.** Moves and solution scoring pseudo-code.

---

```

1: procedure SCORING(initSol, finalSol, sol, bestSol, solMoveList, solScoreList, moveScoreList)
2:   Input: initSol, finalSol, sol, bestSol, solMoveList, solScoreList, moveScoreList
3:   Output: solScoreList, moveScoreList
4:   if  $f(\text{sol}) < f(\text{bestSol})$  then
5:     bestSol  $\leftarrow$  sol
6:     solScore  $\leftarrow$  MAXSCORE
7:   else
8:     SOLSCORING(sol, bestSol, solScore)
9:   end if
10:  UPDATESOLSCORELIST(solScore, solScoreList)
11:  MOVESCORING(initSol, finalSol, solScore, solMoveList, moveScoreList)
12: end procedure
13:
14: procedure MOVESCORING(initSol, finalSol, solScore, solMoveList, moveScoreList)
15:   Input: initSol, finalSol, solScore, solMoveList, moveScoreList
16:   Output: moveScoreList
17:    $i \leftarrow 1$ 
18:    $j \leftarrow 0$ 
19:    $S_j \leftarrow$  initSol
20:    $\Delta \leftarrow f(\text{finalSol}) - f(\text{initSol})$ 
21:   while  $i \leq \text{len}(\text{solMoveList})$  do
22:     if not(solMoveList[ $i$ ]  $\in$  moveScoreList) then
23:       moveScore  $\leftarrow$  DEFAULTSCORE
24:     else
25:       moveScore  $\leftarrow$  GETMOVESCORE(solMoveList[ $i$ ], moveScoreList)
26:     end if
27:      $\delta \leftarrow ((f(S_i) - f(S_j)) \times 100) / \Delta$ 
28:     moveScore  $\leftarrow$  moveScore + (solScore  $\times$   $\delta$ )
29:     UPDATEMOVESCORELIST(moveScore, moveScoreList)
30:      $j \leftarrow j + 1$ 
31:      $i \leftarrow i + 1$ 
32:   end while
33: end procedure
34:
35: procedure SOLSCORING(sol, bestSol, solScore)
36:   Input: sol, bestSol, solScore
37:   Output: solScore
38:    $\Delta \leftarrow (f(\text{sol}) \times 100) / f(\text{bestSol})$ 
39:   solScore  $\leftarrow$  MAXSCORE  $\times$   $\Delta$ 
40: end procedure

```

---

- Takes as input the accumulated move values (moveScoreList) and the current solution  $s_t$ .
- Filters moves based on their applicability to the current solution.
- Creates a probability distribution over moves based on their learned values.
- Returns a list of potential moves weighted by their historical effectiveness.

To guide the search process, the algorithm assigns a selection probability to each move that is directly proportional to its learned score. In other words, moves with higher scores have a higher chance of being selected. This is achieved by including multiple copies of each move in the potential move list, with the number of copies proportional to the move's score (line 24–26). This approach:

- Naturally balances exploration and exploitation – higher-valued moves are more likely to be selected but all moves maintain some probability of selection;

- Adapts to the learning process – as move scores evolve, selection probabilities automatically adjust;
- Maintains simplicity while providing effective value-based guidance.

**Fallback Strategy:** When the value-based agent does not have applicable learned moves (`potentialMoveList =  $\emptyset$` ), the algorithm falls back to the underlying local search heuristic through `SelectAcceptableMove`. This hybrid approach ensures:

- Continuous search progress even when learned knowledge is insufficient;
- Gradual transition from heuristic to learned behavior as experience accumulates;
- Robustness through combination of learning and traditional search strategies.

---

**Algorithm 3.** Value-guided local search template.

---

```

1: procedure LS_RL
2:   Input: Initial solution  $s_0$ 
3:   Output: Final solution  $s_t$ 
4:    $t \leftarrow 0$ 
5:   while stopping criterion not met do
6:     potentialMoveList  $\leftarrow$  SELECTPOSSIBLEMOVES(moveScoreList,  $s_t$ )
7:     if potentialMoveList  $\neq \emptyset$  then
8:       move( $t$ )  $\leftarrow$  RAND(potentialMoveList) ▷ Sample from value-based distribution
9:     else
10:      move( $t$ )  $\leftarrow$  SELECTACCEPTABLEMOVE( $s_t$ ) ▷ Fallback to heuristic
11:    end if
12:     $s_{t+1} \leftarrow$  MAKEMOVE( $s_t$ , move( $t$ ))
13:     $s_t \leftarrow s_{t+1}$ 
14:     $t \leftarrow t + 1$ 
15:  end while
16:  return  $s_t$ 
17: end procedure
18: procedure SELECTPOSSIBLEMOVES(moveScoreList,  $s_t$ )
19:   Input: moveScoreList, current solution  $s_t$ 
20:   Output: List of potential moves weighted by learned values
21:   potentialMoveList  $\leftarrow$  empty list
22:   for each (move, score) in moveScoreList do
23:     if CHECKAPPLICABILITY(move,  $s_t$ ) then
24:       for count  $\leftarrow$  1 to score $\times$ 2 do ▷ Create probability distribution
25:         potentialMoveList  $\leftarrow$  potentialMoveList  $\cup$  {move}
26:       end for
27:     end if
28:   end for
29:   return potentialMoveList
30: end procedure

```

---

The resulting algorithm creates an effective synergy between value-based learning and local search:

- The value-based agent progressively learns which moves are most effective;
- The local search provides a reliable foundation when learned knowledge is insufficient;
- The hybrid approach maintains the benefits of both systematic search and learning-based improvement.

This integration demonstrates how free value-based reinforcement learning can enhance traditional optimization methods while maintaining their robustness and reliability.

### 4. EXPERIMENTS AND DISCUSSION

To evaluate the optimization capability and scalability of our Reinforcement Learning (RL) hybrid methods, we benchmarked against two prevalent local search algorithms: Simulated Annealing (SA) and Variable Neighborhood Search (VNS). These algorithms were selected as baselines due to their well-established efficacy across diverse combinatorial optimization problems [9–12], rendering them standardized measures. This study focus on the NP-Hard Quadratic 3-dimensional Assignment Problem (Q3AP), serving as a test problem for assessing enhancement. In Section 4.3, we provide a comparative study between standalone SA and VNS *versus* SA+RL and VNS+RL respectively SA and VNS versions augmented with a RL agent.

#### 4.1. Running problem

The quadratic 3-dimensional assignment problem (Q3AP) is a extension of the quadratic assignment problem (QAP) and the axial 3-dimensional assignment problem (3AP). It was introduced by Pierskalla [31] and has been rediscovered as a mathematical model of some assignment problems such as the symbol-mapping problem described in [7]. The Q3AP can be formulated as follows:

$$\min \left\{ \sum_{i=1}^n \sum_{j=1}^n \sum_{p=1}^n \sum_{k=1}^n \sum_{n=1}^n \sum_{q=1}^n c_{ijpknq} x_{ijp} x_{knq} + \sum_{i=1}^n \sum_{j=1}^n \sum_{p=1}^n b_{ijp} x_{ijp} \right\}, \tag{1}$$

where:

$$X = (x_{ijp}) \in I \cap J \cap P, \tag{2}$$

$$x_{ijp} \in \{0, 1\}, \quad i, j, p = 1, 2, \dots, n. \tag{3}$$

$I, J$  and  $P$  sets are defined as follows:

$$I = \left\{ X = (x_{ijp}) : \sum_{j=1}^n \sum_{p=1}^n x_{ijp} = 1, \quad \text{for } i = 1, \dots, n \right\},$$

$$J = \left\{ X = (x_{ijp}) : \sum_{i=1}^n \sum_{p=1}^n x_{ijp} = 1, \quad \text{for } j = 1, \dots, n \right\},$$

$$P = \left\{ X = (x_{ijp}) : \sum_{i=1}^n \sum_{j=1}^n x_{ijp} = 1, \quad \text{for } p = 1, \dots, n \right\}.$$

While for the QAP the problem is to find a 2-dimensional permutation matrix that minimizes the objective function, the problem for Q3AP is to minimize a quadratic function over the 3-dimensional assignment polytope  $I \cap J \cap P$ . That is why this problem is referred to as quadratic 3-dimensional assignment problem.

An alternative formulation that we particularly used in our implementations is the permutation-based formulation. In permutation-based formulation, the Q3AP given by equations (1)–(3) can be expressed as [7]:

$$\min \left\{ f(p, q) = \sum_{i=1}^n \sum_{j=1}^n c_{ip(i)p(j)jq(i)q(j)} + \sum_{i=1}^n b_{ip(i)q(i)} \right\}, \tag{4}$$

where  $p$  and  $q$  are permutations over the set  $\{1, \dots, n\}$ . The first term in (4) can be interpreted as the cost of mapping simultaneously the entity  $i$  to the locations  $p(i)$  and  $q(i)$  and the entity  $j$  to the locations  $p(j)$  and  $q(j)$ . The second term can be interpreted as the cost of placing entity  $i$  simultaneously to locations  $p(i)$  and  $q(i)$ .

Regarding the temporal complexity of the Q3AP, and since Q3AP is an extension of the QAP and of the axial 3-dimensional assignment problem both of which are proved to be NP-hard then Q3AP is also NP-hard. While the number of feasible solutions of a QAP of size  $n$  is  $n!$ , the number of feasible solutions of a Q3AP of dimension  $n$  is  $n! \times n!$ .

The Q3AP instances used in our experiments have been derived from instances of the QAPLIB QAP reference library [32] where the six-dimensional cost matrix  $C_{ijpknq}$  of the Q3AP has been built using the relationship suggested by Hahn *et al.* [7]:

$$C_{ijpknq} = f_{ik} \cdot d_{jn} \cdot f_{ik} \cdot d_{pq}; \quad i, j, p, k, n, q = 1, \dots, n, \quad (5)$$

where  $n$  is the size of the instance,  $f_{ik}$  is an entry in the flow matrix  $F$  of the QAP problem, and  $d_{jn}$  and  $d_{pq}$  are entries in the distance matrix  $D$  of the QAP problem.

By generating the cost matrix  $C$  in this way, we generated Q3AP instances that retain some of the cost structure properties of standard QAP benchmark instances. This allows testing the performance of Q3AP algorithms on problems with somewhat similar characteristics to difficult QAP problems [7].

## 4.2. Local search methods

In this section, we will discuss two popular local search methods: Simulated Annealing (SA) and Variable Neighborhood Search (VNS).

Simulated Annealing (SA) is a metaheuristic rooted in statistical mechanics, adept at escaping local minima. Its defining feature is the acceptance of “uphill” moves, namely solutions that are inferior to the current one. This strategy is crucial for preventing entrapment in local optima. The probability of accepting such moves is controlled by a temperature parameter  $T$ , which decreases over time. This decrement facilitates the algorithm’s gradual convergence towards an improved solution [33].

The SA algorithm (as detailed in Algorithm 4) initiates with a randomly generated initial solution, ensuring a diverse starting point for the optimization process. Each iteration of the algorithm involves selecting a new solution from the current solution’s neighborhood. The decision to adopt this new solution is based on the Metropolis criterion: a superior solution is always accepted, while an inferior solution is accepted with a probability calculated *via* the Boltzmann distribution. This probability depends on both the temperature  $T$  and the difference in the objective function values between the current and the proposed solutions [2].

An integral aspect of SA is its geometric cooling schedule. In our implementation, the initial temperature  $T_0$  is set at 500, and it is reduced geometrically over time. The geometric cooling schedule involves multiplying the temperature by a constant factor  $a \in [0, 1]$  at each step, resulting in an exponential decrease in temperature. This methodical reduction ensures an effective balance between diversification, which entails exploring the search space, and intensification, which focuses on honing in on a local minimum. We set our constant factor to 0.95.

Variable Neighborhood Search (VNS), introduced by Mladenović *et al.* [34], dynamically alters neighborhood structures to effectively search for optimal solutions. The process begins with a randomly generated initial solution, ensuring a diverse starting point. VNS progresses through three main phases: shaking, local search, and move. In the shaking phase, a solution from the  $k_t$ th neighborhood of the current solution is randomly selected, serving as the starting point for local search. If a better solution is found, it replaces the original and the algorithm resets to the smallest neighborhood; otherwise, it moves to a larger neighborhood, allowing for broader search space exploration. In our VNS implementation, four specific neighborhood functions have been employed:

- **DoubleSwap:** Denoted as `DoubleSwap( $S, i, j$ )`, where  $S$  is the current solution and  $i, j$  are two distinct randomly chosen indices. This function swaps the elements at positions  $i$  and  $j$  in  $S$ .
- **TripleSwap:** Represented as `TripleSwap( $S, i, j, k$ )`, where  $i, j, k$  are three unique indices in  $S$ . This function permutes the elements at these indices.
- **Optheaderswap:** Notated as `Optheaderswap( $S, i, j$ )`, with  $i, j$  being two indices chosen excluding the first two positions. It swaps the elements at  $i$  and  $j$  in  $S$ .

**Algorithm 4.** Simulated Annealing (SA).

---

```

1: procedure SA
2:   Output: The best reached solution  $s$ 
3:    $s_0 = \text{GENERATEINITIALSOLUTION}();$ 
4:    $T = T_0;$ 
5:   while (stopping criterion not met) do
6:      $s' = \text{PICKATRANDOM}(N(s));$ 
7:     if ( $f(s') < f(s)$ ) then
8:        $s = s';$ 
9:     else
10:      ACCEPT  $s'$  AS NEW SOLUTION WITH PROBABILITY  $p(T, s', s);$ 
11:    end if
12:    UPDATE( $T$ );
13:  end while
14: end procedure

```

---

- **DichotomySwap:** Defined as  $\text{DichotomySwap}(S)$ . This function splits  $S$  into two halves and swaps these halves.

The strength of VNS lies in its dynamic neighborhood strategy, which allows for efficient exploration of large and complex search spaces. Its adaptability to different problems and compatibility with other metaheuristics makes it a robust choice for solving optimization problems. By incorporating these specific neighborhood functions, our VNS approach ensures a comprehensive and effective exploration of the solution space, increasing the probability of finding globally optimal solutions.

**Algorithm 5.** Variable Neighborhood Search (VNS).

---

```

1: procedure VNS
2:   Input: A set of neighborhood structures  $N_k, k = 1, \dots, k_{\max}$ 
3:   Output: The best reached solution  $s$ 
4:    $s_0 = \text{GENERATEINITIALSOLUTION}();$ 
5:   while (stopping criterion not met) do
6:      $k = 1;$ 
7:     while ( $k < k_{\max}$ ) do
8:        $s' = \text{PICKATRANDOM}(N_k(s));$ 
9:        $s'' = \text{LOCALSEARCH}(s');$ 
10:      if ( $f(s'') < f(s)$ ) then
11:         $s = s'';$ 
12:         $k = 1;$ 
13:      else
14:         $k = k + 1;$ 
15:      end if
16:    end while
17:  end while
18: end procedure

```

---

### 4.3. Experiments and discussion

To assess the efficiency and scalability of our proposal, we considered Q3AP benchmarks from the literature [7, 8]. These Q3AP benchmarks have been derived from QAP instances of the QAPLIB library [32]. We have selected two categories of QAP instances from which we derived Q3AP instances: the Nugent instances [35] and the Hadley instances [36]. The Nugent instances are QAP instances known for their varying levels of

TABLE 1. Experimental results for 100 iterations.

Metric	Algo	Nug13	Nug15	Nug18	Nug20	Nug22	Nug25	Nug27	Nug30	Had16	Had18
BKV		<b>1912</b>	<b>2230</b>	<b>17 836</b>	<b>25 590</b>	<b>42 467</b>	<b>37 716</b>	<b>13 266</b>	<b>69 704</b>	<b>52 980</b>	<b>84 932</b>
BFV	SA	<b>1912</b>	<b>2230</b>	18 562	<b>25 590</b>	43 108	<b>37 716</b>	14 813	72 006	<b>52 980</b>	<b>84 932</b>
	SA+RL	<b>1912</b>	<b>2230</b>	<b>17 836</b>	<b>25 590</b>	<b>42 467</b>	<b>37 716</b>	<b>13 266</b>	<b>69 704</b>	<b>52 980</b>	<b>84 932</b>
	VNS	<b>1912</b>	<b>2230</b>	<b>17 836</b>	<b>25 590</b>	43 016	<b>37 716</b>	14 516	72 180	<b>52 980</b>	<b>84 932</b>
	VNS+RL	<b>1912</b>	<b>2230</b>	<b>17 836</b>	<b>25 590</b>	<b>42 467</b>	<b>37 716</b>	<b>13 266</b>	<b>69 704</b>	<b>52 980</b>	<b>84 932</b>
Hits (%)	SA	31	4	0	1	0	2	0	0	7	8
	SA+RL	74	37	9	11	7	27	16	8	46	34
	VNS	28	7	2	2	0	1	0	0	7	9
	VNS+RL	79	52	23	29	11	24	19	11	49	41
MeanBC	SA	2473	2861	20 894	28 461	48 007	39 018	16 898	74 181	55 873	87 818
	SA+RL	2108	2421	18 952	26 433	44 641	38 461	14 108	71 962	53 746	85 843
	VNS	2538	2846	20 431	28 413	48 131	39 156	16 789	74 087	55 793	87 801
	VNS+RL	2046	2381	18 463	25 998	44 382	38 721	14 081	71 067	53 567	85 838
MeanTime (s)	SA	431	198	891	1448	2716	2466	3087	4889	401	585
	SA+RL	181	114	507	989	2101	1798	2188	3997	279	397
	VNS	467	196	867	1403	2756	2468	3085	4884	403	581
	VNS+RL	153	108	464	891	1989	1816	2154	3887	259	381

complexity and are commonly used to test the effectiveness of algorithms in solving QAPs. They are labeled in our experiments by the prefix “Nug” followed by the size of the instance. “Nug15”, for example, designates the Nugent instance of size 15. Similarly, the Hadley instances are denoted by the prefix “Had” and the size of the instance. They are particularly valuable for assessing algorithm performance in more complex scenarios.

The instance sizes range from 13 to 30 for Nugent, 16 and 18 for Hadley, providing a comprehensive evaluation across different scales and complexities. This range ensures that our assessment covers a wide spectrum of scenarios, from relatively simpler to more challenging ones, offering insights into the scalability and adaptability of the algorithms under study.

Table 1 reports some statistics on the results obtained from the various experiments carried out. These statistics are computed from a sample of 100 iterations. The meaning of the metrics is given below:

- **Best Known Value (BKV)**: The best performing solution known for each instance in literature and serves as our reference for performance. This value is presented in bold in the tables for clear identification as a benchmark.
- **Best Found Value (BFV)**: The best solution cost found by the algorithm on all iterations. The BFV that achieves the BKV for an instance is presented in bold to highlight best performance and facilitate direct comparison
- **Hits (%)**: The frequency with which the algorithm has successfully met the BKV.
- **Mean Best Cost (MeanBC)**: The average cost of the best solutions caught by the algorithm on throughout all iterations.
- **Mean Time (MeanTime (s))**: The average total execution time taken by the search process over all iterations. This includes the entire duration from the start of the algorithm to its termination.

As shown in Table 1, the reinforcement learning-augmented local search methods (SA+RL and VNS+RL) consistently outperformed their corresponding standalone versions (SA and VNS) across all Q3AP instances. This superiority is particularly evident in the success rate (Hits), where the augmented versions achieved the best known value (BKV) significantly more frequently over the 100 iterations. The integration of the RL component effectively guides the search towards promising regions, enabling more efficient convergence to optimal solutions. To quantify the improvements achieved by the augmented methods *versus* their standalone counterparts, we defined two metrics: the *cost improvement ratio (IRC)* and the *time improvement ratio (IRT)* using the following relationships:

$$IRC (\%) = \left( \frac{\text{MeanBC(LS)} - \text{MeanBC(LS+RL)}}{\text{MeanBC(LS)}} \right) \times 100 \tag{6}$$

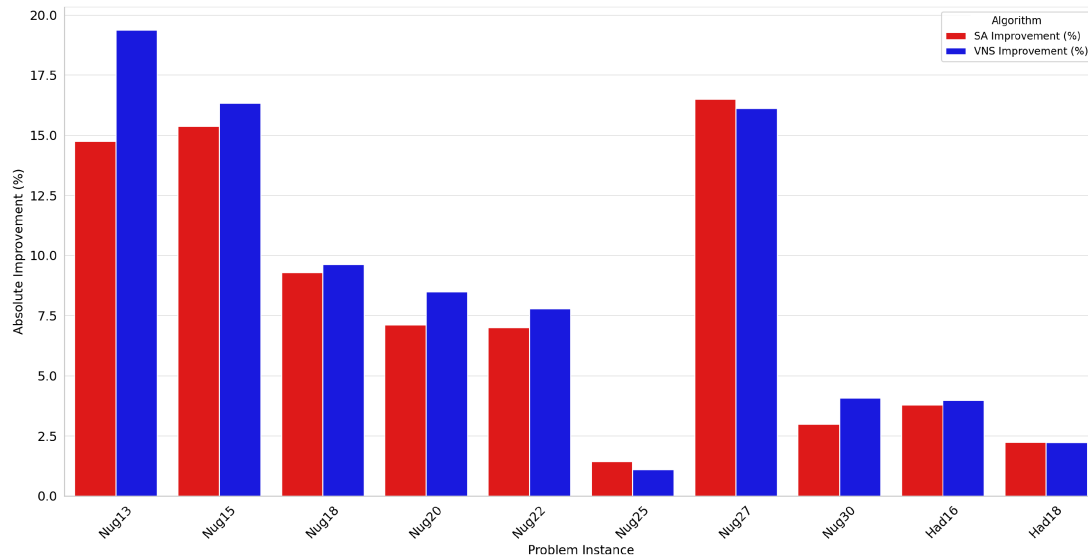


FIGURE 2. IRC (%) for 100 iterations.

and

$$\text{IRT} (\%) = \left( \frac{\text{MeanTime}(\text{LS}) - \text{MeanTime}(\text{LS}+\text{RL})}{\text{MeanTime}(\text{LS})} \right) \times 100 \quad (7)$$

where LS is the standalone local search, *i.e.* SA and VNS in our case, and LS+RL the augmented local search with reinforcement learning agent, *i.e.* SA+RL and VNS+RL.

Figures 2 and 3 illustrate respectively the improvement in average cost (IRC) and average time (IRT) for Q3AP instances over 100 iterations. Both SA+RL and VNS+RL showed consistent improvements over their standalone versions in all instances, with improvement ratios ranging from modest gains in some cases to substantial improvements in others. The time improvements were particularly notable, with significant speed-ups observed across all problem instances. These speedups, whilst counterintuitive given the additional computational overhead of the RL component, stem from the more efficient exploration of the solution space. RL guidance reduces unproductive moves and enhances the exploitation of promising regions, allowing algorithms to reach optimal solutions with fewer iterations. This efficiency gain more than compensates for the overhead introduced by the RL scoring system.

To assess the impact of the number of iterations on the quality of solutions and execution time, we carried out further experiments in which we extended the number of iterations to 300. The experimental results are reported in Table 2.

The results from extending the number of iterations to 300 (Figs. 4 and 5) demonstrate a significant enhancement in performance. The IRC ratios showed substantial improvements compared to the 100-iteration experiments, with larger instances benefiting particularly well from the extended runtime. This improvement results from the RL agent having more opportunities to learn and refine its solution strategy, leading to better exploration of the solution space and increased success in escaping local optima. Regarding execution time, the IRT ratios maintained their positive trend despite the increased number of iterations. Both SA+RL and VNS+RL continued to demonstrate significant time savings across all instance sizes, with particularly notable improvements for larger problem instances. This sustained performance advantage indicates that the RL-enhanced algorithms become increasingly efficient at guiding the search process as they accumulate more experience through additional iterations.

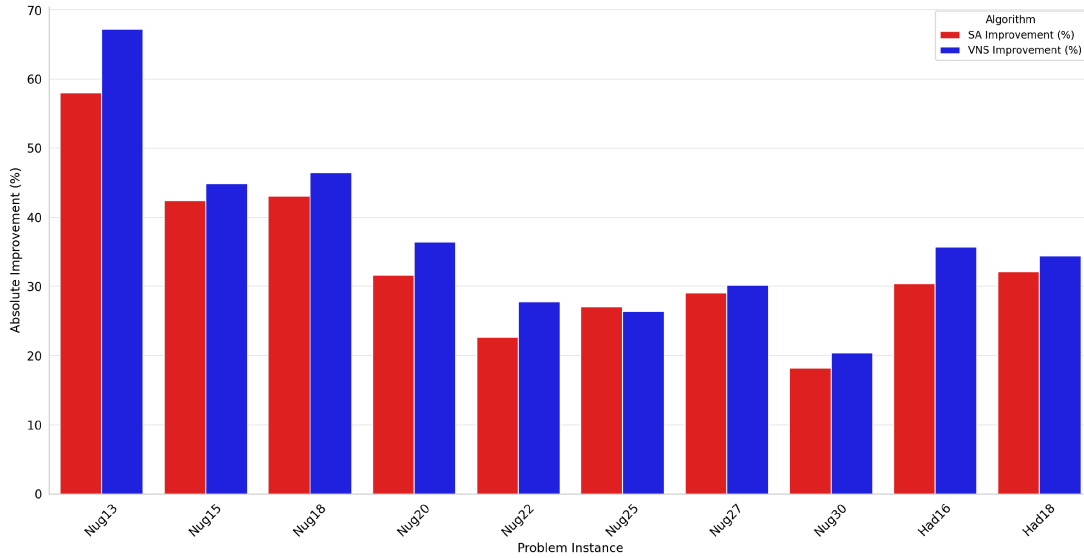


FIGURE 3. IRT (%) for 100 iterations.

TABLE 2. Experimental results for 300 iterations.

Metric	Algo	Nug13	Nug15	Nug18	Nug20	Nug22	Nug25	Nug27	Nug30	Had16	Had18
BKV		<b>1912</b>	<b>2230</b>	<b>17 836</b>	<b>25 590</b>	<b>42 467</b>	<b>37 716</b>	<b>13 266</b>	<b>69 704</b>	<b>52 980</b>	<b>84 932</b>
BFV	SA	<b>1912</b>	<b>2230</b>	18 562	<b>25 590</b>	43 108	<b>37 716</b>	14 813	72 006	<b>52 980</b>	<b>84 932</b>
	SA+RL	<b>1912</b>	<b>2230</b>	<b>17 836</b>	<b>25 590</b>	<b>42 467</b>	<b>37 716</b>	<b>13 266</b>	<b>69 704</b>	<b>52 980</b>	<b>84 932</b>
	VNS	<b>1912</b>	<b>2230</b>	<b>17 836</b>	<b>25 590</b>	43 016	<b>37 716</b>	14 516	72 180	<b>52 980</b>	<b>84 932</b>
	VNS+RL	<b>1912</b>	<b>2230</b>	<b>17 836</b>	<b>25 590</b>	<b>42 467</b>	<b>37 716</b>	<b>13 266</b>	<b>69 704</b>	<b>52 980</b>	<b>84 932</b>
Hits (%)	SA	31.0	4.6	0.0	2.0	0.0	3.0	0.0	0.0	6.2	7.0
	SA+RL	84.6	61.0	44.0	38.0	25.3	48.7	43.0	30.3	70.3	65.3
	VNS	29.3	7.7	2.0	2.6	0.0	1.3	0.0	0.0	5.3	6.3
	VNS+RL	87.0	65.6	33.3	46.6	24.3	33.4	29.8	19.6	63.3	54.3
MeanBC	SA	2443	2841	20 731	29 003	48 113	39 512	17 832	75 322	56 012	88 091
	SA+RL	2003	2361	18 014	26 402	42 971	37 912	13 971	70 132	53 312	85 034
	VNS	2418	2705	20 645	28 113	48 009	39 348	16 912	74 731	56 012	88 034
	VNS+RL	1968	2308	17 994	25 842	42 987	37 891	13 904	70 381	53 162	85 113
MeanTime (s)	SA	433	201	864	1431	2736	2481	3051	4806	398	598
	SA+RL	142	91	413	783	1832	1361	1464	2611	197	287
	VNS	463	194	841	1374	2771	2501	3076	4897	411	592
	VNS+RL	129	97	408	704	1337	1381	1609	2816	214	297

While both SA+RL and VNS+RL outperform their standalone counterparts for all instance sizes, we observe a clear trend indicating that the relative improvement achieved by the RL-augmented methods increases with problem complexity. To better illustrate this, we introduce the Hit Improvement Ratio (HIR), which quantifies the percentage increase in the hit rate of augmented method compared to the standard methods. The HIR is formally defined as:

$$\text{HIR} (\%) = \text{Hits}(\text{LS}+\text{RL}) - \text{Hits}(\text{LS}) \tag{8}$$

where Hits(LS+RL) represents the hit rate of the RL-augmented local search method, and Hits(LS) represents the hit rate of the corresponding baseline local search method. Figure 6 shows the evolution of the HIR on the various Q3AP instances for 300 iterations. The HIR generally increases with problem size, indicating a growing relative advantage of our RL-augmented methods. However, we observe that the HIR plateaus or even decreases

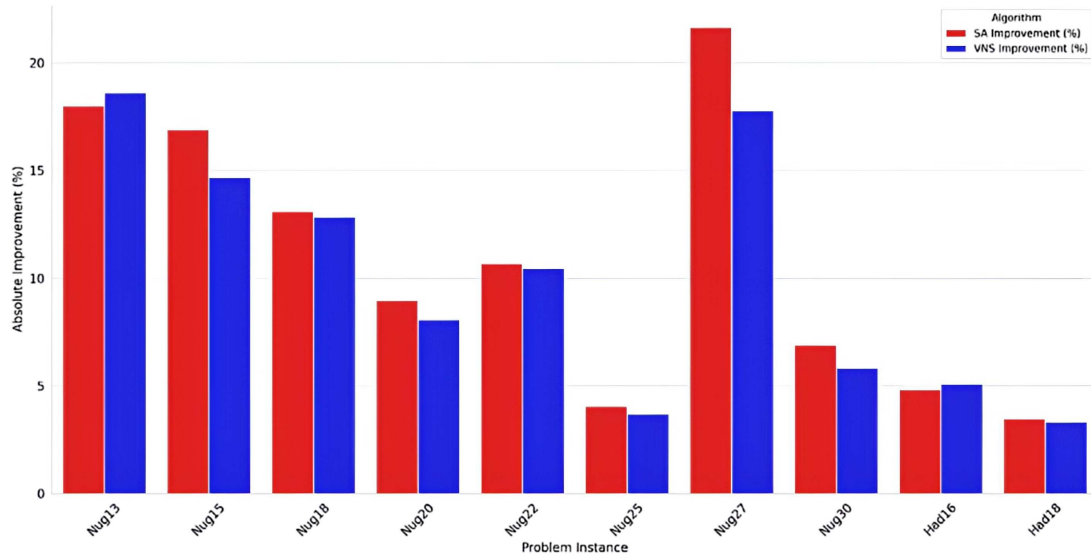


FIGURE 4. IRC (%) for 300 iterations.

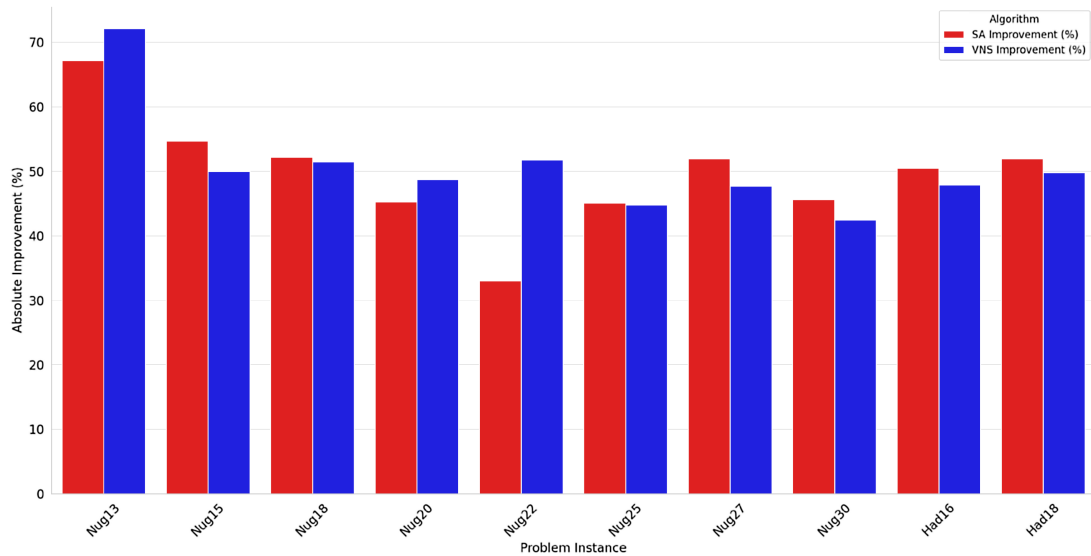


FIGURE 5. IRT (%) for 300 iterations.

slightly at the largest Nugent instances (Nug22–Nug30), suggesting that the rate of relative improvement slows down as the problems get very complex. This plateauing effect in larger instances can be attributed to several factors. As problem size increases, the search space grows exponentially – for example, Nug30 encompasses approximately  $(30!)^2$  possible solutions. This rapid expansion in complexity challenges the RL agent’s capacity to learn and adapt at the same efficiency as observed in smaller instances.

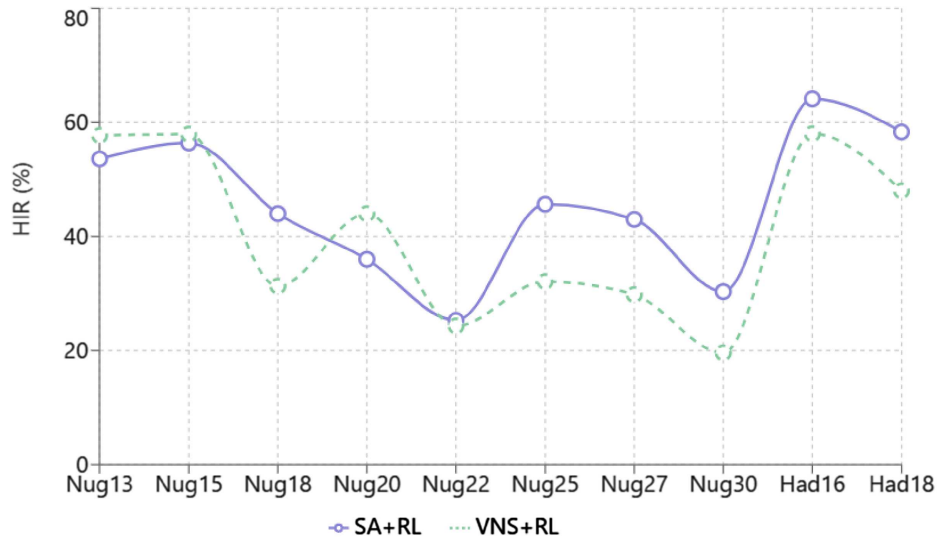


FIGURE 6. HIR (%) for 300 iterations.

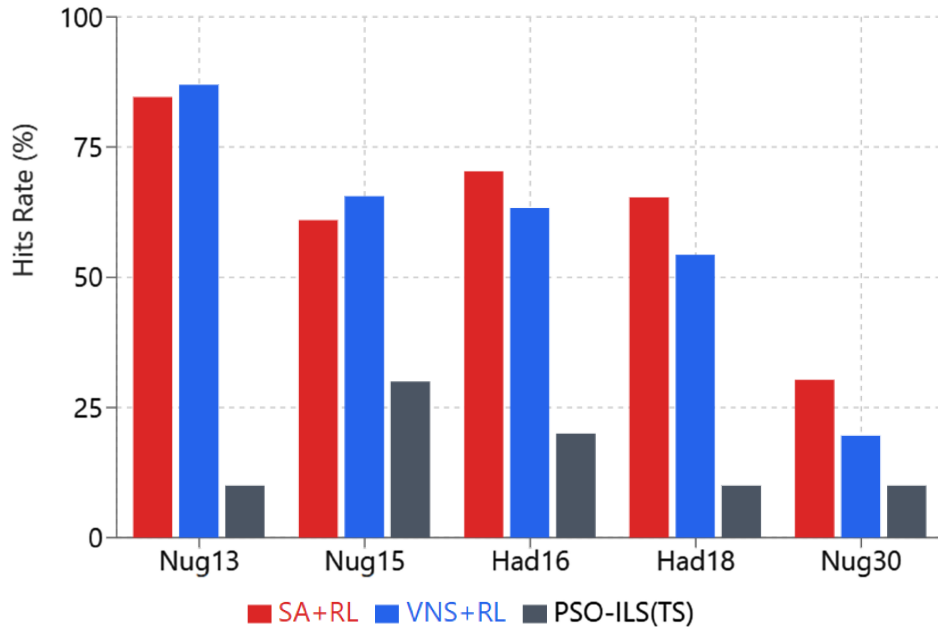


FIGURE 7. Comparison of Hits Rates between SA+RL, VNS+RL, and PSO-ILS(TS) across different instances.

Despite this plateauing, RL-augmented methods continue to deliver substantial advantages over standard approaches, even at the largest instance sizes, underscoring their robust practical value across a wide problem spectrum.

To contextualize our findings, we compared our methods against the PSO-ILS(TS) hybrid approach by Ait Abderrahim *et al.* [28]. While direct comparisons are constrained by differences in experimental setups, including

the number of runs and computational resources, the success rate metric provides a robust basis for comparison as it normalizes performance across different implementations. Figure 7 shows that both SA+RL and VNS+RL consistently achieve 2–3 times higher success rates than PSO-ILS(TS). This performance gap is most pronounced in Had16, where our methods maintain success rates above 60% while PSO-ILS(TS) achieves only 20%. Even in the most challenging instance (Nug30), where we observed the previously discussed plateauing effect, our methods still demonstrate substantially higher hit rates than the PSO-ILS(TS) approach. These comparative results further strengthen our findings about the effectiveness of integrating reinforcement learning with local search methods for solving Q3AP instances.

## 5. CONCLUSION

In this work, we proposed a hybrid method by integrating Reinforcement Learning into heuristic algorithms to enhance their exploration abilities. We tested this on the Quadratic 3-dimensional Assignment Problem using Simulated Annealing and Variable Neighborhood Search as example heuristics. The goal was to address the limitation of traditional heuristics in not exploiting information gathered from prior iterations thus wasting potentially useful data. Our findings confirm this goal, demonstrating that our RL-enhanced algorithms perform better than traditional SA and VNS in solution quality and execution time. This is especially seen in our extended iteration experiments, where RL's contribution to more strategic exploration of the search space led to notable gains. However, our approach has limitations. The reliance on iterative learning and the need for many iterations to achieve significant improvements in larger problems could restrict feasibility for time-critical situations. The general applicability of our RL integration framework across different local search methods still needs further exploration. Future work should focus on validating the integration of RL with various metaheuristics like Tabu Search and Genetic Algorithms and expanding its application to a wider range of combinatorial optimization problems. This will provide insights into its scalability and efficacy but also explore the potential of RL for enhancing decision-making in complex optimization problems.

### CONFLICTS OF INTEREST

The authors declare that there is no conflict of interests regarding the publication of this paper.

### DATA AVAILABILITY STATEMENT

The authors confirm that all data generated or analysed during this study are included in this article.

### REFERENCES

- [1] H.R. Lourenço, O.C. Martin, T. Stützle, F. Glover and G.A. Kochenberger, Iterated Local Search. Springer US, Boston, MA (2003) 320–353.
- [2] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, Optimization by simulated annealing. *Science* **220** (1983) 671–680.
- [3] F. Glover and M. Laguna, Tabu Search. Kluwer Academic Publishers, Norwell, MA, USA (1997).
- [4] P. Hansen, N. Mladenović, E.K. Burke and G. Kendall, Variable Neighborhood Search. Springer US, Boston, MA (2005) 211–238.
- [5] K.P. Bennett and E. Parrado-Hernández, The interplay of optimization and machine learning research. *J. Mach. Learn. Res.* **7** (2006) 1265–1281.
- [6] H. Mittelman and D. Salvagnin, On solving a hard quadratic 3-dimensional assignment problem. *Math. Program. Comput.* **7** (2015) 219–234.
- [7] P.M. Hahn, B.-J. Kim, T. Stuetzle, S. Kanthak, W.L. Hightower, H. Samra, Z. Ding and M. Guignard, The quadratic three-dimensional assignment problem: exact and approximate solution methods. *Eur. J. Oper. Res.* **184** (2008) 416–428.
- [8] L. Loukil, M. Mehdi, N. Melab, E.-G. Talbi and P. Bouvry, Parallel hybrid genetic algorithms for solving Q3AP on computational grid. *Int. J. Found. Comput. Sci.* **23** (2012) 483–500.
- [9] D. Delahaye, S. Chaimatanan, M. Mongeau, M. Gendreau and J.-Y. Potvin, Simulated Annealing: From Basics to Applications. Springer International Publishing, Cham (2019) 1–35.

- [10] C. Koulamas, S.R. Antony and R. Jaen, A survey of simulated annealing applications to operations research problems. *Omega* **22** (1994) 41–56.
- [11] S. Lan, W. Fan, S. Yang, P.M. Pardalos and N. Mladenovic, A survey on the applications of variable neighborhood search algorithm in healthcare management. *Ann. Math. Artif. Intell.* **89** (2021) 741–775.
- [12] P. Hansen and N. Mladenović, Variable neighborhood search: principles and applications. *Eur. J. Oper. Res.* **130** (2001) 449–467.
- [13] M. Karimi-Mamaghan, M. Mohammadi, P. Meyer, A.M. Karimi-Mamaghan and E.-G. Talbi, Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: a state-of-the-art. *Eur. J. Oper. Res.* **296** (2022) 393–422.
- [14] S. Szénási and G. Légrádi, Machine learning aided metaheuristics: a comprehensive review of hybrid local search methods. *Expert Syst. App.* **258** (2024) 125192.
- [15] J. Barrera-García, F. Cisternas-Caneo, B. Crawford, M. Gómez Sánchez and R. Soto, Feature selection problem and metaheuristics: a systematic literature review about its formulation, evaluation and applications. *Biomimetics* **9** (2024) 9.
- [16] B. Bischl, P. Kerschke, L. Kotthoff, M. Lindauer, Y. Malitsky, A. Fréchet, H. Hoos, F. Hutter, K. Leyton-Brown, K. Tierney and J. Vanschoren, Aslib: a benchmark library for algorithm selection. *Artif. Intell.* **237** (2016) 41–58.
- [17] C.P. Gomes and B. Selman, Algorithm portfolio design: theory vs. practice. Preprint [arXiv:1302.1541](https://arxiv.org/abs/1302.1541) (2013).
- [18] V.-A. Darvari, S. Hailes and M. Musolesi, Graph reinforcement learning for combinatorial optimization: a survey and unifying perspective. Preprint [arXiv:2404.06492](https://arxiv.org/abs/2404.06492) (2024).
- [19] Q. Lan, A.R. Mahmood, S. Yan and Z. Xu, Learning to optimize for reinforcement learning. Preprint [arXiv:2302.01470](https://arxiv.org/abs/2302.01470) (2024).
- [20] N. Mazyavkina, S. Sviridov, S. Ivanov and E. Burnaev, Reinforcement learning for combinatorial optimization: a survey. *Comput. Oper. Res.* **134** (2021) 105400.
- [21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, Proximal policy optimization algorithms. Preprint [arXiv:1707.06347](https://arxiv.org/abs/1707.06347) (2017).
- [22] I. Bello, H. Pham, Q.V. Le, M. Norouzi and S. Bengio, Neural combinatorial optimization with reinforcement learning. Preprint [arXiv:1611.09940](https://arxiv.org/abs/1611.09940) (2016).
- [23] A. Garmendia, Q. Cappart, J. Ceberio and A. Mendiburu, Marco: a memory-augmented reinforcement framework for combinatorial optimization. Preprint [arXiv:2408.02207](https://arxiv.org/abs/2408.02207) (2024).
- [24] H. Yang and M. Gu, A new baseline of policy gradient for traveling salesman problem, in 2022 IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA) (2022) 1–7.
- [25] T. Mustakhov, Y. Akhmetbek and A. Bogrybayeva, Deep reinforcement learning for stochastic dynamic vehicle routing problem, in 2023 17th International Conference on Electronics Computer and Computation (ICECCO) (2023) 1–5.
- [26] J. Holder, N. Jaques and M. Mesbahi, Multi agent reinforcement learning for sequential satellite assignment problems, in Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 39. (2025) 26516–26524.
- [27] X. Kong, Y. Zhou, Z. Li and S. Wang, Multi-UAV simultaneous target assignment and path planning based on deep reinforcement learning in dynamic multiple obstacles environments. *Front. Neurobot.* **17** (2024) 1302898.
- [28] I. Ait Abderrahim and L. Loukil, Hybrid approach for solving the Q3AP. *Int. J. Swarm Intell. Res. (IJSIR)* **12** (2021) 98–114.
- [29] P.S. Bagga and A. Delarue, Solving the quadratic assignment problem using deep reinforcement learning. Preprint [arXiv:2310.01604](https://arxiv.org/abs/2310.01604) (2023).
- [30] J.E.R. Staddon, The dynamics of behavior: review of Sutton and Barto: reinforcement learning: an introduction. *J. Exper. Anal. Behav.* **113** (2020) 485–491.
- [31] W.P. Pierskalla, The multi-dimensional assignment problem. Technical Memorandum No. 93, Operations Research Department, CASE Institute of Technology (1967).
- [32] R.E. Burkard, E. Çela, S.E. Karisch and F. Rendl, Quadratic assignment problem library. <https://coral.ise.lehigh.edu/data-sets/qaplib/>. Accessed: 01/01/2024.
- [33] V. Černý, Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. *J. Optim. Theory App.* **45** (1985) 41–51.
- [34] N. Mladenović and P. Hansen, Variable neighborhood search. *Comput. Oper. Res.* **24** (1997) 1097–1100.
- [35] C.E. Nugent, T.E. Vollmann and J. Ruml, An experimental comparison of techniques for the assignment of facilities to locations. *Oper. Res.* **16** (1968) 150–173.

- [36] S.W. Hadley, F. Rendl and H. Wolkowicz, A new lower bound via projection for the quadratic assignment problem. *Math. Oper. Res.* **17** (1992) 727–739.



**Please help to maintain this journal in open access!**

This journal is currently published in open access under the Subscribe to Open model (S2O). We are thankful to our subscribers and supporters for making it possible to publish this journal in open access in the current year, free of charge for authors and readers.

Check with your library that it subscribes to the journal, or consider making a personal donation to the S2O programme by contacting [subscribers@edpsciences.org](mailto:subscribers@edpsciences.org).

More information, including a list of supporters and financial transparency reports, is available at <https://edpsciences.org/en/subscribe-to-open-s2o>.