RAIRO-Oper. Res. 60 (2026) 1103–1126
https://doi.org/10.1051/ro/2026026

RAIRO Operations Research
www.rairo-ro.org

LEVERAGING REINFORCEMENT LEARNING AND EVOLUTIONARY ALGORITHM TO SOLVE BI-LEVEL COMBINATORIAL OPTIMIZATION PROBLEM

Abir Chaabani*

Abstract. Bi-level optimization research area has become increasingly popular, largely due to its effectiveness in modeling and solving real-world problems. This framework provides a hierarchical structure involving two decision-makers (*i.e.*, upper and lower levels) that govern together to find an optimal solution to complex optimization situations. Most resolution methods proposed in the literature adhere to this hierarchical structure, which limit their applicability only to small-scale instances of the problem. Among these resolution strategies, we highlight an interesting evolutionary algorithm known as CODBA, which focuses on decomposing the lower-level search space into several parts that evolve in parallel to address the high complexity of the nested structure. In this paper, we enhance the searching capabilities of CODBA by proposing a novel evolutionary reinforcement learning approach that integrates the core CODBA scheme with a Q-learning strategy, presenting a promising method for training intelligent search algorithms for bi-level optimization problems. The computational statistical experiments are performed on bi-level multi-depot vehicle routing problem, demonstrated the effectiveness of our solution approach in terms of computation time and solution quality compared to existing algorithms.

Mathematics Subject Classification. 90C29, 90C59, 68T05, 90B06.

Received December 13, 2024. Accepted March 2, 2026.

1. Introduction

Bi-level Optimization (BO), a specific case of multi-level optimization considering two nested problems, is one of the most extensively studied branch within this research area [37,46]. This type of problem involves two decision levels: an Upper-Level (UL) problem constrained by the solution of a Lower-Level (LL) task. Each decision level controls its own objectives, decision variables, and constraints, but the decisions made at the upper level explicitly depend on the lower level one, influencing the outcome of the upper problem. Respecting this definition, the lower-level problem optimizes its objectives by treating the upper-level decision variables as fixed parameters.

Many optimization problems and decision-making processes in academic contexts and real-world applications exhibit a hierarchical structure. In this direction, we can cite: supply chain applications [13], electricity

Keywords. Bi-level decision-making, evolutionary algorithms, Q-learning strategy, VNS algorithm.

École Nationale d'Ingénieurs de Carthage (ENICarthage), Université de Carthage, SMART Lab, ISGT, Université de Tunis, Tunis, Tunisie.
*Corresponding author: abir.chaabani@gmail.com

transmission applications, telecommunication applications [2], facility location [23], security applications [51], machine learning [34], scheduling problems [4], just to cite a few. These research studies illustrate the diverse applications of bi-level optimization across different domains, demonstrating then its significance in addressing complex decision-making scenarios. The implications of bi-level optimization model, the relationship between the two decision levels, and their practical applications are currently the key interests for the BO community. While these models enable practical applications across various fields, they also present challenges regarding resolution methods.

Existing bi-level resolution methods can be mainly categorized into two classes: (1) classical methods and (2) evolutionary ones. The classical methods primarily consist of techniques such as the Karush–Kuhn–Tucker approach [27], the complementary pivoting method, the descent method, and the trust region method [43]. The main shortcoming of these approaches is their reliance on the mathematical characteristics of bi-level optimization problems, rendering them less effective for addressing real-world challenges like high dimensionality and non-linearity.

The second family refers to Evolutionary Bi-level Algorithms (EBAs), that have been successfully applied to handle applications that do not adhere to regularities like continuity or convexity. Thanks to these properties of EBAs, attempts have been made to solve bi-level optimization problems [1, 9, 45]. However, the advantages come with a trade-off that the majority of the proposed EBAs represent either a high computational cost or applicable to only small classes of the problem. This is due to the necessity of executing for every upper solution, another Evolutionary Algorithm (EA) at the lower level to find the lower (near) optimal solution.

For these reasons, a number of attempts have been proposed in literature towards finding efficient EBA with less computational cost. In this direction, we can cite the work of Sinha *et al.* [45] who proposed a nested approach based on Quadratic Approximations (BLEAQ) of optimal lower level variables with respect to the upper level variables. BLEAQ performed well on continuous benchmark problems, however, the quadratic approximation idea was not suitable for the combinatorial context.

Recently, Chaabani *et al.* in [9] proposed a new evolutionary algorithm called CODBA that handles three main mechanisms with EBA which are: (1) decomposition, (2) multi-threading and (3) co-evolution to reduce the complexity induced by the lower level. Thanks to the interesting features of CODBA approach, several research studies adopt this method to solve efficiently a wide range of real-world optimization problems. Examples include applications in security [24], feature selection [20], and feature construction [19], etc. However, it remains a nested evolutionary strategy, which entails significant computational costs when dealing with large-scale bi-level optimization problems.

In recent years, there has been a growing research interest in integrating Machine Learning (ML) techniques, particularly reinforcement learning technique, into the evolutionary algorithms for solving optimization problems.

The main motivation behind this integration is to enhance the efficiency and effectiveness of the search process. In this research area, there are two sub-fields that branch out from the integration of RL with EA:

(1) Evolutionary Computation for Reinforcement Learning techniques. Readers interested in this research subfield can refer to the recent surveys by Lin *et al.* [32] and Bai *et al.* [5].
(2) The second subfield which is directly related to this research paper is (2) Reinforcement Learning techniques for Evolutionary Computation. Thus, by leveraging machine learning, evolutionary algorithms can adaptively learn from past iterations, improving solution quality and convergence speed.

To this end, the idea of this paper is to propose an evolutionary reinforcement learning algorithm to solve combinatorial bi-level optimization problem which integrates CODBA scheme with reinforcement learning, to exploit the strengths of both strategies while compensating their limits in such NP-hard optimization context. The main concern is to make the algorithm able to learn how to choose the appropriate search strategy to enhance its performance. Consequently, the main contributions of this paper are summarized as follows:

(1) Design a new lower level Q-learning assisted VNS,

(2) Propose a new efficient evolutionary reinforcement learning algorithm, exploiting CODBA scheme with the lower level Q-learning assisted VNS, called EVORL-CODBA for solving combinatorial bi-level optimization problems;
(3) Investigating and demonstrating the out-performance of EVORL-CODBA over threes recently proposed bi-level algorithms that are CODBA-VNS [11], CODBA [9], and NBEA [35];
(4) Reporting comparative results on the use of the four bi-level algorithms for solving the Bi-level Multi-Depot Routing Problem, which is a well-studied problem in operations research.

2. Overview of machine learning integration with evolutionary algorithms

In this section, we describe the research field related to the integration of machine learning techniques with evolutionary algorithms. We begin by introducing the machine learning techniques discussed in the literature, especially reinforcement learning techniques, followed by a focused overview of the different integration strategies of RL into EA.

2.1. Brief overview of machine learning techniques

Machine Learning (ML) is a branch of artificial intelligence that focuses on developing algorithms and statistical approaches that can learn from data to perform new tasks based on inferred information [50]. These ML algorithms are classified in the literature into three categories:

– **Supervised Learning Algorithms:** In this category, the algorithm assumes that the values of input variables and the corresponding output variable values are known at prior. The main concern of these strategies is to determine the relationship between input variables and output labels, then use them to prediction. We cite: Based on the learning objective, supervised learning algorithms are typically categorized into classification and regression techniques. A wide range of models have been proposed in the literature, ranging from simple linear models to complex deep neural networks.
  • Linear Regression and Logistic Regression [21]: Used for predicting continuous and categorical outcomes, respectively.
  • Support Vector Machines (SVM) [15]: Effective in high–dimensional spaces, particularly when using kernel methods.
  • Naive Bayes [40]: A probabilistic model based on the assumption of feature independence. It utilizes Bayes' theorem for classification tasks. The "naive" aspect refers to the simplifying assumption that all features are independent of each other given the class label.
  • Decision Trees [6]: Provide interpretable models and can handle both categorical and numerical data by splitting data based on features.
  • k-Nearest Neighbors (k-NN) [16]: A non-parametric method that relies on distance metrics for prediction.
  Recent surveys [42] discussed the growing evolution of this research area, highlighting the integration of uncertainty estimation in supervised learning through methods such as Bayesian deep learning and model ensembles and its critical importance in high-task domains. Furthermore, ensemble learning techniques, such as Random Forests and Gradient Boosted Trees, have gained widespread popularity due to their robustness and high predictive accuracy. However, a major limitation of supervised learning is its reliance on large volumes of labeled data, which can be both expensive and time-consuming to obtain. This challenge has stimulated the development of semi-supervised and self-supervised learning approaches, which aim to leverage unlabeled data to improve learning performance [18].
– **Unsupervised Learning Algorithms:** In this category, the algorithm assumes that the values of input variables are available, while the output variables are unlabeled. In this way, the learning algorithms try to identify patterns presented in the input variables. In this class most known approaches are:
  • k-Means Clustering [25]: A partitioning method used to partition a dataset into K distinct, non-overlapping clusters. The algorithm aims to group data points based on similarity, where similarity is typically measured by the distance between data points and cluster centroids,

- Hierarchical Clustering and DBSCAN [38]: Hierarchical clustering builds a hierarchy of clusters, while DBSCAN groups data based on density, identifying clusters of varying shapes and handling noise,
- Multiple Correspondence Analysis (MCA) [3]: It is s a statistical method used to analyze relationships between multiple categorical variables,
- Gaussian Mixture Models (GMM) [26]: Data is generated from a mixture of Gaussian distributions, allowing for soft cluster assignments.

Unsupervised learning offers several advantages, notably the ability to work without labeled data, which is particularly useful when such data is scarce or expensive to obtain. Moreover, these algorithms are highly flexible and can be adapted to various types of data and problem settings. However, unsupervised learning has some drawbacks among which: it is difficult to choose an appropriate algorithm for a specific task and it is difficult to evaluate the performance of the output without a set of clear and objective criteria [44].

– **Reinforcement Learning Algorithms:** One of the most prominent machine learning methods, involves an agent that learns and interacts with environment to make decisions that would maximize a reward or minimize a risk. RL algorithms are classified into model-based and model-free algorithms. The first class uses a specific model for the environment to predict states and rewards (*e.g.* Monte Carlo Tree Search). Model-free algorithms learn directly from interactions with the environment without the need for a model. This category of Model-free algorithms is divided into: (1) on-policy methods, such as SARSA, which learns the value of the current policy by evaluating actions taken. This means that the agent's policy is continuously updated as it explores and learns. (2) Value-based methods such as Q-Learning, which learns the value of the optimal policy using data from various policies. This allows non-policy methods to learn from past experiences and to potentially speed up the learning process through exploratory actions.

2.2. Reinforcement learning with evolutionary optimization

In this section, we focus on reviewing existing works in the literature related to the integration of reinforcement learning techniques to assist evolutionary algorithms, as a promising approach in the field of optimization. The main motivation behind this integration is to overcome the limitations of EAs, which are mainly represented by slow convergence speed. In this context, RL represents an injected component to the EA scheme that exploits data generated through algorithm iterations to obtain useful knowledge to enhance the algorithm's performance [36]. Several studies have explored the use of reinforcement learning with evolutionary algorithms (EAs). According to Song *et al.* in a recent survey in 2024 [48], three main ways can be identified to discuss the integration of RL into EA which are: (1) The integration methods for RL and EA, (2) The RL-assisted strategy and (3) The attribute settings for RL.

(1) **Integration methods for RL and EA:** The integration of RL to assist EA involves mainly two manners: direct integration and indirect integration. Regarding the first case, the action performed by the RL technique is directly applied to the search of the EA, while the agent will use the feedback from the search performance information to update the model. In this context, we can cite the work of Li *et al.* [31] who proposed an integrated RL with Artificial Bee Colony algorithm to solve Job-Shop Scheduling problem with lot-streaming. Moreover, Hu *et al.* [22] include direct solution generation and automated tuning of control parameters. Regarding the indirect integration approach, the RL technique operates and makes decisions on the ensemble strategies, which are utilized then by the EA search. Then, RL updates theses strategies based on their performance. This enhances the algorithm's robustness by simultaneously utilizing diverse strategies within a single framework. We can cite here the work of Zhang *et al.* who proposed a genetic algorithm based on Q-learning for multi-objective sequence optimization [53]. As well, Li *et al.* who designed a deep reinforcement learning for optimization problem [29].

(2) **RL-assisted strategy:** These assisted strategies attempt to enhance the search performance of EA by utilizing:
   – **Generating solution:** The objective of this technique is to generate a complete or partial solution that will be utilized by the EA, similar to how solutions are generated using domain knowledge heuristics [10].

There exist two manners for generating solutions: (1) directly generating solutions for the target problem, and (2) injecting solutions that serve as sub-problems of the target problem. In this direction, we cite the work of Zhang *et al.* [52] who proposed a Q-learning method for controlling solution generation in the particle swarm algorithm.

- **Learnable objective function:** The basic concept of a learnable objective function is to integrate the information gathered during exploration and the characteristics of the problem into the objective function, guiding the algorithm to search more effectively. In this direction, we cite the Inverse Reinforcement Learning (IRL), a specialized type of reinforcement learning, which is frequently utilized to obtain the reward function. Moreover, Liu *et al.* [33] proposed an IRL method based on particle swarm optimization to model driver's steering behavior.
- **Algorithm selection:** The use of RL for algorithm selection has become increasingly popular and is employed on the hyper-heuristic research domain, which focuses on selecting the appropriate Low-Level Heuristic (LLH) for a given problem. In this context, we highlight the Q-learning RL technique, which has been widely used for selecting low-level heuristics (LLHs) [54].
- **Operator selection:** The main concern of this strategy is to employ RL to adaptively select the appropriate operators for EA search process. Specifically, RL primarily performs operator selection among three types: (1) Adaptive Crossover [47], (2) Adaptive mutation [30], and (3) Adaptive local search [55].
- **Parameter adaptation:** In this type of RL assisted EA, RL technique has the ability to dynamically adapt strategies based on the specific problem and the performance of the algorithmic search by an adaptive control parameter adjustment. Several research studies have explored the use of RL for adaptive control adjustment. We cite here only a few representative studies, such as those by Peng *et al.* [39] and Song *et al.* [47].

(3) **Attribute settings for RL:** In this category, attribute settings for RL refer to the parameters and factors (state and reward) that guide how RL interacts with the EA process:

- **Setting of state:** In order to better assist EA with RL, it is important that the state information reflects the latest progress of the algorithm' search. We cite here the work of Cheng *et al.* [12] who proposed a discretized approach for representing continuous states by dividing them into sub-intervals.
- **Setting of reward function:** The reward function may include two forms: direct calculation based on fitness value changes and acquisition through specific mapping relationships. In the first case, Sun *et al.* [49] used the difference in best fitness values between two consecutive generations of populations as the reward. While, Xia *et al.* [50] propose a reward adjustment operator based on the frequency of occurrence in the subspace, which was able to balance exploration and exploitation.

3. Evolutionary reinforcement learning with CODBA algorithm

3.1. Motivation

As we mentioned previously, in this paper we are motivated by the recent success of the application of learning techniques and particularly the RL methods with EA to solve bi-level combinatorial optimization problem. In other words, we are focused on exploiting RL capabilities in discovering optimal decision-making strategies, and improving convergence of bi-level algorithm through iterative feedback. To this end, our proposal is based on the following three main points:

- **Idea 1:** Inherit the core principles of CODBA, which are: (1) Decomposition of the lower search space through the Discrete Space Decomposition Method (DSDM) [9], and (2) multi-threading in order to cope with the complexity induced by the lower level.
- **Idea 2:** Leverage the strengths of Q-learning to search for the optimal policy that enhances the bi-level algorithm's convergence.

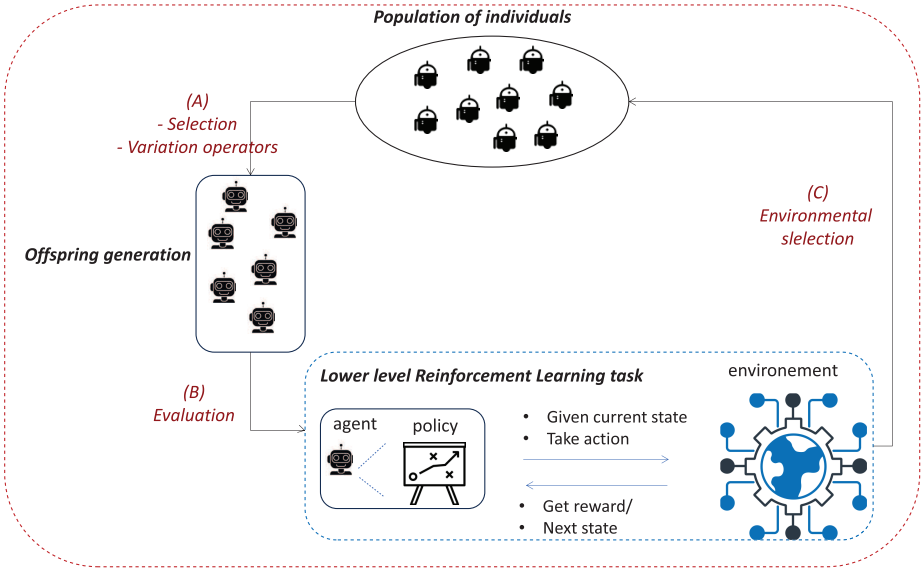

FIGURE 1. General scheme of our EVOLRL_CODBA.

– **Idea 3:** Utilize the knowledge stored in the Q-table as a shared memory to convey insights between the different decomposed parts, enabling the search process to begin not from an arbitrary table, but from the accumulated experience gained during previous learning tasks on one specific solution.

In the following subsections, we focus on describing the algorithmic design of our proposed EVORL-CODBA.

3.2. EVORL-CODBA: basic scheme description

The integration of Q-learning into the basic framework of an EA can be presented at different levels: in the generation of the initial solution or population; at the objective function level to auto-generate the evaluation of solutions; in the core operators of an EA: selection, crossover, mutation or environmental selection to guide them in the search space; or also in the tuning parameters process. In this work, we are interested in integrating the Q-learning technique into the upper level evaluation process. At this stage, the upper EA appeals the lower task to determine the corresponding lower reaction. In the classical scheme, another EA is invoked at this stage to search for or generate the lower solution. In our proposal, we choose to integrate a Q-learning technique at this stage to determine the (near) optimal reaction and to improve the convergence rate, which will enhance the performance of the upper-level procedure as described by Figure 1.

Toward this goal, the step-by-step procedure of our EVORL-CODBA is illustrated through Figures 1 and 2, which are intended to depict different levels of details of our proposed approach. Specifically, Figure 1 presents the general scheme of our methodology, highlighting its main components and overall structure. In contrast, Figure 2 provides a more detailed view, focusing on the specific mechanisms and interactions within the core components of the approach, and showing how reinforcement learning is integrated at the lower level. In addition, the pseudo-code of our proposal is depicted by Algorithms 1–4 to illustrate both levels' instructions. Here, it's important to note that we implement the genetic algorithm as a resolution method within the proposed scheme at the upper level. In fact, any other evolutionary algorithm can be employed in this context. To this end, we begin with the Initialization process (line 3, Algorithm 1), which randomly generates an initial upper-level population of size N, where each individual encodes a candidate solution based on the upper-level problem by identifying the upper decision variables x_u. In fact, in the context of bi-level optimization, this initialization step must also launch a lower-level optimization process to evaluate each randomly generated upper-level solution.

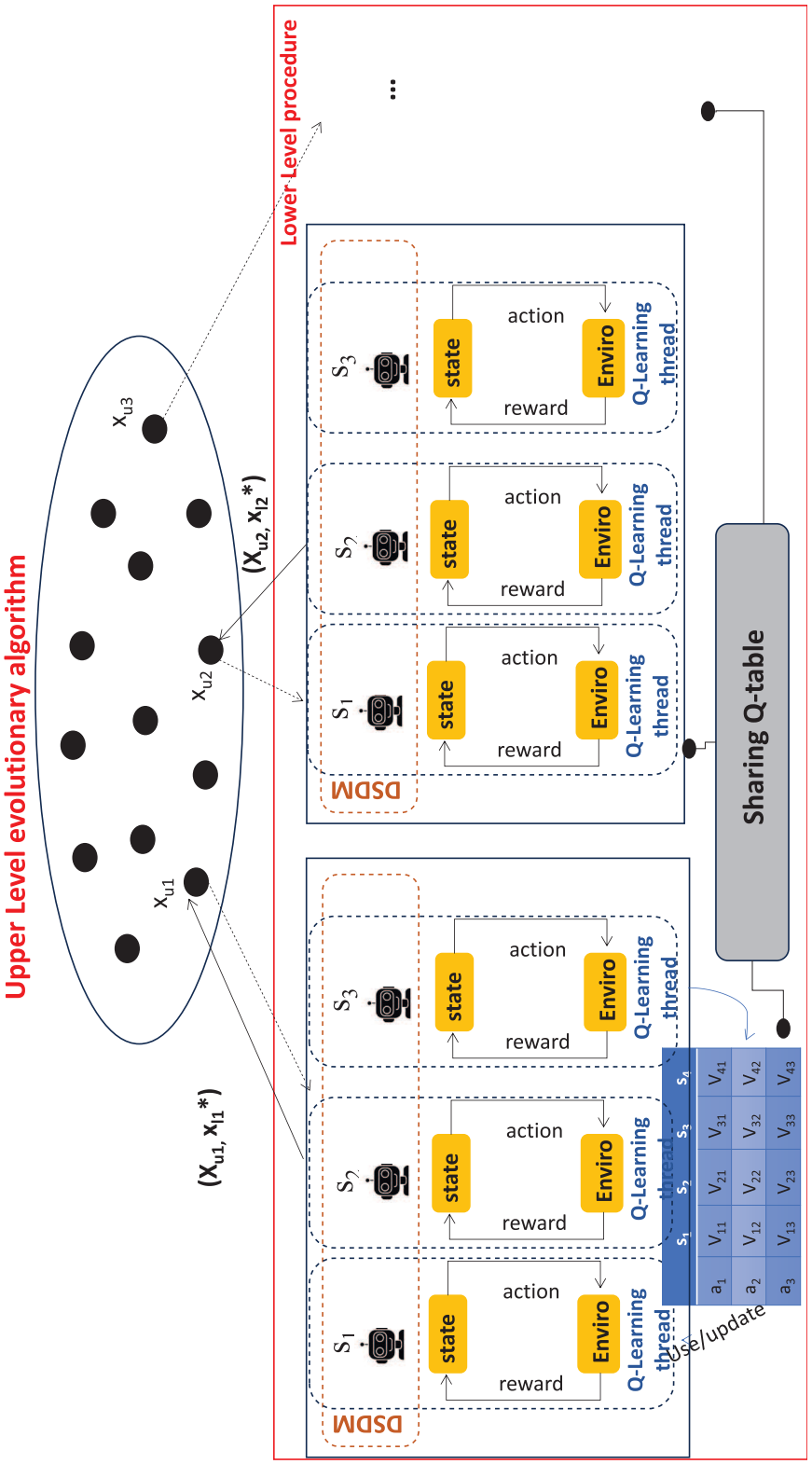

FIGURE 2. Illustration of EVORL-CODBA scheme.

Consequently, the initialization function should take both upper-level and lower-level parameters as input to ensure proper generation and evaluation between the two levels. Based on this population, the algorithm enters an iterative loop consisting of the following steps :

- **UpperSelection (line 5, Algorithm 1):** This step identifies the mating pool set, composed by the parent solutions that will undergo the upper evolution, in which we select promising individuals from the current population based on their fitness values. We adopt a tournament selection mechanism (described by Algorithm 2) to maintain diversity and guide the search toward high-quality regions.
- **UpperVariation (line 6, Algorithm 1):** This function applies evolutionary variation operators (crossover and mutation), to the selected individuals to produce new candidate solutions. By recombining and mutating existing solutions, variation operators enable the algorithm to discover unexplored regions of the solution space, which increases the chances of finding optimal or near-optimal solutions over successive generations. It is important to note that the choice of appropriate operators depends on the characteristics of the optimization problem and is detailed in Subsection 4.1 for the considered experimental setup.

Once the offspring population Q_t is generated (step 6 in Algorithm 1), an evaluation process needs to be performed. At this stage, a lower level learning process applying the Q-Learning technique with a Variable Neighborhood Search (VNS) algorithm is invoked (line 9 and 10, Algorithm 1). The step by step description of our Lower EVOLRL_CODBA is presented in Subsection 3.3.

We note here that, ordinarily, the basic CODBA scheme begins the lower search by generating distant reference solutions within the search space using its DSDM method. Subsequently, it generates a subpopulation around each solution to form diverse subpopulations, on which it runs a population-based algorithm to evolve the different parts in parallel using multi-threading.

In our EVORL-CODBA approach, we start the lower level by a set of reference solutions generated by CODBA' DSDM method that cover the whole lower search space. After that, we execute a VNS procedure for the different generated solutions assisted by a Q-learning technique. The idea here is to hybridize the Q-Learning strategy with the VNS procedure to determine the best search policies corresponding to the best neighborhood operators adopted to find the optimal lower reaction (step 10, Algorithm 1). Our main concern is to exploit the strengths of Q-Learning to assist the VNS procedure in order to create a more effective optimization process, particularly adopted to complex problem such as the bi-level context. It is important to note here that with this Q-learning VNS procedure we maintain the following facts:

- We maintain one common Q-Table structure for the different lunched Q-learning VNS algorithms on each reference point on the lower level task. Indeed, when the lower evolution for one upper solution is completed, the corresponding learning policies stored in this Q-Table will be transferred to a next lower task, transferring then the knowledge acquired from prior tasks (*cf.* Fig. 2).
- For each reference solution, the Q-learning process is executed through a thread, thus the different solution parts will be evolved in parallel using the multi-threading mechanism (*cf.* Fig. 2).

After performing the learning lower task for all mating pool upper individuals, an upper offspring population is generated and filled with the new upper level population using a replacement strategy (step 13, Algorithm 1). In this context, we adopt an elitist environmental selection strategy that ensures the retention of the best individuals, regardless of whether they come from the current parent population or the newly generated offspring. In this way, the upper population is formed with N best solutions from the combined population (parent individuals with offspring solutions) according to an elitist selection to prevent the algorithm from wasting time on re-evaluating previously rejected partial solutions. This process will be repeated until the termination criterion is achieved.

3.3. Lower level learning strategy

In this section, we describe the lower Q-Learning assisted VNS algorithm which is depicted by both Algorithms 3 and 4. As we mentioned previously, we start the lower level by applying CODBA'DSDM method

Algorithm 1: Upper EVOLRL_CODBA.

Input: $PopsizeU$: upper population size, Gu: upper generation, $PoolSize$: Mainting pool size, LP: Lower Parameters, QP: Q-learning parameters, V: variation parameters, $Q-Table$: common shared Q-table
Output: $s*$: Best found solution
01**: Begin**
02: $t \leftarrow 0$;
03: UpperPop←**Initialization**($PopsizeU$, LP);
04: **While** (Not TerminationCriterion) **do**
05: MaitingPool←**UpperSelection**(UpperPop, $PoolSize$:);
06: Q_t←**UpperVariation**(MaitingPool, V);
07: $Q'_t \leftarrow \varnothing$
08: **For** each Ind in Q_t **do**
09: x_u ←UpperSolution(Ind); /extract the upper decision variables/
10: $Q'_t \leftarrow Q'_t \cup$ **Lower EVOLRL_CODBA**(x_u, LP, QP, $Q-Table$);
11: **End For**
12: $Pt+1$←UpperPop $\cup Q'_t$;
13: UpperPop←**EnviromentalSelection**(P_{t+1})
14: $t \leftarrow t+1$;
15: **End while**
16: s*←**BestSolution**(UpperPop);
17: **End**

Algorithm 2: Upper selection function.

Input: $UpperPop$: upper population, $PoolSize$: Mainting pool size, k: tournement Selection Parameter
Output: $Maiting\_pool$
01**: Begin**
02: $MatingPool \leftarrow \varnothing$
03: **While** size($Maiting\_pool$) ¡ $PoolSize$ **do**
04 $TournamentSet \leftarrow \varnothing$
// Step 1: Randomly select k individuals
05 $TournamentSet$←Randomly select k individuals from $UpperPop$
// Step 2: Select the best individual among the k candidates
06 $BestInd$← individual in $TournamentSet$ with best fitness
// Step 3: Add the best to the mating pool
07 Add $BestInd$ to $MatingPool$
08 Reinsert remaining individual in $TournmentSet$ insto the $UpperPop$
09: **End while**
10: return $Maiting\_pool$
11: **End**

in order to generate a set of spread points $RefSols$ representative of all parts of the decision space (step 2, Algorithm 3). For more details about this decomposition method, readers are refereed to [11]. Then the Q-table is initialized before starting the learning process. In this way all Q-values are typically set to a constant value (commonly zero) to encourage learning through exploration (line 05, Algorithm 3). After that, we launch a reinforcement Q-learning algorithm We note that the detailed description of this function is provided Algorithm 4 and further elaborated in Subsection 3.3.1. This procedure is combined with a local search technique (VNS) at every $RefSol$ solution, facilitating parallel execution of the different parts. We note here that the Q-Table is structured with $|RefSols|$ rows corresponding to the number of reference solutions generated by DSDM and M columns corresponding to the number of neighborhood operators that we will adopted with our VNS. At this stage, for each $RefSol$ point we will obtain a new local solution after a number of Q-learning assisted VNS iterations T. If the latter is better than the best stored produced member, we update the global

local best solution and we repeat the process for the different generated $RefSol$ solutions. We note here that each produced solution is evaluated using the considered lower-level objective function and constraints.

Algorithm 3: Lower EVOLRL_CODBA.

Input: upper candidate solution x_u^0, Dxl: range value of lower decision variables, $Fdec$: lower decomposition factor, N: Problem Dimension, α: Learning Rate, γ: Discount Factor, ε: Exploration Rate,$Q-Table$;
Output: Best lower solution (x_u^0, x_l^*)
01**: Begin**
02: $RefSols \leftarrow$**DSDM** (Dxl, $Fdec$, N);
03: $x_l^* \leftarrow$**Ø**; $i \leftarrow$ 1**;**
04: $LocalBestSol \leftarrow$ **Ø**;
05: Q-Table ← InitializeTable(states, actions, initial_value)
06: **For** each sol in $RefSols$ d**o**
07: $LocalBestSol[i] \leftarrow$**Q-Learning**($\alpha$, γ, ε, $Q-Table$, $RefSols[i]$)
08: $i \leftarrow i+1$;
09: **EndFor**
10: $x_l^{*'} \leftarrow$ BestSolution($LocalBestSol$);
11: **If** ($x_l^{*'}$ ¡ x_l^*) **Then**
12: $x_l^* \leftarrow x_l^{*'}$;
13: **End If**
14: **End**

Algorithm 4: Q-learning.

Input: α: Learning Rate, γ, Discount Factor, ε: Exploration Rate, $Q-Table$, $RefSol$,A: set of neighborhood structures or operators
Output: Local Best Solution s'
01**: Begin**
02: **While** (Not TerminationCriterion) **do**
03 $t \leftarrow$Random(0, 1);
04: **If** t ¡ ε **then** /*Choose a random action (exploration)
05: $NeighborhoodMove\ N \leftarrow$ **RandomAction**($Q-Table[RefSol]$))
06: **Else** /* Choose the action with the best Q-value (exploitation)
07: $N \leftarrow$ **ArgMin**_{$a \in Actions$} $Q-Table([RefSol], a)$
08: **End If**
09: $s' \leftarrow N(RefSol)$; /*Apply the selected $neighborhoodMove\ N$ to $RefSol$
10: $R \leftarrow f(s')$-$f(RefSol)$; /*Compute the reward R
11: **If** ($R \leq$**0**)**Then** /* $f(s')$ ¡$f(RefSol)$
12: Q(s',N) ← Q(s',N)+α(R+γ a´ max Q(s´, a´)−Q(s,a))
13: Update($Q-Table$)
14: Set $RefSol = s'$
15: **End If**
16:**End While**
17**: End**

3.3.1. Q-learning assisted VNS adaptations

Before explaining the basic process of Q-Learning assisted VNS, we need to identify the adaptations made to integrate Q-learning with the VNS procedure;

(1) **State representation:** it reflects in our case to the configuration of solution being evaluated at the lower level of the algorithm.

(2) **Action:** Action a ∈ action space A. Actions define a set of neighborhood structures or operators using with the VNS.
(3) **Reward calculation:** A reward function is a mapping from states and actions into real numbers. Thus, a reward indicates how action chosen in particular state improves or worsens a solution to the problem. In our case, we compute the reward value so that it aligns with the improvement or deterioration in the objective function. It is measured by the difference between the fitness of the current solution obtained s' and the fitness of the best solution found during previous iterations $s*$. This reward setting rewards the neighborhood search operator that could enhance the VNS process.
(4) **Exploration and exploitation technique:** The main concern here is how to exploit the principles of Q-learning while balancing exploration (*i.e.* trying new actions to discover their effects and potential reward) and exploitation (*i.e.* using the current knowledge to select the best-known actions that improve the reward) in the search space. In this work, we use the ϵ-greedy policy commonly used to achieve this balance as follows:

$$n' = \begin{cases} \text{ArgMin}_{n \in N}\, Q(s', n) & \text{if random} < 1 - \varepsilon \\ \text{Select randomly from } N & \text{otherwise.} \end{cases} \tag{1}$$

Where, n is a new neighborhood strategy defined in the set N and is ranged in $[0 - 1]$ interval. We note that the initial ϵ parameter is set to:

$$\varepsilon = 0.5\,(1 - \text{FE}_t/\text{MaxFE}). \tag{2}$$

Where FE_t is the number of consumed Function Evaluation (FE), and MaxFE is the predefined maximum number of FFs. Basing on this formula we gradually shifts from exploration to exploitation.

3.3.2. Q-learning assisted VNS basic iteration

The Q-learning assisted VNS could provide useful information for choosing more proper lower-level search neighborhood in the evolutionary optimization process. The main procedure is as follows: We begin by generating a t random value from the interval $[0 - 1]$. If t value is less than ϵ, we choose a random neighborhood move from the list of adopted move with our VNS algorithm. Otherwise, we select the action with the highest Q-value in the Q-Table for the current state $RefSol$. Then, we apply the neighborhood operator N_i (selected action) on the current solution $RefSol$ to produce a new solution s' (step 9, Algorithm 4). We compute the objective function (or fitness) for the new solution s' and then we calculate the induced reward (step 10 in Algorithm 3). If the new solution improves the objective function than the reward will be negative, in this case we use the following Q-learning update formula to obtain the Q-value of s':

$$Q(s', N) \leftarrow Q(s', N) + \alpha(R + \gamma\, N' \max Q(s', N') - Q(s, N). \tag{3}$$

In which α is the learning rate, a parameter that determines how much the new information should be blended with the old information (the current Q-value). R is the computed reward. γ is the discount factor, which ranges between 0 and 1 and determines the importance of future rewards compared to immediate rewards. The term $N' \max Q(s', N')$ defines the maximum expected future reward that we can achieve if we apply an action N' from the newt state s'. Referring to Choong *et al.* [14], we choose in this work to fix the learning rate α as:

$$\alpha = 1 - 0.9 \times \text{FE}_t/\text{MaxFE}. \tag{4}$$

In this formula, the algorithm will start by a high α value and then the value decreases until the search progress, enabling the algorithm to place greater emphasis on past experiences. After that, we update the Q-Table by the new state and Q-value (step 13, Algorithm 4). We note here that to avoid burdening the algorithm, we chose to update the Q-Table and insert the new state only in cases where the reward is improved. We continue iterating this process until a stopping criterion is met.

4. Experimental study

In order to evaluate the potential of integrating Q-learning strategy in the lower level of CODBA scheme, we conduct in this part an experimental analysis of our EVORL-CODBA against:

- **CODBA-VNS:** A co-evolutionary hybrid decomposition-based algorithm which is an EBA algorithm that handles a VNS strategy in the lower level problem [11].
- **CODBA:** The basic co-evolutionary decomposition-based algorithm that uses a GA at the upper and lower level problem [9].
- **NBA:** A nested method that uses hierarchically EA at both levels to handle bi-level optimization problems [35].

The main concern of this experimental study is to identify the benefits of integrating Q-learning with EBA, in terms of convergence rate and solution quality. This is why, we conduct two experimental analysis parts. We focus firstly on comparing the performance (*i.e.* solution quality) of EVORL-CODBA mainly to the baseline scheme *i.e.*, running experiments using CODBA without Q-learning. In the second part, we investigate the performance of the three compared algorithms in the upper level using the average upper level fitness values and in the lower-level decision-making process using the rationality metrics of Legillon *et al.* [28] (*i.e.* evaluating the lower convergence) to highlight how efficiently the proposed integrated version explores the search space.

We should note that all the used algorithms are applied to the same combinatorial optimization problem, the Bi-level Multi-depot Routing Problem (Bi-MDVRP) from supply chain management, which was chosen for comparison purposes with these experiments. We adopt the same components for all three algorithms in this comparison: the stopping criterion, the variation operators, and the initializers across both setups, to ensure fairness in the comparison. As well, all algorithms are coded in Java programming language, and all simulations are performed on the same machine (Intel(R) Core(TM) i3-10110U CPU @ 2.10 GHz 2.59 GHz).

4.1. Benchmark problems: the Bi-MDVRP instances

Several combinatorial optimization problems are cast within the bi-level framework. In this experimental study we choose the bi-level multi-depot vehicle routing problem which is an extension of the traditional vehicle routing problem that reveals two levels of optimization. In the context of supply chain management, the Bi-MDVRP typically involves two levels of decision-making.

- Upper entity: This deals with route optimizing for vehicles transporting goods from multiple depots to customers in order to minimize the delivery cost. To achieve the lowest possible cost for delivering products, this upper level is connected to a lower-level entity.
- Lower entity: This level involves decisions around production plan and quantities. Based on the customer demand identified in the upper level (*i.e.* order received from the upper entity), the production entity seeks the most efficient plan to produce the necessary quantity to meet demand at the lowest possible cost. This ensures that the overall cost of products delivered to the client will be minimized.

According to this definition, we adopted the commonly used test problems within the MDVRP community [7] that allow assessing the performance of any algorithm design with respect to different kinds of difficulties (small and large-sized problems). Indeed, we add plants as there are depots randomly located on the map to create the bi-level instances. To this end the used benchmark problems are the set $S1$ composed by the $Bi-pr$ suite and the set $S2$ which is composed by the $Bi-p$ one.

To fit EVORL-CODBA to the Bi-MDVRP problem, we used the two following perturbation mechanisms at the upper level as crossover and mutation operators, RBX and UMutation respectively. Indeed for the lower level we used the three following neighborhood search strategies:

- One-opt operator which is based on element-exchange operator.
- Insert-Move (k) which changes the position of a segment of k depots for one plant u.
- Exchange (k; $k1$) which swap between two plant indexes between two different depots.

TABLE 1. Design parameters and their levels.

	A	B	C	D
	γ	*LFE*	*Upop*	*UFE*
Low	0.2	2000	20	2000
Medium	0.5	3000	50	2500
High	0.8	5000	100	5000

TABLE 2. The orthogonal array L_9.

Experiments	A	B	C	D
1	A(1)	B(1)	C(1)	D(1)
2	A(1)	B(2)	C(2)	D(2)
3	A(1)	B(3)	C(3)	D(3)
4	A(2)	B(1)	C(2)	D(3)
5	A(2)	B(2)	C(3)	D(1)
6	A(2)	B(3)	C(1)	D(2)
7	A(3)	B(1)	C(3)	D(2)
8	A(3)	B(2)	C(1)	D(3)
9	A(3)	B(3)	C(2)	D(1)

4.2. Parameter tuning and statistical methodology

The main issue in this section is to find the optimal parameter settings used for comparison. In this context, we used the Taguchi's Design Of Experiment (TDOE) to determine which specific configuration of parameters is required with least amount of simulation [41]. This TDOE suggests the use of orthogonal arrays to organize the parameters that affect the algorithm and the levels at which they should vary. In this work, the main parameters that affect the EVOLRL-CODBA performance are: the discount factor γ, the number of FE used by the VNS at the lower level LFE, the upper population size $Upop$, and the number of upper Function Evluations UFE. Table 1 presents the levels defined for each parameter. Indeed, the resulting orthogonal array with four factors and three levels in Taguchi method is $L9(3*4)$, depicted by Table 2. Figure 3 displays the obtained results in terms of average upper level fitness means. Indeed, Figure 4 displays the obtained SNR (Signal-to-Noise Ratio) values for both small and large instance problems.

Regarding both performed experiments and resulting SNR values, the optimal parameters for the small-scale case are: A(3), B(2), C(2) and D(3). Furthermore,the optimal levels regarding the large size instances are A(3), B(2), C(3) and D(3). We note that the same number of FEs is used as a termination criterion to turn these parameters with TDOE. Moreover, the crossover and mutation probability are set to 0.9, 0.1 respectively, as tuned in the literature. Regarding the specific problem parameters such the cost of production of each plant, the production availability and the cost of acquiring good from plant to depot are fixed referring to [7].

To ensure reliable comparisons, in addition to parameter tuning, we adopted a statistical methodology to determine whether there is a significant difference between the results of the compared algorithms. For this reason, we used in a pairwise fashion the Wilcoxon rank test in order to check whether there is a statistical difference between the obtained samples results. The results of the significance tests at level $\alpha = 0.05$ are presented in the form of $(- : no\ significance)$ and $(+ : significance)$ in the order in which the algorithms appear in the Table of result. Thus, for each couple (algorithm, instance), we perform 31 runs as recommended in [17]. Consequently, we can determine whether the performance difference between EVORL-CODBA and one of the other approaches is statistically significant or just a random result. Afterward, the obtained results were

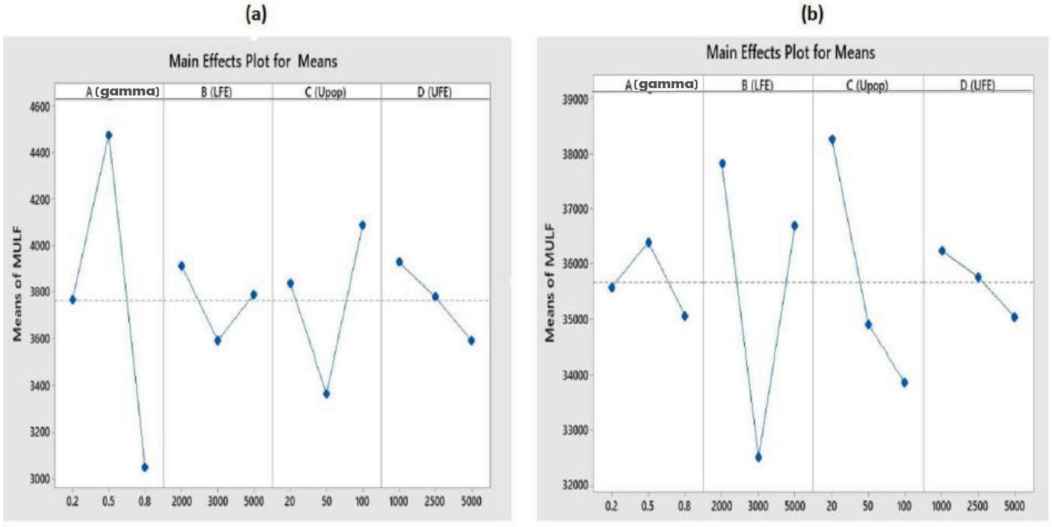

FIGURE 3. The Mean of Upper Level Fitness (MULF) plot for Bi-MDVRP instances in Taguchi methodology: (a) for small scale problems, and (b) for large scale problems.

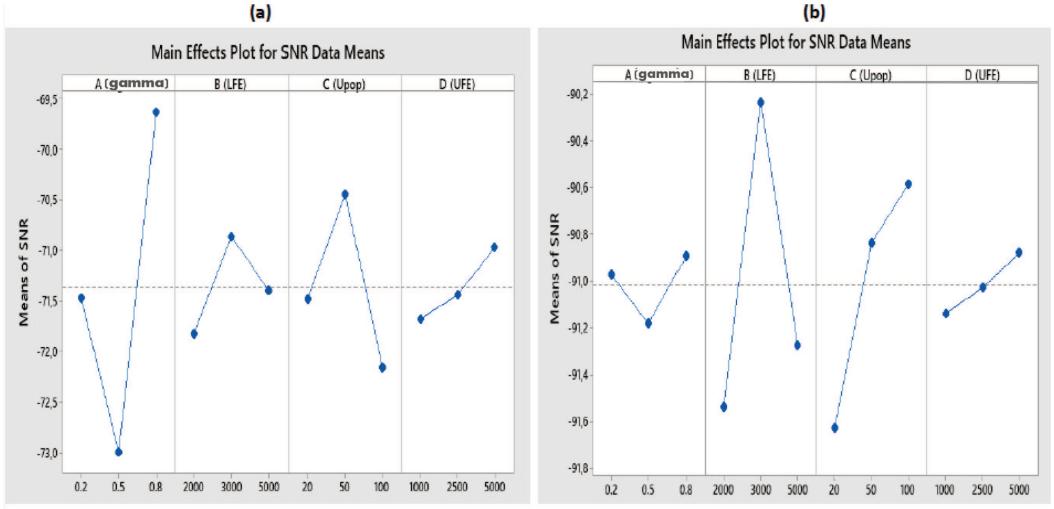

FIGURE 4. The SNR ratio plot for Bi-MDVRP instances in Taguchi methodology: (a) for small scale problems, and (b) for large scale problems.

analyzed using three performance indicators which are: (1) The average as a measure of central tendency, (2) The direct rationality, and (3) The weighted rationality [28] to measure the effectiveness of lower level reactions.

4.3. Experimental results

In this section, we present the statistical experimental results of the four considered algorithms under comparisons. We structured this part into two subsections: In the first one, we focus on highlighting the performance differences of our algorithm when using Q-learning versus without it. Regarding this issue, we report a com-

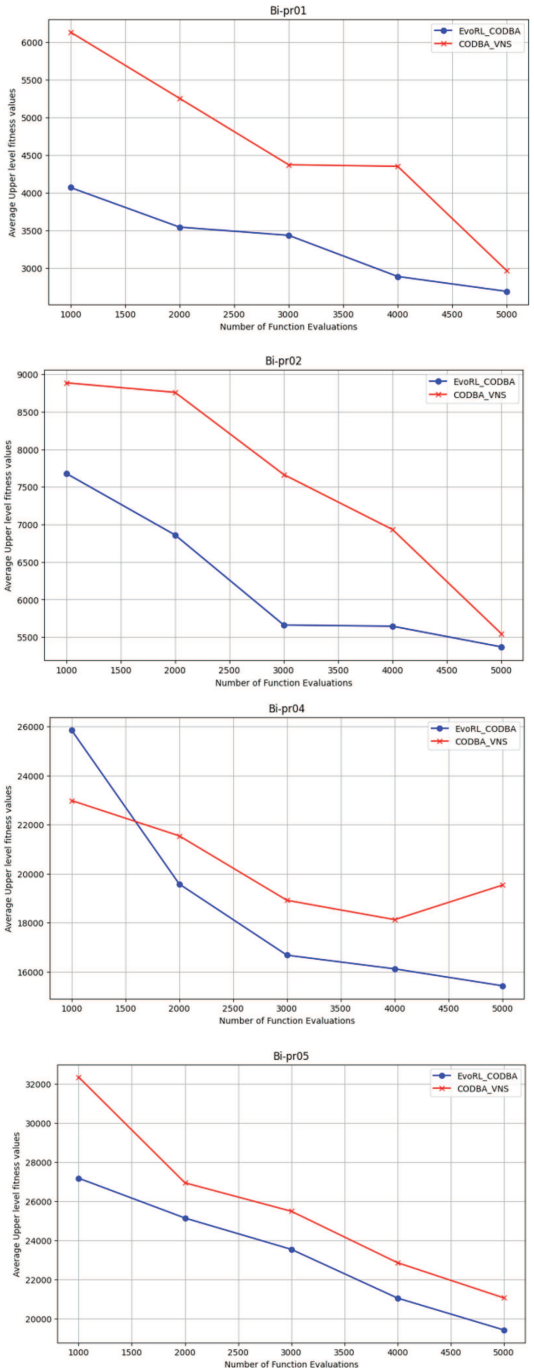

FIGURE 5. Convergence comparison of EvoRL-CODBA and CODBA VNS on small and medium-sized bi-MDVRP instances.

parative analysis between EVORL-CODBA and CODBA (*cf.* Subsect. 4.3.1). In the second part, we present a comparative analysis of our proposal against other established schemes (*cf.* Subsect. 4.3.2).

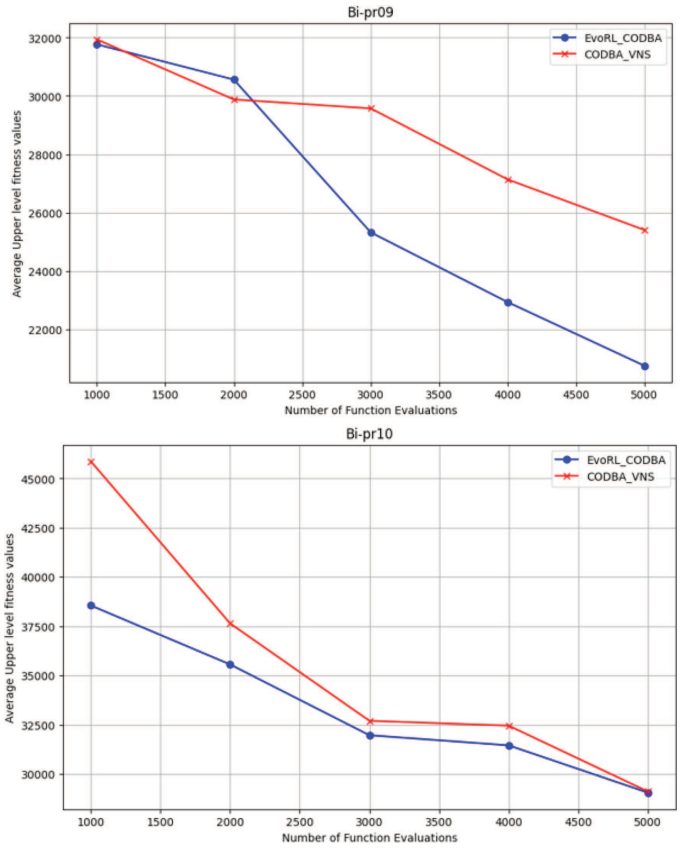

FIGURE 6. Convergence comparison of EvoRL-CODBA and CODBA VNS on large-sized bi-MDVRP instances.

4.3.1. Impact of Q-learning integration on CODBA performance

To show the impact of integrating Q-learning technique with the lower level of CODBA, we study in this section the performance of our EVOLRL-CODBA regarding the baseline version (*i.e.* CODBA without Q-learning). Our main motivation here is to demonstrate the ability of the Q-learning strategy in improving the convergence rate of the whole algorithm and consequently reducing CPU execution time, thereby enhancing the overall efficiency and scalability of the approach. Figures 5 and 6 describe the average upper level fitness values obtained by the two confronted algorithms on small size problem (Bi-pr01 and Bi-pr02), medium large size problem (Bi-pr04 and Bi-pr05) and large size problems (Bi-pr09 and Bi-pr10).

To measure the convergence rate of the two algorithms, we compared the fitness values obtained with different number of functions evaluations values fixed as stopping conditions. The results suggest that the greater the number of FEs, the better the average upper-level fitness value. We can observe from these Figures also that EVORL-CODBA presents the best performance on different termination criteria levels for the used instances. These results demonstrate that by integrating Q-learning, we are adaptively fine-tune the exploration-exploitation balance, enabling our proposal to better find the (near) optimal reaction regarding an upper decision solution. The superior performance of EvoRL-CODBA compared to CODBA-VNS can be attributed to several key improvements introduced by our approach:

– **Reinforcement Learning-Driven Adaptation:** EvoRL-CODBA integrates a reinforcement learning (RL) mechanism at the lower level. This allows the algorithm to learn optimal neighborhood strategies over time,

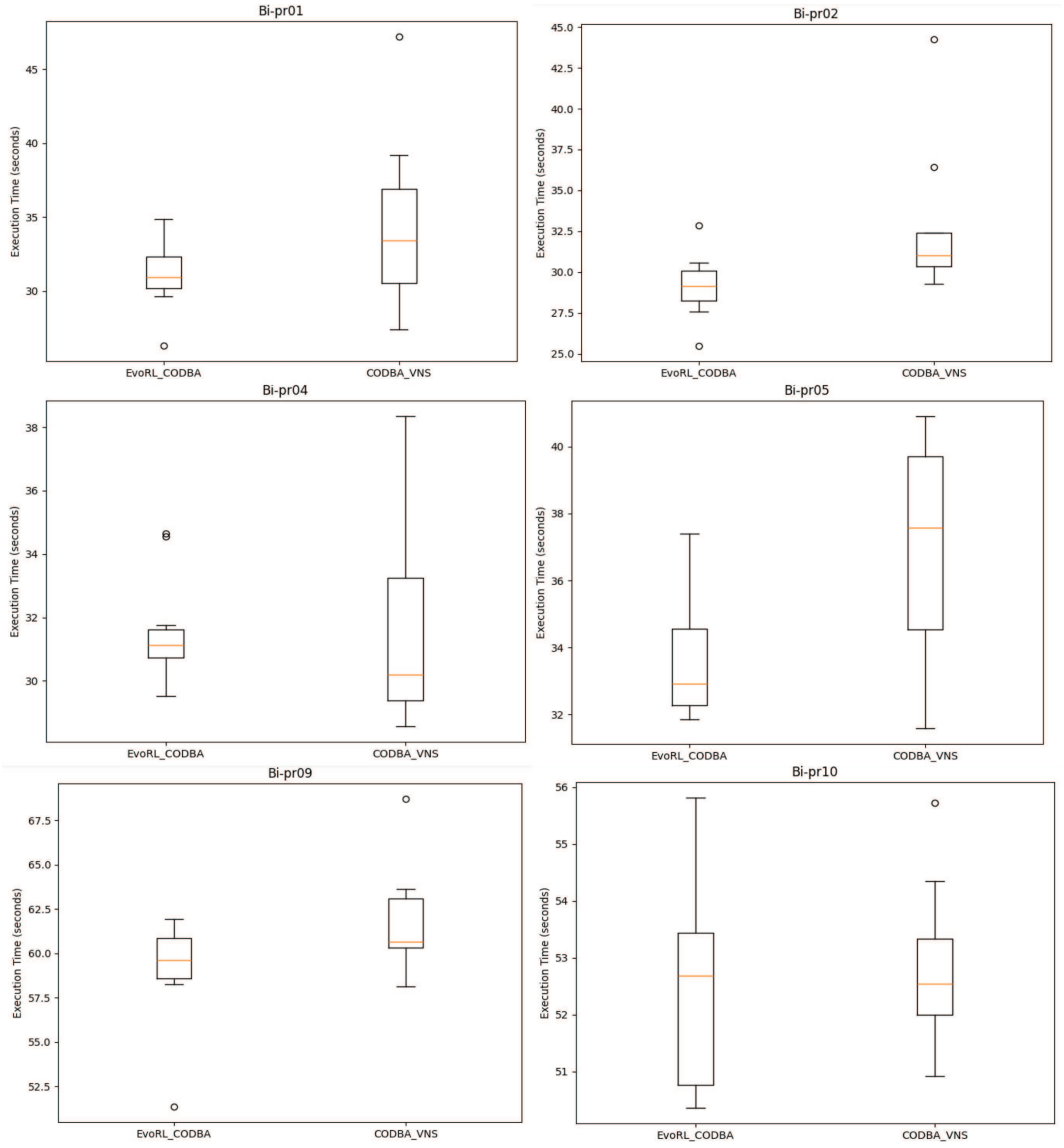

FIGURE 7. CPU time boxplots for Evol-CODBA and CODBA-VNS on small, medium and large scale test problems.

rather than relying on fixed or manually tuned local search operators, as is the case in CODBA-VNS. This dynamic adaptation ensures more efficient exploration and exploitation of the lower-level solution space.

– **Bi-level Co-Design with Learning Feedback:** EvoRL-CODBA leverages information from the learning process to guide upper-level evolution. Specifically, the performance of lower-level strategy, learned through Q-learning, feeds back into the upper-level individuals which may improve convergence quality and accelerates the search process

To further investigate the EVORL-CODBA behavior we present in Figure 7 several box plots for three types of Bi-MDVRP instances. As shown by these Figures, the box plots give an idea about the first quartile,

TABLE 3. Average upper level fitness values of EVORL_CODBA, CODBA-VNS, CODBA and N-BEA on Bi-MDVRP. The symbol "+" means that H_0 is rejected while the symbol "–" means the opposite. The best values are highlighted in bold.

	Instances	EVORL_CODBA	CODBA-VNS	CODBA	N-BEA
Suite Bi-pr	Bi-pr01	**2583** (+++)	2940 (++)	2795 (+)	6470
	Bi-pr02	**5258** (+++)	5539 (++)	5461 (+)	8863
	Bi-pr03	**11 018** (+++)	12 486 (++)	11 527 (+)	14 742
	Bi-pr04	**15 203** (+++)	19 648 (++)	18 883 (+)	20 798
	Bi-pr05	**18 102** (+++)	21 452 (++)	22 576 (–)	23 673
	Bi-pr06	**27 027** (+++)	29 501 (++)	28 690 (+)	31 933
	Bi-pr07	**4530** (+++)	4742 (++)	4679 (+)	15 073
	Bi-pr08	**10 102** (+++)	13 567 (++)	10 844 (+)	22 829
	Bi-pr09	**19 001** (+++)	25 492 (++)	20 988 (+)	36 154
	Bi-pr10	**29 000** (–++)	29 213 (++)	31 337 (+)	40 205
Suite Bi-p	Bi-p01	**1385** (+++)	1479 (++)	1501 (+)	4635
	Bi-p02	**1330** (+++)	1845 (++)	1461 (+)	5509
	Bi-p03	**1852** (+++)	2180 (++)	2324 (+)	8651
	Bi-p04	**3010** (+++)	4572 (++)	3133 (+)	3231
	Bi-p05	**2896** (+++)	3135 (+–)	3108 (+)	3244
	Bi-p06	**2980** (+++)	3624 (++)	3157 (+)	4075
	Bi-p07	**2925** (+++)	3624 (++)	3221 (+)	5635
	Bi-p08	39 852 (+–+)	44 001 (++)	**39 851** (+)	43 711
	Bi-p09	**42 858** (+++)	45 443 (++)	45 755 (+)	44 970
	Bi-p10	**43 715** (+++)	48 557 (++)	62 173 (+)	60 873
	Bi-p11	53 224 (+++)	56 162 (++)	**51 696** (+)	47 703
	Bi-p12	6368 (+++)	6464 (++)	**6112** (+)	12 891
	Bi-p13	**5925** (– –+)	6004 (– +)	5946 (+)	18 410
	Bi-p14	**5799** (+++)	6932 (++)	6119 (+)	18 141
	Bi-p15	**19 496** (+++)	22 682 (++)	19 793 (+)	28 783
	Bi-p16	20 686 (+ – +)	23 947 (++)	**19 957** (+)	24 111
	Bi-p17	22 235 (+++)	23 219 (++)	**19 924** (+)	29 120
	Bi-p18	**46 933** (+–+)	50 351 (++)	47 119 (+)	62 332
	Bi-p19	**46 264** (+++)	48 587 (++)	46 560 (+)	65 013
	Bi-p20	**44 721** (+++)	50 124 (++)	46 244 (+)	71 370
	Bi-p21	**96 660** (+++)	102 773 (++)	98 232 (+)	143 140
	Bi-p22	**97 991** (+++)	99 125 (– +)	99468 (+)	140 894
	Bi-p23	**97 439** (+++)	102 090 (++)	99 462 (+)	133 514

Notes: H_0: median (Algorithm 1) = median (Algorithm 2), H_1: median (Algorithm 1) ≠ median (Algorithm 2).

the third one, the median, the best value, the worst one, and the outliers. This fact provides a clearer view about the algorithms' efficiently. Indeed, we observe that our proposed approach consumes less CPU times than CODBA based on VNS which confirms the potentially of EVORL in generating higher-quality solutions in less computational time.

4.3.2. Comparative results and analysis

In this subsection, we detail the EVORL-CODBA performance along with three competitive algorithms. In our experimental study, the upper-level performance is tested using the average upper level fitness values and the lower level is measured using the direct rationality and the weighted rationality metrics. Tables 3, 4 and 5 illustrate the statistical obtained results. We note here that the statistical result are mentioned with the used comparative metric in each table. As the statistical comparison is conducted for each pair of algorithms,

TABLE 4. Average direct rationnality values of EVORL_CODBA, CODBA-VNS, CODBA and N-BEA on Bi-MDVRP. The symbol "+" means that H_0 is rejected while The symbol "–" means the opposite. The best values are highlighted in bold.

	Instances	EVORL_CODBA	CODBA-VNS	CODBA	N-BEA
Suite Bi-pr	Bi-pr01	**0.95 (– ++)**	1.01 (++)	1.50 (+)	7.89
	Bi-pr02	**2.01 (+++)**	2.31 (++)	2.42 (+)	8.45
	Bi-pr03	**7.54 (+++)**	8.56 (++)	8.02 (+)	12.32
	Bi-pr04	**6.52 (+++)**	6.89 (++)	7.52 (+)	11.09
	Bi-pr05	**3.98 (+++)**	4.58 (–+)	4.98 (+)	8.25
	Bi-pr06	5.60 (+++)	3.05 (++)	3.12 (+)	9.52
	Bi-pr07	**1.65 (+++)**	1.75 (++)	1.89 (+)	10.09
	Bi-pr08	**5.11 (+++)**	5.22 (–+)	5.55 (+)	11.01
	Bi-pr09	**6.20 (+++)**	6.50 (++)	6.25 (+)	10.57
	Bi-pr10	**5.77 (+++)**	5.98 (++)	6.33 (+)	9.43
Suite Bi-p	Bi-p01	**1.59 (+++)**	1.67 (++)	1.88 (+)	6.81
	Bi-p02	2.80 (+++)	1.50 (++)	1.70 (+)	7.34
	Bi-p03	**1.55 (+++)**	1.66 (++)	1.79 (+)	8.52
	Bi-p04	**1.62 (+++)**	1.77 (++)	1.80 (+)	2.67
	Bi-p05	**1.42 (+++)**	1.62 (–+)	1.76 (+)	2.71
	Bi-p06	**1.62 (+++)**	1.70 (++)	2.01 (+)	4.01
	Bi-p07	**2.75 (+++)**	2.85 (++)	2.98 (+)	5.85
	Bi-p08	**5.33 (+++)**	5.45 (++)	5.75 (+)	13.78
	Bi-p09	**6.11 (+++)**	6.32 (–+)	6.25 (+)	14.9
	Bi-p10	**5.34 (+++)**	5.41 (++)	5.80 (+)	15.21
	Bi-p11	**10.01 (+++)**	10.45 (++)	10.90 (+)	16.91
	Bi-p12	6.91 (+++)	5.28 (++)	5.35 (+)	7.59
	Bi-p13	7.01 (+++)	5.20 (++)	5.18 (+)	5.50
	Bi-p14	**6.99 (+++)**	7.69 (++)	7.82 (+)	7.90
	Bi-p15	7.10 (– ++)	7.17 (++)	7.25 (+)	11.95
	Bi-p16	**9.56 (– ++)**	9.67 (–+)	9.98 (+)	12.12
	Bi-p17	**11.01 (+++)**	11.72 (–+)	11.82 (+)	12.43
	Bi-p18	**14.97 (+++)**	15.98 (–+)	15.58 (+)	20.56
	Bi-p19	**16.84 (+++)**	17.10 (++)	16.98 (+)	21.69
	Bi-p20	**17.58 (+++)**	18.11 (++)	18.34 (+)	19.19
	Bi-p21	**18.06 (+++)**	18.89 (–+)	18.48 (+)	30.20
	Bi-p22	**19.01 (+++)**	19.60 (–+)	19.11 (+)	32.30
	Bi-p23	**16.75 (+–+)**	17.52 (++)	17.06 (+)	34.55

Notes: H_0: median (Algorithm 1) = median (Algorithm 2), H_1: median (Algorithm 1) ≠ median (Algorithm 2).

the first cell of one Table (*e.g.* in the column for EVORL-CODBA) contains three symbols: for describing the statistical result between (EVORL-CODBA, CODBA-VNS), then (EVORL-CODBA, CODBA) and the third symbol define the statistical result regarding the couple (EVORL-CODBA, N-BEA). When presenting the results for the second algorithm, it is only necessary to report the statistical test results between CODBA-VNS and CODBA, and between CODBA-VNS and N-BEA, since the pair (CODBA-VNS, EVORL-CODBA) has already been evaluated and reported in the first column, and vice versa. In addition, we note that we use the parameters obtained by TDOE for all algorithms under comparison.

We observe from the Table 3 that for an upper-level performance, the co-evolutionary decomposition-based algorithms (*i.e.* EVORL-CODBA, CODBA-VNS and CODBA) present better performance than the nested evolutionary algorithms on all instance problems. This result highlight the potential of the decomposition mechanism to promote diversity and hence emphasize convergence toward the optimal bi-level solution.

TABLE 5. Average weighted rationnality values of EVORL_CODBA, CODBA-VNS, CODBA and N-BEA on Bi-MDVRP. The symbol "+" means that H_0 is rejected while The symbol "–" means the opposite. The best values are highlighted in bold.

	Instances	EVORL_CODBA	CODBA-VNS	CODBA	N-BEA
Suite Bi-pr	Bi-pr01	**6.01 (+++)**	6.28 (++)	6.78 (+)	299.20
	Bi-pr02	**78.8 (+++)**	119.52 (–+)	120.52 (+)	877.01
	Bi-pr03	**79.12 (+++)**	200.68 (++)	233.98 (+)	383.02
	Bi-pr04	**76.6 (+++)**	180.08 (++)	184.11 (+)	155.35
	Bi-pr05	**64.21 (+++)**	65.02 (++)	66.19 (+)	898.82
	Bi-pr06	82.6 (–++)	**76.87 (++)**	**90.29 (+)**	453.70
	Bi-pr07	**72.4 (+–+)**	163.74 (++)	77.11 (+)	99.10
	Bi-pr08	**77.4 (+++)**	159.81 (++)	170.92 (+)	211.91
	Bi-pr09	**79.7 (+++)**	160.45 (++)	155.10 (+)	459.10
	Bi-pr10	**16.01 (+++)**	18.71 (++)	169.30 (+)	618.20
Suite Bi-p	Bi-p01	**11.54 (+++)**	18.99 (++)	28.91 (+)	341.20
	Bi-p02	20.12 (+++)	23.33 (++)	**19.32 (+)**	477.01
	Bi-p03	**21.01 (+++)**	25.10 (++)	75.10 (+)	382.01
	Bi-p04	**65.52 (+++)**	72.10 (++)	76.01 (+)	392.01
	Bi-p05	**33.85 (+++)**	35.28 (++)	81.50 (+)	320.20
	Bi-p06	**41.75 (+++)**	45.11 (++)	45.38 (+)	354.01
	Bi-p07	**74.21 (+++)**	79.20 (++)	77.57 (+)	329.20
	Bi-p08	111.7 (+++)	**100.40 (++)**	185.20 (+)	356.30
	Bi-p09	**94.9 (+++)**	100.50 (++)	128.21 (+)	398.90
	Bi-p10	**61.22 (–++)**	63.72 (++)	65.45 (+)	392.32
	Bi-p11	**64.3 (+++)**	70.22 (++)	73.12 (+)	229.01
	Bi-p12	81.25 (+–+)	**78.01 (+–)**	79.58 (+)	313.03
	Bi-p13	97.21 (+++)	99.03 (++)	**92.53 (+)**	219.01
	Bi-p14	**87.14 (–++)**	88.40 (–+)	90.63 (+)	232.30
	Bi-p15	**111.84 (+++)**	151.81 (++)	126.44 (+)	242.10
	Bi-p16	**115.0 (+++)**	128.04 (+–)	130.04 (+)	256.02
	Bi-p17	**142.3 (+++)**	145.06 (++)	155.68 (+)	352.01
	Bi-p18	**93.54 (–++)**	94.80 (++)	88.10 (+)	412.01
	Bi-p19	**55.10 (+–+)**	58.92 (++)	55.32 (+)	432.01
	Bi-p20	**111.11 (+++)**	126.18 (++)	132.25 (+)	652.11
	Bi-p21	**131.52 (+++)**	139.95 (++)	119.55 (+)	629.10
	Bi-p22	**120.5 (+++)**	154.05 (++)	141.55 (+)	682.02
	Bi-p23	**295.25 (+–+)**	322.15 (++)	301.25 (+)	994.23

Notes: H_0: median (Algorithm 1) = median (Algorithm 2), H_1: median (Algorithm 1) ≠ median (Algorithm 2).

Besides, incorporating Q-learning with VNS local search procedure with CODBA scheme gives our proposal greater chances to produce good results. Our algorithm has reached the best upper fitness values in 28 out of the 33 instances. For the other seven instances, EVORL-CODBA generates the second best values regarding others competitors. The obtained results highlight the ability of Q-learning in guiding the VNS strategy to select the adequate search strategy that can enhance and improve the convergence rate of the lower-level decision task, making the search process more robust and adaptive. Our designed approach allows the search strategy to dynamically adjust its focus, prioritizing promising regions of the solution space while reducing unnecessary exploration.

To investigate now the EVORL-CODBA behavior on the lower-level, we report in Tables 4 and 5 the statistical results of the involved bi-level algorithms. EvoRL-CODBA is strictly better than CODBA-VNS, CODBA and NBEA. The superiority of EVORL-CODBA over its competitors on most test problems may be explained by

TABLE 6. RPD values of EVORL_CODBA, CODBA-VNS, CODBA and N-BEA on small, average and large Bi-MDVRP benchmarks.

Instances		EVORL_CODBA	CODBA-VNS	CODBA	N-BEA
Small	Bi-pr01	0.04	1.22	3.51	7.36
	Bi-pr02	0.15	0.61	2.61	4.52
	Bi-p03	0.09	0.98	3.66	5.02
Medium	Bi-pr04	0.26	0.78	2.10	4.02
	Bi-pr05	0.11	0.65	2.31	4.56
	Bi-p07	0.26	0.31	4.56	5.89
Large	Bi-pr10	1.08	1.93	2.41	7.32
	Bi-p20	1.02	2.03	3.65	7.45
	Bi-p23	1.37	2.19	3.75	8.65

the fact that the Q-learning information shared between threads and under generations provides potential useful information for choosing the more appropriate search neighborhood operator which participated in enhancing the lower convergence and subsequently the whole evolutionary process.

To further assess the quality of solutions and provide quantitative evidence of the impact on diversification and intensification, we report the Relative Percentage Deviation (RPD) for each algorithm on small, medium, and large instances. This metric measures the deviation of the obtained solution regarding each couple (algorithm, instance) from the best-found solution, after a number of executions (*cf.* Eq. (5))

$$\text{RPD} = \frac{Sol_{\text{algo}} - bestSol}{bestSol}. \tag{5}$$

Table 6 presents the generated results. In fact, we deduce that the proposed EVORL_CODBA outperforms the other algorithms in the 9 considered instances with an average RPD value of 0.4867. The worst performance is then observed with the N-BEA algorithm which presents an average RPD value of 6.09. By considering this metric, we aimed to quantify the added value of incorporating Q-learning in terms of accelerating the convergence of our proposal At last, we can deduce that the superior results observed in terms of average upper-level fitness, direct rationality, and weighted rationality for EVORL_CODBA can be attributed to the combination of several key mechanisms: the decomposition strategy, the multi-threaded execution inherited from CODBA, and the integration of reinforcement learning. The latter introduces a learning-based mechanism that enables the algorithm to adaptively improve lower-level solutions based on feedback, thereby promoting more coherent and effective interactions between the upper and lower levels. Thus, we can conclude from these tables that we have succeeded in designing a novel bi-level strategy, which explains the consistent outperformance of EVORL_CODBA across all evaluated metrics and instance

5. CONCLUSION

In this paper, we have merged reinforcement learning with bi-level evolutionary algorithm to handle combinatorial bi-level optimization problems. Particularly, we integrate Q-learning strategy with the decomposition-based lower level strategy of CODBA algorithm to make the search process more efficient. The statistical experimental results obtained show that our proposed EVORL-CODBA algorithm provides competitive results regarding the algorithms used on the bi-level multi-depot vehicle routing problems. These results demonstrate the potential benefits of the integrated Q-learning strategy compared to the baseline version. This work could be extended in a number of ways:

– First, we are interested in evaluating the Q-learning strategy in the upper level process to better guide the upper- and lower-level optimizations. As well, we aim to tailor reward mechanisms, exploration-exploitation structures, and even self-tuning of parameters to adapt to other problem-specific characteristics.
– Second, we aim to hybridize our proposal with other learning techniques such as deep Q-networks to handle complex, and high-dimensional bi-level problems.
– Finally, we are interested in proposing an extension of our approach to handle multi-objective bi-level problems, which would require balancing trade-offs among multiple objectives in both decision making levels.

Funding

This research received no external funding.

Conflict of Interest

The author declares that there are no conflicts of interest regarding the present study.

Data availability statement

The datasets used during the current study are available on reasonable request. Indeed, the code used in this paper is available online in a Github repository: https://github.com/AbirUser/JavaFBLOP/tree/main/approaches/QCODBA [8].

Author contribution statement

The author confirms being the sole contributor to this work and is responsible for all aspects of the research, including conceptualization, methodology, implementation, analysis, and manuscript preparation.

References

[1] M. Abbassi, A. Chaabani, L.B. Said and N. Absi, An approximation-based chemical reaction algorithm for combinatorial multi-objective bi-level optimization problems, in Proceedings of 2021 IEEE Congress on Evolutionary Computation (CEC) (2021) 1627–1634. https://doi.org/10.1109/CEC45853.2021.9504711.

[2] L.G. Acuna, D.R. Rios, C.P. Arboleda and E.G. Ponzon, Cooperation model in the electricity energy market using bi-level optimization and Shapley value. *Oper. Res. Perspect.* **5** (2018) 161–168.

[3] V. Audigier, F. Husson and J. Josse, MIMCA: multiple imputation for categorical variables with multiple correspondence analysis. *Stat. Comput.* **27** (2017) 501–518.

[4] A. Azzouz, A. Chaabani, M. Ennigrou and L.B. Said, Handling sequence-dependent setup time flexible job shop problem with learning and deterioration considerations using evolutionary bilevel optimization. *Appl. Artif. Intell.* **34** (2020) 433–455.

[5] H. Bai, R. Cheng and Y. Jin, Evolutionary reinforcement learning: a survey. *Intell. Comput.* **2** (2023) 0025. https://doi.org/10.34133/icomputing.0025.

[6] L. Breiman, Random forests. *Mach. Learn.* **45** (2001) 5–32.

[7] H.I. Calvete, C. Gale and M.-J. Oliveros, A hybrid algorithm for solving a bilevel production-distribution planning problem, in Modeling and Simulation in Engineering, Economics, and Management – MS 2013. Vol. 145 of *Lecture Notes in Business Information Processing*. Springer, Berlin (2013) 138–144.

[8] A. Chaabani, EVORL_CODBA: a new efficient evolutionary reinforcement learning algorithm for bi-level combinatorial optimization problem. GitHub repository (2025). https://github.com/AbirUser/JavaFBLOP/tree/main/approaches/QCODBA.

[9] A. Chaabani, S. Bechikh and L.B. Said, A co-evolutionary decomposition-based algorithm for bi-level combinatorial optimization, in 2015 IEEE Congress on Evolutionary Computation (CEC) (2015) 1659–1666. https://doi.org/10.1109/CEC.2015.7257084.

[10] A. Chaabani, S. Bechikh and L.B. Said, A memetic evolutionary algorithm for bi-level combinatorial optimization: a realization between Bi-MDVRP and Bi-CVRP, in 2016 IEEE Congress on Evolutionary Computation (CEC). IEEE (2016) 1666–1673. https://doi.org/10.1109/CEC.2016.7743988.

[11] A. Chaabani, S. Bechikh and L. Ben Said, A co-evolutionary hybrid decomposition-based algorithm for bi-level combinatorial optimization problems. *Soft Comput.* **24** (2020) 7211–7229.

[12] L. Cheng, Q. Tang, L. Zhang and Z. Zhang, Multi-objective Q-learning-based hyper-heuristic with bi-criteria selection for energy-aware mixed shop scheduling. *Swarm Evol. Comput.* **69** (2022) 100985.

[13] A. Cheraghalipour, M.M. Paydar and M. Hajiaghaei-Keshteli, Designing and solving a bi-level model for rice supply chain using evolutionary algorithms. *Comput. Electron. Agric.* **162** (2019) 651–668.

[14] S.S. Choong, L.P. Wong and C.P. Lim, Automatic design of hyper-heuristic based on reinforcement learning. *Inf. Sci.* **436** (2018) 89–107.

[15] C. Cortes and V. Vapnik, Support-vector networks. *Mach. Learn.* **20** (1995) 273–297.

[16] P. Cunningham and S.J. Delany, K-nearest neighbour classifiers-a tutorial. *ACM Comput. Surv.* **54** (2021) 1–25.

[17] J. Derrac, S. García, D. Molina and F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **1** (2011) 3–18.

[18] J. Gui, T. Chen, J. Zhang, Q. Cao, Z. Sun, H. Luo and D. Tao, A survey on self-supervised learning: algorithms, applications, and future trends. *IEEE Trans. Pattern Anal. Mach. Intell.* (2024). https://doi.org/10.1109/TPAMI.2024.3415112.

[19] M. Hammami, S. Bechikh, A. Louati, M. Makhlouf and L.B. Said, Feature construction as a bi-level optimization problem. *Neural Comput. Appl.* **32** (2020) 13783–13804.

[20] M. Hammami, S. Bechikh, C.C. Hung and L.B. Said, Class-dependent weighted feature selection as a bi-level optimization problem, in Neural Information Processing, ICONIP 2020, edited by H. Yang, K. Pasupa, A.C.S. Leung, J.T. Kwok, J.H. Chan and I. King. Vol. 1333 of *Communications in Computer and Information Science*. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-63823-8_32.

[21] T. Hastie, R. Tibshirani and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York (2009).

[22] Z. Hu, W. Gong and S. Li, Reinforcement learning-based differential evolution for parameters extraction of photovoltaic models. *Energy Rep.* **7** (2021) 916–928.

[23] Z. Hu, L. Wang, J. Qin, B. Lev and L. Gan, Optimization of facility location and size problem based on bi-level multi-objective programming. *Comput. Oper. Res.* **145** (2022) 105860.

[24] M. Jerbi, Z.C. Dagdia, S. Bechikh and L.B. Said, Android malware detection as a bi-level problem. *Comput. Secur.* **121** (2022) 102825.

[25] M. Käärik and K. Pärna, On the quality of k-means clustering based on grouped data. *J. Stat. Plan. Infer.* **139** (2009) 3836–3841.

[26] K. Kontolati, D. Loukrezis, D.G. Giovanis, L. Vandanapu and M.D. Shields, A survey of unsupervised learning methods for high-dimensional uncertainty quantification in black-box-type problems. *J. Comput. Phys.* **464** (2022) 111313.

[27] H.W. Kuhn and A.W. Tucker, Nonlinear programming, in Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, University of California Press (1951) 481–492.

[28] F. Legillon, A. Liefooghe and E.-G. Talbi, CoBRA: a cooperative coevolutionary algorithm for bi-level optimization, in 2012 IEEE Congress on Evolutionary Computation (CEC). IEEE (2012) 1–8. https://doi.org/10.1109/CEC.2012.6256620.

[29] K. Li, T. Zhang and R. Wang, Deep reinforcement learning for multiobjective optimization. *IEEE Trans. Cybern.* **51** (2020) 3103–3114.

[30] T. Li, Y. Meng and L. Tang, Scheduling of continuous annealing with a multi-objective differential evolution algorithm based on deep reinforcement learning. *IEEE Trans. Autom. Sci. Eng.* **21** (2023) 1767–1780.

[31] Y. Li, C. Liao, L. Wang, Y. Xiao, Y. Cao and S. Guo, A reinforcement learning-artificial bee colony algorithm for flexible job-shop scheduling problem with lot streaming. *Appl. Soft Comput.* **146** (2023) 110658.

[32] Y. Lin, F. Lin, G. Cai, H. Chen, L. Zou and P. Wu, Evolutionary reinforcement learning: a systematic review and future directions. Preprint arXiv:2402.13296 (2024).

[33] Z.J. Liu and H.N. Wu, Driver behavior modeling via inverse reinforcement learning based on particle swarm optimization, in Proceedings of the 2020 Chinese Automation Congress (CAC). IEEE (2020) 7232–7237.

[34] H. Louati, S. Bechikh, A. Louati, C.C. Hung and L.B. Said, Deep convolutional neural network architecture design as a bilevel optimization problem. *Neurocomputing* **439** (2021) 44–62.

[35] Y. Marinakis, A. Migdalas and P. Pardalos, A new bilevel formulation for the vehicle routing problem and a solution method using a genetic algorithm. *J. Glob. Optim.* **38** (2007) 555–580.

[36] N. Mazyavkina, S. Sviridov, S. Ivanov and E. Burnæev, Reinforcement learning for combinatorial optimization: a survey. *Comput. Oper. Res.* **134** (2021) 105400.

[37] J.A. Mejia-De-Dios, A. Rodriguez-Molina and E. Mezura-Montes, Multi-objective bilevel optimization: a survey of the state-of-the-art. *IEEE Trans. Syst. Man Cybern. Syst.* **53** (2023) 5478–5490.
[38] A. Moreira, M.Y. Santos and S. Carneiro, Density-Based Clustering Algorithms-DBSCAN and SNN. University of Minho, Portugal (2005) 1–18.
[39] L. Peng, Z. Yuan, G. Dai, M. Wang and Z. Tang, Reinforcement learning-based hybrid differential evolution for global optimization of interplanetary trajectory design. *Swarm Evol. Comput.* **81** (2023) 101351.
[40] I. Rish, An empirical study of the naive Bayes classifier, in Proceedings of IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence. Vol. 3 (2001) 41–46.
[41] P.J. Ross, Taguchi Techniques for Quality Engineering: Loss Function, Orthogonal Experiments, Parameter and Tolerance Design. McGraw-Hill, New York (1988).
[42] I.H. Sarker, Machine learning: algorithms, real-world applications and research directions. *SN Comput. Sci.* **2** (2021) 160.
[43] G. Savard and J. Gauvin, The steepest descent direction for the nonlinear bilevel programming problem. *Oper. Res. Lett.* **15** (1994) 265–272.
[44] K. Sindhu Meena and S. Suriya, A survey on supervised and unsupervised learning techniques, in Proceedings of International Conference on Artificial Intelligence, Smart Grid and Smart City Applications (AISGSC 2019), edited by L. Kumar, L. Jayashree and R. Manimegalai. Springer, Cham (2020) 613–623. https://doi.org/10.1007/978-3-030-24051-6_58.
[45] A. Sinha, P. Malo and K. Deb, Efficient evolutionary algorithm for single-objective bilevel optimization. Preprint arXiv:1303.3901 (2013).
[46] A. Sinha, P. Malo and K. Deb, A review on bilevel optimization: from classical to evolutionary approaches and applications. *IEEE Trans. Evol. Comput.* **22** (2018) 276–295.
[47] Y. Song, L. Wei, Q. Yang, J. Wu, L. Xing and Y. Chen, RL-GA: a reinforcement learning-based genetic algorithm for electromagnetic detection satellite scheduling problem. *Swarm Evol. Comput.* **77** (2023) 101236.
[48] Y. Song, Y. Wu, Y. Guo, R. Yan, P.N. Suganthan, Y. Zhang and Q. Feng, Reinforcement learning-assisted evolutionary algorithm: a survey and research opportunities. *Swarm Evol. Comput.* **86** (2024) 101517.
[49] J. Sun, X. Liu, T. Bäck and Z. Xu, Learning adaptive differential evolution algorithm from optimization experiences by policy gradient. *IEEE Trans. Evol. Comput.* **25** (2021) 666–680.
[50] H. Xia, C. Li, S. Zeng, Q. Tan, J. Wang and S. Yang, A reinforcement-learning-based evolutionary algorithm using solution space clustering for multimodal optimization problems, in 2021 IEEE Congress on Evolutionary Computation (CEC). IEEE (2021) 1938–1945. https://doi.org/10.1109/CEC45853.2021.9504896.
[51] N. Yorino, M. Abdillah, Y. Sasaki and Y. Zoka, Robust power system security assessment under uncertainties using bi-level optimization. *IEEE Trans. Power Syst.* **33** (2017) 352–362.
[52] X. Zhang, S. Xia, X. Li and T. Zhang, Multi-objective particle swarm optimization with multi-mode collaboration based on reinforcement learning for path planning of unmanned air vehicles. *Knowl. Based Syst.* **250** (2022) 109075.
[53] D. Zhang, Y. Chen and G. Zhu, Multi-objective hole-making sequence optimization by genetic algorithm based on Q-learning. *IEEE Trans. Emerg. Top. Comput. Intell.* **8** (2024) 1–14.
[54] F. Zhao, S. Di and L. Wang, A hyperheuristic with Q-learning for the multiobjective energy-efficient distributed blocking flow shop scheduling problem. *IEEE Trans. Cybern.* **53** (2023) 3337–3350.
[55] B. Zhou and Z. Zhao, An adaptive artificial bee colony algorithm enhanced by deep Q-learning for milk-run vehicle scheduling problem based on supply hub. *Knowl. Based Syst.* **264** (2023) 110367.