Bol. Soc. Paran. Mat.
©SPM – E-ISSN-2175-1188
SPM: www.spm.uem.br/bspm

(3s.) **v. 2026 (44) 18** : 1–16.
ISSN-0037-8712
doi:10.5269/bspm.82819

Hybrid Genetic Algorithm with Deep Q-Learning Mutation for Enhanced Vehicle Routing Optimization

Manar Fahim and Halima Lakhbab

Abstract: The Capacitated Vehicle Routing Problem (CVRP) is one of the most researched NP-hard problems, dealing with minimizing total costs while respecting vehicle capacity constraints. Genetic Algorithms (GA) provide flexibility but rely on random mutations, which may slow convergence and produce suboptimal solutions. In this work, we propose a hybrid GA–DQN approach where a pre-trained Deep Q-Network (DQN) guides mutation decisions to improve solution quality. This hybridization provides a better balance between exploration and exploitation compared with a standard GA. Experimental results show that the proposed method converges faster and achieves better solutions across several CVRP instances, demonstrating that reinforcement learning can effectively enhance evolutionary optimization methods.

Keywords: Vehicle Routing Problem (VRP), Capacitated VRP, Genetic Algorithm (GA), Reinforcement Learning (RL), Deep Q-Network (DQN), hybrid optimization.

Contents

1. Introduction

The Capacitated Vehicle Routing Problem (CVRP) is a fundamental combinatorial optimization problem in modern logistics and transportation systems. First introduced by Dantzig and Ramser in 1959 [4], the CVRP seeks to determine the minimum-cost set of routes to serve a set of customers while respecting vehicle capacity constraints. With the rapid growth of e-commerce and the expansion of urban distribution networks, CVRP has become increasingly significant, posing major cost and operational challenges for logistics companies [2].

The main difficulty of the CVRP is that routing decisions, vehicle utilization, and multiple operational constraints must be jointly optimized. In addition, there are additional external and practical factors, such as the distribution of customers,traffic regulations and the actual limitations of the logistics industry, adding to the complexity of the problem. This means the CVRP has been categorized as an NP-hard problem [15], and therefore is very computationally difficult to solve and has a great deal of real-world relevance.

Since its inception, there have been many approaches proposed to solve the CVRP. These approaches include exact methods and heuristic and metaheuristic approaches. The use of Genetic Algorithms (GAs)

2020 *Mathematics Subject Classification:* 90C27, 68T20, 68T05.
Submitted March 19, 2026. Published June 19, 2026.

has been widespread because they are highly adaptable and robust as well as an effective way to search through the vast and complex solution space [13]. However, the performance of GA is highly dependent on the design of variation operators, particularly the mutation operator, which is typically applied in a random manner. This unguided mutation can lead to slow convergence and reduced solution quality, especially for large-scale problem instances.

More recently, Deep Reinforcement Learning (DRL) has emerged as a promising alternative for solving routing problems, as it can learn adaptive decision-making policies directly from data. Despite its potential, DRL-based methods often require large neural networks and substantial training time, which can limit their practical applicability for large CVRP instances. These limitations motivate hybrid approaches that combine the exploration strengths of GA with the adaptive decision-making capability of DRL.

In this work, we propose a hybrid GA–DQN approach in which a DQN guides the mutation operator within the GA. The pre-trained DQN evaluates which mutations are likely to improve the current solution, allowing the GA to maintain its population diversity while benefiting from intelligent, learning-driven guidance. Experimental results demonstrate that this hybrid approach improves solution quality and accelerates convergence compared to conventional GA, while maintaining strong generalization across different CVRP instances [11].

The remainder of this paper is organized as follows: Section 2 reviews the state-of-the-art of GA and RL-based methods for CVRP; Section 3 introduces the mathematical modeling of the problem; Section Section 4 details the proposed GA–DQN methodology; Section Section 5 analyzes the experimental results; and Section Section 6 concludes and discusses future research directions.

2. Related work

The Vehicle Routing Problem (VRP) is a classical combinatorial optimization problem and is known to be NP-hard, which implies that finding optimal solutions becomes computationally prohibitive as the problem size increases [21]. Existing approaches for solving the VRP can generally be divided into two main categories: traditional optimization algorithms and learning-based methods.

Traditional approaches mainly include exact algorithms, heuristic algorithms, and meta-heuristic algorithms. Exact methods are relatively easy to formulate and implement, and they are effective for small-scale VRP instances with simple structures.However, Their system performance tends to degrade quickly when solving larger problem instances, particularly with the presence of constraints, which makes them inappropriate to apply to real problems.To overcome these challenges, there is the use of limitations, a number of heuristic algorithms have been proposed and are able to producing high quality solutions within a time that is reasonable from a computational perspective.

In comparison to classical heuristics, metaheuristic algorithms search the solution space reach, More extensively and generally, achieve better quality solutions and remain flexible sufficient to solve various variants of VRP [10]. Common meta-heuristic approaches include tabu search [3], simulated annealing [19], genetic algorithms [16], ant colony optimization [14], and particle swarm optimization [1]. Despite their strong performance, these methods may suffer from premature convergence and can become trapped in local optima, especially for large-scale or highly constrained VRP instances.

In recent years, advances in artificial intelligence have enabled deep reinforcement learning (DRL) to achieve remarkable success in solving sequential decision-making problems. DRL allows an agent to learn optimal strategies by interacting with the environment and updating its policy based on feedback. Although neural networks had previously been explored for combinatorial optimization, the introduction of pointer networks [17] marked the first successful application of deep learning to such problems.

Building upon pointer networks, Nazari et al. [12] proposed a reinforcement learning framework to solve the Capacitated Vehicle Routing Problem (CVRP) using policy gradient methods. Their model is capable of generating high-quality solutions in a short time and generalizing to unseen instances without retraining. Kwon et al. [9] proposed an end-to-end reinforcement learning algorithm that utilized solution symmetry to explore different promising solutions concurrently to approach the optimal solution for the TSP and CVRP efficiently.

Fu et al. [5] examined how supervised learning models can generalize to the Traveling Salesman Problem (TSP), demonstrating that data-driven approaches are capable of learning and exploiting mean-

ingful structural properties of the problem beyond the training set. Yang et al. [18] combined DRL with adaptive large neighborhood search and incorporated multi-head attention and graph neural networks to enhance solution quality. Yu et al. [20] proposed incorporating multi-head attention mechanisms together with local contextual information in order to better model the complex spatial and temporal.

Despite these advances, DRL-based techniques have made good progress, they typically require lengthy training and do not produce the same quality of results as leading heuristic solvers such as LKH3 [6] As a result, there has been an increase in combining and using evolutionary algorithm with DRL to solve VRP problems. Khadka and Tumer [7] proposed the Evolutionary Reinforcement Learning (ERL) framework, combining evolutionary search capabilities with deep reinforcement learning, representing a significant step forward toward an integration of the two fields. Nevertheless, hybrid VRP frameworks still present an opportunity for more research.

More recently, Zhang et al. [21] proposed a hybrid framework combining deep reinforcement learning with evolutionary algorithm-guided imitation for the CVRP. Evolutionary algorithms are used to generate expert solutions that guide the training of a DRL agent through imitation learning, resulting in improved solution quality compared to existing DRL-based methods. However, the reliance on evolutionary demonstrations increases computational cost, especially for large-scale instances.

3. Background

3.1. Problem description and formulation

The Vehicle Routing Problem (VRP) was initially presented in 1959 by Dantzig and Ramser [4], aiming to optimize fuel delivery to service stations. As an extension of the Traveling Salesman Problem (TSP) [8] , the VRP rapidly gained prominence in logistics, merging cost reduction with fulfilling multiple operational restrictions like vehicle capacity. Owing to its NP-hard nature, early exact algorithms were found to be inefficient for tackling large-scale problems. As a result, beginning in the 1980s these methods were enhanced with heuristic and metaheuristic strategies, such as genetic algorithms, tabu search, and ant colony optimization.

VRP refers to a category of problems where a group of customers needs to be served by a fleet of vehicles, each assigned a distinct route. These routes should be planned to optimize one or more goals, like minimizing the overall distance covered or lowering customer wait times. There are variations of the VRP, which introduce extra constraints such as vehicle load limits, time constraints, technical restrictions, or spatial boundaries.

In its basic formulation, the VRP can be defined as follows: given a graph where nodes represent customers and edges represent travel routes between them, the objective is to design feasible routes for a fleet of available vehicles such that all constraints are satisfied while optimizing a predefined objective function.

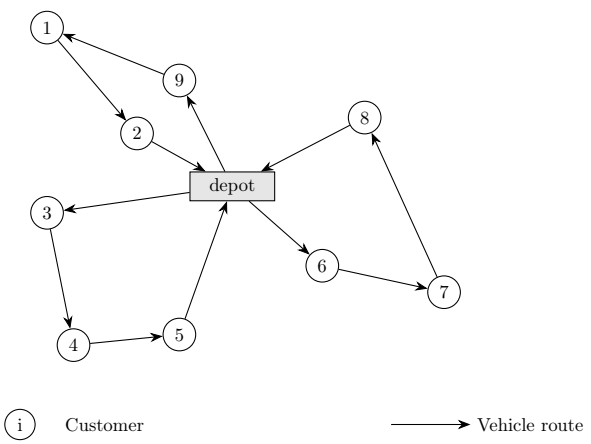


Figure 1: Vehicle Routing Problem

VRP asks how to assign each customer to a route, with one vehicle handling each tour. The task is to figure out the order in which each vehicle visits its assigned customers, while making sure every vehicle stays within its capacity and meets customer demand. The main goal: find routes that keep the total distance as short as possible, using the fewest vehicles. Every route starts and ends at a central depot.

The classical Vehicle Routing Problem with capacity constraints (CVRP) can be modeled as a weighted and directed graph $G = (N, A)$, where:
N denotes the set of nodes, including both customers and depot locations,
A represents the set of directed arcs connecting pairs of nodes $i, j \in N$.

More precisely, let $C = \{1, \ldots, n_c\}$ be the set of customers that require deliveries from the depot. The complete set of nodes is given by $N = C \cup \{0, n_c + 1\}$, where 0 and $n_c + 1$ correspond to the departure and return depot, respectively. Each customer $i \in C$ has an associated demand $d_i > 0$.

A fleet of n_v vehicles, denoted by $V = \{1, \ldots, n_v\}$, is stationed at the depot. All vehicles are assumed to have the same capacity Q (homogeneous fleet), which must satisfy:

$$Q \geq \max_{i \in C} d_i$$

For each pair of nodes $i, j \in N$, the travel cost $c_{i,j}$ is known and generally proportional to the distance between them. To optimize the routing, decision variables are defined to determine the sequence in which customers are visited by each vehicle.

$$x_{i,j}^v = \begin{cases} 1 & \text{if vehicle } v \in V \text{ visits node } j \text{ immediately after node } i, \\ 0 & \text{otherwise.} \end{cases}$$

Let y_i represent the remaining capacity of a vehicle after completing service at customer $i \in C$. Under this notation, the Vehicle Routing Problem can be expressed as the following optimization model:

$$\min \sum_{v \in V} \sum_{i \in N} \sum_{j \in N} c_{i,j} x_{i,j}^v \tag{1.1}$$

Constraints:

$$\sum_{v \in V} \sum_{j \in N} x_{i,j}^v = 1 \qquad \forall i \in C \tag{1.2}$$

$$\sum_{j \in N} x_{i,j}^v - \sum_{j \in N} x_{j,i}^v = 0 \qquad \forall i \in C, \forall v \in V \tag{1.3}$$

$$\sum_{j \in N} x_{0,j}^v = 1 \qquad \forall v \in V \tag{1.4}$$

$$\sum_{j \in N} x_{j,n+1}^v = 1 \qquad \forall v \in V \tag{1.4}$$

$$\sum_{i \in C} d_i \cdot \sum_{j \in N} x_{i,j}^v \leq Q \qquad \forall v \in V \tag{1.5}$$

$$u_i - u_j + Q \cdot x_{i,j} \leq Q - d_j \qquad \forall i \neq 0, \forall j \neq 0, i \neq j \tag{1.6}$$

$$d_i \leq u_i \leq Q \qquad \forall i \in C \tag{1.6}$$

$$x_{i,j}^v \in \{0, 1\} \qquad \forall i, j \in N, i \neq j, \forall v \in V \tag{1.7}$$

The **cost function** of a solution $X = \left(x_{i,j}^v\right)$, $\forall i, j \in N, \forall v \in V$, is defined as:

$$\text{cost}(X) = \sum_{v \in V} \sum_{i \in N} \sum_{j \in N} c_{i,j} x_{i,j}^v \tag{1.8}$$

The number of vehicles used by solution X is defined as:

$$\text{Nb\_vehicles}(X) = \sum_{v \in V} \sum_{j \in C} x_{0,j}^{v} \tag{1.9}$$

Explanation of the Constraints :

The objective function **(1.1)** aims to minimize the total number of vehicles employed across the n_v routes, as well as the cumulative travel cost.
The problem formulation is subject to several constraints:
Constraint(1.2) Eguarantees that each customer will be served by a single vehicle.
Constraint(1.3) States that the arrival vehicle is the same one which departs from that customer.
Constraint(1.4) Enforces that each vehicle leaves the depot only once.
Constraint(1.5) Ensures a unique return to the depot for each vehicle or route.
Constraint(1.6) through **(1.7)** Impose vehicle capacity and integrality constraints.
Constraint(1.8) and **(1.9)** Serve as evaluation measures, respectively quantifying the solution in terms of total distance traveled and the number of vehicles utilized.

Numerous variations of the vehicle routing problem (VRP) have been devised to make it more realistic. The classic Capacitated VRP (CVRP) keeps things simple: each vehicle has a limited load, you start and finish at a central depot, and every customer gets one visit just try to keep the total travel cost down.

3.2. Genetic Algorithm

- **Encoding**: A solution is encoded as a permutation of integers; 0 represents the depot. The clients are numbered from 1 to n. A route is associated with one block of clients between two zeros and starting and ending at the depot, respectively. The number of vehicles is not explicitly coded, the ordering of the blocks implicitly defines the routes. The set of blocks corresponds to a feasible solution, in which the vehicle capacity constraint is respected. This coding is easy to handle, it has a small size and can be handled by standard genetic operators.

- **Initial Population Generation**: In the CVRP, each individual represents a feasible solution composed of routes that respect capacity constraints and start and end at the depot. The initial population can be created randomly or using classical heuristics to ensure diversity. Diversity enables effective exploration of the solution space and avoids premature convergence.

- **Evaluation (Fitness)**: The fitness function is a minimization function since the goal is for all vehicles to travel the shortest distance possible without violating capacity constraints. It is obtained by summing all the distances of the routes. Solutions violating capacity constraints are penalized. The fitness function allows comparison of solutions to favor feasible, low-cost options.

- **Selection**: Selection chooses the most promising solutions for the next generation. Common methods include:

  - **Rank Selection**: Solutions are ranked by fitness and selected based on position.
  - **Roulette Wheel Selection**: Each solution occupies a portion of a "roulette wheel" proportional to its fitness.
  - **Random Selection**: Solutions are selected uniformly at random.
  - **Tournament Selection**: A random subset of solutions is drawn, and the best is selected.

- **Crossover**: Crossover creates new solutions by combining two parent solutions. Methods include:

  - **Partially Mapped Crossover (PMX)**: Exchanges a segment of clients between two parents, filling the rest respecting positions.

- **Order Crossover (OX)**: Copies a segment from one parent and fills the rest according to the other parent's order.
- **Single-Point Crossover**: Cuts at the same random point of each parent and swaps segments beyond that point.

- **Mutation**: Mutation maintains diversity by altering solutions slightly. Techniques include:
  - **Swap**: Exchanges two randomly selected clients in a route.
  - **2-opt (Inversion)**: Reverses the order of clients between two positions.
  - **Insertion**: Moves a client to another position, possibly in another route.
- **Selection of the Best Solution**: Once the stopping criterion is reached, the best solution is selected. It corresponds to the routes with the lowest total cost while respecting CVRP constraints.
- **Stopping Criterion**: Defines when the algorithm stops. For CVRP, it can be a maximum number of generations, a time limit, or no improvement of the best solution over several generations.

Algorithm 1 Genetic Algorithm for CVRP

```
Initialize the population POP with feasible CVRP solutions
Evaluate the population POP by computing the total tour cost
Initialize POP' as empty
Find x such that f(x) = min_i[f(x_i)], 1 ≤ i ≤ N
f_min ← f(x)
x_min ← x
repeat
    Evaluate the population POP
    repeat                                          ▷ Genetic reproduction phase
        Select individuals
        Apply crossover respecting vehicle capacity
        Apply mutation (swap, insertion, 2-opt)
        Repair infeasible solutions if necessary
    until POP' is filled with new individuals
    Select new population from (POP, POP')
    Find x such that f(x) = min_i[f(x_i)]
    if f(x) < f_min then
        f_min ← f(x)
        x_min ← x
until stopping criteria are met
Output: best solution (x_min, f_min)
```

3.3. Deep Q-Network (DQN)

- **Overview**: The *Deep Q-Network* (DQN) is a deep reinforcement learning algorithm that combines classical Q-Learning with deep neural networks. It allows an agent to make decisions in complex environments where the state space is too large to be represented explicitly. In DQN, the action-value function Q is approximated by a neural network, which estimates the quality of each possible action in a given state. The agent chooses actions to maximize the expected cumulative reward. To ensure stable learning, DQN relies on two key mechanisms: *experience replay*, which reuses past interactions, and a *target network*, which provides more stable reference values during training.
- **State (s)**: The state represents the current situation of the agent (vehicle). It may include:

- The current position of the vehicle

- The customers already visited

- The remaining load in the vehicle

- The distance matrix

- The vector of remaining customer demands

- **Action** (a): An action corresponds to the choice of the next customer to visit (or returning to the depot). Actions are selected among the customers not yet served while respecting the vehicle's capacity constraint.

- **Reward** (r): The reward is a feedback signal provided after each action to guide the learning process. In this work:

  - A negative reward proportional to the distance traveled encourages route minimization.

  - Strongly negative rewards penalize constraint violations, such as vehicle overload or infeasible actions.

  - A positive reward is given at the end of an episode when all customers are successfully served with minimized total routing cost.

- **Policy** (π): The policy is modeled by the Deep Q-Network and aims to approximate the action-value function $Q(s, a; \theta)$. For a given state s, the selected action is:

$$\pi(s) = \arg\max_{a \in \mathcal{A}} Q(s, a; \theta),$$

  where $\mathcal{A}$ denotes the set of possible actions. An ϵ-greedy strategy is adopted to ensure efficient exploration of the action space.

- **DQN Update**: The Deep Q-Network is trained using the Q-learning update rule. After executing action a_t in state s_t, receiving reward r_t, and observing the next state s_{t+1}, the update is:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}, a') - Q(s_t, a_t) \right],$$

  where α is the learning rate and γ is the discount factor. This enables the agent to progressively improve its routing decisions by learning from past experiences.

Algorithm 2 DQN for CVRP

Initialize replay buffer R with capacity N
Initialize Q-network with random weights θ
Initialize target network $\hat{Q}$ with $\theta^- = \theta$
for episode $e = 1..M$ **do**
  Get initial state s_1
  for step $t = 1..T$ **do**
    Choose a_t using ϵ-greedy strategy ▷ interaction with environment
    Execute a_t and observe s_{t+1} and r_t
    Add transition (s_t, a_t, r_t, s_{t+1}) to R ▷ sampling phase
    $s_t \leftarrow s_{t+1}$
    if t%LEARNSTEP == 0 **then**
      Sample a mini-batch of size d from R
      for each experience (s_j, a_j, r_j, s_{j+1}) **do**
        Compute prediction $\hat{y}_j = Q_\theta(s_j, a_j)$
        Compute target:

$$y_j = \begin{cases} r_j & \text{if } s_{j+1} \text{ is terminal} \\ r_j + \gamma \max_{a'} \hat{Q}_{\theta^-}(s_{j+1}, a') & \text{otherwise} \end{cases}$$

        Update θ: perform a gradient descent step to minimize $(y_j - \hat{y}_j)^2$ ▷ training phase / train_step
      Update target network $\hat{Q}$ every C episodes

4. The proposed method

4.1. GA-DQN

Although classical Genetic Algorithms (GA) can be effectively employed to solve the Capacitated Vehicle Routing Problem (CVRP), their performance is often constrained by the random nature of the mutation operator. Such randomness may lead to slow convergence and suboptimal solutions, particularly for large-scale instances. In contrast, Deep Reinforcement Learning approaches, such as Deep Q-Networks (DQN), are capable of learning effective routing decisions but usually require high training costs and suffer from limited global exploration when used independently. To overcome these limitations, this work proposes a hybrid GA–DQN approach.

In the proposed framework, the genetic algorithm ensures global and population-based exploration of the solution space, while the DQN is integrated into the mutation phase to provide intelligent, learning-based local improvements. The objective of this hybrid approach is to combine the global search capability of the GA with the learning ability of the DQN in order to enhance the quality of the obtained solutions.

Within this framework, the GA acts as the main optimization engine and controls the evolution of a population of candidate solutions through standard operators such as selection, crossover, and replacement. The DQN is incorporated as a decision-making aid that enhances specific stages of the evolutionary process, particularly those influencing local solution refinement. By combining these two methods, the proposed hybridization reduces the randomness inherent in classical evolutionary operators and promotes faster convergence toward high-quality solutions.

Process The subsection below deals with this in detail. Initial population at each generation of CVRP solutions is assessed by a Fitness function based on the total cost of routings and capacity constraints compliance. The best performing solutions are selected and combined through the crossover operator. To generate new offspring solutions. After this step, each offspring is subjected to a DQN-guided mutation phase, which replaces the traditional random mutation mechanism.

Specifically, a portion of the solution is transformed into a sequential decision-making environment, in which the DQN agent selects the customers to visit according to the current state of the vehicle,

including its position, remaining capacity, and the set of unserved customers. The decisions produced by the agent are used to construct an optimized route segment, which is then reintegrated into the global solution.

The interactions generated during this phase are exploited for the progressive training of the DQN, allowing a continuous improvement of the decision policy across generations. Finally, a replacement strategy is applied to form the population of the next generation, and the process is repeated until the stopping criterion is satisfied.

To enforce feasibility, strong penalty rewards are applied whenever capacity constraints are violated or infeasible actions are selected, such as visiting an already served customer. In addition, a positive terminal reward is granted when all customers are successfully served, reinforcing the construction of complete and optimized routes.

To balance exploration and exploitation during training, an ϵ-greedy strategy is used for action selection. At each time step t, the selected action a_t is defined as:

$$a_t = \begin{cases} \text{random action}, & \text{with probability } \epsilon, \\ \arg\max\limits_{a \in \mathcal{A}} Q(s_t, a; \theta), & \text{with probability } 1 - \epsilon, \end{cases} \tag{4.1}$$

where $\mathcal{A}$ denotes the set of admissible actions. Initially, a high value of ϵ is adopted to encourage exploration. As training progresses, ϵ is gradually decreased, allowing the agent to exploit the knowledge learned from previous experiences.

The Deep Q-Network aims to approximate the optimal action-value function $Q^*(s, a)$ using a neural network with parameters by θ. Given a state s_t, the network produces an output of estimate of the Q-values for every possible actions:

$$Q(s_t, a; \theta). \tag{4.2}$$

Next, the agent performs an action a_t, then observes the reward signal r_t and the next state s_{t+1}. The value of the target used for training the network is calculated as:

$$y_t = r_t + \gamma \max_{a' \in \mathcal{A}} Q_{\text{target}}(s_{t+1}, a'; \theta^-), \tag{4.3}$$

where γ represents the discount factor and θ^- represents the parameters of the target network. The parameters of the networks θ can be adjusted using the following loss function:

$$\mathcal{L}(\theta) = \left(y_t - Q(s_t, a_t; \theta)\right)^2. \tag{4.4}$$

The DQN, through the process of learning, enhances its policy through iterations of to make more efficient routing choices and lower total transportation costs.

4.2. Experimental setup

A number of experiments were performed to thoroughly evaluate the efficiency of the proposed method in terms of solution quality and convergence performance. Analysis of the results obtained is performed on the basis of various statistical measurements such as minimum, maximum, average, median and standard deviation cost. These criteria provide a depth insight to analyze the global performance as well as stability of the algorithm. Experiments are conducted on a workstation with an HP system featuring an Intel Core i5 vPro processor, a dedicated GPU, 256 GB of RAM, and an Intel SSD. All algorithms were implemented based on Python in the programming language.

- **Data Settings and Hyper-parameters:**

These hyperparameters are named in Table 1 have been determined through extensive experimentation in the pararadios project:

Table 1: Values of hyper-parameters

Parameter	**Value**
Genetic Algorithm parameters	
Population size	50
Number of generations	100
Selection method	Tournament selection ($k = 3$)
Crossover rate	0.9
Mutation rate	0.2
Top-k mutations selected	8
Number of independent runs	30
Deep Q-Network (DQN) parameters	
Network architecture	Multilayer Perceptron (MLP)
Hidden layer size	256 neurons
Activation function	ReLU
Learning rate	1×10^{-3}
Discount factor (γ)	0.99
Mini-batch size	256 (pre-training), 64 (online use)
Exploration strategy	ε-greedy
Pre-training episodes	1200

Both GA and GA-DQN were evaluated under the same computational budget (number of generations and population size) to ensure a fair comparison.

Most of the experimental data is taken from CVRPLIB, with some extra test instances from ORTEC and famous academic benchmarks. These instances cover a wide range of problem sizes, from small cases with around fifteen customers to large-scale scenarios involving more than a thousand customers. This array is a variety instructure, which gives us the means to test how much actually rigid and flexible our method is. As many of these problems (in particular the larger ones) are too sophisticated to be solved exactly in a reasonable amount of time, they offer sound motivation for employing metaheuristics and learning-based hybrids.

Having such a wide mix of sizes, customer layouts, and vehicle capacities is crucial; it lets us verify if the policies our algorithm learns can still perform well under different constraints. For instances where the true optimal value isn't available, we use the best-known solutions as our benchmark for comparison. You can find the specific details and characteristics of these selected cases summarized in Table 3.

5. Experimental results and discussion

In this section, we compare the standard Genetic Algorithm (GA) with the proposed hybrid GA-DQN approach on the CVRP. The evaluation focuses on solution quality, convergence behavior, and the structural properties of the generated routes. Particular attention is given to the contribution of the Deep Q-Network in guiding the mutation operator. Finally, route visualizations are provided to qualitatively assess the differences between the two approaches.

The data in Table 2 indicate that the proposed GA-DQN achieves a strong balance between solution quality and search efficiency. While the standard GA remains competitive on smaller instances, the hybrid approach consistently reaches or closely approximates the Best Known Solutions. The advantage becomes more pronounced on larger instances, such as X-n101-k25 and the tai benchmarks. In these cases, it significantly outperforms the basic GA, proving that it is much better suited for high-dimensional search spaces where traditional methods often get lost.

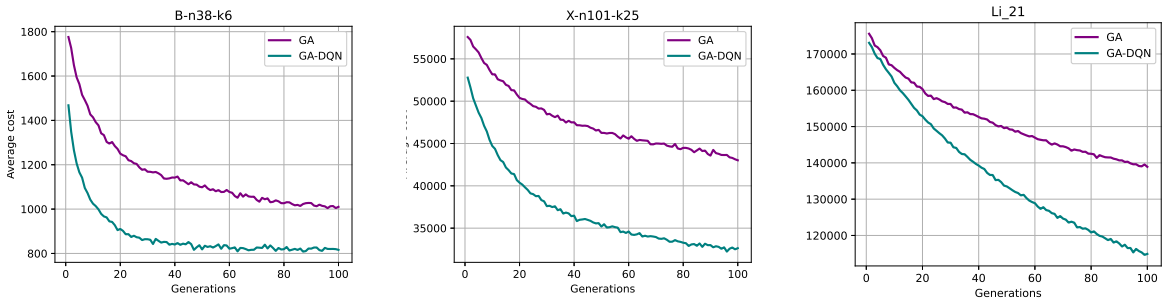
In addition, the integration of the Deep Q-Network allows a more informed mutation process, which contributes to faster convergence in the early stages of the search. This behavior can be observed in the convergence curves, where the GA-DQN shows a steeper reduction in solution cost compared to the standard GA. This suggests that the learning component helps guide the search toward promising regions while reducing the likelihood of premature convergence to local optima. Overall, the hybrid approach provides a more stable optimization process across different instance types.

The proposed approach can also be interpreted as an adaptive mutation strategy guided by reinforcement learning

Table 2: Comparing GA and GA-DQN results across different CVRPLIB.

Instance	Method	MIN	Average	MAX	Median	Std Deviation	Best Known
A-n32-k5	AG	959.02	1072.19	1215.03	1113.21	69.34	784
	AG-DQN	**785.45**	**1007.03**	**1228.61**	**1092.82**	**90.79**	784
A-n65-k9	AG	1942.80	2181.29	2411.35	2210.09	109.19	1174
	AG-DQN	**1274.52**	**1375.76**	**1446.55**	**1382.91**	**41.93**	1174
A-n80-k10	AG	2898.51	3091.32	3327.13	3103.18	89.07	1763
	AG-DQN	**2079.02**	**2174.62**	**2228.61**	**2168.41**	**49.01**	1763
B-n38-k6	AG	941.01	1035.3	1105.21	995.24	61.00	805
	AG-DQN	**810.13**	**820.31**	**832.04**	**818.08**	**13.97**	805
B-n66-k9	AG	1906.65	2098.19	2229.70	2103.09	76.91	1316
	AG-DQN	**1400.07**	**1432.06**	**1491.96**	**1429.71**	**19.53**	1316
B-n78-k10	AG	2085.51	2377.19	2675.39	2427.31	139.88	1221
	AG-DQN	**1410.07**	**1463.69**	**1558.28**	**1459.06**	**36.08**	1221
E-n22-k4	AG	435.72	509.12	582.52	484.40	36.70	375
	AG-DQN	**381.69**	**471.55**	**561.42**	**484.58**	**44.93**	375
E-n76-k10	AG	1449.04	1562.49	1658.26	1597.38	55.66	830
	AG-DQN	**1001.52**	**1045.14**	**1112.25**	**1042.24**	**29.17**	830
E-n101-k8	AG	1805.34	2039.59	2295.88	2071.95	105.01	815
	AG-DQN	**1111.60**	**1202.46**	**1267.04**	**1214.24**	**45.49**	815
F-n45-k4	AG	941.76	1141.30	1290.08	1171.21	98.20	724
	AG-DQN	**730.07**	**749.18**	**781.96**	**744.70**	**15.39**	724
F-n72-k4	AG	515.03	622.67	707.08	672.76	45.15	237
	AG-DQN	**281.41**	**303.65**	**332.55**	**304.77**	**12.85**	237
F-n135-k7	AG	3262.84	3597.98	4015.07	3617.08	181.66	1162
	AG-DQN	**1667.30**	**1901.84**	**2049.81**	**1889.46**	**75.49**	1162
P-n16-k8	AG	454.40	468.09	481.72	460.68	6.83	450
	AG-DQN	**450.34**	**452.36**	**454.40**	**451.95**	**1.02**	450
P-n76-k4	AG	1230.13	1357.41	1529.92	1373.37	56.34	627
	AG-DQN	**774.26**	**825.84**	**884.54**	**845.69**	**25.98**	627
P-n101-k4	AG	1696.26	1891.74	2092.23	1911.45	94.05	681
	AG-DQN	**919.56**	**978.67**	**1052.89**	**977.09**	**40.55**	681
CMT10	AG	4452.01	4691.10	5009.18	4731.02	124.81	1395.85
	AG-DQN	**2749.75**	**2926.15**	**3120.43**	**2901.31**	**90.58**	1395.85
CMT12	AG	1893.72	2850.12	3216.47	2655.10	330.69	819.56
	AG-DQN	**1105.38**	**1855.46**	**2605.53**	**2120.43**	**375.04**	819.56
CMT13	AG	2980.14	3546.37	3962.26	3556.37	229.41	1541.14
	AG-DQN	**1633.54**	**1747.07**	**1846.06**	**1758.33**	**63.71**	1541.14
M-n101-k10	AG	2171.77	2375.14	2564.25	2305.43	100.69	820
	AG-DQN	**1086.52**	**1201.48**	**1297.89**	**1207.91**	**44.46**	820
M-n151-k12	AG	3103.02	3328.90	3665.28	3391.08	124.58	1015
	AG-DQN	**1815.61**	**1982.74**	**2134.79**	**1985.91**	**98.18**	1015
M-n200-k16	AG	4181.26	4605.14	4951.42	4672.21	167.22	1275
	AG-DQN	**2777.55**	**2973.27**	**3093.85**	**2966.13**	**68.48**	1275
X-n101-k25	AG	41800.01	43427.96	45800	43857.89	1214.82	27591
	AG-DQN	**32065.84**	**32450.41**	**33109.23**	**32514.01**	**424.41**	27591
Li_21	AG	133983.62	138105.16	145214.82	138985.06	2694.55	16212.83
	AG-DQN	**111168.85**	**114764.00**	**118066.61**	**115126.97**	**1886.15**	16212.83
Li_23	AG	172070.84	182606.92	188589.69	182798.74	9600.12	18801.13
	AG-DQN	**113514.17**	**143514.17**	**163514.17**	**149731.09**	**22607.55**	18801.13
tai75a	AG	2420.61	2860.12	2878.40	2859.90	102.37	1618.36
	AG-DQN	**2119.56**	**2747.39**	**2764.20**	**2748.60**	**97.38**	1618.36

To measure the impact of hybridization, we compared the evolution of the average cost over generations on several benchmarks

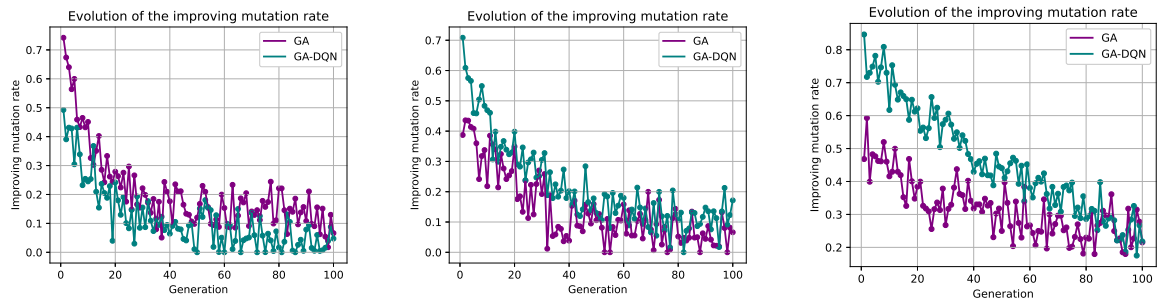


(a) B-n38-k6 (b) X-n101-k25 (c) Li_21

Figure 2: Convergence curves for different CVRP instances

The improved performance of the GA-DQN algorithm can be attributed to its dynamic decision-making mechanism, as illustrated in Figure 3. An analysis of the convergence curves in Figure 2 shows that GA-DQN consistently chieves lower costs compared to the standard genetic algorithm. This advantage is due to the neural network guiding the mutation process, generating more frequent improving moves than traditional stochastic methods.

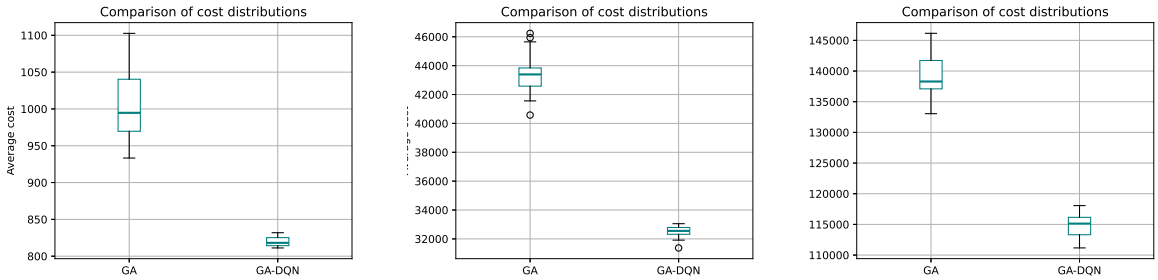
This targeted exploration explains the rapid decline in average costs observed in Figure 2, reflecting a more purposeful movement through the solution space. While the rate of improving mutations naturally tapers off as the search nears convergence a consequence of the diminishing returns of local refinements the GA-DQN maintains a clear edge over the classical genetic algorithm even in later generations. This sustained intelligence prevents the premature stagnation in local optima that so often compromises standard GA performance. Ultimately, Figure 3 provides the mechanistic validation for the results in Figure 2 : the performance gains are not incidental, but the direct result of the DQN acting as an informed selector of mutation strategies, ensuring robustness as problem instances scale in complexity.



(a) B-n38-k6 (b) X-n101-k25 (c) Li_21

Figure 3: Evolution of the mutation rate for GA and GA-DQN on different CVRP instances

The improved mutation effectiveness, being a consequence of the DQN agent, results in a significantly improved success rate for all instances of problems compared to random mutation techniques. Through a strategic creation of variations, this method ensures population diversity and prevents stagnation of search.

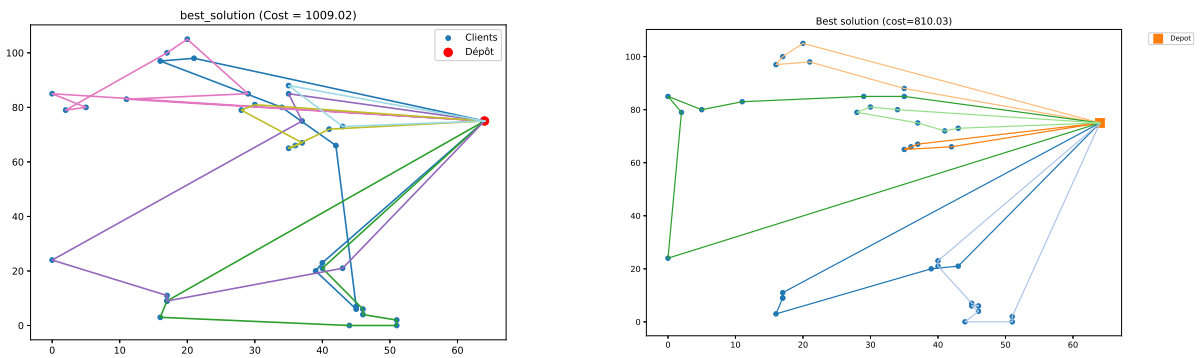


(a) B-n38-k6 (b) X-n101-k25 (c) Li_21

Figure 4: Boxplot comparison of solution cost distributions for GA and GA-DQN across different CVRP instances

The box plots Figure 4 demonstrate very conclusively the statistical edge of the combined methodology in both scenarios, as in all cases, the AG-DQN boxes suggest a clear reduction of mean cost solutions above and beyond what is achieved through the standard GA alone. Most telling, perhaps, however, is simply the fact of these boxes being so small, a clear indication of a reduction in variance or variability of solutions when searching.This lack of variability shows very conclusively that coupling reinforcement learning is more than a way to optimize performance it is a way to bring much greater reliability to the search process regardless of its natural variability.

Even though the model was tested on multiple instances, a smaller instance is shown here to make these differences in routing behavior easier to observe and interpret.



(a) GA (b) GA-DQN

Figure 5: Visual comparison of vehicle routes for instance B-n38-k6 after convergence of GA and GA-DQN

The visualizations in Figure 5 help to better understand the differences between the routing solutions generated by the classical Genetic Algorithm (GA) and the hybrid GA-DQN approach. Although the GA is able to produce feasible and reasonably good routes, the resulting tours often look fragmented, with paths that lack clear spatial continuity between customers.

In comparison, the routes obtained with the GA-DQN method appear more organized and visually coherent. Customers are grouped in a more consistent way, routes are smoother, and route crossings are noticeably reduced. These visual patterns suggest that the DQN-guided mutation helps the algorithm explore the solution space more effectively.

Statistical test

The Wilcoxon signed-rank test was conducted to compare the performance of the classical genetic algorithm with the proposed AG-DQN approach. The results indicate a statistically significant difference

in favor of AG-DQN with a $p-value < 0.01$. This statistical significance suggests that the improvements achieved by the proposed approach are systematic and cannot be attributed to the stochastic nature of evolutionary algorithms.

6. Conclusion

This paper proposes a hybrid GA-DQN approach for solve the Capacitated Vehicle Routing Problem, which is a Deep Q-Network integrated with the genetic algorithm to intelligently guide its mutation operator. Contrary to the classical random mutation, it does allow the proposed mechanism to direct searches to more promising transformations while retaining global exploration capabilities of GA. This hybrid approach tries to improve the quality of the solutions and convergence speed.

Experimental results obtained on a large set of reference instances show that the GA-DQN approach outperforms consistently classical GA, especially when the size of the instances increases. Learning guidance reduces stagnation in local optima, enhances the stability of the results, and produces more spatially consistent solutions. All these observations confirm that reinforcement learning can be used effectively to reinforce some operators of metaheuristics without excessively complicating their structure.

Several contributions arise out of this study it is a method which can be generalized to other forms of VRP, such as time windows multi-depot, dynamic problems, machine learning can assist in other levels of GA, like selection, crossover, more sophisticated models for reinforcement learning have also been explored in order to increase the generalization power in the real-world setting.

Table 3: CVRP Instances Used in the Comparative Study

Instance Name	Clients	Vehicles	Optimal Value
A-n32-k5	31	5	784
A-n65-k9	64	9	1174
A-n80-k10	79	10	1763
B-n38-k6	37	6	805
B-n66-k9	65	9	1316
B-n78-k10	77	10	1221
E-n22-k4	21	4	375
E-n76-k10	75	10	830
E-n101-k8	100	14	1067
F-n45-k4	44	4	724
F-n72-k4	71	4	237
F-n135-k7	134	7	1162
P-n16-k8	15	8	450
P-n76-k4	75	4	593
P-n101-k4	100	4	681
CMT10	199	18	1395.85
CMT12	100	10	819.56
CMT13	120	11	1541.14
M-n101-k10	100	10	82O
M-n151-k12	150	12	1015
M-n200-k16	199	17	1275
X-n101-k25	100	25	27591
Li_21	560	10	16212.83
Li_23	640	15	14499.04
Li_32	1200	11	37159.41
tai75a	75	10	1618.36

References

1. S. Chen et al., *Particle swarm optimization for the vehicle routing problem with time windows*, Expert Systems with Applications **101** (2018), 190–202.
2. N. Christofides and S. Eilon, *The vehicle routing problem*, Naval Research Logistics Quarterly **26** (1979), 223–247.
3. J.-F. Cordeau and G. Laporte, *A tabu search heuristic for the vehicle routing problem with time windows*, Transportation Science **35** (2016), no. 2, 223–247.
4. G.B. Dantzig and J.H. Ramser, *The truck dispatching problem*, Management Science **6** (1959), no. 1, 80–91.
5. Kun Fu, Wei Qiu, and Qiang Zhang, *Generalization in traveling salesman problem with deep learning*, IEEE Transactions on Neural Networks and Learning Systems **33** (2021), no. 8, 3564–3578.
6. Keld Helsgaun, *An extension of the lin-kernighan-helsgaun tsp solver for the cvrp*, Technical Report, Roskilde University (2017).
7. Shauharda Khadka and Kagan Tumer, *Evolutionary reinforcement learning*, Proceedings of the AAAI Conference on Artificial Intelligence (2018).
8. Mustafa Küçük, *Araç rotalama ve çok boyutlu yükleme problemlerine entegre bir yaklaşım*, Ph.D. thesis, Dokuz Eylül Universitesi, Turkey, 2023.
9. Yeong-Dae Kwon, Joonho Choo, Beomjoon Kim, and Inkeun Yoon, *Pomo: Policy optimization with multiple optima for reinforcement learning*, Advances in Neural Information Processing Systems **33** (2020).
10. Gilbert Laporte, *Metaheuristics for vehicle routing problems*, Transportation Science **43** (2009), no. 4, 408–416.
11. S. L. Lee and H. Nazif, *Optimised crossover genetic algorithm for capacitated vehicle routing problem*, Applied Mathematical Modelling **36** (2012), no. 5, 2110–2117.
12. Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takác, *Reinforcement learning for solving the vehicle routing problem*, Advances in Neural Information Processing Systems **31** (2018).
13. C. Prins, *A simple and effective evolutionary algorithm for the vehicle routing problem*, Computers & Operations Research **31** (2004), 1985–2002.
14. L. Reyes-Rubiano et al., *Ant colony optimization for vehicle routing problems*, Swarm and Evolutionary Computation **37** (2017), 1–13.
15. Paolo Toth and Daniele Vigo, *Vehicle routing: Problems, methods, and applications*, Society for Industrial and Applied Mathematics, 2014.
16. Thibaut Vidal et al., *A hybrid genetic algorithm for the capacitated vehicle routing problem*, Computers & Operations Research **115** (2020), 104857.
17. Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly, *Pointer networks*, Advances in Neural Information Processing Systems **28** (2015).
18. Lei Yang and Hao Chen, *A hybrid deep reinforcement learning and large neighborhood search approach for cvrp*, Computers & Operations Research **142** (2022), 105738.
19. B. Yu and Z. Yang, *An improved simulated annealing algorithm for the capacitated vehicle routing problem*, Applied Soft Computing **74** (2019), 1117–1130.
20. Jingwen Yu and Yutong Li, *Attention-based deep reinforcement learning for vehicle routing problems*, Expert Systems with Applications **168** (2021), 114289.
21. Wenqiang Zhang, Xiaomeng Wang, Yashuan Mu, Miaolei Deng, and Peng Li, *Deep reinforcement learning with evolutionary algorithm-guided imitation for capacitated vehicle routing problems*, Applied Soft Computing **184** (2025), 113705.

Manar Fahim,

Laboratory of Mathematical Analysis, Algebra and Applications (LAM2A),

Hassan II University of Casablanca, Faculty of Science, Aïn Chock,

Morocco.

E-mail address: manarfahim12@gmail.com

and

Halima Lakhbab,

Laboratory of Mathematical Analysis, Algebra and Applications (LAM2A),

Hassan II University of Casablanca, Faculty of Science, Aïn Chock,

Morocco.

E-mail address: halimalakhbab@yahoo.fr