Reinforcement Learning Based Optimization for Road-Side-Units Placement Along Highways

Mohammed Saeed*, Shaheer Sherif*, Youssef Mahran* , Lina Ghonim*,
Mohamed Sabry*, Mariam Fathi*, Mohamed A. Ibrahim*, Omar Shehata*
*German University in Cairo (GUC), Egypt
Emails: mohammed.saeed@student.guc.edu.eg, shaheer.aziz@student.guc.edu.eg, youssef.mahran@student.guc.edu.eg,
lina.ghonim@student.guc.edu.eg, mohamed.amer@student.guc.edu.eg, mariam.fathi@student.guc.edu.eg,
mohamed.salahelden@guc.edu.eg, omar.mohamad@guc.edu.eg

Abstract—**This paper proposes a novel Reinforcement Learning (RL) optimization technique for Road-Side-Units (RSUs) placement along a highway. With RSUs playing a crucial role in Intelligent Transportation Systems (ITS), optimizing RSU placement is crucial for efficient communication. This research aims to optimize the cost of the RSU deployment along the highway and minimize the delay in communication between vehicles. For the purpose of this research, two different deterministic RL algorithms were tested, the Deep Deterministic Policy Gradient (DDPG) and the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithms. Both algorithms are model-free off-policy Actor-Critic algorithms with deterministic actions. The algorithms' performance was then compared to three different famous Meta-Heuristic algorithms, Simulated Annealing (SA), Genetic Algorithm (GA), and Discrete Particle Swarm Optimization (DPSO). Training results and Key Performance Indicators (KPIs) show a faster run time for the deterministic RL algorithms for the same number of iterations as the Meta-Heuristic algorithms. KPIs also showcase the better solution cost achieved by the RL agents for the specific optimization problem.**

I. Introduction

The rapid advancement of intelligent transportation systems (ITS) has paved the way for enhanced cooperation between vehicles and infrastructure, aiming to improve road safety, traffic efficiency, and overall driving experience. One of the key enablers of this cooperation is Vehicle-to-Infrastructure (V2I) communication, where vehicles interact with Road-Side Units (RSUs) to exchange critical information such as traffic conditions, road hazards, and navigation data [1].

In order to improve response time and reduce the likelihood of more accidents, RSUs play a crucial role in communicating vital messages, such as accident warnings, to nearby cars and emergency services [2]. RSUs also improve overall network performance, provide internet access, and increase the communication range of automobiles. However, the high cost of deployment frequently limits their use, so strategic placement is required to maximize their coverage and effectiveness, especially in safety-critical applications and remote regions, where their presence can greatly improve traffic management and vehicular communication [3].

A study on RSU deployment optimization in VANETs proposes a cost-efficient strategy using Inertial Navigation Systems (INS) in non-coverage areas to minimize RSUs while maintaining positioning accuracy. Geometric Dilution of Precision (GDOP) is used as a key metric, addressing budget and safety constraints [4]. The authors employ a Simulated Annealing-based Particle Swarm Optimization (SAPSO) algorithm, combining PSO's exploration with SA's global convergence, to optimize RSU placement. This hybrid approach reduces costs and ensures robust positioning, particularly in highway scenarios where full coverage is unnecessary.

A Honey Badger Optimization Algorithm (HBOA)-based RSU deployment scheme (HBOA-RSUD) for VANETs was proposed [2]. Optimizing RSU placement to maximize coverage and minimize costs. Inspired by honey badger foraging, HBOA uses digging and honey modes to identify optimal RSU positions, particularly at urban intersections. A multi-objective fitness function ensures efficient deployment, improving data delivery and energy efficiency. Results show HBOA-RSUD outperforms existing methods in throughput, coverage, and energy savings. While focused on urban intersections, its approach could be adapted for highways, aligning with our research focus.

The problem of RSU placement on highways was investigated where the objective was to minimize network latency while adhering to the total budget designated for deployment [3]. Both vehicle-to-vehicle and vehicle-to-RSU communication was considered where each vehicle could access an RSU using direct access or multi-hop relaying. An RSU placement strategy called Delay Minimization Problem (DMP) is proposed by formulating the problem as an Integer Linear Programming (ILP) model. Simulation results exhibit the superiority of the DMP model as compared to the popular Uniform Distribution model in almost all of the tested cases. Although the paper addresses minimizing latency which was discussed minimally in previous literature, it only considers one numerical optimization technique to address the RSU placement problem and does not assess any meta-heuristic or reinforcement-based optimization techniques.

A two-step technique called ODEL is proposed to consider RSU placement optimization in delay-sensitive applications while reducing the deployment cost of the RSUs [5]. The first step involves determining the RSUs initial positions based on traffic data instead of simply randomizing the initial positions, while the second step utilizes Dijkstra's algorithm and the Genetic Algorithm (GA) to determine the optimal RSU positions. Results show that application delays were reduced by up to 84% while computation performance was reduced by up to 79%. Although this paper strategically initializes RSU positions for faster performance, it only addresses one meta-heuristic algorithm and provides no comparison to any other meta-heuristic or machine learning-based algorithms.

An RSU placement technique based on a memetic algorithm along with a Distributed ML Intrusion Detection System (IDS) for network security was implemented in [6]. The memetic algorithm is essentially a genetic algorithm with an incorporated local search to avoid local minima. Results indicate improved delay as compared with D-RSU and GARSUND algorithms, while the ML model exhibited a detection accuracy of 89.82%. In this paper, however, machine learning techniques were not incorporated in the RSU placement optimization problem which is one of the aims of this paper.

This paper aims to introduce a novel RL approach for optimizing RSU placement along a highway. With the literature focusing on Meta-Heuristic approaches, this paper aims to utilize the Deep Deterministic Policy Gradient (DDPG) and the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithms. The performance of the RL agents is compared with three famous Meta-Heuristic algorithms; Simulated Annealing (SA), Genetic Algorithm (GA), and Discrete Particle Swarm Optimization (DPSO). Utilizing RL in optimizing RSU placement has never been proposed in the literature before.

II. Methodology

The aim of our optimization problem is to find the optimal RSU placement along highways. The challenge is that there are two conflicting objectives which are minimizing transmission delays and minimizing the RSU deployment cost since minimizing the first would entail adding as much RSUs as possible. However, this would simultaneously increase the deployment cost and thus, a balance between the two objectives is desired. Within the context of this paper, the following assumptions are considered. The first is that the RSUs can only be placed on one side of the road. The second is that V2X communication is allowed meaning that the vehicles can communicate with each other as well as with the RSUs. Hence, there are two types of communication which are direct-access communication and multi-hop communication. Direct access communication happens when the vehicle communicates with the RSU in its same road segment directly. Multi-hop communication on the other hand, involves vehicle to vehicle communication then vehicle to RSU communication since it is assumed that the RSUs' transmission range is only within the length of the segment.

A. Problem Formulation

1) Objective Function: The objective of the RSU deployment problem is to minimize latency while simultaneously minimizing the cost of deployment of the RSUs. Hence, the objective function consists of the weighted sum of two elements which are the transmission delay between the vehicle and its corresponding RSU and the deployment cost of the RSUs. The mathematical formulation of the aforementioned objective function is shown in Eq.1 [5].

$$F = \alpha\, F_1 + \beta\, F_2 \tag{1}$$

Where F_1 and F_2 represent the terms of the objective function associated with the deployment cost and the transmission delay respectively. Additionally, α and β are the weights assigned to each of these terms which reflect the relative importance of each of the elements of the objective function. The expression of the deployment cost F_1 is available in Eq.2.

$$F_1 = \min \sum_{i=1}^{k} a_i R_i \tag{2}$$

Where a_i represents the deployment cost of the i^{th} RSU. R_i represents whether or not an RSU is present in the j^{th} segment where $j \in \{1, 2, 3, \ldots, k\}$, and k is the maximum number of road segments. The variable R_i is 0 if no RSU is deployed in the j^{th} road segment, and 1 if an RSU is deployed. The second term in the objective function associated with the transmission delay can be observed in Eq. 3 [5].

$$F_2 = \min \sum_{j=1}^{k} \delta_j \tag{3}$$

Where δ_j represents the transmission delay for a vehicle in road segment j to communicate with its neighboring RSU and is defined as follows:

$$\delta_j = \begin{cases} D_{\text{max}} & if\ y_j = -1 \\ D_{\text{direct}} & if\ y_j = j \\ D_{\text{multi-hop}} & if\ y_j = j+1 \text{ or } y_j = j-1 \end{cases} \tag{4}$$

Where y_j denotes the index of the RSU segment transmitting messages to vehicles in road segment j. If no RSU communicates with segment j, then $y_j = -1$ and the transmission delay is D_{max}. On the other hand, if segment j communicates with the RSU present in its own segment through direct communication, then $y_j = j$ with delay D_{direct}. Finally, if vehicles in road segment j transmit messages to an RSU present in another road segment through multi-hop communication, then y_j is set to the index of that adjacent segment, either $j-1$ or $j+1$ and the transmission delay is $D_{multihop}$.

2) Problem Constraints: The problem is subject to the following constraints which govern the deployment and operation of RSUs and can be expressed mathematically as [3]:

$$R_i \in \{0,1\}, \forall i \in \{1,2,3,\dots,k\} \tag{5}$$

$$T_{th} = 1 \tag{6}$$

$$0 \leq y_j < k-1 \tag{7}$$

$$R_i = 1 \Rightarrow y_j = j \tag{8}$$

$$\sum_{j=1}^{k} [y_j = i] \leq 1, i \in \{1,\dots,k\} \tag{9}$$

The RSU deployment constraint (Eq.5) enforces that only one RSU can be installed in each road segment. A value of $R_i = 0$ indicates that no RSU deployment in segment i, while $R_i = 1$ indicates the presence of an RSU in segment i. Additionally, the multi-hop communication constraint (Eq.6) indicates that at any given time, a multi-hop communication link can be maintained by at most T_{th} (RSUs), where T_{th} is a positive integer representing the maximum allowable hop count for multi-hop relaying. The communication range constraint (Eq.7) restricts a vehicle to communicate with RSUs only within the predefined k segments. Furthermore, the direct access condition (Eq.8) specifies that a vehicle can access an RSU directly only if the vehicle and the RSU are in the same segment. Finally, the RSU access limitation (Eq.9) restricts each RSU to communicate with one vehicle either through direct or multi-hop communication.

B. RL Network Structure

With the optimization problem and its constraints formally defined, two Reinforcement Learning (RL) algorithms, Deep Deterministic Policy Gradient (DDPG) and Twin Delayed Deep Deterministic Policy Gradient (TD3), are used to efficiently solve the RSU placement problem. Both are model-free, off-policy, and deterministic algorithms, making them well-suited for finding a single deterministic solution to this problem.

The TD3 algorithm consists of two actor networks and four critic networks while the DDPG consists of only one actor and two critics. The actor network takes as an input the states and outputs the action while the critic network takes as an input the states in addition to the action and outputs the Q-values. The action space (A) of the two algorithms consists of the solution for the k-segment optimization problem as shown in Eq. 10.

$$A = [R_0, R_1, \dots, R_k] \text{ where } R_i \in \{0,1\} \tag{10}$$

The state or observation space consists of the optimization problem parameters as shown in Eq. 11.

$$S = [k, V_{pl}, C_R, D_{Direct}, D_{multihop}, D_{max}] \tag{11}$$

Where k represents the number of segments, V_{pl} represents the number of vehicles per lane, C_R represents the deployment cost of each RSU, D_{Direct} represents the direct access delay, $D_{multihop}$ represents the multi-hop delay and D_{max} is the maximum delay of a dead vehicle.

The neural networks of both algorithms consist of 4 layers. An input layer of 6 nodes for the actor networks. Two hidden layers of 400 and 300 nodes respectively with *ReLU* activation function and an output layer with *tanh* activation function. Due to the *tanh* activation in the output layer, the actor output is scaled to the range of -1 and 1. The output is then mapped to zeros and ones with the logic presented in Eq. 12.

$$A_i = \begin{cases} 0 & if\ A_i \leq 0 \\ 1 & if\ A_i > 0 \end{cases} \tag{12}$$

C. RL Reward Function

Reinforcement learning frequently uses two kinds of reward functions: sparse and dense reward functions. Sparse reward functions typically offer large rewards that are only accessible in specific scenarios. On the other hand, with dense reward systems, smaller rewards are awarded more frequently. For this research, a dense reward function was used as shown in Eq. 13.

$$reward = \frac{a * k}{f} \tag{13}$$

$$a = 10000 \tag{14}$$

$$k \in \{5, 20, 400\} \tag{15}$$

Where f is the fitness value of the current solution, k is the number of highway segments in the optimization problem, and a is a weighting term where its value shown in Eq. 14 is obtained through trial and error. Since RL problems aim to maximize the reward, while the optimization problem seeks to minimize the fitness value f, minimizing f is equivalent to maximizing its reciprocal $\frac{1}{f}$.

D. Algorithm Hyper Parameters

1) Deep Deterministic Policy Gradient (DDPG): The hyperparameters of any RL algorithm have a direct impact on how effective it is. To maintain stability, a small learning rate was used. The agent is encouraged to choose future rewards over immediate ones as the discount factor is high. To promote exploration over exploitation, a normal action noise with a standard deviation of 0.2 was introduced. Table I lists the hyperparameters that were applied to the DDPG algorithm.

2) Twin Delayed Deep Deterministic Policy Gradient: The same hyperparameters used for the DDPG algorithm were used for the TD3 to maintain consistency in the RL results. However, the TD3 is an upgraded version of the DDPG where it utilizes three main features not previously present in DDPG which are summarized as follows [7]:

- Clipped Double-Q Learning is the first additional feature where TD3 learns two Q-functions rather than one where the targets in the Bellman error loss functions are formed by the lesser of the two Q-values.
- The second feature is "delayed" policy updates. Compared to the Q-function, TD3 changes the policy and target networks less frequently. Hence, for every two Q-function modifications, one policy update is done.

- Target Policy Smoothing also distinguishes TD3 from DDPG. By adding noise to the target action, TD3 smoothes down Q-values along action changes, making it more difficult for the policy to take advantage of Q-function mistakes.

The hyperparameters of the TD3 algorithm used are shown in Table I.

3) Simulated Annealing: The annealing process in materials science serves as the inspiration for the stochastic optimization method known as Simulated Annealing (SA) [8]. Under the guidance of a temperature parameter α that progressively drops, it explores the solution space by accepting both improvements and occasionally worse solutions. Each iteration lowers the temperature by a predetermined amount until a minimum temperature α_f is reached when using a linear cooling rate β. Important hyperparameters include the cooling rate β, the number of iterations per temperature step, the starting temperature α_i, and the lowest temperature α_f. This strategy helps SA in successfully achieving a balance between exploration and exploitation. The set of hyperparameters used for the SA algorithm is shown in Table I.

4) Genetic Algorithm: Genetic Algorithm (GA) is an evolutionary optimization technique inspired by natural selection [9]. It operates on a population of candidate solutions, evolving them through selection, crossover, and mutation. Selection prioritizes the fittest individuals, crossover combines features from pairs of solutions, and mutation introduces random changes to maintain diversity. Key hyperparameters include the population size p, crossover rate χ, mutation rate μ, and elitism rate ϵ, which determines the proportion of top individuals preserved between generations. GA is well-suited for solving complex optimization problems by iteratively refining solutions over generations. The set of hyperparameters used for the GA is shown in Table I.

5) Discrete Particle Swarm Optimization: Discrete Particle Swarm Optimization (DPSO) is an optimization technique tailored for problems with discrete solution spaces, inspired by the collective behavior of swarms [10]. Each particle in the swarm represents a potential solution and moves through the solution space by updating its position based on its own best-known solution and the best-known solution of the swarm. The update is governed by a velocity component adapted to discrete domains, often involving probabilistic decisions for position changes. Key hyperparameters include the swarm size s, inertia weight ω, cognitive weight c_1, social weight c_2, and the maximum number of iterations n. The set of hyperparameters used for the DPSO is shown in Table I.

E. Training Machine

The training procedure followed the conditions and parameters demonstrated above. The training platform was a python based optimization problem implementation. The training ran on a linux-based machine running Ubuntu 20.04. Training was accelerated by using NVIDIA CUDA on a GeForce RTX 3070 GPU with 8GB VRAM, 32GB RAM and a Ryzen 7 5800H CPU which allowed for the quick completion of multiple episodes.

TABLE I
HYPERPARAMETERS USED IN TRAINING

Algorithm	Hyperparameter	Value
Deep Deterministic Policy Gradient (DDPG)		
λ	Learning rate	0.001
B	Replay buffer size	$1,000,000$
–	Steps before learning starts	100
N	Minibatch size	256
τ	Update coefficient	0.005
γ	Discount factor	0.99
f	Training frequency	1 episode
σ_a	Exploration noise	0.2
n	Training steps	1000
Twin Delayed Deep Deterministic Policy Gradient (TD3)		
σ_t	Target policy noise	0.2
–	Target noise clip	0.5
Simulated Annealing (SA)		
α_i	Initial temperature	1500
α_f	Final temperature	0.01
β	Cooling rate	2.0
n	Number of iterations	1000
–	Iteration per temperature drop	1
Genetic Algorithm (GA)		
p	Population size	50
g	Number of generations	1000
ϵ	Elitism rate	0.1
μ	Mutation rate	0.1
χ	Crossover rate	0.8
Discrete Particle Swarm Optimization (DPSO)		
s	Swarm size	50
n	Number of iterations	1000
ω	Inertia weight	0.4
c_1	Cognitive weight	0.2
c_2	Social weight	0.3

F. *Key Performance Indicators*

The performance of the Meta-Heuristic and RL algorithms was evaluated using key performance indicators (KPIs) that were measured by running each algorithm for 10 runs. The runtime and fitness values for each run were recorded. Runtime measures the time taken by the algorithm to complete a run, providing insight into computational efficiency. It is recorded for each run and summarized using average runtime to assess consistency. Fitness values represent the quality of solutions generated by the algorithm in terms of the objective function being optimized. For each run, the fitness values are recorded, and the average fitness and standard deviation are calculated to evaluate solution quality. Together, these KPIs offer a comprehensive view of the algorithm's efficiency and effectiveness.

III. Results

A. *5-Segment Highway Optimization Problem*

Firstly the algorithms were run for 1000 iterations in a small-scale optimization problem. The problem had the following parameters:

- 5 Segments
- 20 Vehicles per Lane
- 800 Units Deployment Cost
- Direct Delay (D_{direct}) of 2 seconds
- Multi-hop Delay ($D_{multi-hop}$) of 5 seconds
- Max Delay (D_{max}) of 150 seconds (Dead Vehicle)
- $\alpha = 0.2$ and $\beta = 0.8$ to favor delay minimization

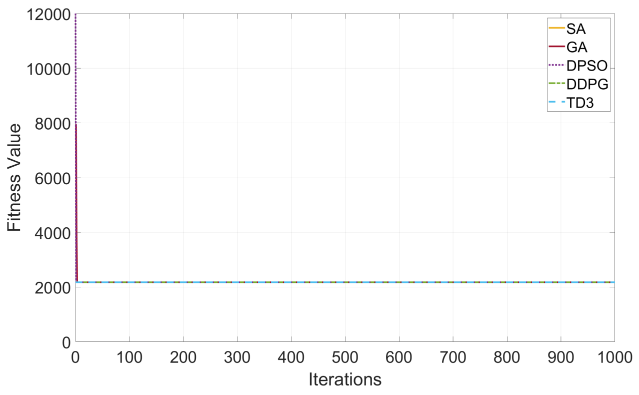

Fig. 1. 5-Segment highway optimization problem convergence curve for SA, GA, DPSO, DDPG, and TD3 algorithms

Fig. 1 shows the convergence curve of the 5 algorithms. All algorithms converged to the same optimal solution from the first iteration due to the small scale of this optimization problem.

B. *20-Segment Highway Optimization Problem*

The algorithms were then run for a larger-scale problem with a highway consisting of 20 segments and the same parameters as the previous problem.

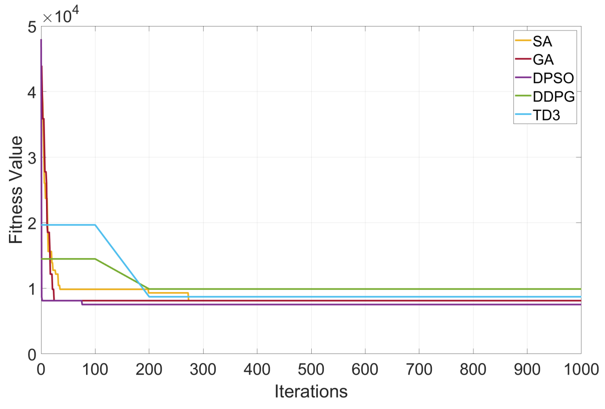

Fig. 2. 20-Segment highway optimization problem convergence curve for SA, GA, DPSO, DDPG, and TD3 algorithms

Fig. 2 shows the DPSO converged to the most optimal solution relatively compared to other algorithms. The GA and SA reach the second most optimal solution with the GA having a faster convergence followed by the TD3. The DDPG achieved the poorest solution.

RL is designed to solve complex problems that need high computational effort which normal techniques fail to solve. This is due to the presence of neural networks which are powerful function approximators. In small-scale problems, such as a 20-segment highway problem, the conventional techniques may achieve a better performance since the problem is not complex enough. However, when increasing the problem size to a full-sized real-life problem, the power of RL algorithms starts to appear.

C. *400-Segment Highway Optimization Problem*

In order to test the algorithms effectively, a large-scale real-life problem was formulated with a highway of 400 segments. Since highways usually link cities and rural areas, they are usually long in length and contain a large number of segments.

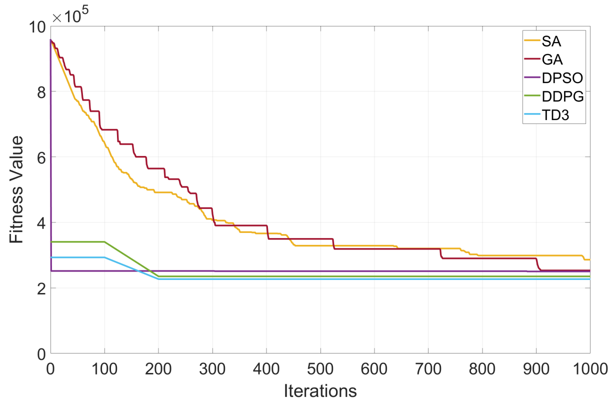

Fig. 3. 400-Segment highway optimization problem convergence curve for SA, GA, DPSO, DDPG, and TD3 algorithms

Fig. 3 shows that the SA started to suffer under a large-scale complex problem with it reaching the poorest fitness value. The

TD3 reached the lowest fitness value and most optimal solution in only 200 iterations. The DDPG also reached the second most optimal value in 200 iterations. The DPSO converged to a value early on however, it was a poorer solution compared to the DDPG and TD3 algorithms.

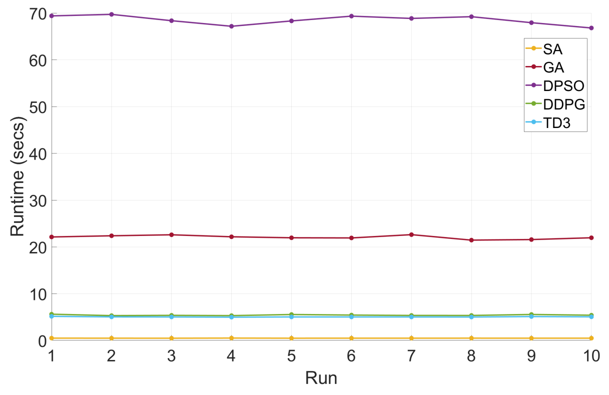

Fig. 4. 400-Segment highway optimization problem runtime KPIs for SA, GA, DPSO, DDPG, and TD3 algorithms

To further solidify the TD3 and DDPG advantage, Fig. 4 shows that although the DPSO posed a challenge in terms of fitness value, it failed to compete in terms of run time. The DPSO, the best performing Meta-Heuristic, averaged a runtime of 68.5 seconds. A very lengthy and demanding runtime relative to the other algorithms. The second best performing Meta-Heuristic, genetic algorithm, averaged 22 seconds of runtime which is still relatively demanding. The DDPG and TD3 on the other hand averaged a runtime of 4.9 seconds proving the RL agents' ability to solve complex problems with ease. The SA was the fastest algorithm with an average runtime of 0.48 seconds however, it had the poorest performance. A summary of the runtime KPIs is presented in Table II.

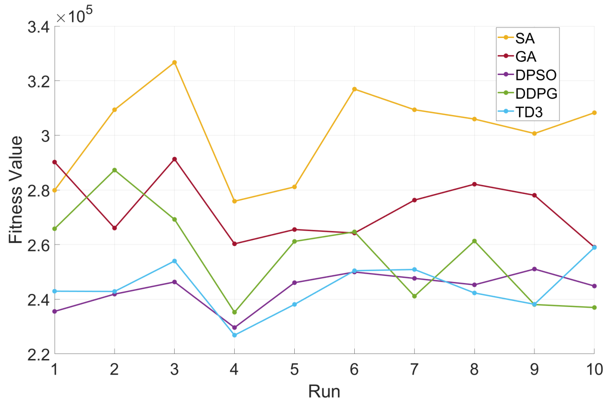

Fig. 5. 400-Segment highway optimization problem fitness KPIs for SA, GA, DPSO, DDPG, and TD3 algorithms

Fig. 5 portrays the fitness value obtained during each run. It is apparent that the SA had the poorest performance while the TD3 had the best performance followed by the DPSO. The DDPG posed a challenge for the DPSO however it couldn't reach a better solution. A summary of the fitness KPIs is presented in Table II.

TABLE II
FITNESS & RUNTIME KPIS FOR SA, GA, DPSO, DDPG AND TD3 ALGORITHMS

Algorithm	Iterations	Average Runtime	Average Fitness	Standard Deviation
SA	1000	0.48 sec	301,395	16,125
GA	1000	21.4 sec	273,297	11,341
DPSO	1000	66.7 sec	243,806	6,266
DDPG	1000	5.2 sec	256,070	16,477
TD3	1000	4.9 sec	244,534	8,818

IV. CONCLUSION

This paper introduced a novel RL-based optimization technique for RSUs placement along a highway. Utilizing deterministic RL algorithms, specifically the DDPG and TD3 algorithms, the proposed agents obtained a similar performance to the Meta-Heuristic algorithms in small-scale problems. However, in large-scale problems, training results showcased the better performance of the DDPG and TD3 algorithms in contrast to Meta-Heuristics. Performance KPIs showcased the much faster runtime for the RL techniques and better average fitness.

REFERENCES

[1] R. Cai, Y. Feng, D. He, Y. Xu, Y. Zhang, and W. Xie, "A combined cable-connected rsu and uav-assisted rsu deployment strategy in v2i communication," in *ICC 2020-2020 IEEE international conference on communications (ICC)*, pp. 1–6, IEEE, 2020.

[2] J. Sengathir, M. Deva Priya, A. Christy Jeba Malar, and S. Sam Peter, "Honey badger optimization algorithm-based rsu deployment for improving network coverage in vanets," in *Micro-electronics and telecommunication engineering: proceedings of 6th ICMETE 2022*, pp. 179–193, Springer, 2023.

[3] Z. Ahmed, S. Naz, and J. Ahmed, "Minimizing transmission delays in vehicular ad hoc networks by optimized placement of road-side unit," *Wireless Networks*, vol. 26, no. 4, pp. 2905–2914, 2020.

[4] S. Zhang, Y. Shi, R. Zhang, and L. Shen, "Roadside units non-full coverage optimization deployment based on simulated annealing particle swarm optimization algorithm," in *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*, pp. 544–549, 2018.

[5] S. Mehar, S.-M. Senouci, A. Kies, and Z. M. Maaza, "An optimized roadside units (rsu) placement for delay-sensitive applications in vehicular networks," in *12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, (New York, United States), pp. 121–127, 2020.

[6] S. Anbalagan, A. K. Bashir, G. Raja, P. Dhanasekaran, G. Vijayaraghavan, U. Tariq, and M. Guizani, "Machine-learning-based efficient and secure rsu placement mechanism for software-defined-iov," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13950–13957, 2021.

[7] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," 2018.

[8] A. G. Nikolaev and S. H. Jacobson, "Simulated annealing," *Handbook of metaheuristics*, pp. 1–39, 2010.

[9] J. H. Holland, "Genetic algorithms," *Scientific american*, vol. 267, no. 1, pp. 66–73, 1992.

[10] K. Rameshkumar, R. K. Suresh, and K. M. Mohanasundaram, "Discrete particle swarm optimization (dpso) algorithm for permutation flowshop scheduling to minimize makespan," in *Advances in Natural Computation* (L. Wang, K. Chen, and Y. S. Ong, eds.), (Berlin, Heidelberg), pp. 572–581, Springer Berlin Heidelberg, 2005.