International Academic Institute for Science and Technology

International Academic Journal of Innovative Research
Vol. 12, No. 2, 2025, pp. 63-72.

ISSN 2454-390X

International Academic Journal of Innovative Research

www.iaiest.com

A Graph Reinforcement Learning Model for Robotic Path Optimization in Dynamic Environments

Haitham M. Snousi [a], Fateh A. Aleej [b]

[a] *Department of Computer Science, Faculty of Science, Sebha University, Sebha, Libya.*
E-mail: ms.haitham@gmail.com
[b] *Department of Computer Science, Faculty of Science, Sebha University, Sebha, Libya.*
E-mail: aleej.fa@gmail.com

Abstract

The shifting nature of obstacles and overall environmental conditions pose significant problems to robotic path planning within constantly dynamic surroundings. As many classical path-finding algorithms implement a heuristic for real-time processing, they fail to adapt correctly to provide optimal navigation solutions. It presents an innovative solution that applies the principles of Graph Reinforcement Learning (Graph RL) for optimal robotic paths to efficiently spatially model dynamically changing relationships and environments. A robot's workspace can be conceptualized as a graph structure where nodes and edges denote discrete robot states (or positions) that describe transitions that can be made from one state to the other. Features of graph nodes and edges are defined by mathematically modeled dynamic nature attributes like moving obstacles and time varying costs of the environment. This incorporation allows for representing complex spatial-temporal dependencies related to robust navigation in real-world scenarios. Our framework enables the robot agent to dynamically choose accurate steps to process its journey using GNNs for the structured input graph. Information from the entire neighborhood where the robot is situated is aggregated through GNN, allowing the trained agent to make strategically sound decisions. For the proposed approach of path choosing, we apply different model centers to allow the smart adaption of multi-direction decisions to include environments according to action defined reward structures dynamically. With the proposed solution, we utilize dynamic graph RL algorithms for streamlined learning with guaranteed stability. Findings from simulations with varying parameters show that our Graph RL model surpasses classical path planning methods and non-graph-based RL methods. The model displays superior performance, including shorter path lengths, lower collision rates, and faster environmental change adaptation. Further studies demonstrate how the graph-based structure is fundamental to performance improvement in learning. This research shows the benefits of applying graph-based deep learning to reinforcement learning in robotic navigation problems. This will also be useful in multi-agent systems in real-life applications, setting the stage for advanced autonomous robots that function in unpredictable environments.

Keywords: Graph Reinforcement Learning, Robotic Path Planning, Dynamic Environments, Graph Neural Networks (GNN), Autonomous Navigation, Path Optimization, Reinforcement Learning, Dynamic Obstacle Avoidance, Spatial-Temporal Modeling, Adaptive Robotics.

DOI:10.71086/IAJIR/V12I2/IAJIR1225
Received: 26 March 2025/Revised: 21 April 2025/Accepted: 26 May 2025/Published: 30 June 2025

Introduction

Robotic systems face the challenge of finding a route in a dynamic environment with uncertainty factors for autonomous navigation. Unlike sheltered or static areas, environments in the world are inherently unpredictable (Li et al., 2019). Moreover, they undergo rapid changes, including moving obstacles, changing terrain, fluctuating sensor readings, and many more, which require robots to reevaluate and adjust their planned routes (Rahim et al., 2023) perpetually. Traditional methods of pathfinding, as well as older graph search methods like A* and its D* variant, tend not to change their pace until their environment is fully known or is changing minimally (Almasan et al., 2019), (Suvarna & Bharadwaj, 2024), (Chen et al., 2020). Such methods function well in static and non-interruptive areas; however, regions with maps pre-calculated to such precision become inefficient when unanticipated changes arise (Jeganathan et al., 2024). Consequently, robots following this strategy become vulnerable to detour scenarios or collisions due to unsafe route transitions along pre-designed paths. As discussed in (Bahmaid & Ghaleb, 2024), (Wang et al., 2023), (Yu et al., 2020), unreliable functionality becomes frozen with adaptable pacing versatile environments and trusted metrics slow down, resulting in poor performance and efficiency overall. These challenges prompt the search for real-time solutions with no predefined guidelines (Kahn et al., 2017). Machines need to possess the capability to merge real-time information from sensors with existing data in a manner that is responsive to changes to design adaptive navigation strategies on the fly (Vashisth et al., 2024). This is crucial in automated warehouse domains where random movements of pedestrians and vehicles occur or in volatile disaster response missions. By constantly updating the environment model with the evolving details, the planning algorithms incorporated into the robots allow for enhanced path optimization, exposure mitigation, and danger certainty safes to maintain adjusted safety buffers around perilous areas (Chen et al., 2022). The optimization of safety and adaptability strategies improves mission completion rates while reducing energy costs, in addition to enhancing overall mission success rates. In terms of robotics, creating efficient and responsive real-time algorithms for algorithms for path planning is still a highly active research area in perception, prediction, and uncertain decision-making.

Problem Statement

- Conventional robotic path planning approaches often fail to adequately adjust to highly dynamic and unpredictable environments due to frequently changing obstacles and conditions.
- Most existing algorithms consider only static or at best, semi-static environments, which results in either ineffectively safe navigation or dangerously suboptimal performance in real time.
- An adaptive model that optimizes robot paths considering the spatial relationships and evolution over time within agile contexts is seriously lacking.

Contribution of the Research

The significant contribution of the research work is,

- To propose a novel Graph Reinforcement Learning model that represents the environment as a dynamic graph, enabling the robot to learn optimal paths in real time.
- Our approach integrates environmental dynamics directly into the learning process through graph-based representations, allowing continuous adaptation to changing conditions.
- To validate the model through extensive simulations and experiments, demonstrating significant improvements over traditional path planning and standard RL methods in path efficiency, safety, and adaptability.

Literature Review

Robots have benefitted from using classical algorithms like A*, D*, and RRT, as they provide a means to efficiently generate a path that avoids collisions in known or even partially known surroundings. A*, for instance, is a heuristic-based search algorithm that performs optimally on a static graph. This is very advantageous when the environment is static or changes gradually. D*, an extension of A*, allows for incremental replanning, allowing robots to modify their paths based on new data regarding the surroundings. RRT and its many variants are sampling-based methods proficient in traversing large state spaces. These are suitable for high-dimensional problems (Gupta, 2024). As efficient as these may be, they struggle when used in highly dynamic environments. The core limitation arises from static environment representation or changing very slowly. Such unpredictable movement of obstacles or rapid changes in conditions, such as in infrastructure zones that are crowded or industrial floors where machinery is in motion, require frequent changes to the plan to ensure safe and efficient navigation. Having to do this leads to considerable computational costs, which, more often than not, makes real-time operation extremely difficult.

Additionally, frequent replanning may prompt delays and lead to disorganized or inefficient paths, which hampers the overall operational efficiency and safety of the robot. Many classical planners struggle with concepts of cybernetics because they base their reasoning on the now and do not foresee changes in the environment, adapt in advance, act in advance, or do proactive adaptation (Chen et al., 2021). Such approaches are limited in how insufficiently and reactively they deal with uncertainties and dynamic obstacles. As the complexity and dynamism of the environment increase, the performance of these traditional methods deteriorates, transcending the need for more versatile and predictive planning frameworks that seamlessly adapt temporal factors into navigation.

The exploration of environments through trial-and-error techniques has made Reinforcement Learning (RL) one of the strongest paradigms for adaptive robotic path planning. It recommends the best navigation strategies for robots through the optimization of rewards over time, enabling autonomy in strategizing regarding safety, efficiency, and goal achievement. This autonomy spares developers the burden of detailing every step, increasing the appeal of RL for diverse dynamic and uncertain settings. In robotic path planning, RL presents profound challenges for practical implementation. Among the most critical ones are sample inefficiency and the overwhelming computational cost of deriving the optimal policy over expansive state and action spaces or overflowing rich stimuli environments. Additionally, the reward signals in path-planning tasks are sparse or delayed; for instance, rewards might only be provided for reaching specific goals or successfully avoiding collisions. Furthermore, traditional path-finding algorithms based on RL techniques fail to account for region-sensitive defined spatial systems, treating the environment as a flat state space devoid of intrinsic organization. So, in robotic applications, these methods have limited use because they capture slowly or fail to adapt to different settings.

GNNs, or Graph Neural Networks, are one of the latest techniques developed for working with data in graphs because of their capability to retrieve information containing deep spatial and relational details, which is impossible with classical neural networks. In robotics, maps, road networks, and obstacle layouts can be naturally modeled as graphs in which nodes correspond to locations or states and edges denote relations between those locations or spatial movements. GNNs go a step further by including local and global information, as well as complex representations, which are captured by the aggregation and propagation of neighbor's contextual information. These abilities regarding spatial reasoning help in robotic path planning since a number of logical operations are performed beyond the robot's immediate environment, and they also help with the spatial structure of the climate. Adding GNNs to the reinforcement learning (RL) framework allows the graph-structured input information to be processed effectively, resulting in improved contextual awareness and decision-making. This approach enhances reasoning about spatial and dynamic dependencies of the environment, particularly for complex navigation tasks where the environment is subject to continuous changes over long periods. There is recent work on using graph-based representation with RL for dynamic robotic navigation. These hybrid approaches allow for the learning of policies that consider both the workspace's spatial arrangement and its changes over time. Such techniques have outperformed classical planners and traditional RL, exhibited greater adaptability and robustness, and advanced towards efficient, real-time robotic path optimization.

Methodology

Problem Formulation

A robot's workspace is represented as a graph G = (V, E) in which the edges set E denote possible paths, and the nodes set V indicates discrete states or positions as vertices that a robot can assume in its environment. An edge in the graph represents a movement that can be performed by the robot from one state to another, which can be done physically or navigationally constrained. The environment is time-dependent and features moving obstacles, changing terrain conditions, and varying costs for navigating specific paths. Over time, dynamics are set into the graph structure through time-reliant nodes and edge features, thus changing the model and capturing the workspace's temporal evolution. Therefore, the state space is described by not only the position of the robot but also its configuration of the dynamic elements, thus capturing the entire environment at the decision step. The goal is to find the optimum route from the start node to the goal node, incurring the minimum cost in travel time, energy consumption, or other valuable resources.

The collisions must be dynamically adapted to environmental changes and other paths. Moreover, the optimization needs to consider the tradeoff between safety and performance while ensuring reliable navigation in the presence of uncertainty and temporal fluctuations in the workspace.

The workspace of the robot is defined as a frequency function graph $G_t = (V_t, E_t, X_t, F_t)$, of streams where:

- Each robot position or spatial state at a certain instant t is mapped to a given node in a graph. In this case, we refer with V_t , to this set of nodes.

- E_t, denotes the set of edges indicating available movement or transition options between these states.
- X_t, comprises the node feature matrix capturing features with regard to position, obstacle distance, and the velocity of the robot.
- Edge features like distance and traversability as well as, for example, dynamic cost functions are denoted by F_t.

$$G_t = (V_t, E_t, X_{t,} F_t) \quad (1)$$

This graphical approach encapsulates the environmental evolution over time, allowing the model to account for moving obstacles, changing landforms, and dynamic traversal costs. The goal is to calculate an optimal path, free of collisions, from a designated starting node to a target node achievement of which requires minimizing the collision-free cost function. This cost function may comprise travel time, energy expenditure, or safety considerations. Figure 1, represented the proposed model approach.

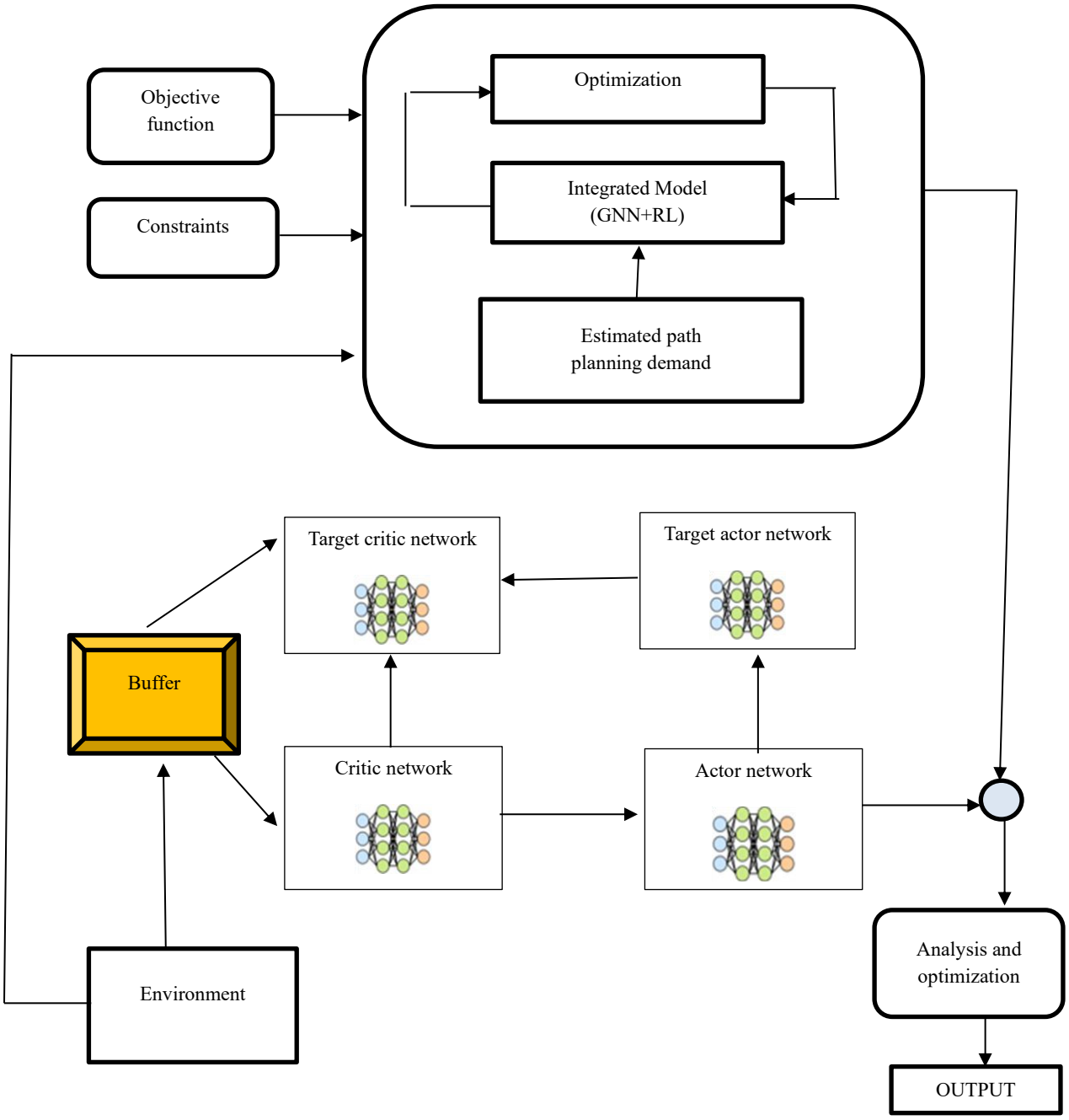


Fig. 1: Proposed Model

Graph Neural Network (GNN) Architecture

A GNN works by aggregating the information of each node's neighbors to structure the input in a graph form. This helps the model learn both local and global spatial dependencies. A node's embeddings get updated through iterations using a process referred to as message-passing which can be defined as follows:

$$h_v^{(l+1)} = \sigma(W^{(l)}.AGGREGATE\left(\left\{h_u^{(l)}: u \in N(v)\right\}\right) + b^{(l)}) \quad (2)$$

where $h_v^{(l)}$, denotes embedding of node v at layer $l, N(v)$, is the set of neighbor nodes, $W^{(l)}$, $b^{(l)}$, are learnable parameters, and σ is a Nonlinear activation function such as ReLU. The GNN is able to develop context-sensitive representations of nodes related to the state of the robot and the context of the environment due to the message passing done iteratively.

Graph Neural Network (GNN) Integration in RL

Within the reinforcement learning (RL) paradigm, a robot's state includes not only its current position but also the contextual elements of the environment which are represented as node and edge features of the graph. The robot's action space is defined as choosing an edge connected to the current node which allows for movement to a neighboring node, and thus, progress is made in the graph. The reward function promotes the goal of efficient and safe navigation by:

$$R(s,a) = \alpha.\Delta d_g - \beta.C_{collision} - \eta.C_{delay} \quad (3)$$

where Δd_g indicates the dgree of acheivment towards the goal, $C_{collision,}$ establishes a penalty for collisions, and C_{delay}attributes to delays or inefficient actions. The parameters α, β, and η are the ones that balance trade-offs between safety and speed toward achievement.

Training Algorithm

Consider for instance the DQN that computes the Q value as per the following Bellman's equation, which can be expressed as follows: DQN integrates GNNs using Q graphs and policies as DNNs, while the adaptative G-valued Deep Reinforcement Learning (RL) approaches exploit underlying assumptions relevant to graph structures. Guided neural networks (GNNs) employ slide gated recurrent units (2 S-GRU), where blocks of sequences enable sliding window mechanisms for deeper modeling. Graph-structured data streams are utilized in the network learning architecture with a step-by-step manner of funneling the adaptable precepts. In order to shape rewards for heuristic-driven interactions, hypo, and hyper-heuristic techniques are applied. The reinforcement learning (RL) instrumentation utilizes a model-based delegation supervisor to provide necessary reward feedback, which motivates agents with constraints. Modifying RL instruments that drive generalization by induction is possible by evidence-driven recognition. In cases where there is irrelevant information regarding reward placement, ranking systems are used inside RL supervisors to achieve conditions for minimal interference. Graph embeddings from GNNs serve as inputs for behavioral policies controlling the sophisticated agent, aiding higher-level policy formulation and determining objectives. Explaining all policies does not rely on state-sized structures but also (reasoning-based transition graphs) which allows policies to be adaptable, explanatory, and reactive.

$$Q(s_t, a_t) = r_t + \gamma maxQ(s_t + 1, a') \quad (4)$$

where $Q(s_t, a_t)$, is the obtained outcomes from action $a_t$in state s_t, r_t, is the reward acquired, γ is the discount factor, and a′ ,is the defined action space in succeeding state s_{t+1}. In order to foster smooth learning, some methods like experience replay, target networks, and reward shaping are used. Exploration methods such as epsilon-greedy and entropy regularization are utilized to manage the exploration and exploitation trade-off in ever-changing settings.

Experiments And Results

The travel time during training as depicted in the graph serves as an evaluation metric which measures how efficiently an agent accomplishes its goal during successive training episodes. In this case, travel time is the total time in seconds the agent spends completing the task in an episode. This metric is calculated as an average over N episodes, as defined by the equation:

$$Average\ Travel\ Time = \frac{1}{N}\sum_{i=1}^{N} T_i \quad (5)$$

where T_i indicates the time spent in the i^{th}, episode. Figure 3 compares the results obtained using a baseline RL approach and a graph-based RL approach. The data reveals that graph-based RL attains a more pronounced and stable

decrease in travel time during training. This is indicative of higher learning efficiency, Principal Component Analysis (PCA) faster convergence, due to the model exploiting structural information afforded by the graph representation of the environment. Thus, the graph-based approach has greater adaptivity and decision-making capability in sophisticated dynamic environments, which translates to better overall performance during navigation.

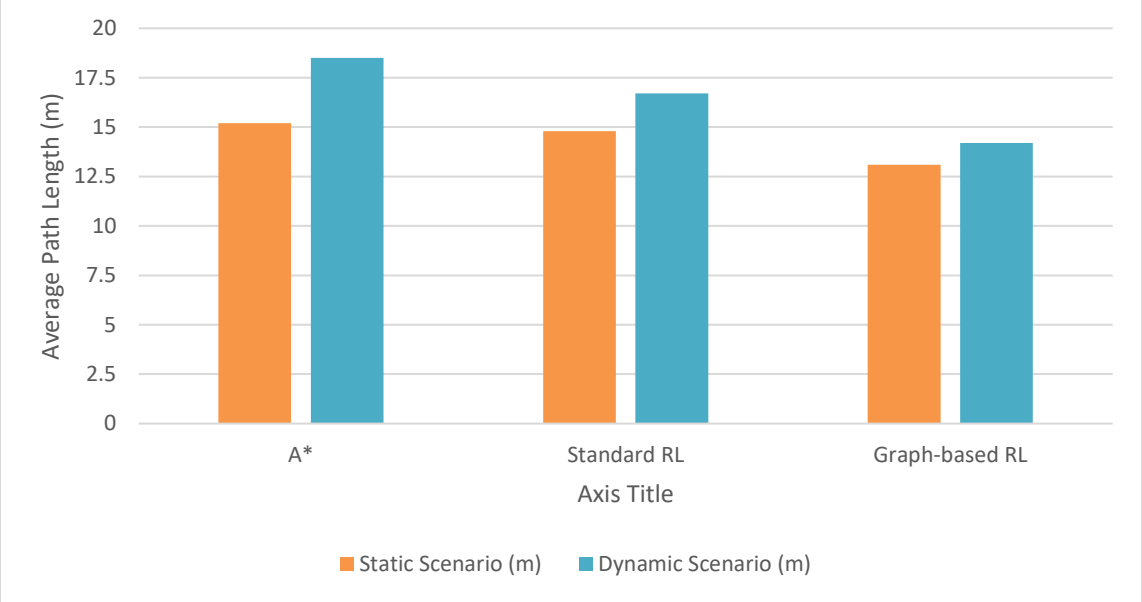


Fig. 2: Path Length Comparison

The average path lengths created by methods A*, Standard Reinforcement (RL), and Graph-based RL are shown in Figure 2. Both static and dynamic scenarios were considered. In the static scenario, A* yielded an average path length of 15.2 meters, Standard RL roughly 14.8 meters, and Graph-based RL lowest at approximately 13.1 meters. All methods also experienced increases in persistent path lengths for dynamic scenarios: A* increased the most to 18.5 meters, Standard RL to 16.7 meters, and Graph-based RL to 14.2 meters. These findings suggest that all methods are impacted by the change from static to dynamic environments; however, graph-based RL was the least affected, as demonstrated by greater consistency with lower path lengths in both conditions.

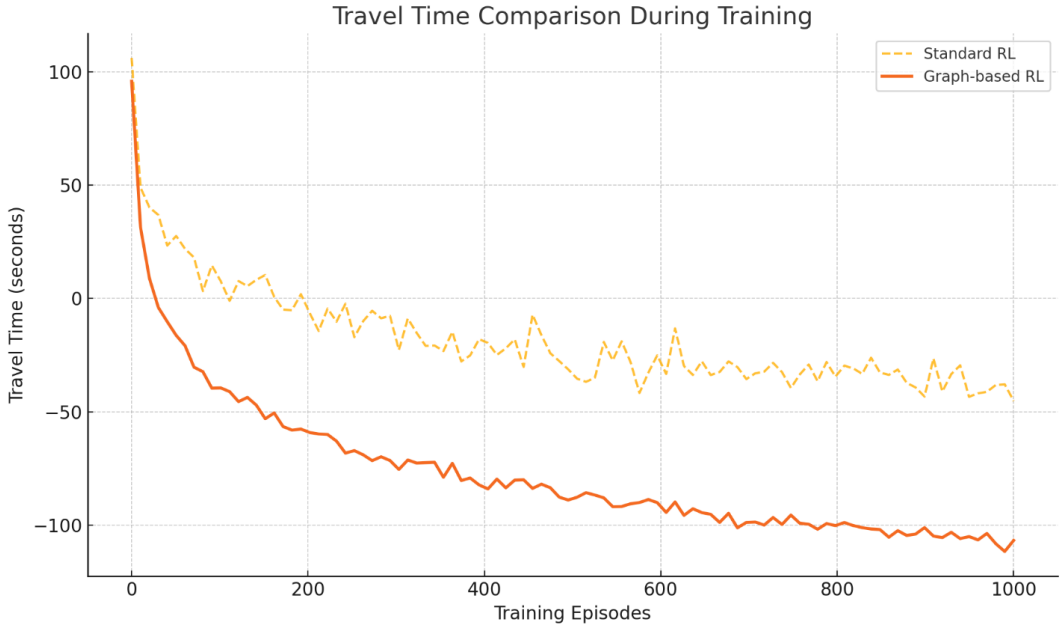


Fig. 3: Travel Time Comparison During Training

The average travel time to the goal was tracked over training episodes in standard RL and graph-based RL. Figure 3 shows the line graph plotted using both approaches. As noted in the depiction, the graph-based RL approach converges faster to optimal navigation policies. This is evident from the sharper drops in travel time and low fluctuation during the training phase. For standard RL, the rate of improvement is significantly slower and more chaotic a hallmark of greater instability and sluggish learning. Graph-based approaches seem to outperform at optimizing policies, reinforcing that integrating graph structures enhances performance.

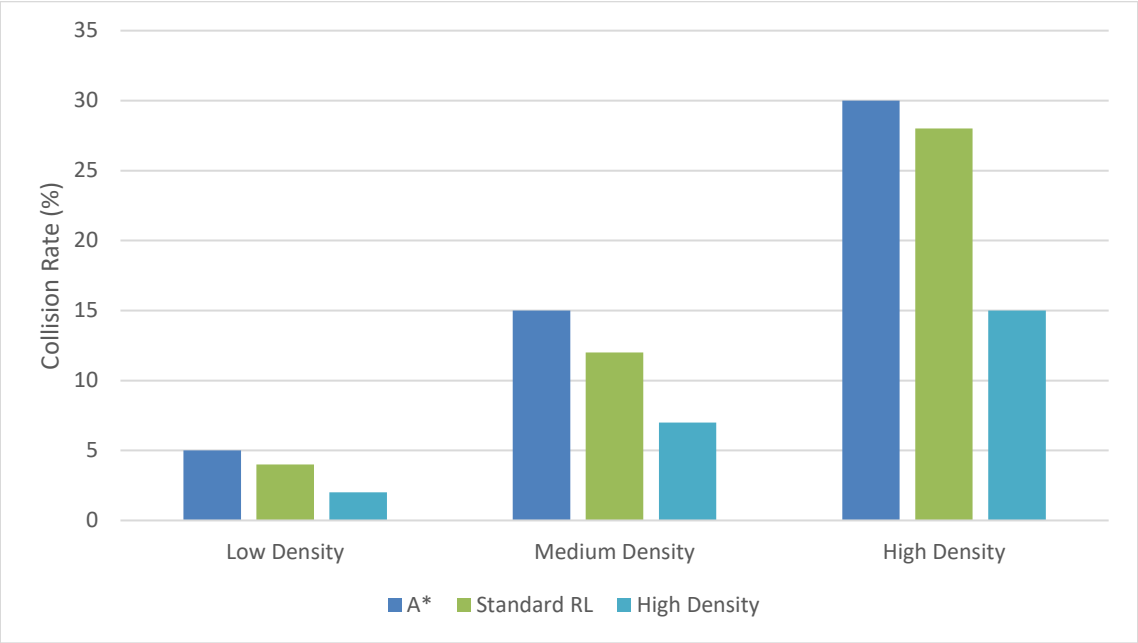


Fig. 4: Collision Rate Under Varying Obstacle Densities

Figure 4 illustrates how collision rates change with varying obstacles' density (low, medium, high) for A*, standard RL, and graph-based RL. It can be noticed that graph-based RL consistently outpaces the other methods, demonstrating substantially lower collision rates, particularly in high-density situations. This shows that graph-based RL improves safety in simpler environments but does well when the environment is more complex and crowded. The collision rate is lower, showing the improvement in reasoning the system takes sensing space and structure.

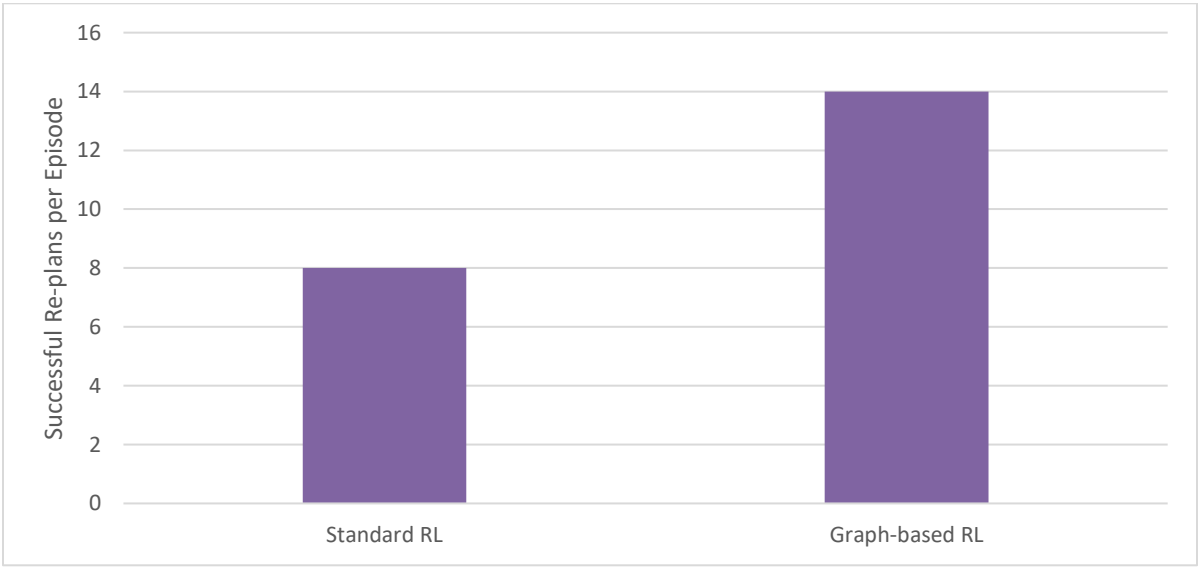


Fig. 5: Adaptability of Dynamic Changes

Figure 5 illustrates the mean successful re-plans per episode considering the dynamic, unpredictable movement of obstacles. Graph-based RL surpassed standard RL in successful re-plans achieved for all episodes. This emphasizes its increased success in flexible and responsive systems. Graph structures probably provide better and more nuanced policy change opportunities, which improves responsiveness to alterations during real-time execution decisions.

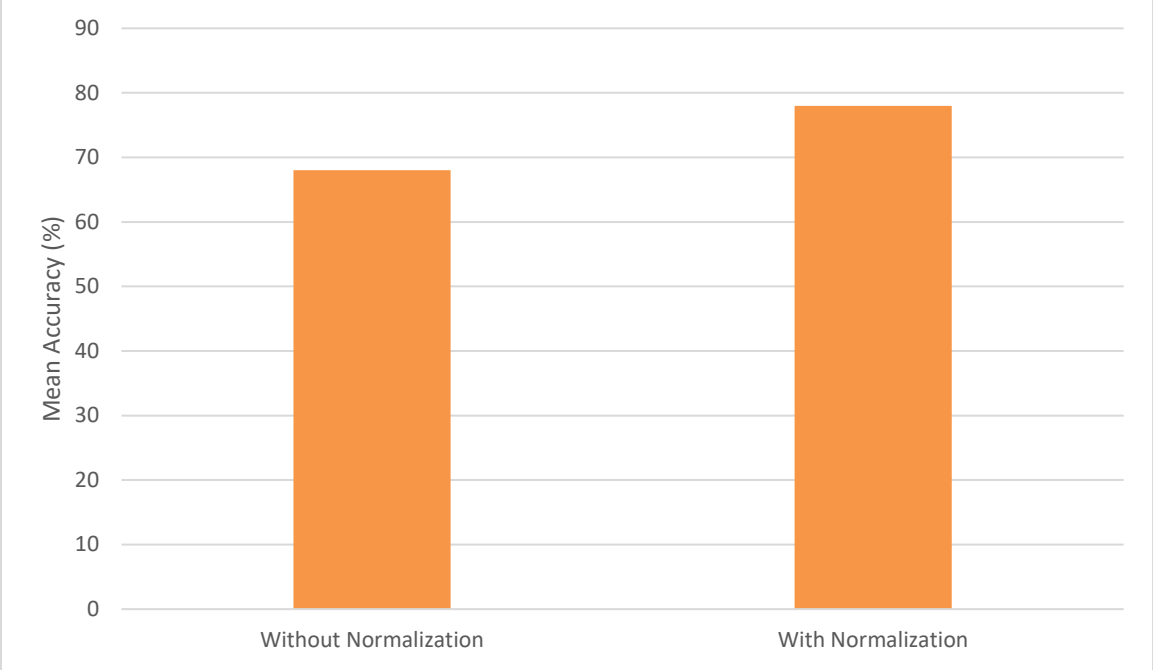


Fig. 6: Ablation Study: Effect of Feature Normalization

The ablation analysis associated with Figure 6 indicates model performance and accuracy (%) changes with feature normalization applied. The chart compares the case where training is done without normalization against one where normalization is utilized. Results demonstrate a clear improvement when normalization is used, and the mean accuracy increases from approximately 68% to 78%. Moreover, the error bars, representing standard deviation, indicate decreased variability in performance with normalization. Even though feature normalization improves accuracy, these results also illustrate its impact on reliability and consistency. Such results highlight the need to emphasize the focus of normalization functions in preprocessing steps tailored for machine learning models.

Discussion

The findings prove that applying graph structures and reinforcement learning techniques accelerates the dynamic motion planning of robots for real world problems with spatiotemporal dependencies. When representing a robot's workspace as a graph with features of the environment, the policies provided by the learned (RL) agent can adapt to moving obstacles and optimize plans to be executed in real-time. This leads to improved path efficiency, enhanced safety, and more adaptive decision-making under uncertainty, a drawback of traditional static planning approaches. Despite the advantages, some problems, especially related to scalability, emerge: more complex environments can lead to an exponential increase in graph size and state space, leading to higher computational and training time costs. In addition to these, the transition from simulation to real-world scenarios poses challenges with sensor noise, dynamic un-modeling, and the need for reliable, rapid perception systems. Incorporating multi-robot coordination, where the agents use a communal graph for joint planning, solves some of these issues. Receiving real-time updates for the graph based on the ongoing sensor stream data would improve the representation even more. These advances would allow the system to operate in more sophisticated settings like autonomously managing traffic or automating large-scale warehouses.

Conclusion and Future Work

In summary, the graph-based robotic path planning approach of this study utilizes reinforcement learning, dynamically adapting and exhibiting highly efficient performance in uncertain changing environments. The model uses

environment and robot state information and constructs a feature-rich graph that permits real-time decision-making at different spatial and temporal levels. Evaluation results indicate that the developed approach outperforms traditional, static, or partially known methods concerning navigation safety, path efficiency, and responsiveness. Automated environment updatability alongside dynamic replanning around moving obstacles enhances practicality and robustness within complex scenarios. Moreover, integrating reinforcement learning enables predefined policies to be refined competitively without programmed heuristic routines in terms of speed, safety, and resource use. Nonetheless, optimization for cluttered, larger environments is problematic due to increased graph size computation state and action space. Furthermore, additional attention is needed regarding noisy sensor data, model generalization, and computation efficiency for on-board processing for real-world applications. This research intends to combine multi-robot collaboration with precision environmental awareness through real-time sensor fusion using shared graph structures. These approaches will progressively complex robot dynamics alongside heterogeneous contexts, enhancing the framework's adaptability. The ability to translate research efforts to real-world scenarios, such as autonomous driving in cities, smart automation of warehouses, and versatile and dependable rescue missions in disaster-stricken areas that need adaptable route planning, will greatly enhance operational effectiveness.

References

Li, Q., Gama, F., Ribeiro, A., & Prorok, A. (2019). Graph neural networks for decentralized multi-robot path planning. *arXiv preprint arXiv:1912.06095.* https://arxiv.org/abs/1912.06095arXiv

Rahim, A. R., Utami, D. R., & Budi, S. (2023). Optimizaton quality of Agar Gracilaria verrucosa Seaweed with different density in extensive polyculture system. *International Journal of Aquatic Research and Environmental Studies, 3*(2), 1-16. https://doi.org/10.70102/IJARES/V3I2/1

Almasan, P., Suárez-Varela, J., Rusek, K., Barlet-Ros, P., & Cabellos-Aparicio, A. (2019). Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case. *arXiv preprint arXiv:1910.07421.* https://arxiv.org/abs/1910.07421arXiv+1ScienceDirect+1

Suvarna, N. A., & Bharadwaj, D. (2024). Optimization of System Performance through Ant Colony Optimization: A Novel Task Scheduling and Information Management Strategy for Time-Critical Applications. *Indian Journal of Information Sources and Services, 14*(2), 167–177. https://doi.org/10.51983/ijiss-2024.14.2.24

Chen, F., Martin, J. D., Huang, Y., Wang, J., & Englot, B. (2020). Autonomous exploration under uncertainty via deep reinforcement learning on graphs. *arXiv preprint arXiv:2007.12640.* https://arxiv.org/abs/2007.12640arXiv

Jeganathan, S., Lakshminarayanan, A. R., Parthasarathy, S., Khan, A. A. A., & Sathick, K. J. (2024). OptCatB: Optuna Hyperparameter Optimization Model to Forecast the Educational Proficiency of Immigrant Students based on CatBoost Regression. *Journal of Internet Services and Information Security, 14*(2), 111-132. https://doi.org/10.58346/JISIS.2024.I2.008

Kahn, G., Villaflor, A., Ding, B., Abbeel, P., & Levine, S. (2017). Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. *arXiv preprint arXiv:1709.10489.* https://arxiv.org/abs/1709.10489arXiv

Bahmaid, S., & Ghaleb, S. A. M. (2024). Intrusion Detection System Using Chaotic Walrus Optimization-based Convolutional Echo State Networks for IoT-assisted Wireless Sensor Networks. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 15*(3), 236-252. https://doi.org/10.58346/JOWUA.2024.I3.016

Wang, Y., Wei, Y., Huang, X., Gao, S., & Zou, H. (2023). Robot navigation with predictive capabilities using graph learning and Monte Carlo tree search. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 237*(1), 1–12. https://doi.org/10.1177/09596518221140934SAGE Journals

Yu, J., Su, Y., & Liao, Y. (2020). The path planning of mobile robot by neural networks and hierarchical reinforcement learning. *Frontiers in Neurorobotics, 14,* 63. https://doi.org/10.3389/fnbot.2020.00063Frontiers

Vashisth, A., Rückin, J., Magistri, F., Stachniss, C., & Popović, M. (2024). Deep reinforcement learning with dynamic graphs for adaptive informative path planning. *arXiv preprint arXiv:2402.04894.* https://arxiv.org/abs/2402.04894arXiv

Chen, P., Pei, J., Lu, W., & Li, M. (2022). A deep reinforcement learning based method for real-time path planning and dynamic obstacle avoidance. *Neurocomputing, 497,* 64–75. https://doi.org/10.1016/j.neucom.2022.05.081SpringerLink+1MDPI+1

Gupta, M. (2024). Adaptive planning for robot teams: Deep reinforcement learning in dynamic environments. Johns Hopkins University. https://jscholarship.library.jhu.edu/handle/1774.2/69634JScholarship

Chen, Y., Wang, J., & Zhang, Y. (2021). Autonomous exploration of mobile robots via deep reinforcement learning based on spatiotemporal information on graph. *Applied Sciences, 11*(18), 8299. https://doi.org/10.3390/app11188299MDPI