Reinforcement Learning-enhanced Path Planning for Mobile Cranes in Dynamic Construction Environments: A Virtual Reality–Simulation Approach

by

Rafik Lemouchi

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Construction Engineering and Management

Department of Civil and Environmental Engineering

University of Alberta

ABSTRACT

This work presents a novel approach to constructing site crane path planning using reinforcement learning and virtual-reality simulations. The approach involves a comprehensive simulation model that includes an agent, actions, states, environment, and a reward system. After extensive training over millions of episodes, the crane agent learns optimal path-planning techniques that improve lifting time, manage energy consumption, and enhance collision detection. The methodology consists of a multi-stage process that begins with creating a realistic virtual environment resembling a complex construction site. In this environment, the crane agent navigates through various scenarios, learning to adapt its path planning to dynamic changes, obstacles, and spatial constraints typical of construction projects. Dynamic changes are related to the placement of different equipment in the construction site and the movement of workers and materials within the site. Through continuous reinforcement learning, the agent refines decision-making, prioritizing efficient lift paths that minimize time and resource utilization, such as fuel consumption.

The study results show significant improvements in terms of lifting time, lifting complexity, and energy consumption, which was achieved through reinforcement learning-based path planning for crane operations. The evolution of the crane agent from initial exploration to peak efficiency is demonstrated through cumulative rewards and decreasing simulation times. The study highlights the agent's ability to maneuver dynamically changing environments and optimize crane operations, showcasing the practicality and effectiveness of the proposed approach. Additionally, the integration of virtual-reality simulations enhances the agent training process by providing realistic scenarios and spatial awareness, which is crucial for crane operators in real-world settings. The study emphasizes the potential of combining advanced technologies such as reinforcement learning and virtual reality to revolutionize crane path planning, ultimately leading to safe, efficient, and environmentally sustainable construction practices.

In conclusion, this research advances crane operations by introducing an innovative methodology that leverages state-of-the-art technologies to optimize path planning and enhance performance in dynamic construction environments.

PREFACE

This thesis is the original work of Rafik Lemouchi. Five articles have been published that are relevant to this research, as well as four articles that are less relevant to this work.

The following publications are directly relevant to the context and scope of this research:

- **Lemouchi, R**., Assaf, M., Bouferguene, A., Al-Hussein, M. "Reinforcement learning-enhanced path planning for mobile cranes in dynamic construction environments: A virtual reality simulation approach." Accepted (Mar., 2024) for publication in *Proceedings, 41st International Symposium on Automation and Robotics in Construction and Mining*, Lille, France, June 3–5, 2024.
- **Lemouchi, R**., Boutouhami, K., Bouferguene, A., Al-Hussein, M. "Reinforcement learning for design." Accepted (Apr., 2023) for publication in *Proceedings, Canadian Society for Civil Engineering Annual Conference*, Moncton, NB, Canada, May 24–27, 2023.
- Boutouhami, K., **Lemouchi, R**., Assaf, M., Bouferguene, A., Al-Hussein, M. (2024). "Hybrid approaches for handling mobile crane location problems in construction sites." *Proceedings, Winter Simulation Conference*, San Antonio, TX, USA, Dec. 10–13, 2023, pp. 2722–2733.
- **Lemouchi, R.,** Assaf, M., Boutouhami, K., Bouferguene, A., Al-Hussein, M. (2023). "Safety training for rigging using virtual reality." *Proceedings, International Conference on Construction Applications of Virtual Reality*, Florence, Italy, Nov. 13–16, 2023, pp. 185–195.
- Aghajamali, K., **Lemouchi, R**., Rahimi, A., Metvaei, S., Lei, Z., Bouferguene, A. "Enhancing safety and efficiency in crane operations: Addressing communication challenges and blind lifts." Submitted (Apr. 2024) for publication in *Proceedings, 2024 Winter Simulation Conference*, Orlando, FL, USA, Dec 15–18, 2024.

The following publications are less relevant to the context and scope of this research:

- **Lemouchi, R**., Correa, W., Rivera, A., Chen, X., Bouferguene, A., Hamzeh, F. "Optimizing safety and productivity with discrete event and agent-based hybrid simulation." Accepted (Apr. 2024) for publication in *Proceedings, Canadian Society for Civil Engineering Annual Conference*, Niagara Falls, ON, Canada, June 5–7, 2024.
- Assaf, M., **Lemouchi, R**., Al-Hussein, M., Li, X. "Virtual reality-based blockchain application for optimized collaborative decisions of modular construction." Accepted (Mar., 2024) for publication in *Proceedings, 41st International Symposium on Automation and Robotics in Construction*, Lille, France, Jun. 3–5, 2024.
- Assaf, M., **Lemouchi, R**., Al-Hussein, M., Li, X. (2023). "A collaborative planning model for offsite construction based on virtual reality and game engines." *Proceedings, Conference on Construction Applications of Virtual Reality*, Florence, Italy, Nov. 13–16, 2023, pp. 46–56.

- Assaf, M., Assaf, S., Correa, W., **Lemouchi, R**., Mohamed, Y. (2024). "A hybrid simulation-based optimization framework for managing modular bridge construction projects: A cable-stayed." *Proceedings, Winter Simulation Conference*, San Antonio, TX, USA, Dec. 10–13, 2023, pp. 3094–3105.

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to my supervisors, Dr. Ahmed Bouferguene and Dr. Mohamad Al-Hussein. Dr. Ahmed Boufergene, your exceptional guidance and steadfast support throughout my MSc journey have been pivotal in my academic and personal growth. Your insights and expertise have enriched my research, and I sincerely appreciate your mentorship. Dr. Mohamad Al-Hussein, your consistent support and invaluable guidance have been instrumental in completing my thesis. Your patience and dedication have inspired me, and I am immensely thankful for your encouragement and advice.

A special acknowledgment goes to Nour Elhouda Rabah Benabbas. Nour, your exceptional support, assistance, and patience have been beyond compare. Your understanding and encouragement during the most trying times have been invaluable. Your kindness and empathy have provided comfort and strength, and I am profoundly grateful for your presence and support. You have been a true friend and ally throughout this journey.

I am also profoundly grateful to Mohamed Assaf for his enduring support, assistance, and guidance during my thesis research. Your readiness to answer my questions, provide constructive feedback, and share your experiences has not just benefited my work but significantly enhanced the quality of my research. Your contributions have been invaluable, and I am sincerely thankful for your help.

To my friends, William, Amira, Alejandro, and Mohamed, your constant support and encouragement have not just been my pillar of strength, but the very foundation of my journey. Your friendship has been a continuous source of motivation and joy. You have uplifted my spirits during challenging times and celebrated my successes as if they were your own. Your companionship has made this journey enjoyable and memorable, and I am deeply grateful for each one of you.

I want to extend my deepest gratitude to my mom. Mom, your unwavering love, encouragement, and support have been the cornerstone of my strength throughout this journey. Your belief in me has been a constant source of motivation, and your sacrifices have not gone unnoticed. You have been my greatest supporter and my strongest advocate. Thank you, Mom, for your endless love and for always being there for me. Your presence has made all the difference, and I am eternally grateful.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Full Expression
BC	Behavioural cloning
BIM	Building information modelling
CL	Curriculum learning
CVE	Curiosity value estimate
DES	Discrete-event Simulation
DOF	Degree of Freedom
EVE	Extrinsic Value Estimate
GA	Genetic Algorithm
GAIL	Generative Adversarial Imitation Learning
GAILVE	Generative Adversarial Imitation Learning Value Estimate
GANS	Generative Adversarial Network
IL	Imitation Learning
MNS	Maximum Number of Steps
RL	Reinforcement Learning
RRT	Rapidly-exploring Random Tree
VR	Virtual Reality
UAV	Unmanned Aerial Vehicles

1. INTRODUCTION

1.1. Research motivation

The use of cranes in construction is increasingly important, especially with the rise of off-site construction methods that rely on these machines to transport and install prefabricated modules. However, integrating new technologies into crane operations has been challenging despite ongoing efforts to modernize the industry (Hu et al., 2021a) . Research has shown that heuristic methods can optimize crane operations, but these methods can be time-consuming and computationally expensive and often need to be revised in dynamic environments (Boutouhami et al., 2023) . Recent advancements in algorithms have shown promise in improving crane operation efficiency. However, challenges remain in adapting to complex environments with multiple degrees of freedom (DOFs). Detailed visualization and improved planning capabilities have brought about advancements, but they also struggle to adapt to dynamic scenarios on construction sites, posing challenges in real-time decision-making and adjustment. Despite these challenges, the construction industry is making significant strides in balancing the age-old challenges of efficient planning and execution with the rapid advancements in automation and digital technologies.

1.2. Research objectives

The following are the research objectives underlying this research, the aim of which is to directly benefit and enhance the work of crane operators, construction project managers, and researchers in the field:

1. To develop improved tools for crane operators that efficiently plan lift paths in virtual environments. The focus is on reducing complexity, improving reliability, and enhancing path planning methods to address the unique challenges of mobile crane operations in congested construction sites to achieve successful and effective lifts.

2. To create an automated path planning process with adaptive learning, considering all lifting approaches, adjusting to site conditions, and establishing optimized procedures for changes during construction.

3. To explore innovative technologies and algorithms that can help achieve an optimal lift path and increase crane operation efficiency.

4. To develop a path planning tool that can adapt to environmental changes on construction sites, mainly focusing on the dynamic nature of lift paths.

The methodology is designed to address the gap with respect to automated path-planning methods for mobile cranes and dynamic construction environments. The focus is on enhancing understanding and providing robust solutions for complex building projects.

1.3. Thesis organization

Chapter 2 (Literature review) provides a detailed summary of the lift planning research conducted in previous works, ranging from using heuristics, meta-heuristics, BIM, and virtual reality (VR) to improve the lift path planning of cranes.
In Chapter 3, the approach employed in the research presented herein is explained in detail, beginning with the development of the simulation logic and continuing through the development of the model and development of the case study.
Chapter 4 discusses the results obtained from the four different approaches. Each approach is assessed, and its performance is compared with the other approaches to provide the best tool for each usage.
Chapter 5 provides the conclusions drawn, as well as the research contributions, study limitations, and recommendations for future work.

2. LITERATURE REVIEW

2.1. Introduction

The construction industry continues to evolve amid efforts to maximize efficiency and minimize costs. In recent years, off-site construction has gained traction due to its ability to save time and money. This approach involves transporting prefabricated modules to the construction site for installation, making cranes an essential component.

Despite numerous attempts to improve crane operations, less efficient tools and planning methodologies still need to be used. According to recent studies (Hu, Fang, and Bai 2021; Tak et al. 2021) , the current practice is for lift engineers to generate static, CAD-based 2D and 3D simulations of various lift scenarios. The planning process is typically time-consuming and based largely on trial and error, where most optimal solutions may be overlooked in the case of complex, congested sites.

2.2. Heureustic search methods

Among the first study to tackle this issue was that by (Reddy & Varghese, 2002) , which used heuristic search methods paired with a configuration space approach to enhance the lift path planning process. Their work paved the way for the automation and optimization of crane operations using various tools, such as heuristic, metaheuristic, and BIM, to tackle the path planning task.

To address the path planning problem, many studies have explored the use of automated planning tools and information technology to enhance path planning practice. The early works in this domain focused on the use of deterministic algorithms for path planning, as presented by (Sivakumar et al., (2003, who employed two heuristic search methods—hill climbing and A*—to automate the path planning task. (ElNimr et al., (2016) , meanwhile, used the A* algorithm in conjunction with 3D visualization, discrete-event simulation (DES), and a mesh generation mechanism to create a framework to enhance the lift path planning process. However, these methods have proved time-consuming and often become stuck in local optima rather than finding the optimal solution.

Several studies have explored using metaheuristic algorithms to improve crane path planning. (Wang et al., (2011) used ant colony optimization to achieve collision-free path planning for mobile cranes. While various works have used the genetic algorithm (GA) to tackle the path

planning task, this use of GA was initiated by (M. S. A. D. Ali et al., (2005, who presented a novel approach for automated path planning of cooperative crane manipulators using GA. The advances made by (M. S. A. D. Ali et al., (2005) paved the way for other works to explore the use of GA in crane operations, notably (Cai et al., (2016), who employed GA to plan lifts in complex environments. At the same time, other works (Boutouhami et al. 2023) with similar results as previous works. However, solutions produced through GA cannot identify the optimal solution and can be heavily influenced by initial conditions, which are probabilistic in nature. Additionally, the computational costs related to implementing GA often render it unfeasible for real-life implementation.

Another common approach amongst the recent works to tackle the path planning task is the use of a rapidly exploring random tree (RRT) algorithm, which was introduced by (Zhou et al., (2020), who developed an approach to guide crane operation to improve the planning operations. Additionally, (Zhou et al., (2021) compared the performance of their approach with methods using bi-directional random and concluded that RRT enhances performance when compared with its predecessors. (Zhou et al., (2021) expanded their work on RRT use for path planning by improving the RRT algorithm performance. However, the major issue with the approaches developed using RRT is the simplification of the crane movement scheme and reducing the degrees of freedom (DOFs) to only three, which is an oversimplification of the task and ignores the complexities and challenges arising from the increased number of DOFs a mobile crane possesses.

The Dijkstra algorithm is another commonly used algorithm in crane operation optimization. (Kayhani et al., (2021) developed a heavy mobile crane lift path planning approach using Dijkstra to plan lifts in congested modular industrial plants using a robotics approach. (Mousaei et al., (2021) presented an automated lift path planning system for mobile cranes leveraging space discretization and obstacle avoidance techniques from robotics and the Dijkstra path planning algorithm to enhance crane operations. (Aghajamali et al., (2023)(used the Dijkstra algorithm to develop a lift planning framework to plan complicated lifts involving walking operations. Despite the improved results using Dijkstra, it has shown many limitations related to computation time and optimal solution development. Additionally, it is well known that the Dijkstra algorithm cannot find solutions with negative edge weights. The negative edge weights could result from changes in elevation, as well as changes in the environment. Thus, to plan lifts in dynamic environments, Dijkstra is unsuitable despite some of the adjustments used in (Aghajamali et al., (2023) to

enhance its performance in such environments.

2.3. Hybrid 3D and BIM simulation

In recent years, many researchers have turned to hybrid 3D/4D simulations and Building Information Technology (BIM) to simulate and generate feasible solutions for lift planning. (Tak et al., (2021) integrated 4D crane simulation and BIM to manage operations on a construction site, while Han et al. 2021 presented a data-driven crane management system for industrial projects. (Dutta et al., (2020) developed a methodology that enables automatic re-planning of lift paths for robotized tower cranes in dynamic BIM environments to address this issue. They used a GPU-based parallelization approach for discrete and continuous collision detection. (Shahnavaz et al., (2020) used BIM and simulation for multi-crane lift animation for collision avoidance and planning, while (Tian et al., 2021) combined BIM with unmanned aerial vehicles (UAV) to monitor crane operations in a steel bridge construction project. Although BIM-based simulations offer detailed visualization, BIM-based approaches struggle to handle dynamic scenarios, making them less adaptable to changes to construction sites and their dynamic nature.

Heuristic search methods, demonstrated by Reddy and Varghese (2002), and other algorithms have significantly advanced automated crane operations, making path planning more efficient and collision-free. However, heuristic methods like hill climbing, A*, and genetic algorithms have drawbacks, such as getting stuck in local optima, being heavily influenced by initial conditions, and having high computational costs. The RRT algorithm oversimplifies crane movement, and the Dijkstra algorithm struggles with computation time and negative edge weights.

Hybrid 3D and BIM simulations and data-driven crane management systems have significantly advanced crane operations on construction sites. These advancements include integrated 4D crane simulation with BIM for comprehensive management, automatic re-planning of lifting paths for robotized tower cranes, and combining BIM with UAVs for detailed visualization and monitoring of crane operations. Despite the benefits, BIM-based approaches need help to handle dynamic scenarios, making them less adaptable to frequent changes on construction sites. BIM-based simulations provide detailed visualization but often need to catch up in dynamically adapting to real-time changes in the construction environment. This limitation reduces their effectiveness in fast-paced or unpredictable scenarios, which is crucial for maintaining efficiency and safety in crane operations

2.4. Reinforcement learning

Reinforcement Learning (RL) is a groundbreaking subset of machine learning, as (G. M. Ali et al., 2023) described in 2023. It is a departure from conventional methods that heavily rely on existing datasets, instead focusing on a learning paradigm centered around experience and feedback. This unique approach enables RL agents to emulate human-like decision-making processes, learning optimal strategies through continuous interaction with their environment. Insights provided by (Lemouchi et al., (2023) shed light on the fundamental principles of RL. Unlike traditional machine learning tools that require vast historical data, RL agents learn through trial and error, gradually refining their actions based on the rewards or penalties received. This iterative learning process is similar to how humans acquire skills and expertise, making RL a promising avenue for developing intelligent systems. However, due to insufficient initial data, RL's effectiveness hinges on extensive training. Researchers such as (Sutton & Barto, (2018) have emphasized the agent's exploration–exploitation dilemma, whereby exploration of new actions as a means of discovering optimal strategies must be balanced with the leveraging of known successful actions. This dynamic nature of RL necessitates careful parameter tuning and optimization to ensure the agent's effectiveness.

The practical applications of RL are diverse and impactful. One notable example is AlphaGo, developed by (Silver et al., (2017) , which demonstrated the capability of RL to master complex games like Go through self-play and RL techniques.

In the realm of construction and engineering, RL has also made significant strides. (Jeong & Jo, 2021) spearheaded the integration of RL with Convolutional Neural Networks (CNNs) for structural design, leveraging industry standards such as those provided by the American Concrete Institute (ACI) in 2022. This approach streamlines the design process and ensures compliance with regulatory requirements. Similarly, (Hayashi & Ohsaki, (2020) applied RL to optimize the design of plane frames under static loads, showcasing the versatility of RL in tackling engineering challenges. Furthermore, (Lemouchi et al., (2023) extended RL's capabilities to designing complex structural systems based on regional standards such as Canadian regulations (Canadian Standards Association, 2019; Government of Canada, 2020) This adaptation underscores the adaptability of RL algorithms to diverse contexts and regulatory frameworks.

Additionally, RL has been instrumental in enhancing decision-making processes during the planning phase of construction projects, as explored by (BuHamdan et al., 2020) . By leveraging RL techniques, stakeholders can make informed decisions that optimize project outcomes while minimizing risks and costs. In the realm of crane operations, (G. M. Ali et al., (2023) introduced a hybrid RL approach for optimizing crane mat layouts, showcasing the applicability of RL in optimizing logistics and operational efficiency. This same approach can be extended to address complex challenges such as crane path planning, highlighting the scalability and versatility of RL techniques.

2.5. Research gaps

For a more detailed analysis of the major works developed for lift path planning, Tables 1 and 2 are presented below. For the purposes of the present work, 11 main criteria were identified in each of the works developed. The 11 criteria are movement criteria, crane configuration, lift simulation, visualization, lift and walk, crab walking, dynamic construction sites, path complexity, energy efficiency, optimality, adaptability to a complex environment, and learning from feedback. Earlier works mainly focused on using simplified methods of path planning, which are easier to model and present fewer challenges in terms of computational cost. For instance, using simplified movement schemes was considered a necessary abstraction level to produce near-accurate results with only a fraction of the calculations. Another main component of previous works is visualization, where most works used 3D models to visualize the lifts. The visualization component was used for collision detection and visual planning of lifts in 3D environments.

Lifting a payload while walking is a possible feature in mobile cranes. However, this feature has been given relatively little attention in the literature (see Table 2). Lift walk minimization means that walking scenarios are considered suboptimal and attempts are made to minimize such operations (unless necessary to perform lift tasks). The logic underlying this is that only a small portion of projects required lifting and walking. "In the PCL construction database, it is reported that crane walking with the load is not commonly performed, accounting for only about 10% of operations. However, in cases where walking is necessary, the lifting process is crucial and becomes the project's bottleneck, impacting its timely completion." (Aghajamali et al. 2023). However, 10% is a significant percentage that cannot be ignored during planning. Thus, the use

of lift and walk must be included in every path planning tool regardless of its implementation in a specific project. Another type of lift walking is crab walking, where the crane's main boom faces a different direction from the crane tracks. According to the literature analysis, this type of lift has been overlooked in existing studies (Table 2).

The next major criterion in construction sites is the dynamic nature of construction sites, where changes occur daily, and the lifting process could be significantly influenced by such unforeseen circumstances. This dynamic nature of construction sites has been ignored or tackled with limited efficiency. (Aghajamali et al., (2023) sought to account for dynamic changes in construction sites within their path-planning approach. However, their approach uses the Dijkstra algorithm. The Dijkstra algorithm is well known for being useful in finding the shortest path between two nodes. However, it cannot find a solution with negative edge weights. Negative edge weights could be a result of environmental changes such as changes in height of the lifted payload. Thus, it is unable to handle changes to the environment that could arise from dynamic obstacles, wind, and other dynamic changes in the environment. Another algorithm that is more flexible toward changes could provide significant results, as explored herein.

Table 1: Criteria addressed in each major work (part 1).

Criterion	Movement Continuity	Crane Configuration	Lift Simulation	Visualization	Lift & Walk	crab walking
Wang et al. 2011	Discrete	No	Yes	3D	No	No
Lei et al. 2013	Discrete	Yes	No	3D	No	No
Lei et al. 2014	Discrete	Yes	No	No	Yes	No
Lin et al. 2014	Discrete	Yes	Yes	3D	No	No
ElNimr et al. 2016	Discrete	No	Yes	3D	No	No
Zhou et al. 2020	Discrete	Yes	Yes	3D	No	No
Kayhani et al. 2021	Semi-Continuous	Yes	Yes	3D	minimized	No

Mousaei et al. 2021	Discrete	Yes	No	3D	No	No
Zhou et al. 2021	Discrete	Yes	Yes	3D	No	No
Aghajamali et al. 2023	Discrete	Yes	No	3D	yes	No
Present work	Continuous	Yes	Yes	3D+VR	Considered & Minimized	Yes

*Discrete is using fixed increased values to move the crane, intermediate values are not considered.

The criterion discussed in this section is path complexity, which requires more in-depth analysis. In this regard, the need was identified for the development of more realistic and manageable lifting patterns, which can be viewed in a virtual environment and assessed prior to the project execution stages. Most previous work in this domain has focused on identifying a short path between two nodes and selecting the path with the lowest cost as the best path. Although some of these developed paths are shorter, they may require many changes in movement and elevation, which would hinder their implementation in a real construction site.

Lift path optimality is discussed extensively in the literature, although gaps remain. For instance, in most of the relevant works, discrete lifting values are used, decreasing the likelihood that the optimal solution will be attained. (The use of discrete lifting values means that a finite number of points will be employed to decrease the number of potential scenarios.) In terms of computation time, discrete values can be beneficial, but this comes at the cost of accuracy. Another deficiency of existing studies in this domain has to do with the use of suboptimal algorithms such as the Dijkstra algorithm, which is proven to produce results that are not well accustomed to dynamic construction site conditions. A* is also well-known for getting stuck in local optimal, and the best solutions are effectively ignored. RL could be used for discovering the environment efficiently, where it could achieve the best solutions and exploring alternatives that could be unforeseen by human users.

One major upset of all of the previous works is the use of discrete movements in lift paths, resulting in shorter calculation times for path determination. However, using discrete values necessitates the use of interpolation which might lead to ignoring potential solutions. The reason for this suboptimality is the presence of intermediate values that could lead to a better solution;

meaning that the optimal solution could be overlooked when using discrete values. In this context, the present work considers the use of continuous lifting coordinates as a means of accounting for all possible solutions, thereby attaining the optimal solution.

No other works have considered crab walking. Hence, solutions that contain crab walking are ignored and not presented in the final lift path. Additionally, previous works have not included the adaptability to complex and changing environments and learning from feedback criteria in their lift path planning methods. These criteria can be incorporated through the use of RL, which is capable of facilitating the customization of the lift according to the construction site properties and is more efficient in responding to unforeseen changes. RL can also learn from the environment through feedback unlike any other approach present in the literature (see Chapter 3). For instance, agents can interact with the environment through sensors, through which they can learn the locations, velocity, and all of the important properties to make decisions when preferring the lift. The ability to learn from the environment makes RL well suited to construction sites, which are continuously changing during the execution phase, and as a result, the produced lift is more efficient, reliable, and implementable in the lifting process.

Table 2: Criteria addressed in each major work (part 2.

Criterion	Dynamic	Path Complexity	Energy Efficiency	Optimality	Adaptability to a complex environment	learning from feedback
Wang et al. 2011	No	No	No	No	No	No
Lei et al. 2013	No	No	No*	No	No	No
Lei et al. 2014	No	No	No	No	No	No
Lin et al. 2014	No	No	No	No	No	No
ElNimr et al. 2016	Yes	No	Yes	No	No	No
Zhou et al. 2020	Yes	No	Yes	No	No	No
Kayhani et al. 2021	No	No	No	No	No	No
Mousaei et al. 2021	No	No	No	No	No	No
Zhou et al. 2021	Yes	No	Yes	No	No	No
Aghajamali et al. 2023	No**	Yes	Yes	No***	No	No
Present work	Yes	Yes	Yes	Yes	Yes	Yes

Notes:

*Each crane movement type consumes a different amount of energy, an aspect that is not considered in the present work.

**The Dijkstra algorithm cannot handle stochastic changes to the environment that could arise from dynamic obstacles, wind, and other dynamic changes in the built environment.

***the proposed work uses the Dijkstra algorithm, which finds the shortest path between two nodes. However, it cannot find solutions with negative edge weights that could arise from dynamic environmental changes.

2.6. Research Motivation

This study explores several areas of improvement where further work is needed, including the following:

1. Better tools must be provided for crane operators to develop practical and achievable lift paths where less redundant and efficient lifts are desired. They are effectively reducing the lift complexity produced in the virtual environment to produce more reliable lift paths.

2. Current path planning methodologies need to account for the unique complexities of mobile crane operations and their planning procedures. These complexities relate to the congested nature of construction sites and the requirements needed to perform a successful lift effectively.

3. A fully automated path planning process that explores all possible lifting approaches while considering the changing nature of construction sites needs to be developed.

4. There is a need for better-optimized procedures that can learn from the built environment and adjust to changes that occur during construction. Therefore, a framework with an adaptive learning approach is required.

5. An optimal lift path is required in order to increase the efficiency of crane operations, and this can be achieved through the use of more advanced algorithms and technologies such as RL. The present work aims to explore more innovative technologies to solve the path planning optimality problem.

6. Lift paths need to be continuously changed as changes occur in the construction site. Thus, there is an urgent need for a path planning tool that can adapt to environmental changes in the construction site.

2.7. Research Objectives

This methodology combines virtual reality (VR) simulations with RL to address previously mentioned research gaps. As such the research presented herein addresses the need for a more comprehensive understanding of optimal solutions in complex and dynamic environments. By integrating 3D environmental elements, the methodology aims to identify precise solutions that consider aspects such as time, complex movement, and realistic scenarios. Furthermore, using RL, the methodology seeks to enhance exploration within construction sites, particularly in the automated path planning of mobile cranes, which can be complex and challenging to adapt to dynamic site conditions. This integration of VR simulations and RL allows a thorough evaluation of diverse alternatives, ultimately selecting the most optimal solutions. The methodology fills the gap left by the absence of fully automated path-planning methods tailored specifically for mobile cranes and dynamic construction settings. Ultimately, this methodology aims to enhance understanding of complex construction environments in order to adapt to changes during the construction phase, leading to better solutions for complex and dynamic building projects. In summary this methodology integrates virtual reality simulations with reinforcement learning to address research gaps in developing optimal solutions for complex and dynamic construction environments, mainly focusing on the automated path planning of mobile cranes.

3. METHODOLOGY

3.1. Overview

The primary objective of this research is to enhance the lift path planning of mobile cranes in construction projects. To achieve this, a framework for a reinforcement learning (RL) based path planning using virtual reality (VR) is introduced, and a comprehensive research methodology is illustrated in Figure 1hereafter.

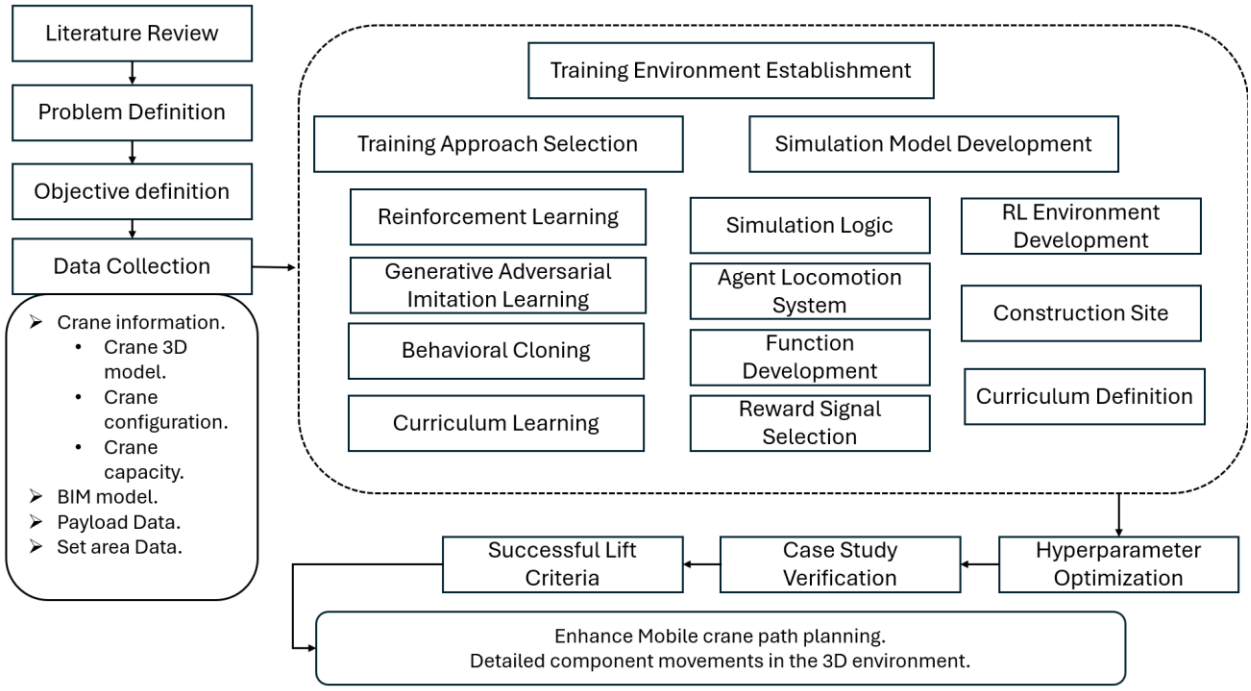


Figure 1: Research methodology.

The research methodology consists of three main components. First, a VR training environment is built to represent the different components encountered during a lift. This consists of collecting data about the various requisite components in order to generate accurate and detailed path planning, starting with a comprehensive 3D model of the crawler crane (data with regard to which is collected from crane catalogues provided by crane manufacturers). Next, a set of 3D models pertaining to the built environment is imported in FBX format from Revit into a secondary 3D modelling software, Blender. The BIM model represent the 3D models of the buildings used in the case study. Next, and after initiating the necessary modifications to the BIM model such as decreasing the internal level of detail, as well as improving the overall format and size of the BIM

models, the models are imported into the Simulation environment and are set with the necessary collision and physical parameters. Still within the data collection process, the necessary information related to the payload—mainly consisting of information related to the dimensions, and physical attributes of the most common payloads in construction projects—is collected. Additionally, irregular shapes of payloads are considered in order to assess the feasibility of lifts in complex scenarios. An example of irregular payload shapes is given in Figure 3, where the water container features an irregular shape.

To assess the validity of the developed methodology, a real construction site is considered as a case study. The case study site contains three different buildings, the 3D models for which were imported from BIM, along with a variety of construction equipment and other elements that may be encountered in a construction site. The case study is discussed in detail in section 4.1.2.

Finally, the following outputs are obtained from this research:

- A detailed path planning decision support system that enables planners to assess a variety of lifting scenarios in a virtual environment using VR.
- An avenue for complete lift path planning automation through the production of a detailed set of coordinates and configurations which can be used to model of the interaction between the mobile crane and the construction site.
- a comprehensive summary of the lift properties throughout the simulation, enabling the user to assess the behaviour of the payload when hoisted in the air and enable the safety and efficiency practices.

3.2. Definitions of lift path planning criteria

The main focus of this work is to provide a lift plan that satisfies the following criteria:

3.2.1. Lifting time

Crane operators have a crucial responsibility of ensuring the safe and efficient transport of payloads. To achieve this, they rely on lifting times as a crucial factor. The longer the payload spends in the air, the higher the risk of accidents, which is why it is essential to keep the lifting time to a minimum. Additionally, the operator must follow the shortest and safest paths during transport to reduce the risk of damage to the payload or any surrounding structures. It is also preferable to maintain the payload's elevation level throughout the lift, as sudden changes in height

can cause instability and increase the risk of accidents. According to a recent study by (Kayhani et al., (2021b) , heavy lifts are often planned such that the payload is keep at a low elevation until the crane is close to their destination, ensuring maximum safety and efficiency during transportation.

3.2.2. Energy consumption

There exist three distinct alternatives to move a load in a sustainable manner. The first one is walking, which involves transporting the object from point A to point B while the mobile crane moves. This type of movement is the least favourable type of lifting primarily because it significantly increases planning costs. The need for crane mats for crane movement is a major contributor to the high costs. One of the most common measures to mitigate mat costs is to plan the lifts in a project such that the crane does not need to walk while performing the lifting procedure (Kayhani et al., 2021b).

The duration of the lifting process is critical in ensuring a safe and efficient transport of the payload when operating a crane. The primary objective of the crane operator is to move the load from one location to another using the shortest and safest route possible while minimizing lifting time. To further enhance safety measures, it is recommended to keep the payload at a low elevation until it reaches its destination. A recent study by (Kayhani et al., 2021b) revealed that heavy lifts are often planned to maintain a low elevation during transport.

The equipment's Hook Movement feature is equipped with a primary hook that offers a single degree of freedom (DOF) in the vertical direction. This exceptional feature allows the hook to effortlessly lift or lower attached objects, making it an incredibly versatile and efficient tool suitable for a wide range of applications.

3.2. Data collection

In order to realistically replicate the lifting planning process, the authors determined that three main types of data are necessary: Crane data, Lifting Module data, and Building data.

3.2.1. Crane model

The crane model used in the simulation is a highly accurate replica of a crawler crane widely used in industry. The model boasts a boom length of 75 m, a track width of 12 m, and a maximum capacity of 3,000 tons, making it a highly versatile and reliable piece of machinery for a wide range of construction projects. The 3D rendition of the crane model is presented in Figure 2,

showcasing the equipment's intricate details and precise specifications.

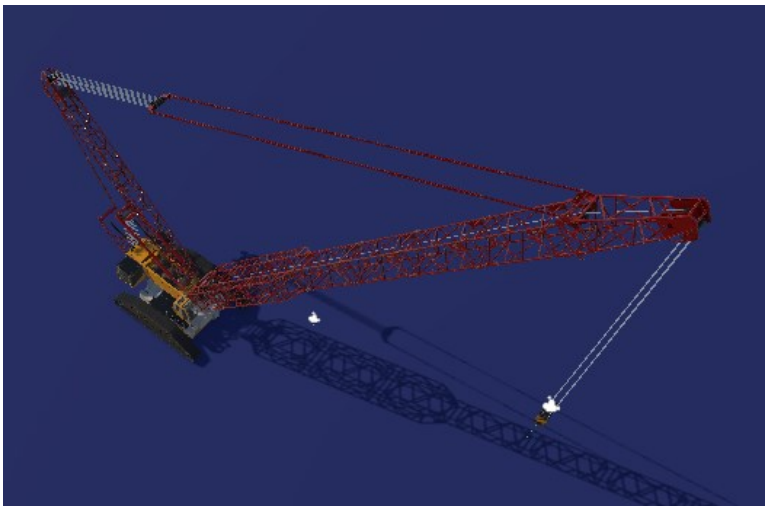


Figure 2: 3D model of crawler crane.

3.2.2. Modules and buildings

Regarding the modules, a diverse array of modules was brought into a 3D format, each possessing unique dimensions, weight, and physical attributes. The buildings were imported via a BIM format, with the models' measurements and characteristics preserved, albeit with simplified component properties and a reduced level of detail to optimize simulation performance.

Figure 3 contains a sample of the payloads used in this simulation. Different types of payloads were intentionally selected to encompass a variety of scenarios, showcasing diverse challenges in loading. These payloads encompass irregular shapes and sizes, each presenting unique lifting-related difficulties. The deliberate inclusion of such varied payloads allows for the complexities associated with different load configurations in crane operations to be explored and addressed.

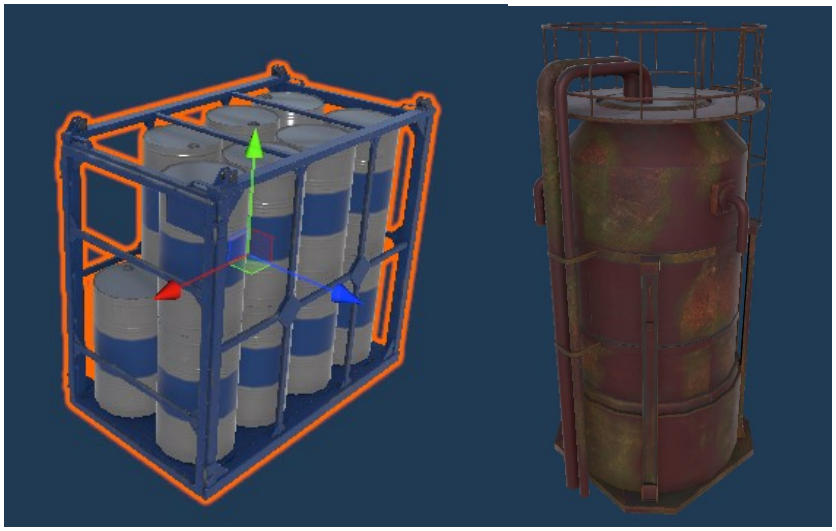


Figure 3: 3D models of payloads.

3.2.3. Rigging components

Creating modelling and rigging components for a VR training environment involved several stages that utilized specialized tools. The authors chose a combination of 3ds Max and SOLIDWORKS because of their compatibility with the Unity 3D game engine, known for its effectiveness in simulation applications. Before beginning the modelling process, the authors carefully identified the rigging components that needed replicating in the virtual environment. This included slings, shackles, hooks, spreader bars, and eyebolts—all essential elements in crane rigging operations.

3ds Max was crucial in designing environmental components and potentially some rigging elements. It likely handled the visual aspects of the training environment, providing a platform for creating immersive surroundings. SOLIDWORKS was specifically employed for creating detailed 3D models of the rigging components. Known for its precision in mechanical design, SOLIDWORKS ensured that the rigging equipment was accurately represented in the virtual space, considering factors such as dimensions.

The modelling process focused on achieving accuracy and realism. This involved careful attention to detail to ensure that the 3D models closely resembled their real-world counterparts. The goal was to create authentic crane rigging operations. 3ds Max and SOLIDWORKS were chosen for their seamless compatibility with the Unity 3D game engine. This streamlined the integration process, allowing for the smooth incorporation of the 3D models into the VR training environment developed in Unity 3D.

Overall, the process involved a systematic approach to tool selection, component identification, detailed modelling, accuracy checks, compatibility considerations, and optimization for training purposes. The result was a comprehensive set of 3D models that formed the backbone of the VR crane rigging training environment, as shown in Figure 4.

Figure 4: Rigging system components.

3.2.4. Hardware selection

To develop the simulation model, various tools, such as a VR setup equipped with a set of computer hardware, were employed. For running the simulations efficiently and smoothly, a high-performance computer featuring an I9 processor I9-2900K, RTX 4090TI graphics card, and a RAM capacity of 64 GB for rendering and visual representation of the developed lifting schemes, was employed. For the VR segment, the HTC VIVE headset was used. This software can present realistic environments in 3D. Furthermore, the HTC headset has eye-tracking features that can track the user's focus throughout the lifting process. The final results produced by the RL and VR model can be used for training purposes, where the worker endeavors to imitate the generated lifts in the virtual environment. The hardware used to develop this model can be seen in Figure 5.

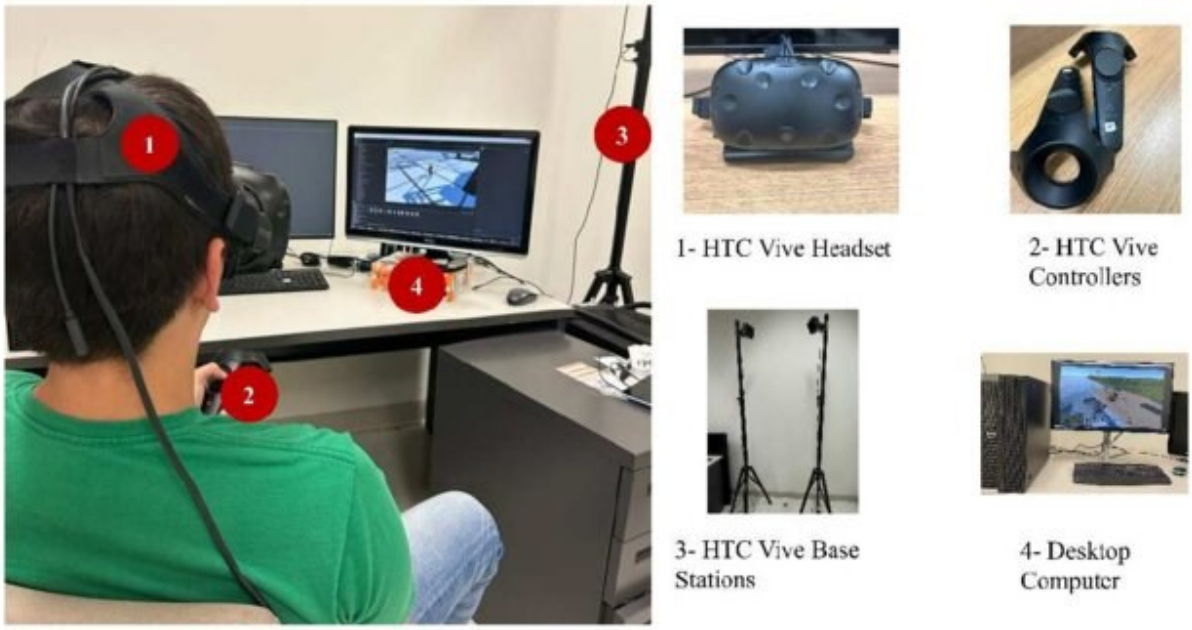


Figure 5: Hardware used to develop the VR simulation model.

3.3. Training environment development

3.3.1. Selection of training approach

To perform a lift, various lifting approaches were considered to take into account the different levels of knowledge that the users have. Each approach provides a set of advantages and disadvantages of its own, and it is up to the user to select the most appropriate tool. The pure RL approach presents a fully automatic approach where no user interaction is required. generative imitation learning requires examples of lifts to be used for training, and the RL agent is expected to replicate the examples and then improve upon them initially. The behavioural cloning approach is very similar to the imitation learning approach; however, more emphasis is given on replicating the examples. Finally, the curriculum learning (CL) approach defines a strategy that can be used for the agent's learning. Each approach requires a level of human intervention, starting with the pure RL approach, which requires no human intervention.

3.3.1.1. Pure reinforcement learning approach

This particular approach to lift planning involves the absence of any sort of prior knowledge. Consequently, the algorithms involved must be capable of selecting an appropriate path without any prior knowledge of the path planning process. This process requires a long phase of trial and

error during which the agent learns about the environment and combines that information with the possible action sets.

This approach offers numerous advantages, the first of which is the ability to plan from scratch without any assistance from external factors. This is expected to provide solutions that are different from those produced by engineers. Additionally, this approach could theoretically provide a valuable resource for planning tasks that are too complex for engineers to assess manually. It is also highly flexible and adaptable, allowing it to achieve tasks without any human intervention. Furthermore, this approach has an increased level of exploration, making it capable of identifying novel solutions.

This approach is particularly well-suited for continuous control problems, such as autonomous driving, robotics, and mobile crane path planning. However, the pure RL approach does come with a set of setbacks. Due to its increased level of exploration, it requires a large set of samples and episodes to train, which is further exacerbated if the task is increasingly complex. The complexity can arise from dynamic changes in the construction site. Additionally, the reward signal design and selection present a major obstacle during the development phase, as it is extremely complex to balance between exploration and exploitation while optimizing the overall performance of the training. Finally, the hyperparameter selection process is a long and tedious, and the agent is highly sensitive to changes made in the parameters, which can destabilize the system.

3.3.1.2. Imitation Learning Approach

This approach involves presenting the agent with a guide, the agent is then expected to follow the directions and produce a similar lift that the guide presented. Later, the agent is expected to continue its pure RL approach, where it would continue improving upon the guide. This approach has many advantages, such as the ability to mimic expert behaviour by imitating the policies and strategies present in the expert guidance while providing safer outcomes that account for risk (with the agent having the capacity to learn safe policies and explore potential risky alternatives).

Imitation learning also provides a data-efficient approach, unlike the pure RL approach; it only requires a fraction of the samples to learn the same task. It also has the added benefit of needing less time to obtain a good policy and mainly focuses on improving the policy rather than inventing it from scratch.

However, imitation learning heavily relies on expert-based demonstration, which means that the demonstration's quality markedly influences the overall results. Moreover, since the agent is attempting to learn a predetermined policy, it will not have the capacity to explore other solutions simultaneously. Thus, the exploration portion of this approach is less favourable. Finally, the environment and the demonstration must be broadly similar; in the possibility of a mismatch between them, the agent could have difficulty learning and implementing the proper policy.

Imitation learning is a different type of learning explored in this research, where the agent is expected to learn from a teacher, demonstrating how the task is expected to be implemented. This approach is very similar to supervised learning methods in other machine learning tasks. Generally, in RL problems, the agent is only given hints (rewards) to understand and define the behaviour and policy necessary to solve the task. While this approach usually produces results unique to those produced by humans, it may require an extended period to produce better results than those produced by humans.

However, Imitation learning includes humans in the learning loop, ensuring the agent obtains more feedback when learning, which in turn decreases the time needed to master the task significantly. The approaches discussed in this section include combining RL and imitation learning in producing detailed lift path plans. The combination of RL and IL is expected to produce the best results regarding analysis time, extrinsic rewards, and lift paths.

In order to implement this approach, a variety of tools and algorithms were considered. In the following subsections, each approach is described, and the advantages and disadvantages of that particular approach are listed.

3.3.1.3. Generative Adversarial Imitation Learning

Generative adversarial imitation learning (GAIL) is an algorithm generally combined with RL. However, unlike traditional RL, GAIL leverages the framework of generative adversarial networks (GANS) to comprehend and learn from demonstrations. According to this definition, GAIL is a type of imitation learning since it uses expert opinion to train the agent's policy.

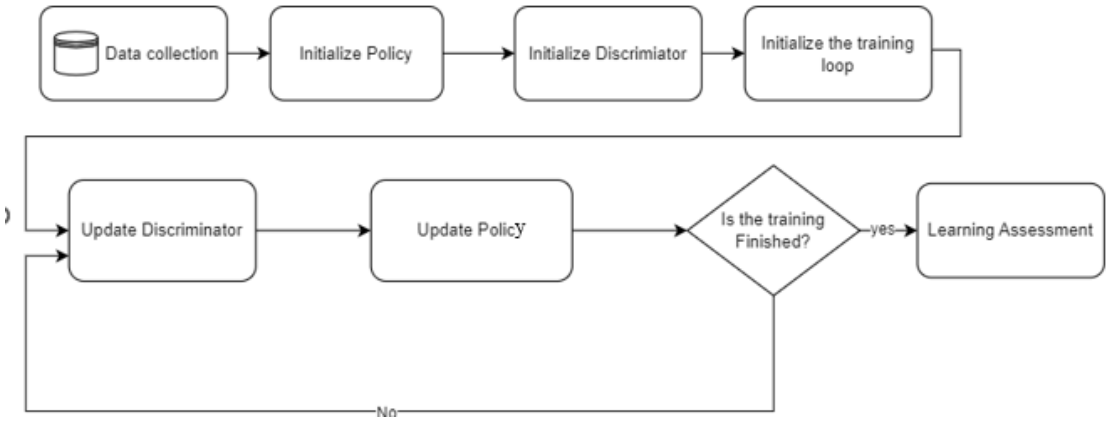


Figure 6: General framework of GAIL.

The general framework used in the GAIL methodology is presented in Figure 6, starting with the initial step of collecting lift-related data, similar to behavioral cloning, which is done through expert demonstration. Next, the agent's policy is initiated, which is initially randomized, and the discriminator follows suit. The training loop is initiated based on updating the discriminator and the policy at each turn. The discriminator uses a binary classification to determine whether the expert takes the action or is generated by the policy. Afterward, the feedback from the discriminator is fed to the policy, whose objective is to generate actions and trajectories that are indistinguishable from those of the demonstration. The training loop is repeated several times until the number of iterations is met or a convergence criterion has been established. Finally, an evaluation process is initiated to assess the training results based on task completion and reward acquisition.

GAIL presents a variety of advantages when compared to traditional RL or BC. For instance, GAIL is sample efficient and requires fewer training episodes to achieve the same tasks as RL. Additionally, unlike RL, GAIL is less reliant on the reward function since it learns from implicit demonstrations provided prior to training. Compared to BC, GAIL also presents a more flexible tool in terms of adaptability to different learning environments and addressing new problems independently.

However, GAIL presents similar disadvantages to BC regarding data dependency, where the agent relies heavily on the quality of the demonstrations provided. Additionally, it is well known that using GAIL requires more knowledge of hyperparameter tuning and environment modelling. GAIL is a complex tool that requires training both the agent and the discriminator, which requires optimal parameters to achieve a stable convergence process. Furthermore, GAIL is

computationally expensive, specifically when the environment is large-scale and complex. Finally, GAIL is susceptible to model collapse, i.e., the scenario in which the agent fails to explore the full range of the environments and actions, which could result in the invariability of the training performance. The level of complexity and computational burden of GAIL were encountered firsthand during the development of the simulation model.

3.3.1.4. Behavioural Cloning

Behavioural cloning is a supervised learning technique involving demonstrations and a mimicking agent whose objective is to replicate the demonstrations provided. Figure 7 presents the general methodology used in behavioural cloning, where the initial stages consist of collecting data from experts; the data consists of actions, states, and sequences needed to perform a lift. The data is then exported, and a function approximator is trained. In order to solve the lift path planning problem, the author assessed a variety of neural network combinations to select the optimal function approximator, the agent is then expected to replicate the expert behaviour.

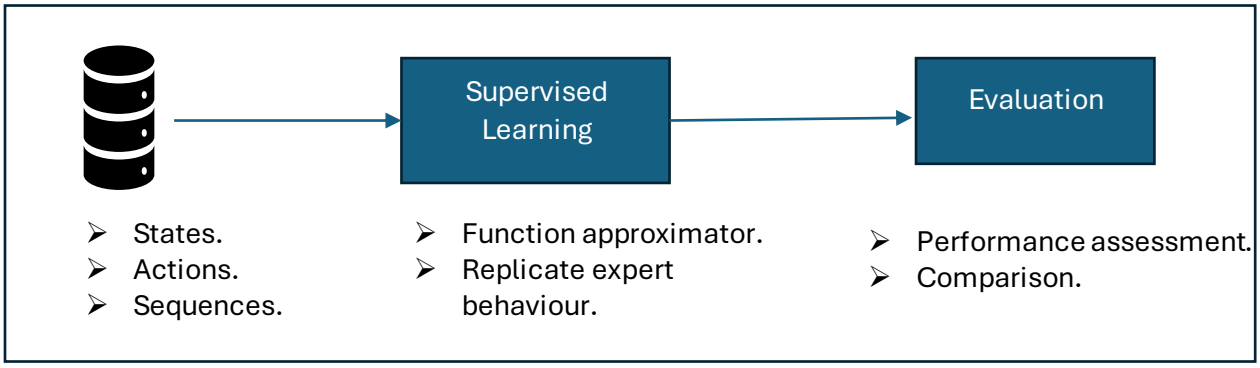


Figure 7: Behavioural cloning methodology.

Finally, to evaluate the actions taken by the agent, each action is compared to the expert demonstration, and the agent's performance is assessed accordingly.

BC presents various advantages that can prove vital for the path-planning task. Firstly, it provides a more efficient and faster approach to training; this ensures that the agent does not linger in the exploration loophole, which requires large amounts of computational power and time. The second advantage of BC. The BC approach was found to produce results consistently with the training examples. Finally, BC is known for its stability during training, where exploration does not hinder the learning process, and the number of necessary samples is significantly fewer than those of RL-based methods.

However, BC is not without flaws; in many cases, these flaws have deterred researchers from using this complex approach. The first major flaw is data dependency, where the agent is incapable of learning without data as opposed to RL. Furthermore, the data quality can significantly affect the agent's learning. Flexibility also poses a significant issue for BC-based training methods, where the agent is incapable of learning if the agent encounters states that differ significantly from the ones presented in the demonstration—and finally, the lack of adaptability where the agent is incapable of handling novel scenarios.

The disadvantages of BC present an opportunity to use hybrid tools and approaches. In many cases, RL and GAIL, among other techniques, to enhance the training process of different agents and produce high-level training outcomes in terms of performance.

3.3.1.5. Curriculum Learning

CL in machine learning is an approach that involves presenting an agent with a series of tasks of increasing complexity. This encourages the agent to progressively improve its skills, from simple tasks to complex ones. This process facilitates a smooth transition from an elementary level of efficiency to a more advanced level of mastery. Moreover, this approach presents a countermeasure to avoid the problem of overfitting.

For the purposes of the present work, CL was deemed to be a suitable alternative due to its capacity to create a hierarchical learning process, whereby the goals and objectives of the agent can be set prior to training. Additionally, it is an excellent tool for comprehending the logic used to develop the path plan, where each step can be viewed and approved individually. It also presents other advantages, such as a robust learning process in terms of exposing the agent to various tasks and planning how to tackle those tasks.

However, the CL approach presents some disadvantages in that the curriculum design process is challenging; this task is often seen as complex, and the knowledge and expertise of an external entity (lifting examples from experts) is generally needed to aid in designing and identifying the best curriculum. Moreover, the curriculum itself could be detrimental to the flexibility of the agent, where the agent is required to learn the curriculum to improve its performance, which might result in neglecting other solutions that need to be presented in the curriculum.

3.4. Simulation logic

In this section a brief description of the methodology used to develop the simulation model is presented. It consists of five main components which are the agent, the actions, the states, the environment, and the rewards. The interaction between the different components is represented in Figure 8.

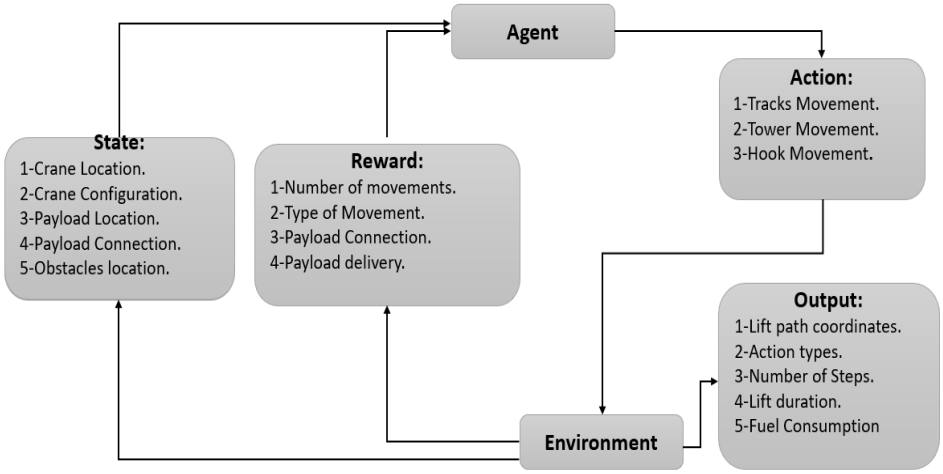
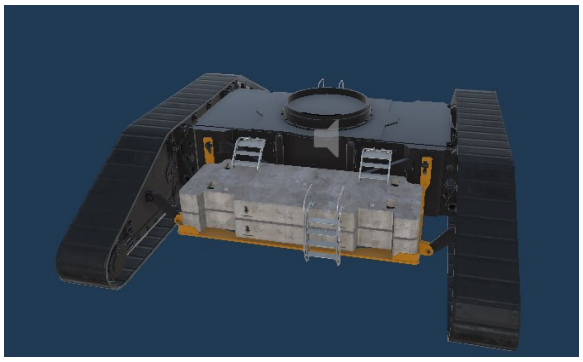


Figure 8: Reinforcement learning model.

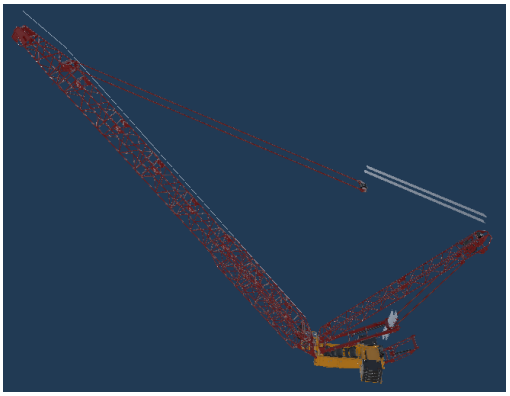
3.4.1. Agent

Initially a crane agent is needed to take different actions, to transport the payload from its original loading point to a predetermined set-point. To achieve this task a proper locomotion system is needed. It should be noted in this regard that a mobile crane's motion system is more complex than that of a tower crane, being made up of many parts that the crane operator needs to coordinate in order to perform a lift. For the present work, three components' movements were modelled and used in the training process, as shown in Figure 9.



A) Crane tracks

B) Main hook



C) Crane boom

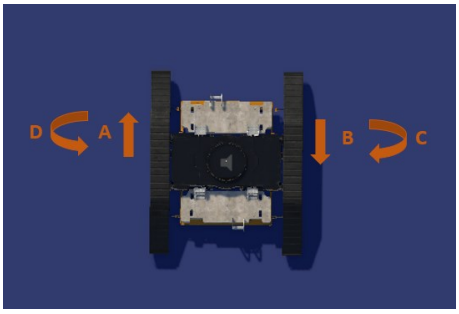
Figure 9: Main crane components.

The first component is the main boom, which can rotate around the crane's central axis and lower or increase its lift angle. Both movements were considered, and the angle's impact on the loading capacity was also considered.

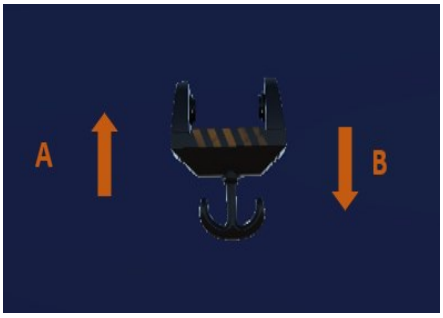
The second component is the crane's tracks, which enable it to move forward and backward and rotate around its center. Finally, the crane's hook's movement was considered. This component has a relatively simple movement: it moves either up or down depending on the lift phase.

3.4.2. Actions

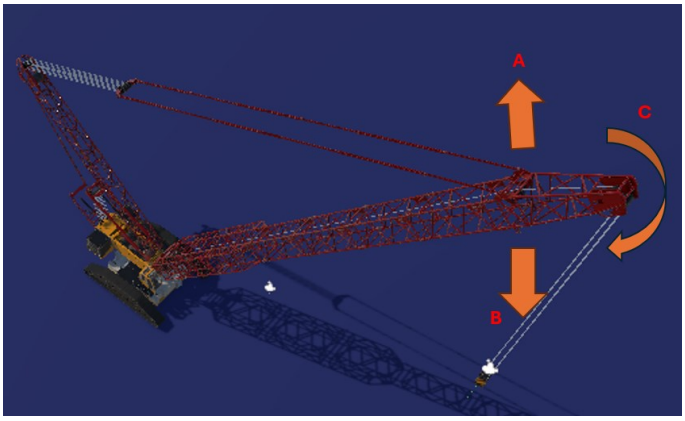
Since the agent comprises three main components, each component was given one DOF.



A) Crane tracks



B) Main hook



C) Crane boom

Figure 10: Degrees of freedom of the main crane components.

The tracks were assigned two DOFs, one rotation around the *y*-axis, and one translation along the *x*-axis, as seen in Figure 10 A. For the agent, this constitutes four different actions that can be taken.

The main hook was given one DOF along the *y*-axis, as seen in Figure 10 B, which allows the agent to take two different additional actions. Finally, the main boom was given two DOFs and two rotations (one rotation around the *y*-axis, and one around the *z*-axis), as seen in Figure 10 C. This constitutes four additional actions.

3.4.3. Environment

Once the crane agent is instantiated with the environment, it is necessary for it to interact with the different components within that environment. This type of interaction is achieved through the use of sensors, where a variety of Lidar-like sensors are projected from multiple points on the crane to detect nearby objects.

Once an object is detected, its physical dimensions are saved and added to the list of observations. Additionally, the local rotation, velocity and importance is added to the list. The

importance of an object is determined through the use of Tags and Layers, where there are four main tags Human, Obstacle, and Objective, Crane. A sample of the collected data can be seen in Figure 11.

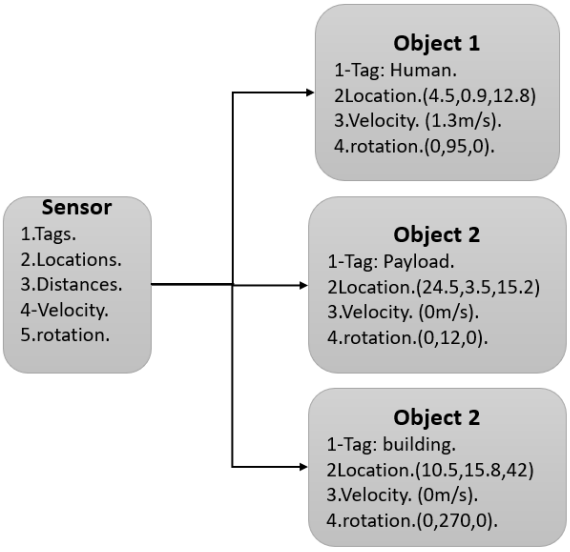


Figure 11: Interaction among sensors and the various components in the environment.

3.4.4. States

The simulation is run for a predefined number of iterations/episodes. In each episode, the location of each crane component is stored alongside the element's velocity local rotation.

In the initial runs of the simulation, and prior to adding a penalty to the collision of the crane model with the payload, the agent could move the payload by pushing it towards the set-point to gain the final reward. To penalize the suboptimal behaviour, an additional variable was added for the payload, which is a Boolean variable. The Boolean value represents whether the object is being lifted; once the object is connected to the hook, the Boolean value is turned to true.

Finally, the environment is reset to its initial configuration in three cases:

1. The episode will end if the maximum number of steps has been attained and the payload is yet to be delivered.
2. If the agent leaves the environment the episode would end.
3. If the payload is delivered to the set location, the final reward is given, and the episode is terminated.

3.5. Functions

3.5.1. Reset function

The reset function is used to restart the training variables and parameters after the end of each training episode, this function for the consistency and accuracy of the training process. This type of functions is generally used in RL problems, to prevent the agent from falling into two main problems. The first problem and the most important one is that if the agent starts each episode with a completely different set of variables, that will result in a state of completely unique training episode at each instance. This renders the training process ineffective and usually produces results that cannot be used. For instance, the agent's increased curiosity in the initial stages generally means that the agent would tend to explore the environment for an extended period of time; this would result in an agent being far away from its payload, which would then either extend the training duration or render it pointless. The second problem is that if an agent's movement and actions change the environment in a way that breaks down some of the conditions necessary for training, the agent would then continue to train with broken conditions. For instance, if the agent (in this case the crawler crane) flips the payload upside down in the first iterations, the next iterations would be incapable of lifting the payload since there is no attachment location.

Thus, the main task of a reset function is to ensure that the agent properties are restored at the start of each iteration (mainly location, velocity, angular velocity, hook placement, hook local rotation, etc.). Table 3 presents a sample of the data.

The reset function is called in a variety of scenarios. The overall code logic is presented below.

The first scenario occurs when the agent has already spent the maximum amount of time allotted for training. Prior to training, the lifting task was performed in the virtual environment, with the amount of time needed to complete the task, referred to as "maximum number of steps" (MNS), being recorded. The MNS was determined by trial and error, and this value was then stored and used as a threshold for the agent's training as a termination condition.

Table 3: Case study construction site reset properties.

Element	*x*-coordinate (m)	*y*-coordinate (m)	*z*-coordinate (m)	Angular velocity	Velocity
Crane base	2.559	−0.568	−1.052	0.000	0.000
Main boom	0.002	0.844	−0.093	0.000	0.000
Hook	−0.005	14.320	25.433	0.000	0.000
Forward sensors	0.000	−0.500	5.850	0.000	0.000
Upper sensors	0.065	11.890	25.423	0.000	0.000
Backward sensors	0.000	−0.500	−6.830	0.000	0.000
Cargo	−56.000	3.200	70.000	0.000	0.000

The second scenario is when agent changes some of the critical properties of the environment, for example when an agent collides with a building and manages to change its properties, the environment is rendered ineffective and thus requires a reset. Another example of this is when an agent collides with the payload, or worker during the lifting process. Although, the author initially toiled with making the agent incapable of making changes to the environment, it later proved more beneficial for the training process. An example of a change that the agent can produce is the destruction of the construction site.

Similar to the previous scenario, the agent is given complete control and freedom over its movements, thus it is capable of leaving the training area, which is an additional condition for the reset function.

The last scenario is when the agent has finished the task. This is done through a set of conditions, related to the location and rotation of the payload compared to those of the set location.

```
Function Environment Reset ():

Input: Maximum number of Steps, Episode Number, Environment Boundaries, Crane Data,
Payload Data, Loading Conditions.

    Ep = Ep + 1

    iteration Number = 0.

    iteration Number = Step Number +1.

    Reset Crane data.

    Reset Payload data.

    Reset loading conditions.

    If iteration number == Maximum Number of Steps.

        Reset all object properties.

    If Payload properties are changed.

        Reset all object properties.

    If Agent Leaves Training Area:

        Reset payload properties.

        Reset Crane properties.

Reset loading conditions.

    Move Payload to a new position.

    Reset Payload physical properties.

    Enable specific object's visibility.

    Set initial states.

End Function
```

3.5.2. Reward function

The reward signal selection process was the most important and complex tasks in developing this RL environment. Additionally, and in order to select the most appropriate reward signal, the simulation must have clear and concise objectives to draw from. After identifying these objectives, the reward selection is a more straightforward task since the aim is to account for the determined requirements. In the present work a variety of reward signals were considered, as discussed in the following subsections. Finally, the optimal reward signal that satisfies the objectives is discussed in section 3.5.2.6.

3.5.2.1. Sparse Rewards

The first reward signal considered is a sparse reward system. This entails offering rewards only at specific successful milestones, such as successfully connecting the payload to the hook or delivering the payload to the designated location. This can encourage the agent to focus on critical actions. The advantage of this approach from a development perspective is its ease of implementation and straightforwardness. In terms of performance, it also provides a variety of advantages, such as concentrated learning, where the agent is only focused on attaining the main milestones.

Additionally, this type of reward ensures a simple learning process. However, this type of reward could provide suboptimal results if the training environment is too complex. Also, the lack of reward-signaling rewards could confuse the agent because it would no longer have a path to follow, resulting in a slower learning curve. Additionally, the high focus on achieving the main milestones would result in a lack of exploration, leading the agent to become stuck in suboptimal solutions. Finally, the most significant disadvantage this type of reward presents is a high probability of stagnation, whereby, in the case of the agent being incapable of finding a solution, it becomes irrelevant which action to take, and, in some cases, the agent spends a significant portion of the training moving about in an endless loop. Pseudo-code representing this rewarding approach is given in Figure12.

```
Input:
  Crane initial location (Cl)
  Payload initial location (PL)
```

```
 Crane capacity (Cc), D: distance between
from crane to the object, Dps: Distance
between payload and set point.
 Payload weight (Pw)
for iteration = 1 to Number of iterations
  for iteration = 1 to max iteration
    if PL is disconnected
      Compute D
      if D > 0.5m
        Act from possible actions.
      else if D < 0.5m
        Apply connection.
        Add Lift reward.
    if PL is connected
      Compute Dps.
      if Dps > 0.5m:
        Act from possible actions.
      else if Dps < 0.5m:
        Set payload.
        Add Final reward.
  Call Reset environment ().
```

Figure 12: Pseudo code for sparse reward.

3.5.2.2. Dense rewards

The second reward signal considered is a dense reward system. This type of reward entails providing continuous rewards at each step an action is taken to increase the proximity to the goal, which incentivizes progress towards the final objective continuously. The advantage of this approach is the constant feedback the agent receives, whereby the agent would receive feedback after each action, making the learning process much smoother; this also aids the agent in determining which actions are favourable and which are not. Another major advantage of this reward system is the increased exploration, which results in a more comprehensive coverage of the training environment and the action space. Finally, the dense reward system is known to provide faster learning, and less stagnation during the training.

However, this approach has its own disadvantages such as the problem of overfitting and local optima, where the agent focuses largely on immediate rewards, leading to overfitting. The agent may also fail to satisfy the ultimate objective of delivering the payload by favouring solutions that decrease the distance between the payload and the set-point in scenarios in which the best path in fact requires drawing further away from the payload, at least temporarily (see Figure 13 . Such a scenario would present a challenge to the agent if the reward is distance-based.

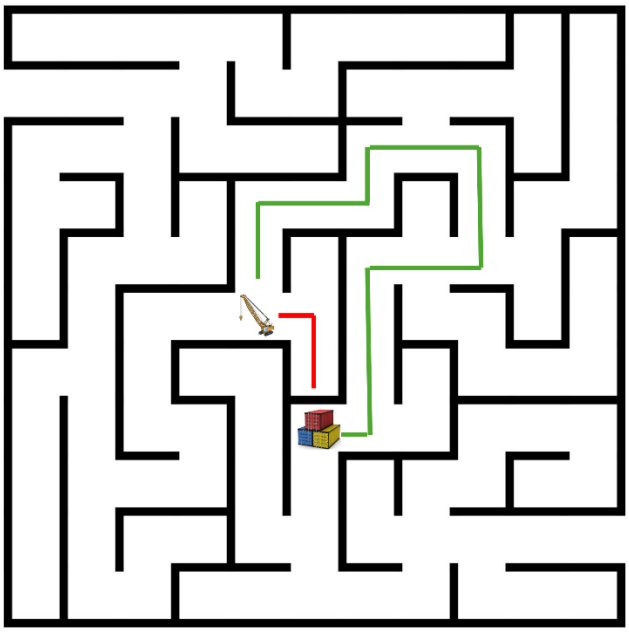


Figure 13: 2D representation of a map in which the agent must walk in the opposite direction to find the shortest path.

```
Input:
Crane initial location (Cl), Crane capacity (Cc),
Payload initial location (PL), D: distance between
crane and object, Dps: Distance between payload
and set point, Payload weight (Pw)
for iteration = 1 to the Number of iterations
   D=0, D+1 =+infinity, Dps=0, Dps+1 = infinity.
   for iteration = 1 to max iteration
      Act from possible actions.
```

```
if PL is disconnected
  Compute D
  if D < D+1
    Apply Distance reward.
      D+1 = D
  else if D > D+1
    Apply distance penalty.
    D+1 = D
  If D< 0.5:
    Apply Connection
if PL is connected
  Compute Dps.
  if Dps < Dps+1:
    Apply Distance reward.
    Dps+1 = Dps
  else if Dps > Dps+1:
    Apply Distance penalty.
    Dps+1 = Dps
  If Dps< 0.5:
    Set Load
Call Reset environment ().
```

Figure 14: Dense reward pseudo code.

In Figure 13 The agent (crane) has three possible actions, to go left, right or up. The option to go left is disregarded since the agent came from that direction and is attempting to find a short path to the target. In the case of a short-sighted policy the agent would select the red path, since it is closer to the target and would reap the rewards based on proximity to the target. However, an optimal policy would focus on long-term rewards where the green path would eventually collect a better final reward despite the initial penalized actions.

In the scenario presented in Figure 13, the agent might develop a strategy which would maximize the reward but decrease the effort, this phenomenon is well known in RL problems as reward hacking. In the previously discussed scenario, the agent could spend the majority of its time at the

closest distance to the payload in order to maximize the final reward. Pseudo-code representing the dense rewarding approach is given in Figure 14.

3.5.2.3. Penalty-based Rewards

The third possible approach to incentivize desired actions and discourage unwanted behaviour in an RL system is by implementing a penalty-based reward system. In such a system, penalties are given to the system for undesirable actions or collisions with objects and obstacles. For instance, if the system collides with critical components of the environment, it will receive a penalty, while successful actions would still be rewarded. This approach offers several benefits, such as increased safety by training the agent to avoid collisions and obstacles and providing constant feedback for effective learning. However, a penalty-based approach also has some drawbacks, which include the risk of over-penalizing the system, thereby preventing it from exploring certain areas of the environment and missing out on optimal solutions. Additionally, the penalty-based approach may only provide sparse feedback for positive actions, which may not encourage overall positive learning. To address these downsides, this approach could be combined with other methods to improve overall performance. For a more detailed breakdown, refer to Figure 15 Error! Reference source not found. for the pseudo code of this approach.

```
Input:
 Crane initial location (Cl)
 Payload initial location (PL), D: distance between
crane and object, Dps: Distance between payload
and set point
 Crane capacity (Cc)
 Payload weight (Pw)
for iteration = 1 to Number of iterations
  for iteration = 1 to max iteration
    if PL is disconnected
      Compute D
      if D > 0.5m
        Act from possible actions.
        check for collision.
```

```
        if collision is true
          Apply collision penalty.
      else if D < 0.5m
        Apply connection.
        Add Lift reward.
    if PL is connected
      Compute Dps.
      if Dps > 0.5m:
        Act from possible actions.
        check for collision.
        if collision is true:
          Apply collision penalty.
       else if Dps < 0.5m:
        Set payload.
        Add Final reward.
  Reset environment.
```

Figure 15: Pseudo code for penalty-based reward.

3.5.2.4. Time-based Rewards

Similar to the dense reward approach, there is an approach which relies on the amount of time the agent spends in the environment. This approach focuses on encouraging the agent to complete the required tasks within a time frame. It also incentivizes the agent to finish tasks in a short period of time and penalizes the actions that prolong the overall training procedure. The advantages of using time-based rewards are many, for example it provides efficient result in which the agent finds the shortest possible path, which in turn improves resource utilization and energy efficiency. It also produces paths which are less redundant and easier to implement (thereby overcoming some of the issues often encountered in path planning automation processes).

```
Input:
  Crane initial location (Cl)
  Payload initial location (PL), D: distance between crane
and object, Dps: Distance between payload and set point
  Crane capacity (Cc)
```

```
 Payload weight (Pw)
for iteration = 1 to Number of iterations
  for iteration = 1 to max iteration
    Act from possible actions
    Apply Time penalty.
    if PL is disconnected
      Compute D
       if D < 0.5m
         Apply connection.
         Add Lift reward.
    if PL is connected
      Compute Dps.
      if Dps < 0.5m:
         Set payload.
         Add Final reward.
  Reset environment.
```

Figure 16: Pseudo code for time-based reward.

However, the time-based approach might produce lower quality paths since it prioritizes task completion over accuracy and quality. Additionally, since the agent learns through the reward system, it becomes hellbent on finishing the task as fast as possible while neglecting the premise of exploration, which in turn might result in ignoring some better alternatives. Also, safety could present an issue for this approach, where in many cases mobile cranes are used in congested construction and industrial areas.

3.5.2.5. Stability-based Rewards

The stability-based rewards signal is heavily reliant on assessing the internal rotations of the crane as well as the payload during the training phase. Where the agent (Crane) moves through the environment and at each step, the *x*- and *y*-rotations of the crane are assessed and compared with a threshold value. The threshold value is chosen based on a trial-and-error approach.

In contrast, if encountered on the construction site, the maximum threshold is expected to cause a significant incident. This approach provides a large emphasis on safety in its practices. For a more detailed breakdown, refer to Figure17 For the pseudo-code of this approach.

```
Input:
  for iteration = 1 to max iteration
    Act from possible actions.
     Check Crane Velocity.
     Check Crane Rotations (Z, X).
     If Min rot <Crane Rotation (Z, X) <Max rot
        Apply Stability Penalty.
    Else if Crane Rotation (Z, X) > Max rot
       Apply Tip over Penalty.
      Reset environment.
     if PL is disconnected
       Compute D
         if D < 0.5m
           Apply connection.
           Add Lift reward.
     if PL is connected
       Compute Dps.
     If Min rot <Payload Rotation (Z, X) <Max rot
       Apply payload Stability Penalty.
    Else if Crane Rotation (Z, X) > Max rot
      Apply payload Tip over Penalty.
     Reset environment.
      if Dps < 0.5m:
          Set payload.
          Add Final reward.
  Reset environment.
```

Figure 17: Pseudo code for stability-based reward.

3.5.2.6. Mixed Approach

While the agent is training, it is essential to select the appropriate reward; this task is perhaps the most critical. Moreover, selecting a reward that balances penalties and rewards is necessary. For instance, if the agent is being over-penalized, the behaviour resulting from the training would be suboptimal. For the present work, various rewarding strategies were explored, and the final

selected strategy was selected based on its enhanced performance in the final training results.

In order to attain the previously determined objectives, which are the optimization of the lifting time, energy consumption, and collision detection, three different types of rewards were built into the training process. The first reward is related to the lifting time, which is highly affected by the number of actions taken and the overall time needed to deliver the payload to its set location. Furthermore, each action has its parameters, such as movement speed, damping ratio, and interaction with the previous movement. For instance, when the crane is moving, no other action can be taken until the crane is at a complete stop. All the previously mentioned parameters are combined to calculate the time needed to perform the lift, and the subsequent reward signal associated with the movement is developed to penalize the agent for each time spent lifting and transporting the object. This penalty ensures that the resulting lift uses the shortest lifting time. The penalty amount was selected after multiple experiments.

Table 4: Summary of rewards and penalties.

Task	**Reward**
Attach Hook	+100
Set payload	+200
Take action	−0.01 × per frame
Distance	−0.01 × distance ÷ maximum distance
Leave training area	−100 + end episode
Collision	−1 × collision type
Stability	−0.01 × instability type
Critical Stability	−10 + end episode

The second reward signal used is related to collisions or leaving the training area. The objective of the training is to discover all of the possible alternatives that can be used to perform a lift. It was decided that the agent would leave to explore the whole construction site. However, in some instances, the agent elects to leave the training area; in those instances, a significant penalty is used to deter the agent from leaving the area. Furthermore, the episode is instantly terminated since,

based on the author's experience with the model, the agent does not find its way back to the payload once it has explored the extremities of the training area.

```
Input:
 Crane initial location (Cl)
 Payload initial location (PL), D: distance
between crane and object, Dps: Distance
between payload and set point, Crane
capacity (Cc),Payload weight (Pw)
for iteration = 1 to Number of iterations
  for iteration = 1 to max iteration
    if PL is disconnected
      Compute D
      if D > 0.5m
        Act from possible actions
        Apply Move penalty × D.
        check for collision.
        if collision is true
          Apply collision penalty.
      else if D < 0.5m
        Apply connection.
        Add Lift reward.
    if PL is connected
      Compute Dps.
      if Dps > 0.5m:
        Act from possible actions
        Apply Move penalty × Dps
        check for collision.
        if collision is true:
          Apply collision penalty.
      else if Dps < 0.5m:
        Set payload.
```

```
        Add Final reward.
    Reset environment.
```

Figure 18: Pseudo code for mixed approach.

Additionally, using the same approach, the agent sometimes collides with the surrounding obstacles and buildings despite the built-in sensors. An additional penalty is added based on the collision tag in those instances. The episode is terminated in some instances, when the agent collides with either a human or a building. The collisions with humans function allows the crane agent to avoid workers on its own. The other collisions are only penalized to allow the agent to train for an extended period.

The third type of reward is related to the type of action taken, where specific actions are preferred. For instance, since pick-up and walk operations are less favourable due to their increased cost, crane movements are more penalized than boom and hook movements. Moreover, main boom movements are expected to consume more energy than hook movements (based on expert opinion). Thus, they are penalized more than hook movements. Finally, hook movements are given the lowest penalty.

Finally, the main reward signals for the crane are those related to lifting and delivering the payload. The first portion of the reward is connecting the payload to the hook; once the agent lifts the payload, it receives an enormous reward. The second portion relates to the agent setting the payload in the set place where the final reward is given, and the episode is terminated. If the agent lifts the payload and fails to deliver it, the episode is terminated, and the environment is reset. All of the rewards incorporated in the training process are described in Figure18.

3.6. Curriculum definition

In order to execute the CL approach, it is necessary to establish a learning path of increasing difficulty, which the crane agent can use to improve its understanding of the environment and learn its policy. Additionally, the agent must achieve two different tasks to perform a successful lift. The first task involves finding the payload and attaching it to the hook. The second task with the highest reward is related to delivering the payload to the final set-point.

The secondary rewards related to lifting time, obstacle avoidance, and stability are also present but are not specified in the curriculum.

The learning approach was divided into nine different steps beginning with an easier learning scenario in the first step and then becoming increasingly difficult; the information related to the components relevant to the agent's learning for each of the steps are described in Table 5.

Table 5: Curriculum learning branches.

Parameters	**Crane Parameters (m)**	**Hook parameters (m)**
Step 1	Crx = (−54.99), Cry = (1.17), Crz = (49.00)	Hx = (−54.99), Hy = (3.20), Hz = (25.41
Step 2	Crx = Uniform (−54.99, −49.99), Cry = Uniform (1.17,1.176), Crz = Uniform (49.00, 49.00)	Hx = Uniform (−54.99, −49.99), Hy = Uniform (3.20,8.20), Hz = Uniform (25.41,25.41)
Step 3	Crx = Uniform (−44.99, −49.99), Cry = Uniform (1.17,1.176), Crz = Uniform (49.00, 49.00)	Hx = Uniform (−44.99, −49.99), Hy = Uniform (13.20,8.20), Hz = Uniform (25.41,25.41)
Step 4	Crx = Uniform (−44.99, −39.99), Cry = Uniform (1.17,1.176), Crz = Uniform (49.00, 49.00)	Hx = Uniform (−44.99, −39.99), Hy = Uniform (13.20,18.20), Hz = Uniform (25.41,25.41)
Step 5	Crx = Uniform (−34.99, −39.99), Cry = Uniform (1.17,1.176), Crz = Uniform (49.00, 49.00)	Hx = Uniform (−34.99, −39.99), Hy = Uniform (23.20,18.20), Hz = Uniform (25.41,25.41)
Step 6	Crx = Uniform (−34.99, −29.99), Cry = Uniform (1.17,1.176), Crz = Uniform (49.00, 49.00)	Hx = Uniform (−34.99, −29.99), Hy = Uniform (23.20,27.20), Hz = Uniform (25.41,25.41)
Step 7	Crx =Uniform (−24.99, −29.99), Cry = Uniform (1.17,1.176), Crz = Uniform (49.00, 49.00)	Hx = Uniform (−24.99, −29.99), Hy = Uniform (23.20,27.20), Hz = Uniform (25.41,25.41)
Step 8	Crx = Uniform (−24.99, −19.99), Cry = Uniform (1.17,1.176), Crz = Uniform (49.00, 49.00)	Hx = Uniform (−24.99, −19.99), Hy = Uniform (23.20,27.20), Hz = Uniform (25.41,25.41)

Step 9	Crx = (−19.99), Cry = (1.17), Crz = (49.00)	Hx = (−19.99), Hy = (27.20), Hz = (25.41)

where:

Crx, Cry, Crz: Crane *x*-coordinate, Crane *y*-coordinate Crane *z*-coordinate.

Hx, Hy, Hz: hook *x*-coordinate, hook *y*-coordinate, hook *z*-coordinate.

3.7. Hyperparameter optimization

In order to train the agent efficiently there has to be an ideal configuration of variables and parameters that work together to ensure the smoothest learning process. In this section, some of the main parameters necessary for training are described, while the ranges for which they explored are outlined in Table 6.

Table 6: Hyperparameters considered for optimization.

Parameter	Range	Suggested value	Optimal value
Maximum no. of training episodes	[2 M, 10 M]	2 M	5 M
Initial learning rate	[1e−5, 1e−2]	1e−3	5e−2
No. of hidden units for reward-based signals	[64×3, 512×3]	128×3	512×3
No. of hidden units for reward-based signals	[32×1, 64×3]	64×2	32×3
Behavioural cloning strength value	[0.01, 0.50]	s0.1	0.1
GAIL strength value	[0.01, 0.50]	0.1	0.1
Demonstration numbers	[1, 5]	5	3

3.7.1. Maximum number of training episodes

Perhaps the most basic, yet the most important parameter to determine prior to starting the training process is to determine how long would the training process take. This is determined through identifying the complexity of the given task and assessing the amount of time necessary to determine the optimal time, and the most computationally efficient value for the final training process. In the present case, since the model contains 5 continuous actions, and 13 total

observations, it was determined that the number of episodes to be used would range between 5 million and 10 million iterations. This value was determined by trial and error after assessing the learning process of the agent based on the initial training, which can be seen in Figure 19.

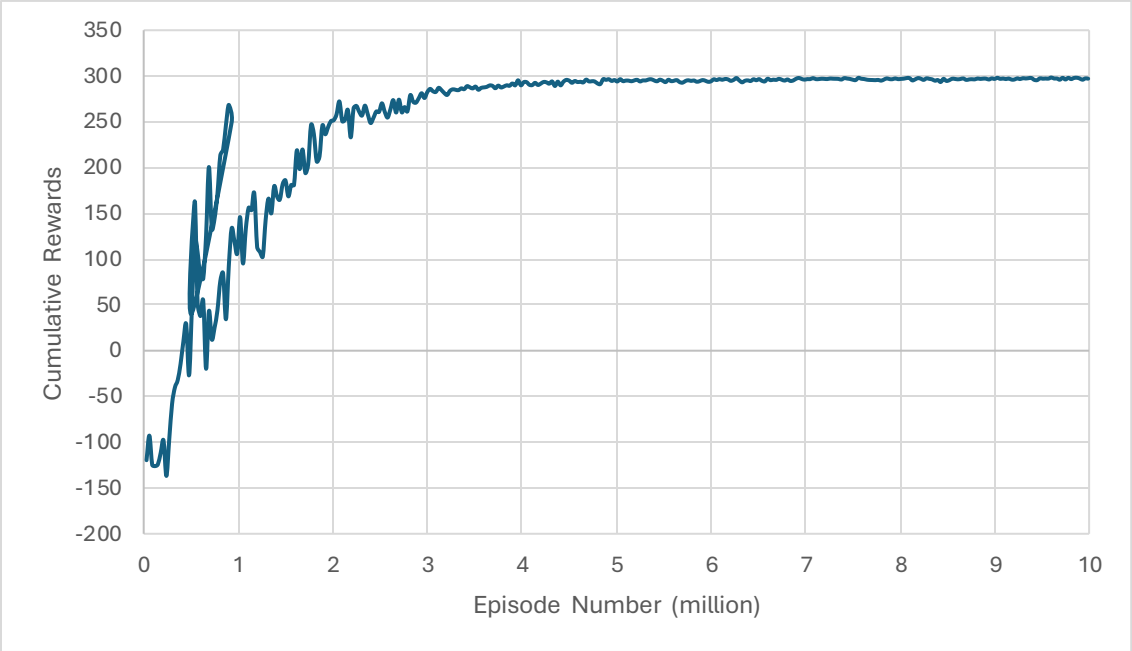


Figure 19: Agent's learning process in response to the number of training episodes.

A preliminary model was used to assess the number of training episodes needed; the preliminary model is a simpler model containing most of the parameters that would be a part of the final model, except for the detailed construction environment. Figure19 shows that in the first 3 million episodes, the agent is progressively learning. It managed to improve its results from a negative reward to approximately 275 points by the 3 millionth episode. The reward signal is based on a 300-point scale, as discussed in detail in the rewards portion 3.5.2 below. Although 275 points present good behaviour, improvements can be implemented to reach a more optimal behaviour. In the subsequent 2 million iterations, the agent manages to reach 295 points and spends the last 5 million episodes improving the results until it reaches a reward signal of 298 points. Although the last 5 million episodes improve the behaviour slightly, it can be expected that the optimal solution will always be determined in that range of episodes. Due to its complexity, more than 10 million episodes could be required in the final training environment. However, this portion of the work allows us to understand the time the agent needs to learn the policy efficiently, which can be increased if the training is insufficient.

3.7.2. Initial learning rate

This parameter determines how the model's parameters are updated after each iteration. The selection process of this parameter could determine the convergence speed of the learning process, as well as the stability of the learning process. For instance, when a high initial rate ratio is selected, the convergence rate is improved significantly. However, this comes with the risk of destabilizing the convergence process due to overshooting the minimum loss value. Overshooting could have a detrimental impact on the final solution, where a possible divergence from the optimal solution can occur, as well as failure to converge. Furthermore, the increased time to converge would also increase the computational power and time needed to reach a final solution.

Thus, finding the optimal balance between convergence speed and stability is essential. Additionally, the learning rate has a significant impact on the model's overall performance, where it could affect the agent's behaviour between quality-driven and high reward-driven approaches. In the present case, it was observed that, with a high learning rate, the agent is more likely to focus on exploiting the knowledge collected without exploring the environment extensively (in contrast with the case of a lower learning rate). Finally, the type of environment and the level of complexity of the environment could influence the learning rate selection process. Simple environments could benefit from quick convergence, whereas more complex ones yield a lower convergence rate. For the present study, a range of [1e−5, 1e−3] was tested to obtain the best parameter value for the agent. The analysis of each alternative is highlighted in Figure 20.

In Figure 20 both learning rates (i.e., 1e−5 and 1e−4) were tested; for the 1e−5 learning rate, the agent was found to be incapable of learning; although the agent was given 5 million iterations, the agent continued to produce negative rewards. Moreover, although the agent occasionally managed to finish the first task of the training, which was to hook the payload, the agent never capitalized on those iterations; furthermore, since the training process requires the agent to repeat an action multiple times before learning it. This agent should have learned the first task within the minimal number of training actions necessary. Moreover, since the agent could only finish the first task occasionally, it has yet to reach the second task, which is to deliver the payload to its final destination.

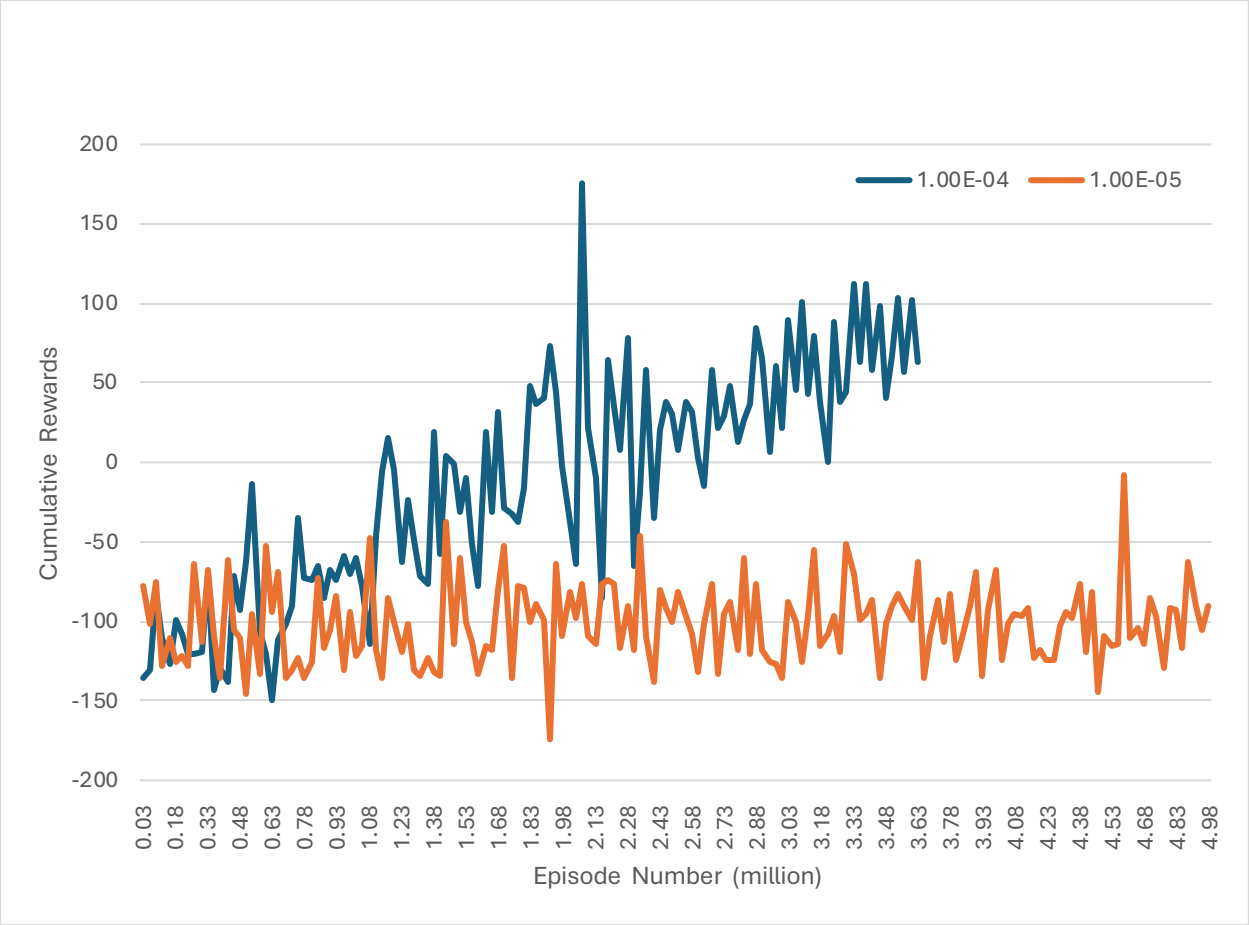


Figure 20: Impact of initial learning rate on overall learning performance of crane agent part 1.

As for the 1e−4 learning rate, the agent improved its learning rate significantly, where it managed to obtain positive rewards within 2 million episodes; the positive rewards did not surpass 100 points, which suggests that the agent only managed to learn the first task. Also, the agent needed to be more consistent in producing such a performance, even a perfect 100-99 point score, and only managed to produce a suboptimal method to achieve the first task. Although. On some occasions, the agent managed to reach the final task but did not repeat the task enough times to learn it.

For the next portion of the analysis, dividing the curves into three phases would help understand each parameter's performance. The first phase is the initial exploration phase, where the agent is not producing any lifts and is consistently producing fluctuating within a negative reward range. The second phase is exponential learning, where the agent learns a good policy and obtains near-

optimal solutions. The last phase is the optimization plateau phase, where the agent starts optimizing the discovered solution to ultimately reach the optimal solution.

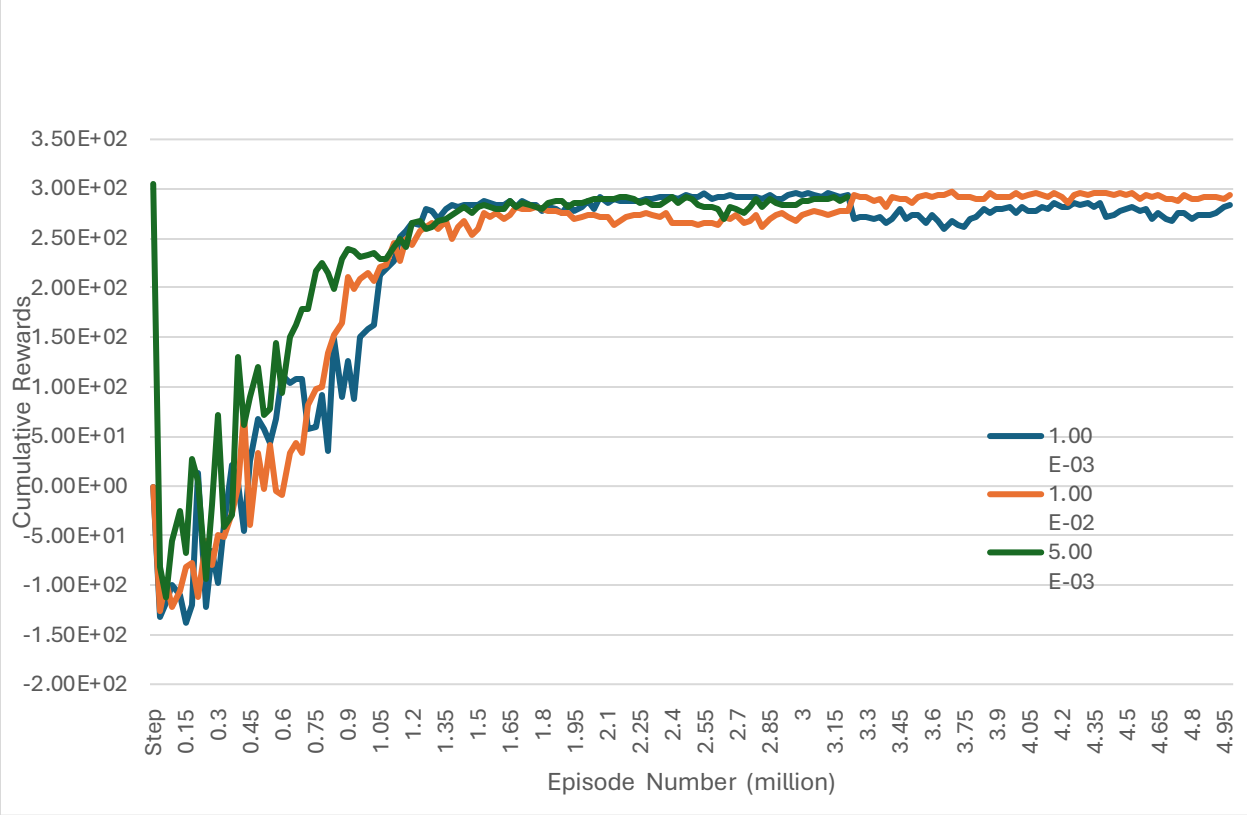


Figure 21: Impact of initial learning rate on overall learning performance of crane agent part 2.

It was necessary to increase the learning rate; in Figure21. it can be observed that using a 1e−2 as a learning rate produces excellent results. The agent was capable of finishing phase one within 0.5 million episodes. The agent was also found to be capable of achieving the first task consistently within 0.8 million episodes, which is a significant improvement. By around the 1.5 millionth episode, the agent had obtained 262.42 points and thus successfully finished the exponential learning phase and entered the final phase. However, the rest of the episodes did not accelerate the learning process significantly, with the agent only managing to obtain a maximum of 279.65 points around the 1.75 millionth episode. The subsequent 2 million episodes did not produce a better solution; only around the 4.5 millionth episode did the agent obtain a reward of 284.48 points, resulting from the optimization phase.

On the other hand, after exploring the 1e−3 learning rate, the agent finished the first phase within 450,000 episodes. Additionally, the first primary operational task was achieved consistently within

0.69 million iterations, comparably a faster convergence than the 1e−2 rate. In the next portion of the training, the agent performed poorly compared to the 1e−2 learning until the 1.15 millionth episode, where both parameters produced similar results. At around the 1.5 millionth episode, the agent reached 279 points and finished the exponential learning phase, meaning that performance here was better than in the case of the 1e−2 parameter. Finally, and within the optimization phase, the 1e−3 model performed consistently better than the 1e−3 model and reached a maximum reward of 294.28 points.

Finally, since both models, the 1e−3 model, performed significantly better than the other models, the authors wanted to explore an intermediary value that could produce better results, and thus, it was deemed that the 3e−5 learning rate would be tested. In Figure 21, the results clearly outline the improvements in the overall performance; the agent has successfully finished the exploration phase within 390,000 iterations, faster than all of the other models. Additionally, the agent was capable of finishing the first task consistently within 600,000 episodes, and it finished the exponential learning phase around the 1.5 millionth episode, similar to the other models; however, throughout the exponential phase, it managed to produce better results in all of the episodes. Finally, although the results were similar to those seen in the 1e−3 model, the agent could achieve slightly better training results of 294.91 points, which suggests that this parameter is better suited to the complexity of the crane training model.

3.7.3. Number of hidden units for reward-based signals

The reward level of complexity significantly influences the method the agent uses to learn its policy, which means that the agent's brain—or, in other words, the neural network architecture—must change accordingly. For this particular problem, the architecture of the neural network responsible for generating rewards based on curiosity also decides the agent's level of exploration and effectiveness. For the present work, several rewards-based hidden units between 32 and 512 and a range of layers between 1 and 3 were explored. The analysis of these results is presented in Figure 22 and summarized in Table 7.

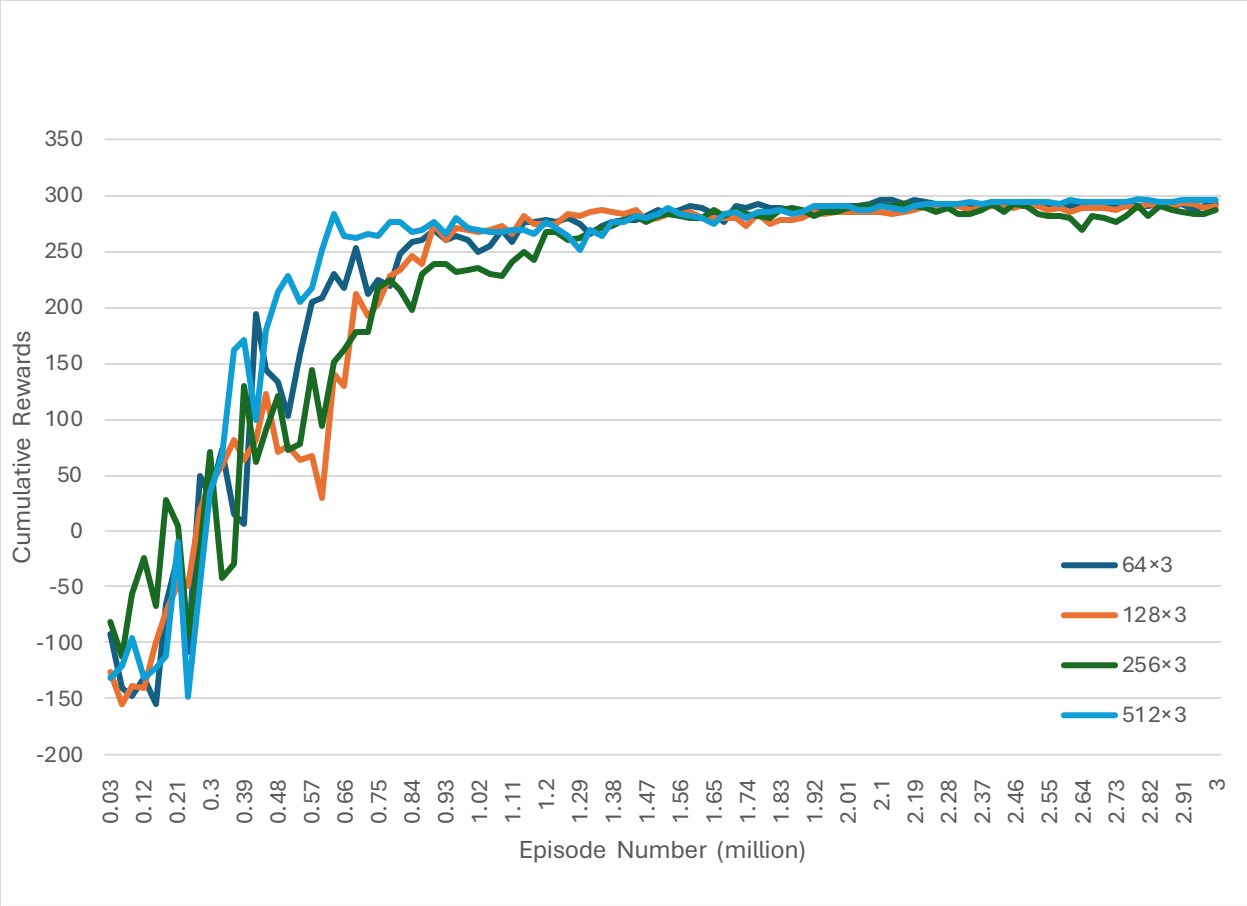


Figure 22: Number of hidden units representing the impact of reward-based signal on agent learning.

Based on the results presented in Figure 22 and Table 7, the models' performance is very similar. However, when taking a closer look, it can be seen that the 512 hidden layer-based model performs best. For instance, the 512-hidden-layer model finished the initial exploration phase around the same time as the other models. However, it could finish the first task at around 390,000 episodes. In contrast, the other models finished the first task at 450,000–660,000 episodes, where the 128 hidden-layer model performed the most poorly, followed by the 256- and 64-hidden-layer models. The second task also proved the effectiveness of the 512-hidden-layer model, where the agent managed to finish the second task in 840,000 episodes, effectively finishing the exponential learning phase. The 64,128 and 256 models required more time to finish the second phase, where most agents took between 1.1 million and 1.2 million episodes to learn the second task. Finally, for the whole learning process overall, it was observed that the 512 models produced the highest

reward of 297.24 points, followed by the 128 models with 297.08 points, and then the 64 and 256 models with 294.52 points and 294.92 points, respectively.

However, another factor to consider is the in-environment training time, where the 256 and 512 models took the most time to produce the required results (1.44 and 1.06 days, respectively). The models using fewer hidden units took between 22.4 h and 20.95 h each, meaning that there was less of a computational burden. Nonetheless, it can be concluded that the model featuring 512 hidden layers is the most favourable.

Table 7: Hidden units for assessing impact of reward-based signals on performance.

Model config. (neurons × number of layers)	NE to achieve initial exploration phase	NE to consistently achieve Task 1	NE to achieve the exploration phase	NE to achieve the best lift	Best lift reward
64×3	300,000	420,000	960,000	2,130,000	294.5272
128×3	300,000	630,000	1,380,000	4,740,000	297.0862
256×3	330,000	570,000	1,680,000	4,410,000	294.9142
512×3	330,000	390,000	1,530,000	4,770,000	297.2379

3.7.4. Number of hidden layers

In RL problems, the use of neural networks can help map the connections between the observations and actions. This mapping represents the policy used by the agent to learn the best decisions. The agent collects information through the interactions it has with the environment, and this knowledge is then used by the neural network to improve the policy and the decision-making process. In other words, the neural network serves as the brain for the agent, determining the best strategy for learning. One of the parameters that is essential to improve training is the size of the neural network, which is selected by determining the number of layers as well as the number of units in the hidden layers. Generally, a complex problem requires a more extensive network of neurons. For the purpose of the present study, a number of hidden layers were explored as well as a range of layers between 1 and 3. The analysis of these results is described in Figure23.

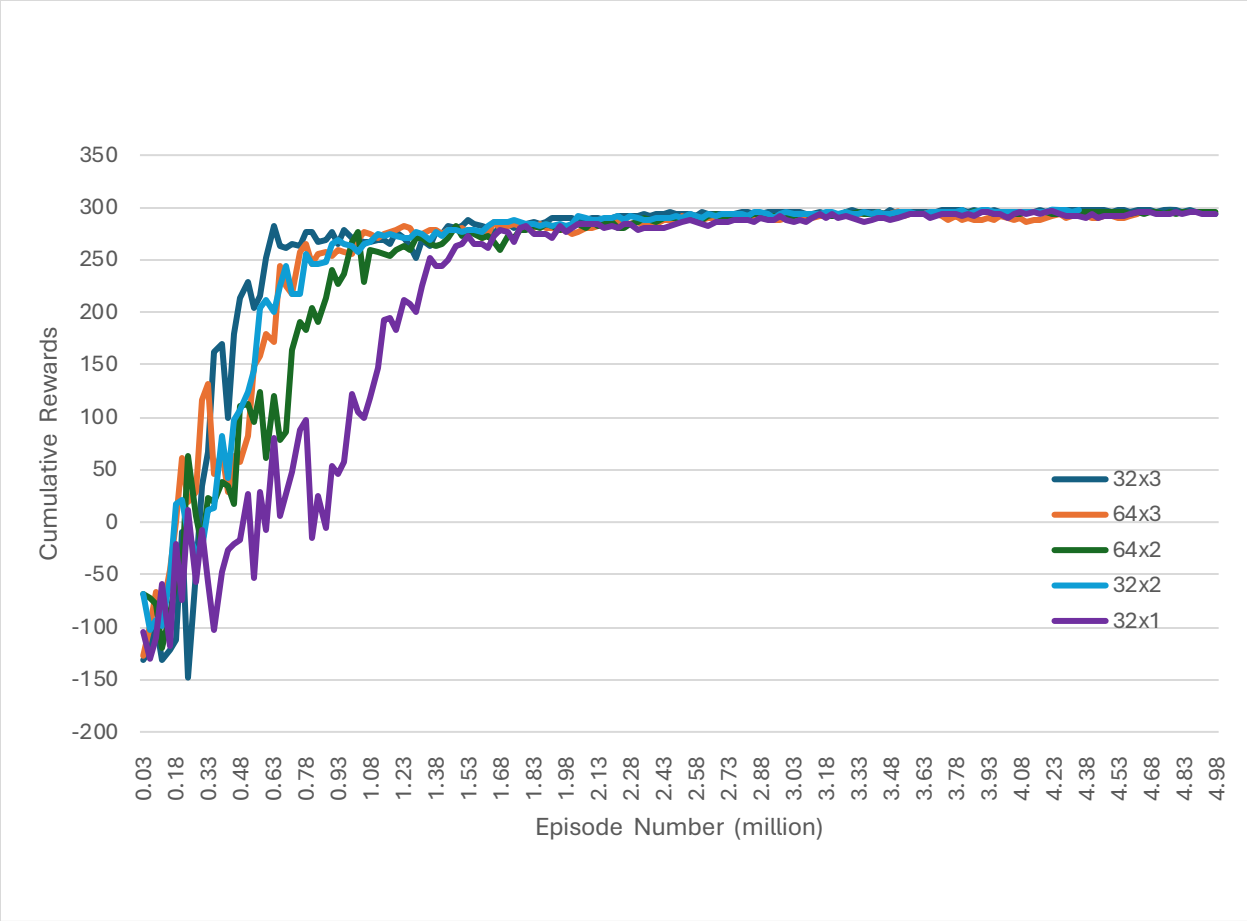


Figure 23: Impact of the number of hidden layers and overall learning performance of crane agent.

Based on the results presented in Figure23. and Table 8, the models' performance is very similar. However, when taking a closer look, it can be seen that the 32×3 hidden layer-based model performs best. The 32×3 hidden-layer model reached the initial exploration phase around the same time as 64×2, it was only beaten by the 64×3 model which only needed 210,000 episodes to reach the initial exploration phase. However, the 32×3 hidden layer-based model could finish the first task at around 390,000 episodes which is much faster than all of the other model, due to the significant learning obtained in the 60,000 episodes. In contrast, the other models finished the first task at 510,000 episodes and 1.08 million episodes, where the 32×1 hidden-layer model performed the most poorly, followed by the 64×2 and 64×3 hidden-layer models. The second task also proved the effectiveness of the 32×3 hidden-layer model, where the agent managed to obtain the highest reward of 297 points, where all of the other models maximized their rewards at 295–296 points. The 32×2, and 32×1 obtained total rewards in the upper range of 296 as opposed to the other

models which performed less favourably. Thus it can be concluded that the 32×3 model is better suited for the final training.

Table 8: Model performance per configuration and number of episodes.

Model config. (neurons × number of layers)	NE to achieve initial exploration phase	NE to consistently achieve Task 1	NE to achieve the exploration phase	NE to achieve the best lift	Best lift reward
32×1	570,000	1,080,000	1,800,000	4,230,000	296.800
32×2	360,000	510,000	1,290,000	4,260,000	296.978
32×3	330,000	390,000	1,530,000	4,770,000	297.238
64×2	330,000	720,000	1,500,000	4,560,000	296.032
64×3	210,000	540,000	1,230,000	3,510,000	295.011

The overall neural network architecture is presented in Figure 24, where there are three input layers—the observation layer, the state layer, and the reward signal layer. The input layers feed into the hidden layers, ranging between 1 and 3 layers with a number of neurons that could range between 32 and 64 for each layer. Finally, the output layer is the action layer, which decides on how to move the crane parts to maximize the rewards.

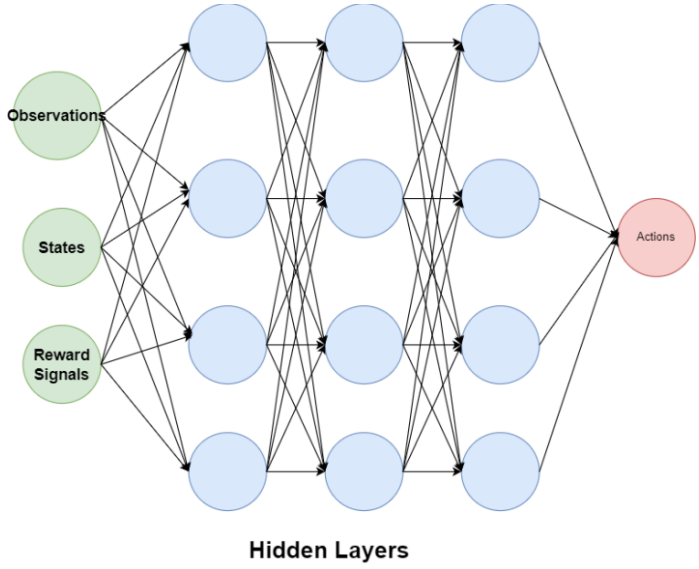


Figure 24: Neural architecture of the model.

4. CASE STUDY AND RESULTS

4.1. Overview of results

The results chapter is divided into four sections; each section will discuss the details related to one of the simulations approaches previously explained in the methodology section:

- The pure reinforcement learning (RL) approach.
- The generative adversarial imitation learning (GAIL) approach coupled with RL.
- The GAIL approach coupled with behavioural cloning and RL, referred to as the RL mixed approach.
- The curriculum learning (CL) approach.

Each approach will show results related to the cumulative reward value per episode, episode length, and the curiosity value estimate when curiosity-based training is used. In conclusion, a general summary of the results is presented, along with a discussion of the advantages and disadvantages. Additionally, implementation strategies for each approach are suggested accordingly. To effectively understand and analyze the results of each approach, simple terminology is displayed to explain the different phases that an RL agent would go through during its training. These phases can be seen in Figure 26.

4.1.1. Case study overview

The main objective underlying the case study is to automate the path planning process, and this consists of transporting a set of payloads from a pick-point to a set-point while avoiding collisions of any sort, specifically those with irreplaceable elements. Generally, the transportation process comprises three steps. The first step is to locate the payload in its original state, determine the shortest path towards it, and lower the hook to a certain distance, later allowing the workers to install the necessary rigging components and successfully attach the payload to the hook. This attachment phase is done through a set of combinations, which are highly dependent on the type of load being transported, its physical properties, and the type of crane being used for transportation purposes.

After payload attachment comes the most crucial task of determining the optimal path between the pick- and set-points. The definition of an optimal path is that the path must minimize the airtime, that the load must be close to the ground at all times, except when no other alternative can be found

as well as the emphasis on minimizing crane walking, which requires a set of arrangements that would increase the cost significantly.

For the case study, a simulation of a large-sized construction site was used; it comprises three constructed buildings, as seen in Figure 25. Each building has its own scaffolding set, which is considered when performing the lift. Additionally, to the buildings, other equipment is being used in the construction site. Mainly a tower crane, which is near building number 2, as well as excavators. Each of these equipment is given a high priority in collisions, meaning that the agent is heavily penalized if any collisions occur with this equipment among other portions of the environment. Another critical component of the construction site is the presence of different materials and containers, as seen in Figure 25.

Additionally, a variety of obstacles are added to the construction site, such as construction materials, holes, and barricades, which all have tuned collision parameters. The agent is expected to learn their locations and avoid them during the lifting process.

Figure 25: 3D model for case study.

The construction site exhibits a complex arrangement of its different components. The placement and distribution of each element was carefully selected to simulate congestion and tight spaces. As a result, the path planning process becomes more challenging for the agent training, but it provides a more realistic training ground. A sampled breakdown of the coordinates of all the critical components of the construction site is presented in Table 9. This table also includes the relevant geometric and physical features of each component.

Table 9: Construction site data sample.

Element	*x*-coordinate (m)	*y*-coordinate (m)	*z*-coordinate (m)	Rotation (degrees)
Building 1	27.000	24.000	−25.750	−90.000
Building 2	−30.750	32.250	−15.750	0.000
Building 3	−79.500	64.000	44.000	0.000
Mobile Crane	−54.790	1.500	48.680	0.000
Tower Crane	4.000	43.000	−13.000	0.000
Cargo	−56.000	3.200	70.000	0.000
Set Location	−54.000	3.200	15.000	0.000
Forklift 1	−30.769	0.127	19.123	19.237
Forklift 2	8.880	0.182	49.850	−204.819
Cargo container 1	−1.640	2.800	56.940	187.340
Cargo container 2	1.450	2.800	58.290	178.336
Cargo container 3	2.960	0.185	57.530	169.655
Storage Pipe	−34.750	0.204	37.750	0.000
Entrance	−50.139	0.194	71.477	−26.954

4.2. Pure Reinforcement Learning approach

4.2.1. Overview

In this section, the emphasis is on the results obtained purely through RL. To provide a basis for comparison, the results of the hyperparameter optimization environment are first presented. Both the hyperparameter model and the case study are composed of 4 main sub-sections, which are discussed in detail. These sub-sections consist of the cumulative reward and episode length. As for the results related to the policy, three indicators are used: extrinsic value estimate, curiosity value estimate, and the curiosity reward section.

4.2.2. Hyperparameter model results

The hyperparameter model is the test model discussed in section 3.2.2. It contains simple lifting problems without obstacles. Prior to running the case study, all of the performance was tested in the hyperparameter model, which is discussed in this section.

4.2.2.1. Cumulative Rewards

This section presents the findings with respect to the implementation of RL for mobile crane path planning. Hyperparameters were optimized for the case study as discussed in detail in section 3.2.2. Figure 26 presents the cumulative rewards achieved per episode for the case study. During the first 0.24 million episodes, the agent had some challenge in developing a policy to solve the lifting problem. As a result, the initial episodes mainly consisted of repetitive interaction between the agent and environment. Using these interactions, the agent managed to reach a positive reward of 6.69 points within 0.33 million episodes, as shown in Figure 26.

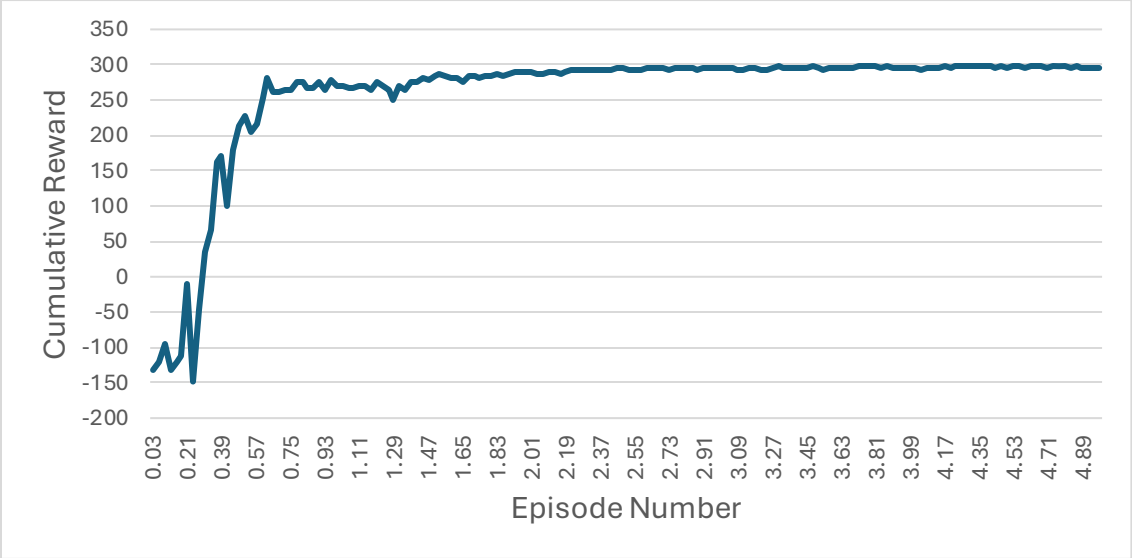


Figure 26: Pure reinforcement learning approach for hyperparameter optimization model.

In the hyperparameter model, an agent was trained to find the best sequence of actions to maximize rewards while maintaining stability, and consistently collecting rewards. The first major task the agent needed to accomplish was to find and attach a payload to a loading hook. The agent successfully avoided all obstacles within 0.33 million episodes and then moved on to the primary task of finding and attaching the payload. The reward for successfully attaching the payload was 100 points, and the agent was able to master this task within 0.39 million episodes with a cumulative reward of 99.4 points on average. This means the agent was capable of improving the rewards collection process by approximately 100 points within 0.06 million episodes. The study also observed that the agent tended to complete the exploitation phase, which involved efficiently discovering the environment, within 0.78 million points, with a total cumulative reward of 276.6

points. The agent demonstrated significant improvement in the optimization portion of the training, and this optimization phase starts from episode 0.76 million till episode 5 million. The agent could improve the results from 276.6 points in episode 0.78 million to a reward of 297.24 points in episode 4.77 million, with an overall improvement of approximately 11% or a total of 21.64 points. The improvements encountered in the hyperparameter optimization were less significant in the last 1.2 million episodes. The agent was capable of attaining a total reward of 296.8 in episode 3.77 million, whereas the rest of the episodes only provided marginal improvements. Since the hyperparameter model was only tested for the purposes of obtaining the best hyperparameters, 5 million episodes was sufficient (whereas a larger number of episodes could be used in the final model).

4.2.2.2. Episode length

In RL, episode length is the number of steps or actions taken from the start to the end of an episode, where an episode is a complete sequence of interactions between an agent and the environment ending in a terminal state (Sutton & Barto, 2018)(Sutton & Barto, 2018)(Sutton & Barto, 2018)(Sutton & Barto, 2018)(Sutton & Barto, 2018). This section presents an analysis that focuses on the episode length criterion for evaluating the lift path planning tool used by the crane agent. The primary goal is to minimize episode length to find the shortest, safest, and most efficient path. The effectiveness of the lift is evaluated using the "number of actions" metric, which measures the agent's task efficiency by penalizing actions that prolong execution time. Therefore, the emphasis is on reducing the number of actions required to accomplish the task.

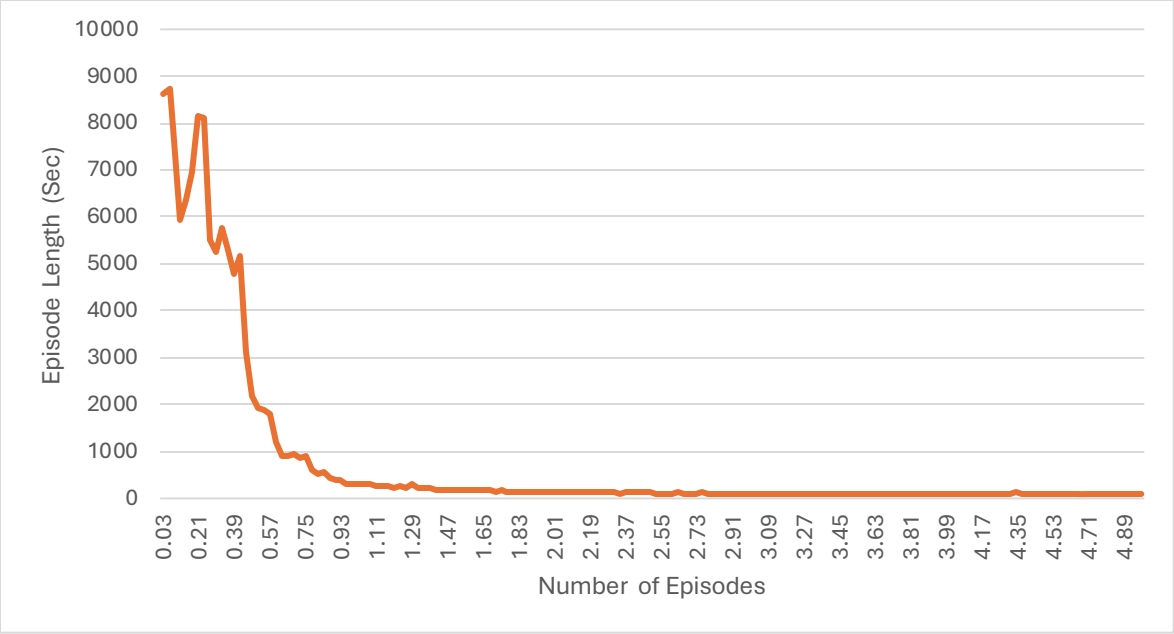


Figure 27: Episode lengths for hyperparameter optimization model

The subsequent discussion outlines the impact of the pure RL approach on episode length and the overall progression concerning the number of actions taken per episode in the hyperparameter optimization model (Figure 27). Unlike the cumulative reward graph, the episode length graph displays a distinctive trend in which the agent's objective is to decrease the episode length. Initially, within the episode length graph, the number of actions steadily decreased until approximately the 0.24 millionth episode, where the episode length was 8,070 s. This phase predominantly aligns with the exploration phase seen within the same number of episodes in the cumulative rewards section. The following episodes provide significant improvement in episode length where the agent manages to decrease the episode length from 8,070 s to 225.5 s within the next 1.08 million episodes: this improvement correlates closely with the exploration phase present in the cumulative rewards graph. Finally, the improvement rate decreases in the subsequent episodes, where it has an all-time low of 94.65 s in episode 4.83 million; it is seen that in the last 1.2 million episodes, the agent was still capable of improving the episode length partially. However, that improvement did not significantly affect the rewards. Hence, the experiment was stopped in episode 5 million.

4.2.2.3. Extrinsic Value Estimate

Extrinsic value estimates in RL refer to rewards provided by the environment, guiding the agent's behaviour toward achieving specific tasks or goals. These rewards are external to the agent's

internal processes and are used to shape the learning policy.(Lillicrap et al., 2015)(Lillicrap et al., 2015)(Lillicrap et al., 2015)(Lillicrap et al., 2015)(Lillicrap et al., 2015). The crane agent uses the Extrinsic Value Estimate (EVE) indicator to forecast and assess the quality of its actions. Its primary objective is to maximize cumulative rewards while maintaining a balance between short-term and long-term decision-making processes, which is crucial for developing successful policies to accomplish tasks. The EVE process results are depicted in Figure 28.

As Figure 28 illustrates, the agent initially exhibits an overestimation of action values for the first 0.24 million episodes compared to the actual values presented Figure 25. This overestimation can be attributed to the agent's limited knowledge or experience during the exploration phase. However, during the primary exploitation phase, the agent's extrinsic value estimates show significant improvement, mirroring the trend observed in the cumulative rewards graph. These estimates continue to increase dramatically until the 1.23 millionth episode, from −4.119 to 175.9 points (a notable 180.2-point increase), consistent with the cumulative rewards graph. Over the final 3.8 million episodes, the reward estimation experiences improvements at a slower rate, aligning with trends observed in the cumulative rewards graph.

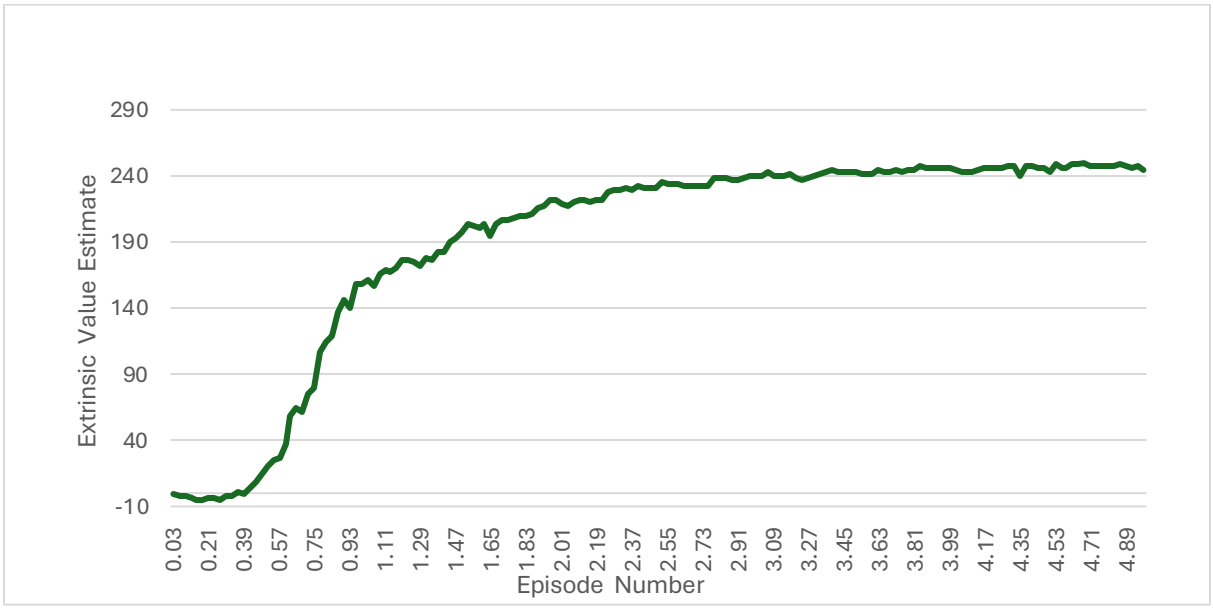


Figure 28: Extrinsic reward estimate for hyperparameter model.

4.2.2.4. Curiosity rewards

A fundamental challenge in RL problemsis assessing the efficacy of the exploration phase during training, a curiosity reward indicator is employed that quantifies rewards obtained solely from exploration. The influence of this indicator on training outcomes is depicted in Figure 29.

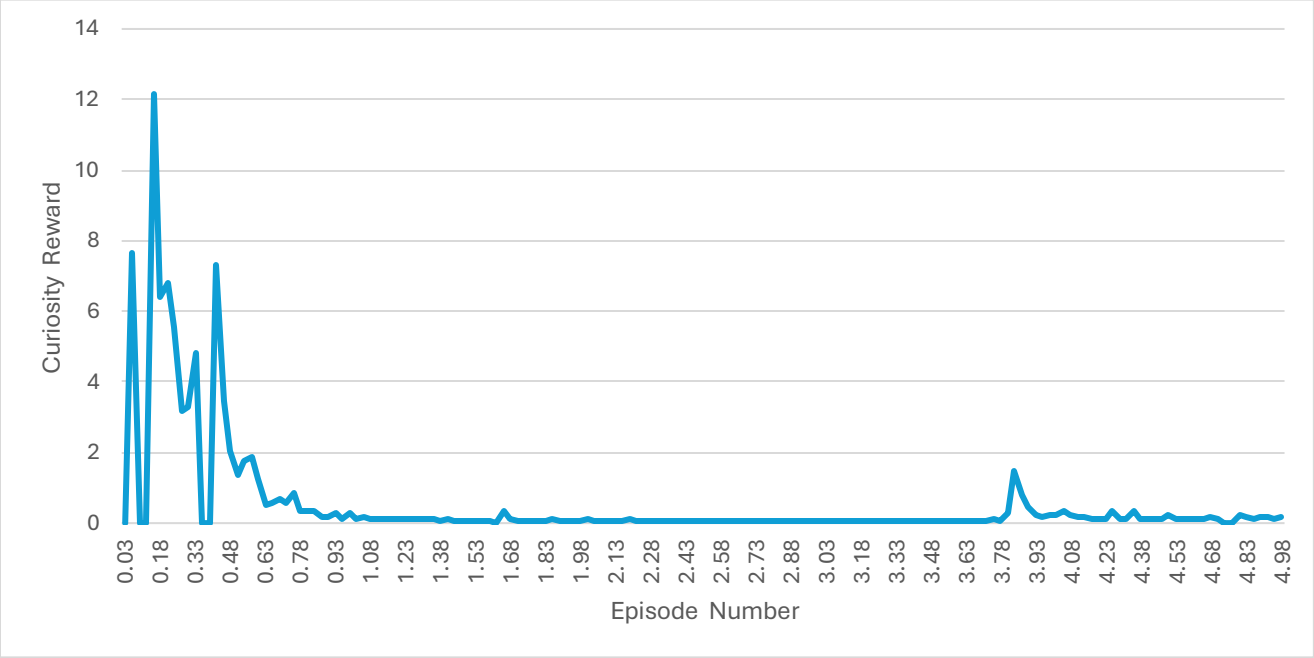


Figure 29: Impact of curiosity on rewards in hyperparameter model.

In RL problems, the agent initially lacks knowledge about its environment. This curiosity significantly affects the agent's ability to accumulate rewards during learning. For example, in a hyperparameter optimization model, the agent's early episodes saw a substantial reward increase, primarily due to the curiosity parameter, reaching a peak value of 6.81 points over the first 0.21 million episodes. However, over the next 0.79 million episodes, rewards derived from curiosity sharply declined, eventually stabilizing at 0.2972 points. This decline in curiosity-based rewards is a natural progression as the agent transitions towards an exploitation phase around the 0.24 millionth episode, as is evident in the cumulative rewards graphs. However, a sudden surge in curiosity-based rewards occurred in the 3.87 millionth episode, with a value of 0.773 points, attributed to the agent discovering a new approach to performing the task. Although this type of occurrence is expected, the surge in curiosity did not improve the reward collection process, as the total reward decreased from 297 to 295 points due to curiosity-based exploration. As the agent develops a deeper understanding of the environment, curiosity's impact on overall reward

collection diminishes, indicating a reduced significance of curiosity in exploration. Throughout the training process, the agent accrues rewards from exploitation, learning rate adjustments, and the curiosity factor.

4.2.2.5. Curiosity value estimate

The curiosity value estimate pertains to the agent's evaluation of exploration returns in terms of rewards. The curiosity value estimate in RL refers to an intrinsic reward signal that motivates an agent to explore states and actions that are novel or uncertain, encouraging learning about the environment. It is used to improve the agent's knowledge and policy (Pathak et al., 2017)(Pathak et al., 2017)(Pathak et al., 2017)(Pathak et al., 2017)(Pathak et al., 2017). Initially, in the first 0.78 million episodes, there is an underestimation of the rewards achievable through curiosity-based exploration where the estimates fluctuate between 0.1 and 0.4 points, where the actual rewards obtained through curiosity are significantly higher; it was only in episode 0.78 million that the curiosity value estimate and the curiosity rewards converged at around 0.33 points. However, in subsequent episodes, a new trend emerged: the overestimation of rewards obtained through curiosity, where the estimate continued to increase up until it reached a final value of 1.46 points in the final episodes, where the curiosity reward value followed the opposite trend as discussed in the previous section. This overestimation stems from the policy's continuous exploration-based approach, wherein the agent is consistently prompted to explore the environment or actions by inflating the curiosity value estimate. The rise in the estimate is attributed to the need for more actual rewards obtained through curiosity, thereby failing to provide a proper incentive for the agent. The overall behaviour of the curiosity value estimate is depicted in Figure 30.

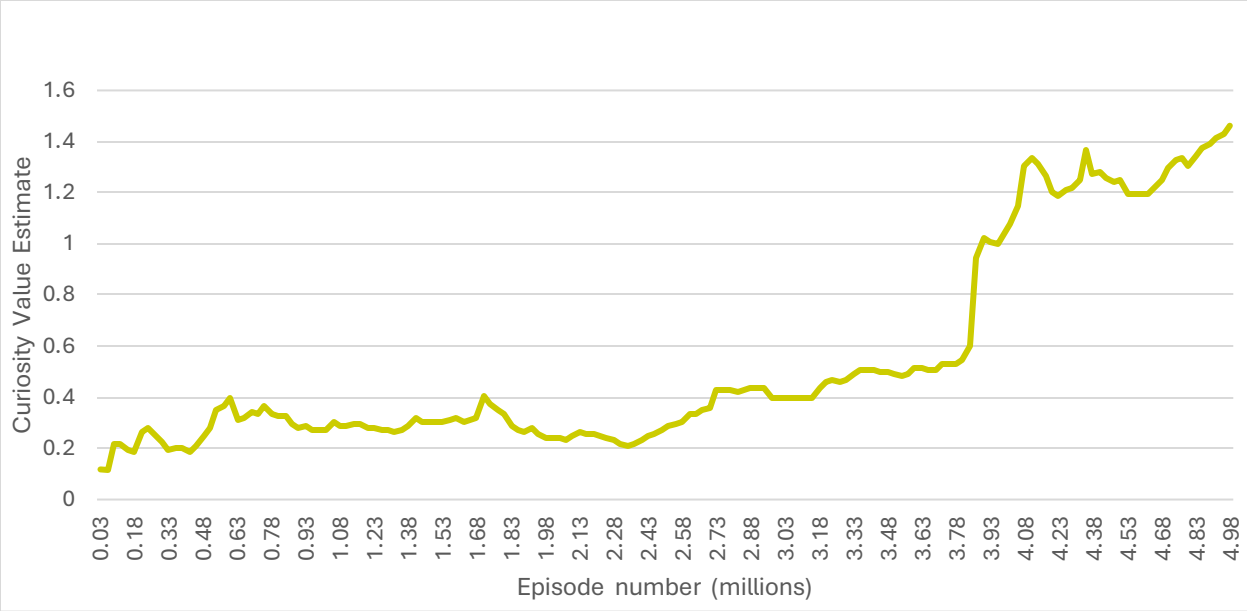


Figure 30: Curiosity value estimates in hyperparameter optimization model.

4.2.3. Case study model results

4.2.3.1. Cumulative rewards for pure RL approach

This section presents the findings of the implementation of RL for mobile crane path planning. Hyperparameters were optimized for the case study as discussed in detail in section 3.2.2. Figure 30 presents the cumulative rewards achieved per episode in the case study. During the first 2 million episodes, the agent was challenged in developing a policy to solve the lifting problem due to the presence of a wide range of obstacles in the environment. This difficulty was not present in the initial hyperparameter optimization environment. As a result, the initial episodes mainly consisted of repetitive collisions between the agent and various environment components. Due to these obstacles, it took the agent 300,000 episodes to accumulate rewards that surpassed 100 points in the hyperparameter model, as shown in Figure 31.

In contrast, the agent in the case study struggled significantly due to the presence of obstacles and could only achieve a positive reward of 21.8 points by the 2.28 millionth episode. This highlights the impact of obstacles on learning and how it can significantly affect the agent's performance.

Over the course of 2.28 million episodes, the agent was confronted with the challenge of navigating an environment fraught with collisions. In the case study model, an agent was trained to navigate through obstacles and find and attach a payload to a loading hook. The agent successfully avoided

all obstacles within 2.46 million episodes, demonstrating its ability to learn and adapt. It then moved on to the primary task of finding and attaching the payload, achieving a reward of 100. The agent's ability to master this task within 2.67 million episodes, with a cumulative reward of 110.01 points, further underscored its learning capabilities. The study also observed that the agent tended to complete the exploitation phase, which involved efficiently discovering the environment, between 20% and 40% of the total time needed. The pure RL model was also tested, and the agent showed significant improvement in learning within 0.65 million episodes after the rewards were enhanced from negative to positive outcomes. The improvement is illustrated in Figure 30, where the agent obtained a cumulative reward of 289 in the 3.48 millionth episode, representing a 179-point increase in just 0.81 million. The steep decrease in the reward collection process indicates the end of the agent's exploitation phase, highlighting its ability to optimize its performance over time. In the context of optimizing hyperparameters for an environment model, it has been observed that the presence of obstacles can considerably slow down the training process. Specifically, the impact of obstacles can lead to a significant delay in the exploitation phase, which might require the agent to undergo roughly 2.35 million additional training episodes. This represents a whopping 302% increase in the duration of the training process, highlighting the importance of carefully evaluating the impact of environmental factors on the agent's performance. As for the accuracy of the training, the complexity of the environment did not affect the agent's performance during the optimization phase of the training; the agent undergoes the process of optimizing the lifting process with small increments.

In the case of the pure RL approach, it was observed that the agent kept optimizing the lift until it reached an impressive score of 299 points in the 8.76 millionth episode. The rewards obtained in the range of 3.48 to 8.76 million episodes indicate that the agent improved its reward by 8.61 points or an equivalent of 3% in the episodes between 3.54 and 9.0 million. This significant improvement is a testament to the agent's ability to adapt and evolve during its training. Moreover, in the last million training episodes, the agent continued to optimize the lifting process and reached a total reward of 299.1 in episode 9.81 million. Theoretically, the agent could continue to improve the results within a small margin indefinitely. However, it was observed that since the last million episodes could only improve the training by 0.003%, it was better to end the training at the ten millionth episode. Overall, the optimization phase is a crucial part of the training process, and the

agent's impressive performance in optimizing the lifting process demonstrates its ability to learn and adapt to the given environment.

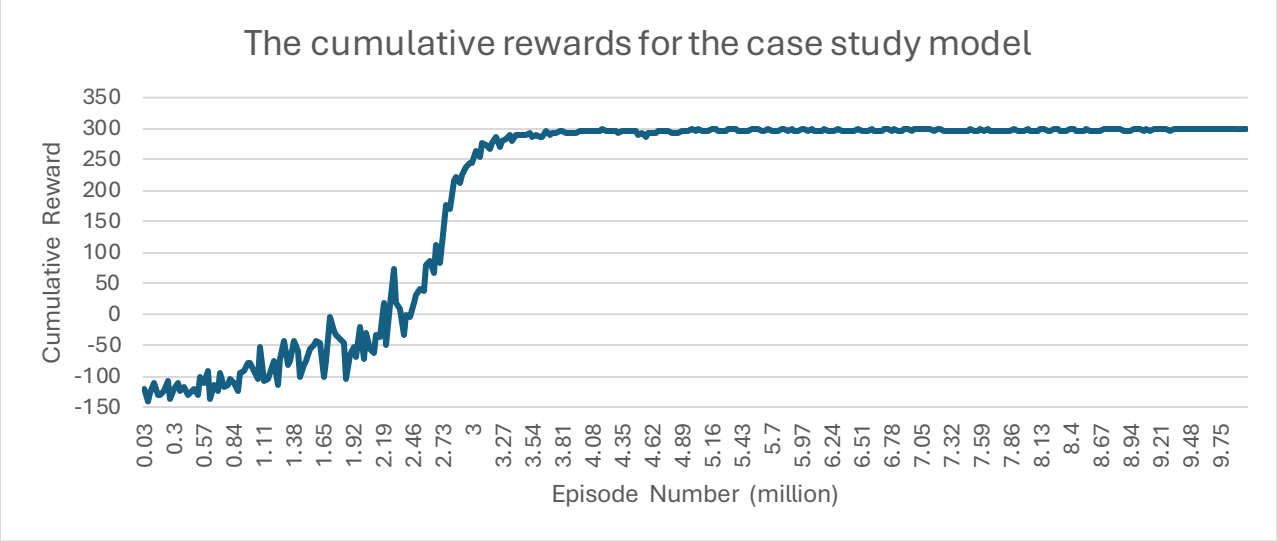


Figure 31: Cumulative rewards for case study model.

4.2.3.2. Episode length

The episode length criterion indicates the efficiency of the lift path planning tool; the objective of the crane agent is to find the shortest, safest, and most efficient path. Hence, the objective is to decrease the episode length until no further improvements can be made or until the agent can no longer improve the path without putting the crane in a hazardous scenario.

The following section discusses the impact of the pure RL approach on episode length and the overall progress made regarding the number of actions taken in each episode. The episode length graph demonstrates a distinct trend from the cumulative reward graph, as the agent's objective is to decrease the number of actions taken rather than to maximize the reward. This explains the initial portion of the episode length graph, where the number of actions taken continued to increase proportionally until the 2.16 millionth episode. This phase is closely associated with the exploration phase, as described in section 4.4.4. The fluctuations observed in the episode length graph within the range [0, 2.16 million] are due to the stability and obstacle collision termination conditions and the agent leaving the training area termination condition, explained in the rewards signal section 3.5.1.

To validate this theory, the episode length of the hyperparameter optimization environment was compared to the case study model, where the agent consistently decreased the episode's length without any fluctuations related to the stability and obstacle termination conditions. The only condition that was present in both environments was the agent leaving the training area termination condition.

The first significant improvement in the episode length was observed in the 2.16 millionth episode, where the overall episode length started to decrease considerably from an all-time high of 8,955 s to an average of 321.4 s per episode, which is only 3.5% of the initial number of episodes. This trend continued until the 3.63 millionth episode, after which the number of episodes decreased at a slower rate. The 3.63 millionth episode correlates closely with the rewards obtained around the same period, where the agent significantly improved the reward by decreasing the number of actions needed. However, between the 3.48 and 3.63 millionth episodes, it was observed that the number of actions continued to drop with a lesser rate of improvement in terms of rewards obtained, indicating that the agent had reached the optimization stage of the training.

In the training process of an agent, a significant improvement was observed between the 3.63 and 9 millionth episodes. The agent completed the task with an average duration of 88.91 s, resulting in a notable decrease of 72% with a reduction of 212.45 s. This marked improvement can be attributed to the agent's initial utilization of different lift paths with varying success rates. The subsequent stage in training included selecting the best paths and improving them, leading to a significant decrease in the number of actions required to complete the task. However, after the nine millionth episode, the agent's performance improvements were marginal. The agent had already learned the best paths, and the remaining episodes mainly involved reiterating the same policy with minor changes. As a result, the agent could only improve its performance by 7.45 s, which accounts for a 10% improvement. This improvement was considered marginal due to the cumulative reward section where the agent only managed to improve the rewards obtention strategy by 0.1 points. Although this could be significant in specific scenarios, it was not so in the current case study.

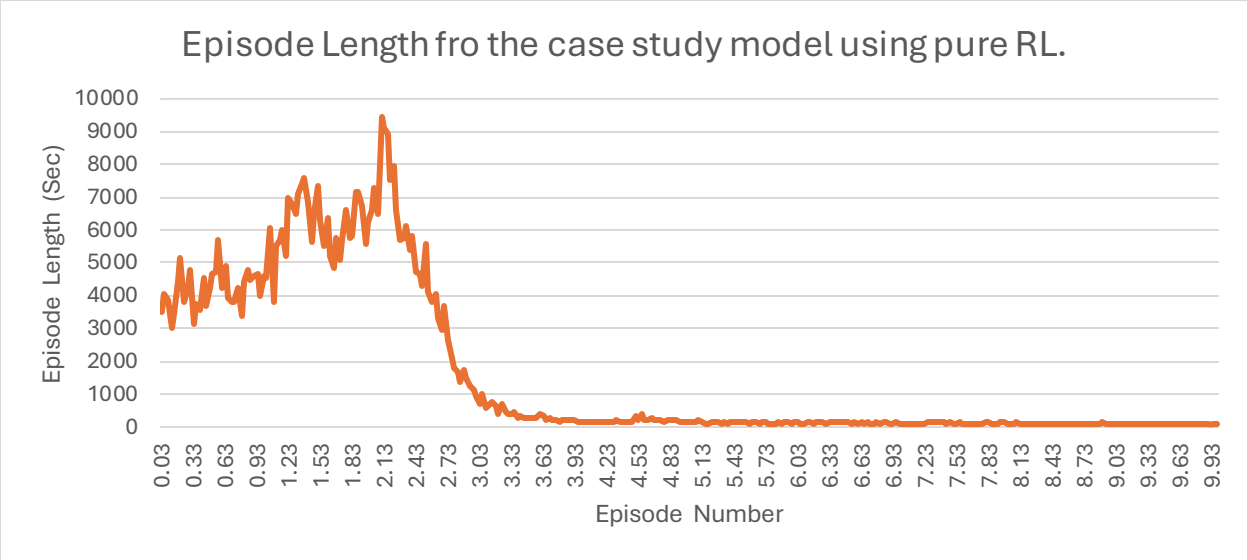


Figure 32: Total episode length relative to number of episodes.

4.2.3.3. Extrinsic value estimate

The crane agent employs the EVE decision-making tool to make predictions and estimates about the quality of its actions. The agent's objective is to maximize its cumulative rewards while also striking a balance between its short-term and long-term decision-making processes. This balance is critical for the agent to develop successful policies that enable it to accomplish its task. The results of the EVE process are presented in Figure 33.

As depicted in Figure 33, the agent initially overestimates the value of its actions for the first 2.25 million episodes compared to the actual actions value. This overestimation can be attributed to the agent's lack of knowledge or experience during the exploration phase. However, during the primary exploitation phase, the agent's extrinsic value estimates improve significantly, following a similar trend to the cumulative rewards graph. The estimates continue to improve dramatically until the 4.05 millionth episode, whereby they increase from −2.90 to 206.47 points, with an increase of 209.37 points, which is similar to the cumulative rewards graph. In the final 5.95 million episodes, the reward estimation experiences minor improvements, similar to the cumulative rewards graph.

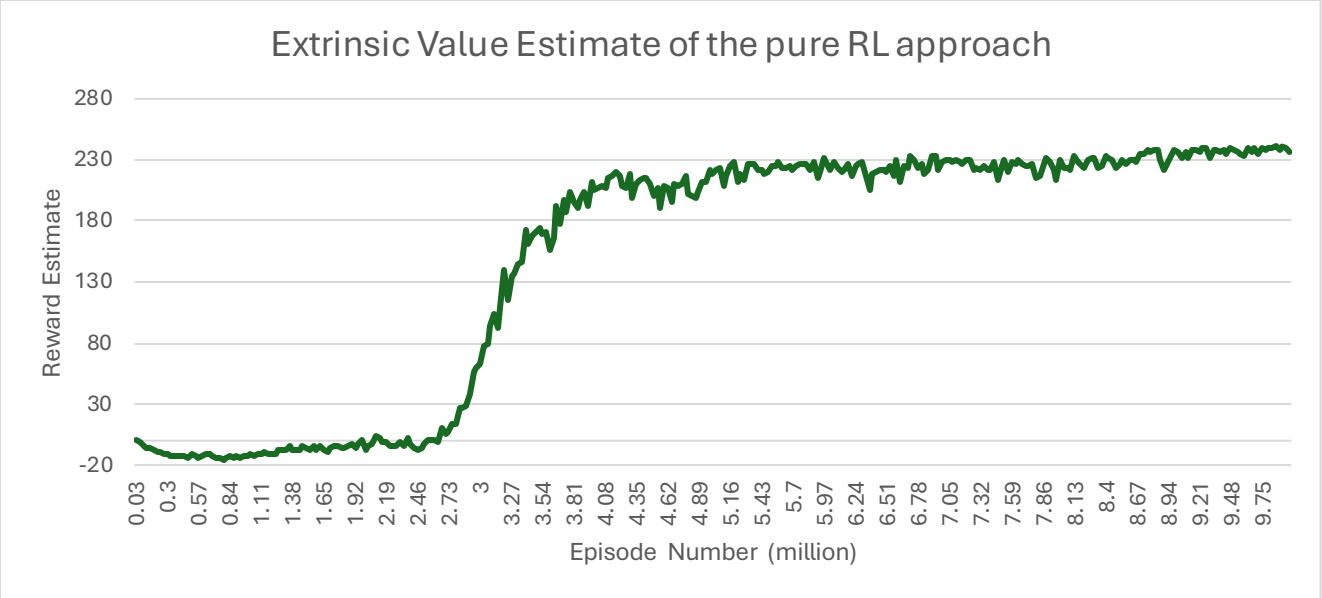


Figure 33: Extrinsic value estimate in pure RL approach.

Nevertheless, one notable difference between the EVE and cumulative rewards models is that the estimated value is always lower than the actual reward obtained. This discrepancy is a common occurrence in RL problems and can be attributed to various factors, such as approximation errors, uncertainty, and trade-offs between exploitation and exploration.

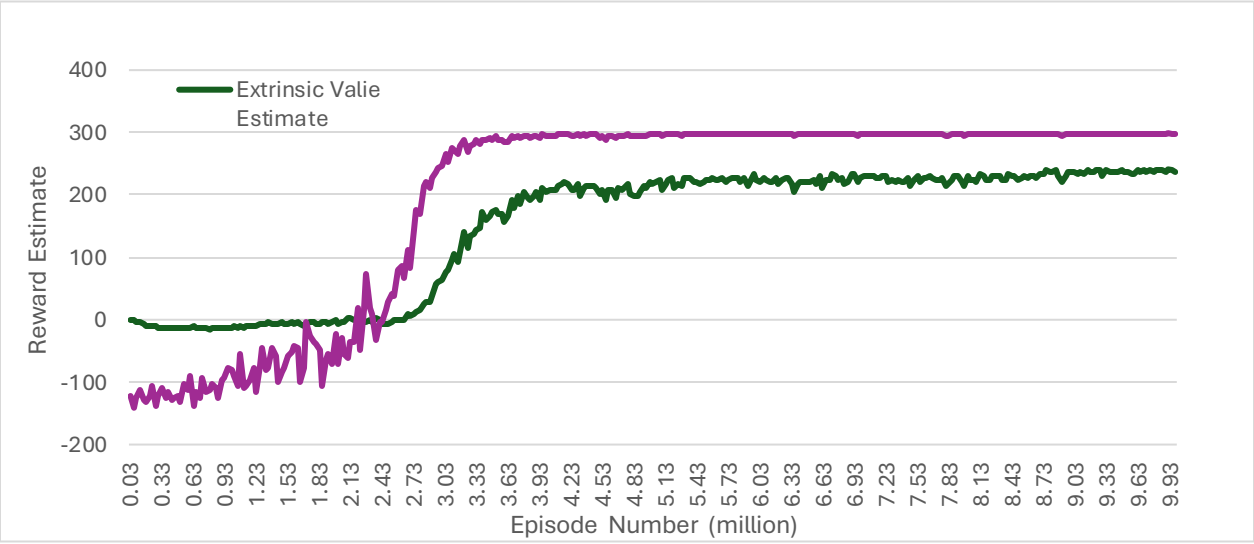


Figure 34: Extrinsic reward estimate versus cumulative reward.

4.2.3.4. Curiosity rewards

As already noted in Chapter 3, achieving a balance between exploration and exploitation is one of the most challenging aspects of any RL problem. To evaluate the effectiveness of the exploration phase during the training process, a curiosity reward indicator is employed that measures the rewards obtained solely from exploration. The impact of this indicator is illustrated in Figure 35, which demonstrates how it influences the training outcomes.

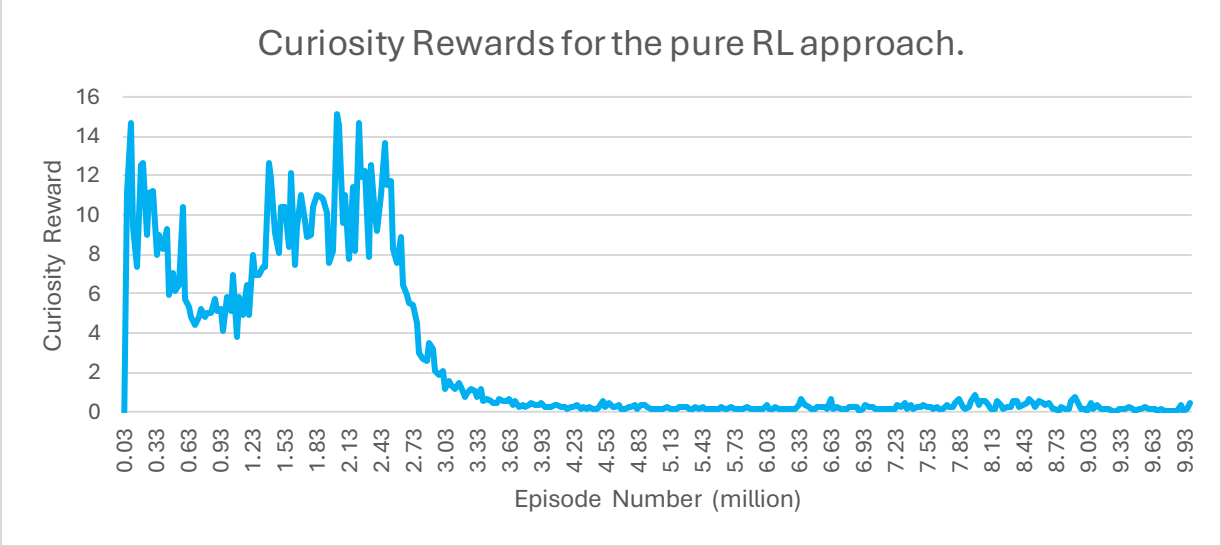


Figure 35: Impact of curiosity rewards in case study model.

In RL problems, an agent's initial episodes are typically characterized by a lack of knowledge about the environment. During this learning phase, the agent's curiosity can significantly affect its ability to collect rewards. This is illustrated in a case study where the agent's first 2.52 million episodes showed a marked reward increase due to the curiosity parameter alone, with the highest peak value observed being 14.6 points. However, following this initial phase, the rewards obtained through curiosity decreased sharply over the next 1.46 million episodes, eventually reaching a value of 0.278 points. This decline in curiosity-based rewards is a natural development since the agent begins to shift towards an exploitation phase around the 2.46 millionth episode, as shown in the cumulative rewards graphs. In subsequent episodes, the impact of curiosity on the overall rewards collection process is minimal due to the agent's deep understanding of the environment. This indicates that curiosity is no longer largely impactful for exploration. Throughout the training process, the agent earns rewards from a combination of exploitation, learning rate, and curiosity factor.

4.2.3.5. Curiosity value estimate

The curiosity value estimate is related to the agent's assessment of exploration return in terms of rewards; the initial value in the first 3 million episodes underestimates the rewards that can be obtained from curiosity-based exploration. However, for the rest of the episodes, there is an overestimation of the rewards obtained through curiosity; this overestimation can be attributed to the policy's continuous exploration-based approach, where the agent is continuously encouraged to explore the environment or the actions by overestimating the curiosity value estimate; the increase of the estimate, meanwhile, is due to the lack of actual rewards obtained through curiosity to provide a proper incentive for the agent. The overall curiosity value estimate behaviour is shown in Figure 36.

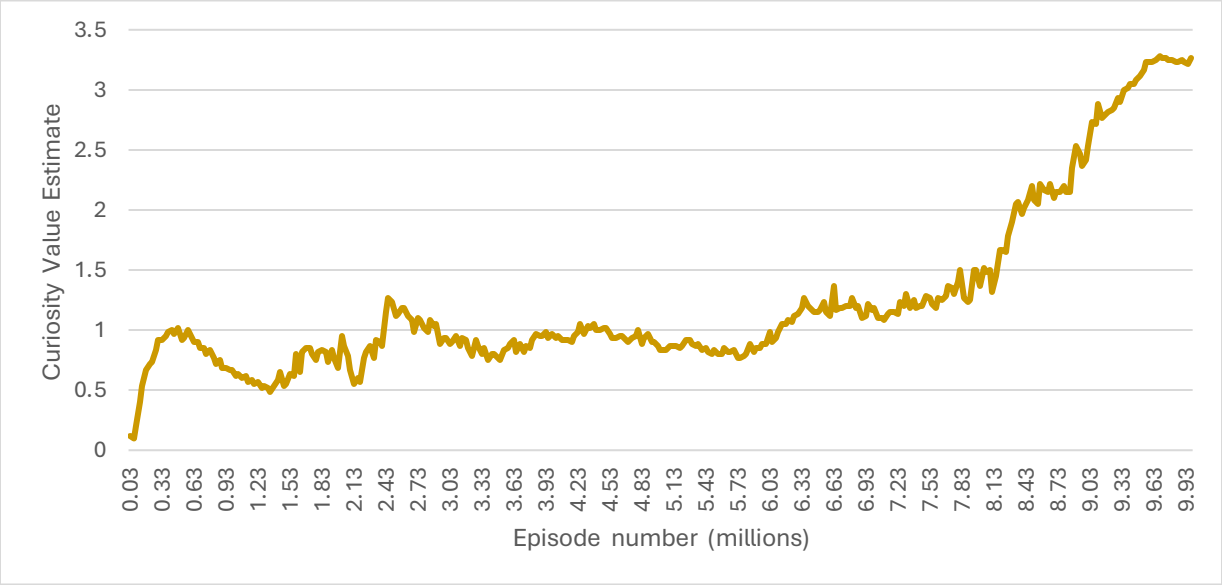


Figure 36: Curiosity value estimate for case study model using pure RL approach.

4.3. Generative adversarial imitation learning approach

4.3.1. Cumulative rewards

This section presents the findings concerning the application of RL to mobile crane path planning within the GAIL framework. The approach involved optimization of hyperparameter to ensure optimal results, as discussed in section 3.2.2. The post-training results exhibited promising outcomes.

Figure 36 shows the cumulative rewards attained per episode, providing insights into the training process. Across the initial 3.18 million episodes, the agent encountered challenges in formulating a policy to address the lifting problem. Consequently, these early interactions primarily consisted of repetitive exchanges between the agent and the environment. Through these interactions, the agent achieved a positive reward of 4.59 points within 3.18 million episodes, as depicted in Figure 37. It is worth noting that the amount of time needed by the agent to finish the exploration phase is far longer in this case than for any of the other models. This is because of the need to strike an appropriate balance between training the agent properly and training the discriminator. This balance often presents a challenge for the overall training process. This challenge added 0.8 million training episodes for the case study (to achieve results similar to those achieved by the pure RL model).

Within the hyperparameter model, the agent was tasked with discovering the optimal actions to maximize rewards while ensuring stability and effectiveness. Its initial milestone involved locating and attaching a payload to a loading hook. Remarkably, within 4.41 million episodes, the agent circumvented all obstacles and accomplished the primary task of payload attachment. This achievement garnered a reward of 105 points, with the agent mastering and stabilizing its training. Notably, the agent demonstrated a monumental improvement in the rewards collection process, increasing by approximately 101 points within 1.25 million episodes, underscoring the significance of carefully selecting each hyperparameter.

The largest difference between the GAIL model, and the pure RL model is the exploitation phase, which generally is followed by an optimization phase in the pure RL model. However, the GAIL model presents a different structure to its training, where the exploitation and optimization phases are mixed within one phase. This could provide better solutions; however, in the present case, the agent obtained a maximum cumulative reward of 263.69 points, achieved in episode 8.07 million. This is significantly lower than the pure RL approach, and the agent itself needed to be more capable of providing a better despite being provided a set of demonstration to achieve the task.

Within this context, it is seen that the advantage of the GAIL method lies in the insurance it provides, where the agent will always manage to find a solution, as opposed to the pure RL approach, in which the agent might not find a solution. The downside of the GAIL approach is its inability to improve the solution found compared to the pure RL approach. The lack of

improvement is explained by the demonstration itself, which is very subjective, meaning the agent would learn an approach that may not be the optimal path; it will then attempt to improve it. If the path can not be further improved, it will attempt to find a different path, resulting in more extended training without any assurances of whether it will find a better solution than that found in the pure RL approach.

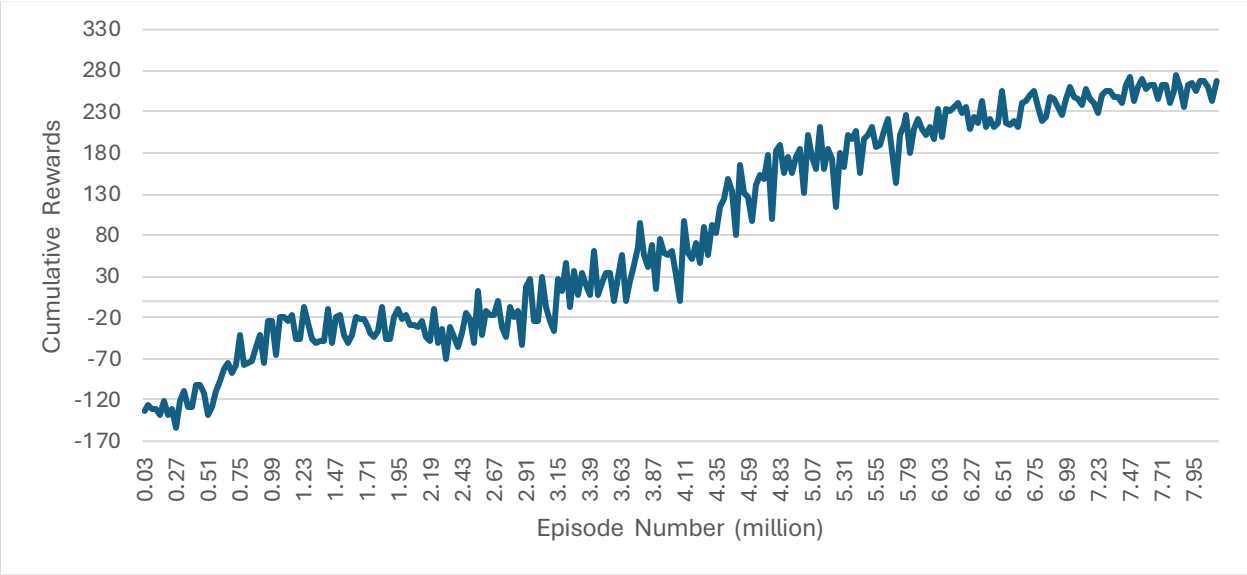


Figure 37: Cumulative rewards for Generative Adversarial Imitation Learning approach.

4.3.2. Episode length in the GAIL model

This section provides an in-depth analysis of the episode length criterion within the context of the GAIL model used for crane path planning. The ensuing discussion delineates the impact of the GAIL approach on episode length and the overall progression regarding the number of actions taken per episode. Contrary to the cumulative reward graph, the episode length graph exhibits a distinct trend, prioritizing reducing actions over maximizing rewards. Within the episode length graph, the number of actions steadily increased from an initial value of 4,350 s until it reached an all-time high of 9,720 s approximately in the 2.1 millionth episode. This phase aligns predominantly with the exploration phase observed within the cumulative rewards section.

Subsequent episodes witnessed significant improvements in episode length, with the agent managing to decrease it from 9,720 s to 1,268 s in episode 6.96 million; this level of improvement is lower than the one seen in the pure RL model, which can be explained by the difficult transition from the demonstration-based path, to a more optimized one. This improvement correlates closely

with the exploration phase depicted in the cumulative rewards graph. However, the rate of improvement tapered off in subsequent episodes, reaching an all-time low of 668.89 s in episode 7.74 million, which constitutes a 48% improvement from episode 6.96 million, or an equivalent decrease of 616.55 s. In the subsequent episodes, the agent could not improve the episode length and, in some instances, performed more poorly due to the attempt to find a different policy unrelated to the demonstration provided. The training was finalized in episode 8 million. Since no significant improvement was observed, an additional parameter was used to enhance the performance of the GAIL model, as discussed in the following subsections.

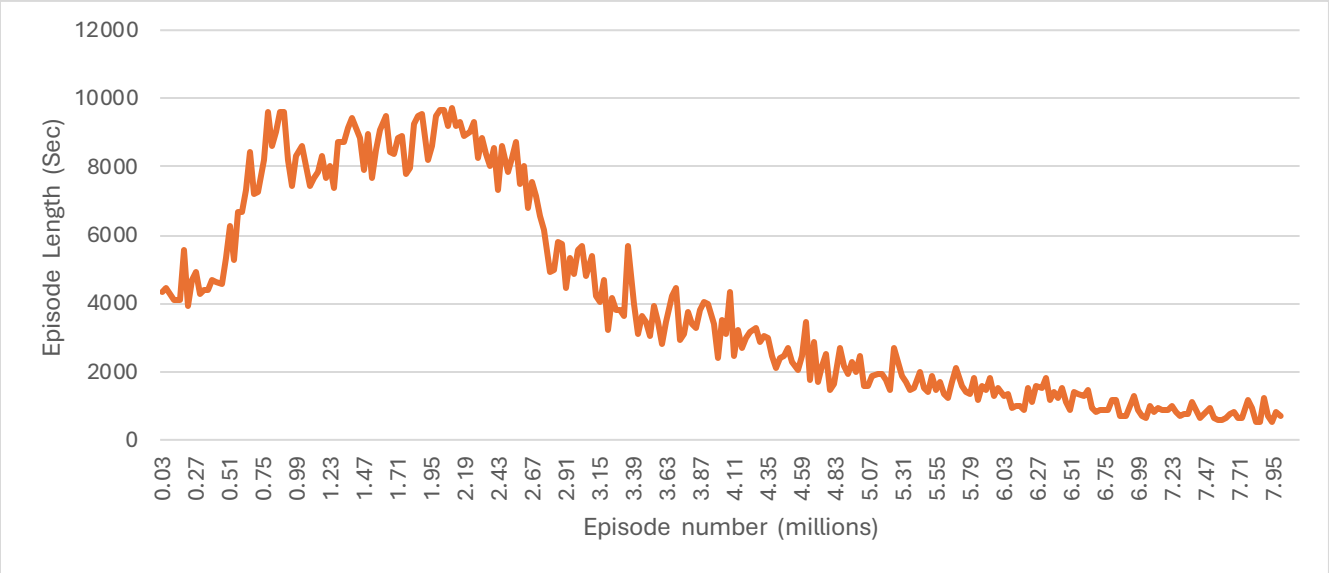


Figure 38: Episode length in GAIL model.

4.3.3. Extrinsic value estimate for the GAIL model

In the context of the GAIL model, the crane agent uses the EVE decision-making tool to forecast and assess the quality of its actions. The results of the EVE process within the GAIL framework are depicted in Figure 39.

As Figure 38 illustrates, the agent initially overestimated action values for the first 3.18 million episodes compared to the actual values. This overestimation can be attributed to the agent's limited knowledge or experience during the exploration phase within the GAIL model. However, during the primary exploitation phase, the agent's extrinsic value estimates demonstrate significant improvement, mirroring a similar trend in the cumulative rewards graph. These estimates continue to increase dramatically until the 7.71 millionth episode, from −9.75 to 109.20 points (a notable

118.95-point increase), consistent with the cumulative rewards graph. However, after the 7.71 millionth episode, it can be seen that the estimates dipped for a short period, where the overall estimate reached a low of 75.01 in episode 7.89 million, which can be explained by the increase in episode length observed around the same period in Figure 38. The episode length increase was due to the agent attempting to discover a new approach to solving the lifting problem. However, the performance was adversely affected. In the rest of the episodes, the EVE regained its value, and the agent continued training using the same approach. If the agent is given more training episodes, it could provide an improved solution; however, due to the poorer performance of the GAIL model in comparison with the pure RL, it can be inferred that further modification could enhance its performance significantly, as discussed in the following subsections.

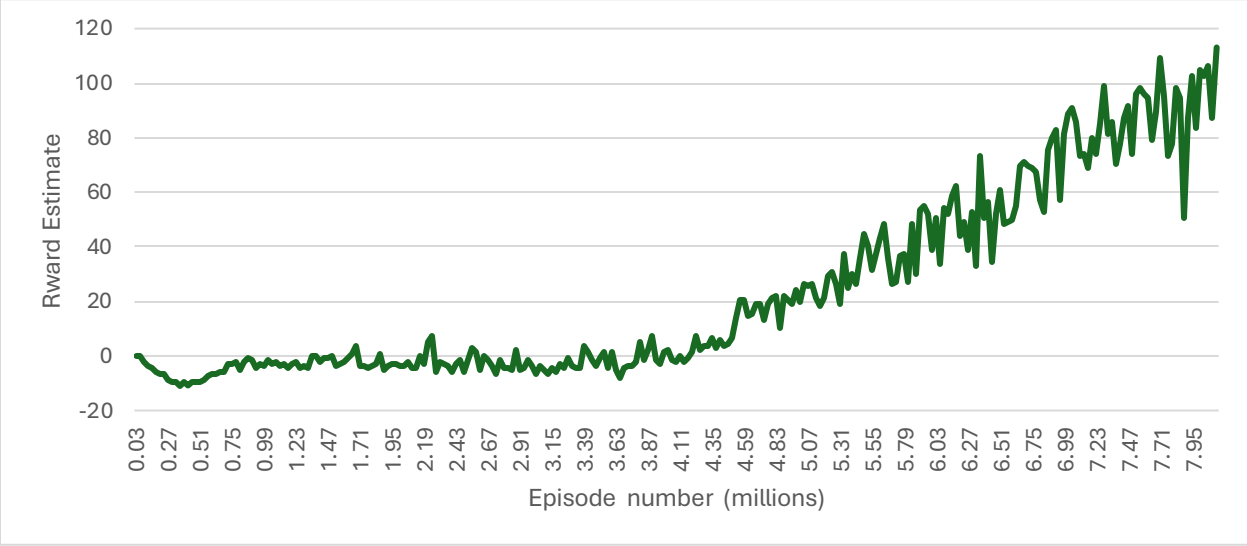


Figure 39: Extrinsic reward estimate in GAIL model.

4.3.4. Curiosity rewards in GAIL model

To gauge the effectiveness of the exploration phase during training within the GAIL framework, a curiosity reward indicator is employed that quantifies the rewards solely derived from exploration. The impact of this indicator on training outcomes is illustrated in Figure 40.

In the developed model, the initial episodes of the agent are marked by a need for more environmental knowledge. During this learning phase, the agent's curiosity is pivotal in its ability to amass rewards. This phenomenon is explained within the GAIL model, where the agent's initial 2.61 million episodes witnessed a notable reward surge solely attributed to the curiosity parameter, peaking at 19.76 points. However, after this initial surge, rewards stemming from curiosity

experienced a sharp decline over the ensuing 5 million episodes until it reached an all-time low of 0.52 points in episode 7.95 million. The decline in curiosity-based rewards adversely affected the overall improvement of the cumulative reward collection process despite the agent being incapable of reaching higher rewards, which was proven to be possible in the pure RL model. This lack of exploration is one of the primary reasons for the sharp decrease in improvements. However, this problem can be fixed when paired with a BC approach as a means of taking into consideration different lift paths besides those seen in the demonstrations.

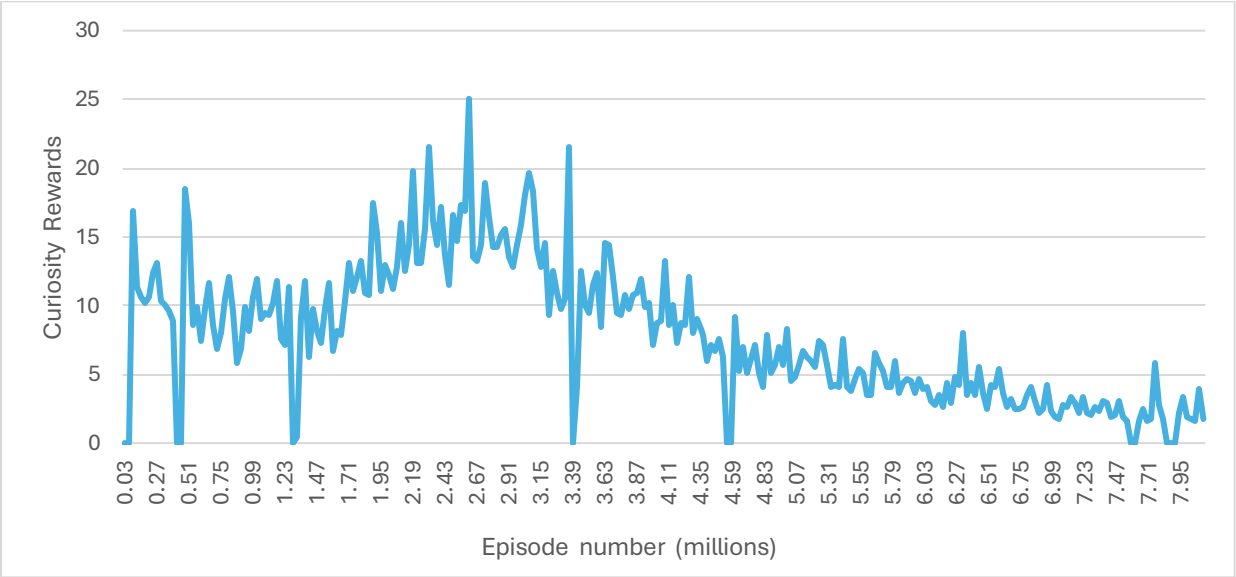


Figure 40: Curiosity rewards in GAIL model

4.3.5. Curiosity value estimate

The curiosity value estimate pertains to the agent's evaluation of the exploration's payoff in terms of rewards. Figure 40 shows that the CVE estimate for the GAIL model is almost identical to that of the pure RL model, which can be explained by the RL components present in the GAIL model. The rewards obtained from curiosity, seen in Figure 41, differ entirely from those present in the CVE. This can be explained by the agent's curiosity for RL portion exploration but not the GAIL

portion of the model. In this regard, two parameters for GAIL are discussed in the following subsections: the GAIL expert estimate and the GAIL reward.

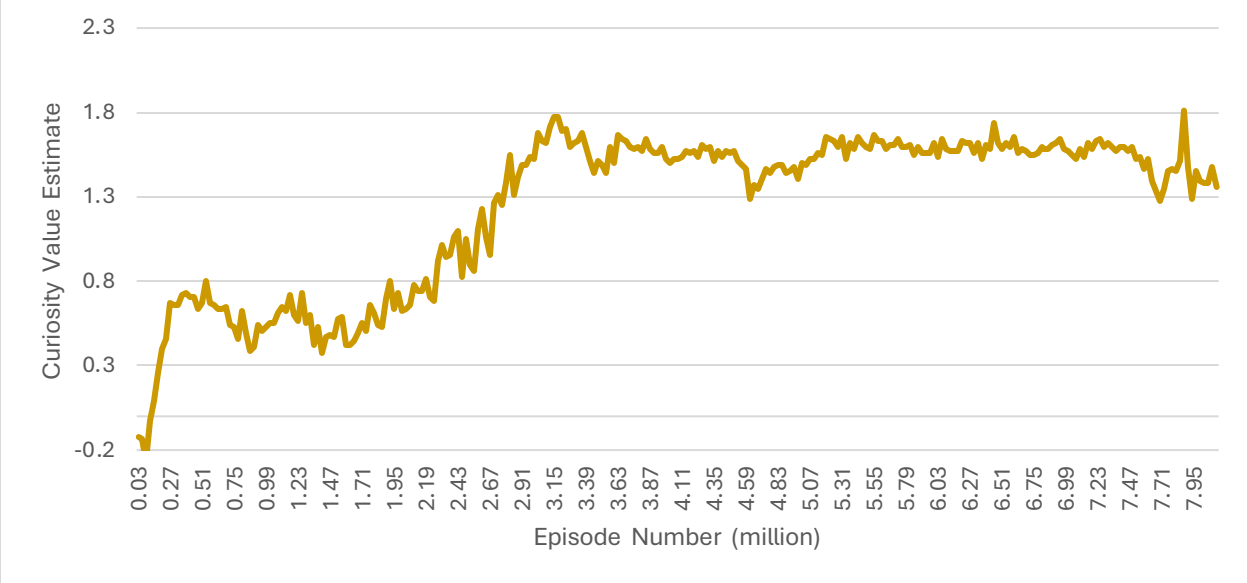


Figure 41: Curiosity value estimate for GAIL model.

4.3.6. GAIL reward

This section presents the findings concerning the implementation of GAIL and RL for mobile crane path planning. Hyperparameters for the case study were optimized as discussed in detail in section 3.2.2 (also see Figure 42). The GAIL rewards figure 42 provides an indicator of the impact GAIL has on the rewards collection process, thereby shedding light on the discriminator training process and its efficiency for training. During the first portion of the training, the agent is heavily reliant on the GAIL portion to develop a policy, the demonstrations are then used to accumulate the majority of the initial rewards, as shown in Figure 41, where the agent can be seen to have obtained a total of 103.9 points in episode 0.87 million. The 103.9 points of rewards obtained through GAIL are evened out with the cumulative rewards obtained through the RL portion, and total reward of −56 is obtained. The fact that the cumulative reward value is negative despite the large GAIL contribution to the training is significant, particularly considering that the RL model lags behind the GAIL model in the initial portion of the training. In the next 3 million episodes the GAIL-based rewards decrease in value from 103.9 to 27.9 points, a result that can be explained by the training gained through the RL model within the same time period. The result for the period between 0.87 and 3.9 million episodes can be explained by the increased reliance of the agent on

the RL model as the primary reward-collecting tool. The reliance on the GAIL algorithm continues to drop in the last 4 million episodes until it reaches an all time low of 3.93 point in episode 8.07 million. The pattern depicted in Figure 42 indicates that the agent initially focuses on replicating the demonstrations as a main approach to collect rewards, once the decimators and the agent are trained efficiently. The agent becomes less reliant on the provided demonstration and becomes more reliant on the collected information from the RL model.

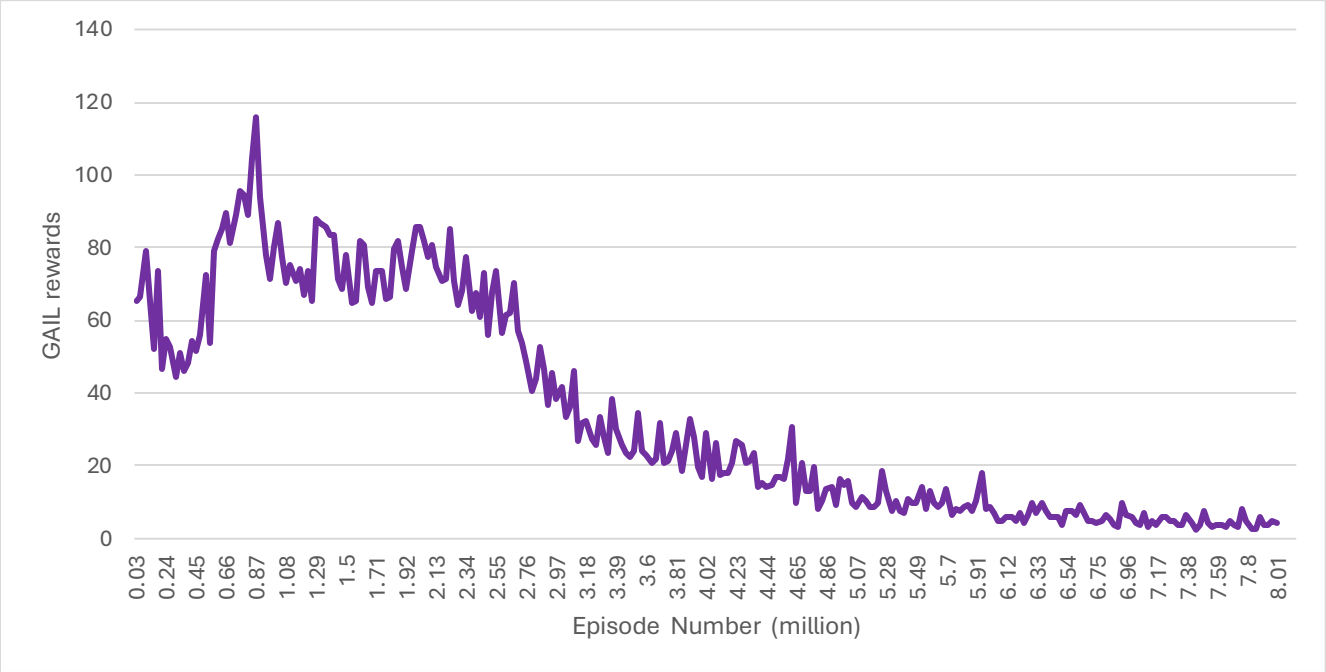


Figure 42: Rewards for GAIL approach.

4.3.7. GAIL Value Estimate

To gauge the effectiveness of the GAIL model on the training model, the agent makes a set of predictions based on the rewards which could be obtained from the GAIL algorithm. The GAIL estimate values can be divided into two phases. The first phase is an increased GAIL rewards estimate, where the first 0.66 million represent an overall positive trend of 0.65, which suggests that the GAIL model would obtain a large number of rewards from the GAIL algorithm. This aligns with the findings with respect to GAIL rewards (Figure 43). Similar to the GAIL rewards figure 42, the GAILVE started decreasing proportionally, where the GAILVE decreased from 0.65 to a minimum of 0.2031 (in the 7.81 millionth episode). This coincides with the decreased impact of the GAIL algorithm on the overall training seen within the same range of episodes in the cumulative rewards section.

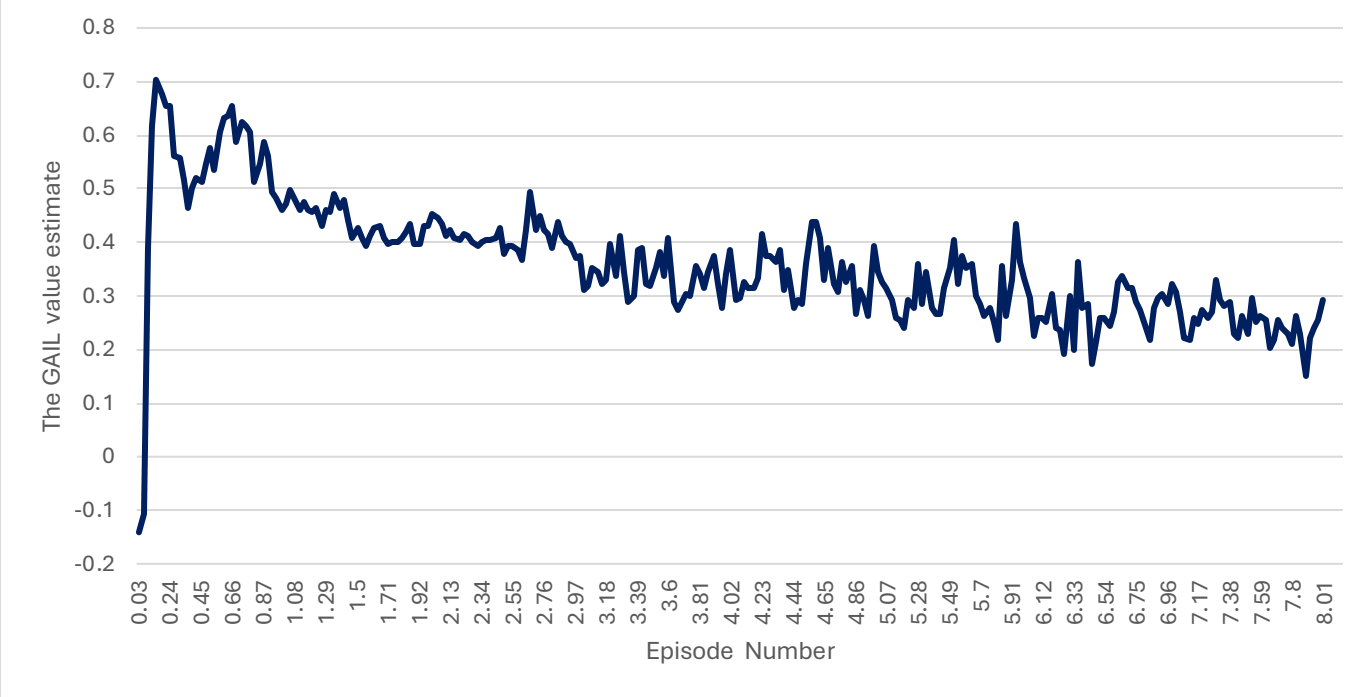


Figure 43: Value estimate for GAIL approach.

4.4. Mixed learning approach

As noted in previous sections, the pure RL model performed significantly better than the other models, such as the GAIL and BC models. However, the RL requires a large set of training episodes, which could deter users from implementing it on real-life projects. Furthermore, the RL model does not ensure the finding of a solution within a specified number of episodes, which is a significant downside to its implementation. On the other hand, the GAIL model presented a set of suboptimal solutions compared with the solutions found through RL. In some cases, the GAIL model would necessitate a more extensive set of episodes than the RL model. However, the GAIL model presents an advantage over the RL approach in that the user can determine the method with which they achieve tasks and have a more hands-on approach to their training procedure.

Furthermore, the GAIL approach presents more certainty for the user since it will always manage to find a solution for the problem presented. Therefore, using a mixture of GAIL and RL can provide better solutions. Based on practical experimentation, it was observed that the BC approach, despite its downsides, could be highly effective for training if properly implemented alongside RL and GAIL. Thus, this section discusses the implementation of a hybrid approach combining RL,

GAIL, and BC within one model. This section is divided into six subsections: cumulative rewards, episode length, curiosity rewards, curiosity value estimate, GAIL rewards, and GAIL value estimate (GAILVE).

4.4.1. Cumulative rewards for the RL mixed approach

This section presents the findings concerning the application of the RL mixed approach to mobile crane path planning within a mixed GAIL, BC, and RL framework. The approach began with optimization of the hyperparameters as discussed in section 3.2.2. Figure 44 shows the cumulative rewards attained per episode, providing insights into the training process. The first observation in analyzing the rewards obtained by the mixed RL approach is the similarity between these results and those of the pure RL approach. The cumulative rewards graph is divided into three portions, which are the exploration, exploitation, and optimization phases. This is identical to the pure RL approach and in contrast to the GAIL approach, which has only two phases. The exploration phase, during which the agent continuously attempts to discover the environment through a simple cycle of trial and error, took approximately 1.74 million episodes to yield a positive solution. It is worth noting that the Pure RL model needed 2.25 million episodes to achieve the same task, which is a 0.5-million-episode improvement. Furthermore, the GAIL approach needed 3.15 million episodes to end the exploration phase.

The first task needed to advance the agent's training is locating the payload and attaching it to the hook; for this task, the agent took approximately 1.98 million episodes to master the tasks of locating the payload, successfully attaching it to the hook, and, most importantly, avoiding all obstacles to obtain a cumulative reward of 121.6 points. The first task completion criterion provided a more significant challenge for the pure RL model, which needed 2.73 million episodes to achieve a 107.5 reward, an additional 0.75 million episodes compared with the mixed RL approach. Finally, the GAIL model performed poorly in comparison with the mixed approach in terms of achieving the first task, where it took the agent 4.41 million episodes to gain 105.3 rewards; this indicates that the GAIL model requires a whopping 2.43 million episodes to achieve the same task as the mixed approach.

The next task, which coincides with the end of the exploration phase, is to compare the different models in terms of performance with respect to the given criteria. The mixed approach finished the exploration phase in 2.88 million episodes, with a total reward of 280. The rate of improvement

for the agent training in the episodes between 1.74 and 2.88 million for the mixed RL approach is 3327%, with a total improvement of 291.8 points. The other models fail to present similar improvements compared to the mixed RL approach, where the pure RL approach managed to finish the exploration phase in 3.21 million episodes and a total reward of 277.4 points. The rate of improvement for the pure RL approach between episodes 2.25 and 3.21 million is +2,337%, with a total reward improvement of 289.80 points. The GAIL model does not present an exploitation phase.

The last phase, which is the optimization phase, displayed similar results between the mixed RL and the pure RL approaches. In the mixed RL approach, the training procedure was run for 8 million episodes, 5.12 million of which were used for optimization. The agent optimized the solution found in episode 2.88 million of 280 points to an all-time high of 298.56 points in episode 8.37 million with an improvement rate of 5% or 15.531 points. The agent training was stopped in episode 8.5 million since the agent only managed to improve the performance by 1% from episode 4 million till episode 8.5 million. The pure RL model behaved similarly, and it managed to improve its results from 277.4 points in episode 3.21 million to 299.015 points in episode 9.9 million with a rate of improvement of 8%, which is more significant than the 5% improvement rate seen in the mixed approach. The training of the pure RL approach was allowed to run longer since it displayed better potential in the 5 million episodes, during which 2.12 points improved the results. Finally, the GAIL model could have performed better than both models since It presented a mixed phase of exploitation and optimization with which the maximum reward obtained is 263.69 points, which is far inferior to the other models. The training could have been continued indefinitely; however, it was observed that 8 million episodes was sufficient to understand the performance of the GAIL model, which performed poorly in all phases and tasks compared with the pure RL and mixed RL models.

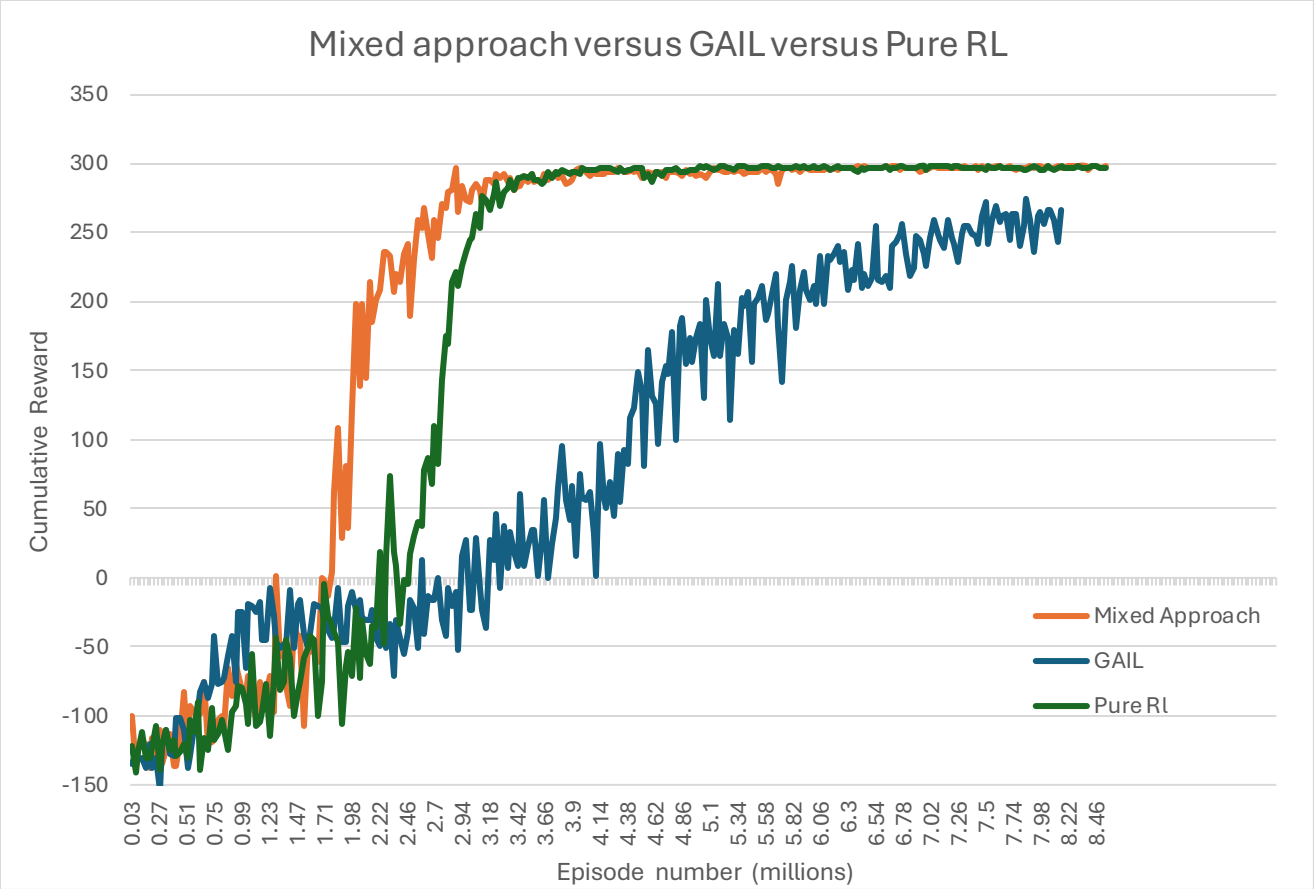


Figure 44: Cumulative rewards for mixed RL approach versus GAIL versus Pure RL.

4.4.2. Episode length in mixed RL model

This section provides an in-depth analysis of the episode length criterion within the context of the mixed RL model used for crane path planning. The episode length graph is divided into three portions, similar to the cumulative rewards graph. The first phase is the exploration phase, during which the agent attempts to discover the environment through various actions; this phase generally corresponds with an increase in episode duration, with the agent reaching an all-time high of 6,696.2 s within the first 1.55 million episodes. The agent is then incentivized to find a solution that improves the rewards; to do so, the agent decreases the episode length significantly within the exploitation phase, where the episode length reaches a value of 511.83 s within the next 1.32 million episodes. The episode length becomes 8% of the initial value in episode 1.56 million, constituting a 5,776.35 s decrease in episode length.

In the final phase, which is related to the optimization phase the agent continues to decrease the episode length up until the shortest episode length is obtained of 89.517 s is achieved in episode 8.37 million with an improvement of 412.3 s in episode length or an 81% improvement. However, in the last 0.37 million episodes the improvement rate was less impactful, where the episode length only decreased by 1.25 s, or a total improvement of 1%, in the 0.37 million last episodes. Additionally, after the best result was obtained, the agent only increased the episode length since it could not improve the solution found; it attempted to divert from that solution, which increased to 23.1 s, or a total increase of 24%, in the last 0.13 million episodes, indicating that the agent was no longer capable of improving the developed path.

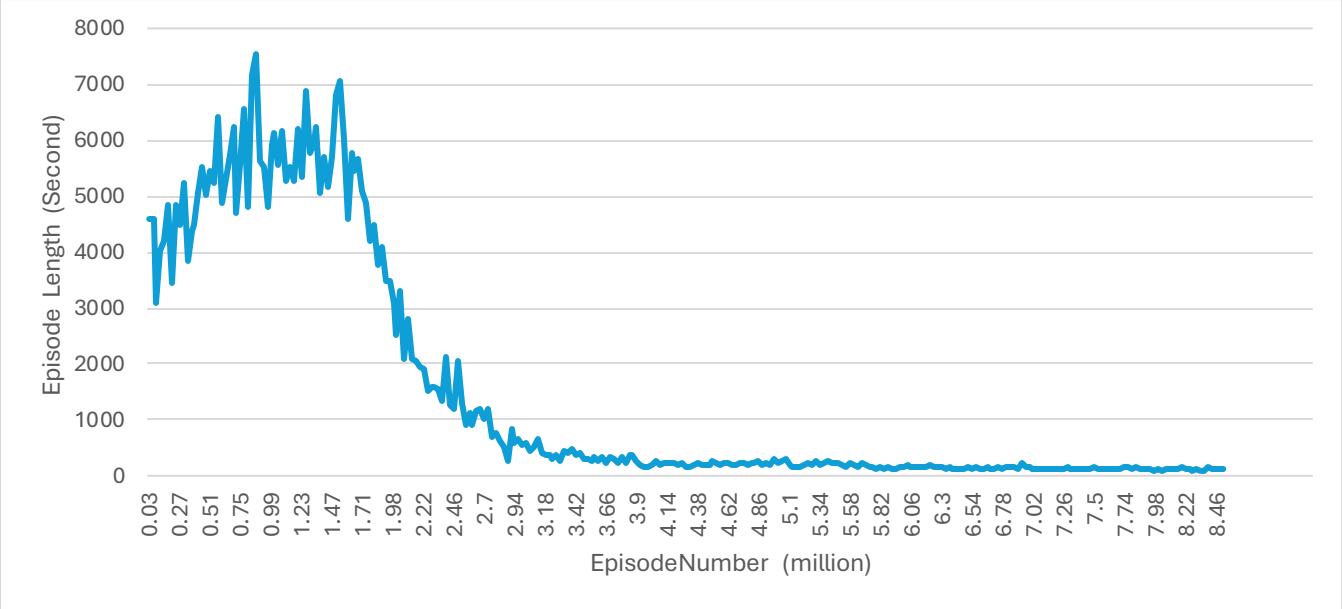


Figure 45: Episode length for mixed RL approach.

4.4.3. Extrinsic value estimate in mixed RL model

In the mixed RL model, the crane agent uses the EVE decision-making tool to forecast and assess the quality of its actions. The results of the EVE process within the mixed RL framework are depicted in Figure 46. As Figure 46shows, the mixed RL approach generates similar results to those of the pure RL approach in terms of EVE, where the graph is divided into three phases. The first phase is the exploration phase, where the agent tends to overestimate the rewards obtained through its actions, where the first 1.74 million episodes generally present estimate values superior to the actual rewards obtained in training. The overestimation ends in episode 1.74 million with an estimated reward of −5.484, an underestimate of the cumulative rewards obtained in the training.

The episodes following 1.74 million all present an underestimate of the cumulative reward; however, the graph follows the same trends found in the cumulative rewards graph, with the exploitation and optimization phases present in both graphs. However, the most significant estimate for the agent is seen in episode 8.37 million with a total reward estimate of 234.25 points, with the remaining 130,000 episodes having lower estimates of 229.066 points due to the lack of improvements that can be implemented on the developed path.

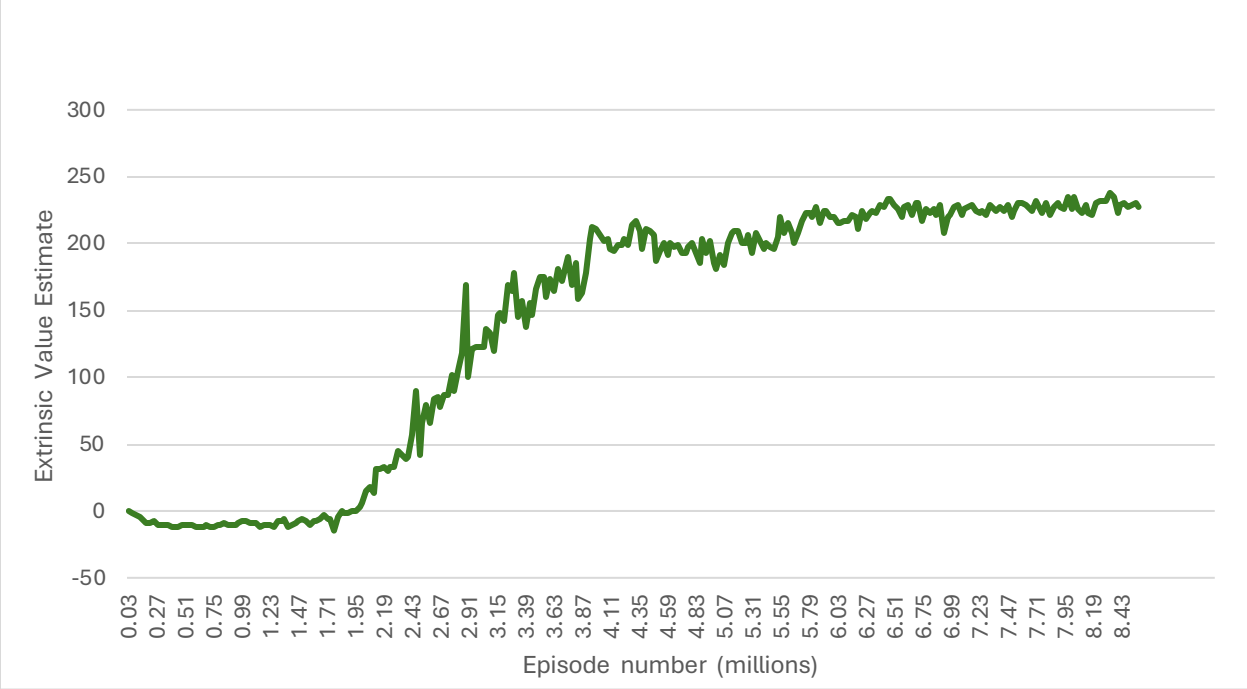


Figure 46: Extrinsic value estimate for mixed RL approach.

4.4.4. Curiosity rewards in mixed RL model

To gauge the effectiveness of curiosity-based exploration during training within the mixed RL framework, a curiosity reward indicator was employed that quantifies rewards solely derived from exploration. The impact of this indicator on training outcomes is illustrated in Figure 47.

In the developed model, the initial episodes of the agent are marked by a need for more environmental knowledge. During this learning phase, the agent's curiosity is pivotal in its ability to amass rewards. This phenomenon is explained within the mixed RL model, where the agent's initial 1.56 million episodes witnessed a notable curiosity-based rewards surge solely attributed to

the curiosity parameter, peaking at 13.63 points. However, after this initial surge, rewards stemming from curiosity experienced a sharp decline over the ensuing 7 million episodes until it reached an all-time low of 0.08 points in episode 8.46 million. The decline in curiosity-based rewards is related to the lack of better alternative paths that the agent could implement.

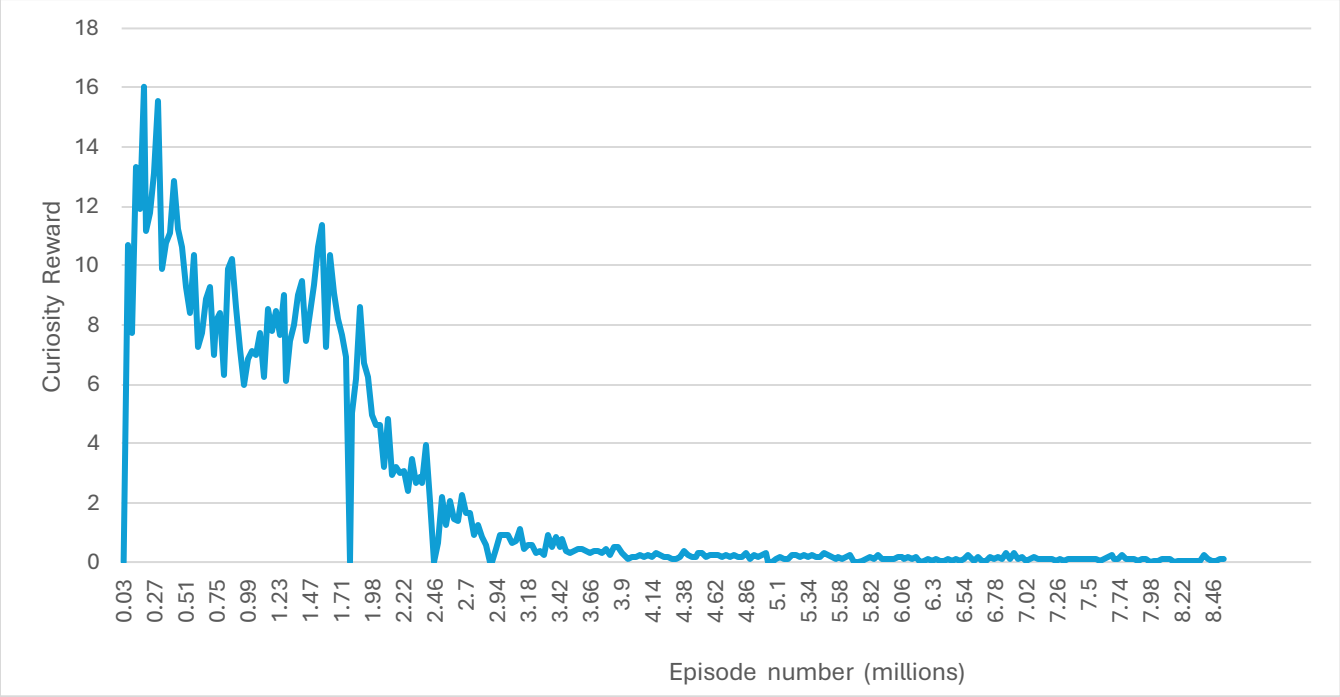


Figure 47: Curiosity rewards in mixed RL model.

4.4.5. Curiosity value estimate

The curiosity value estimate pertains to the agent's evaluation of the exploration's payoff in terms of rewards. Figure 48 shows the CVE for the Mixed RL model, where the general trend in CVE observed for models using GAIL is the underestimation of the curiosity-based rewards throughout the training process. This indicates that the curiosity rewards collection process is affected by other

parameters. These parameters are those related to GAIL—the GAIL expert estimate and the GAIL reward—which are discussed in the following sections.

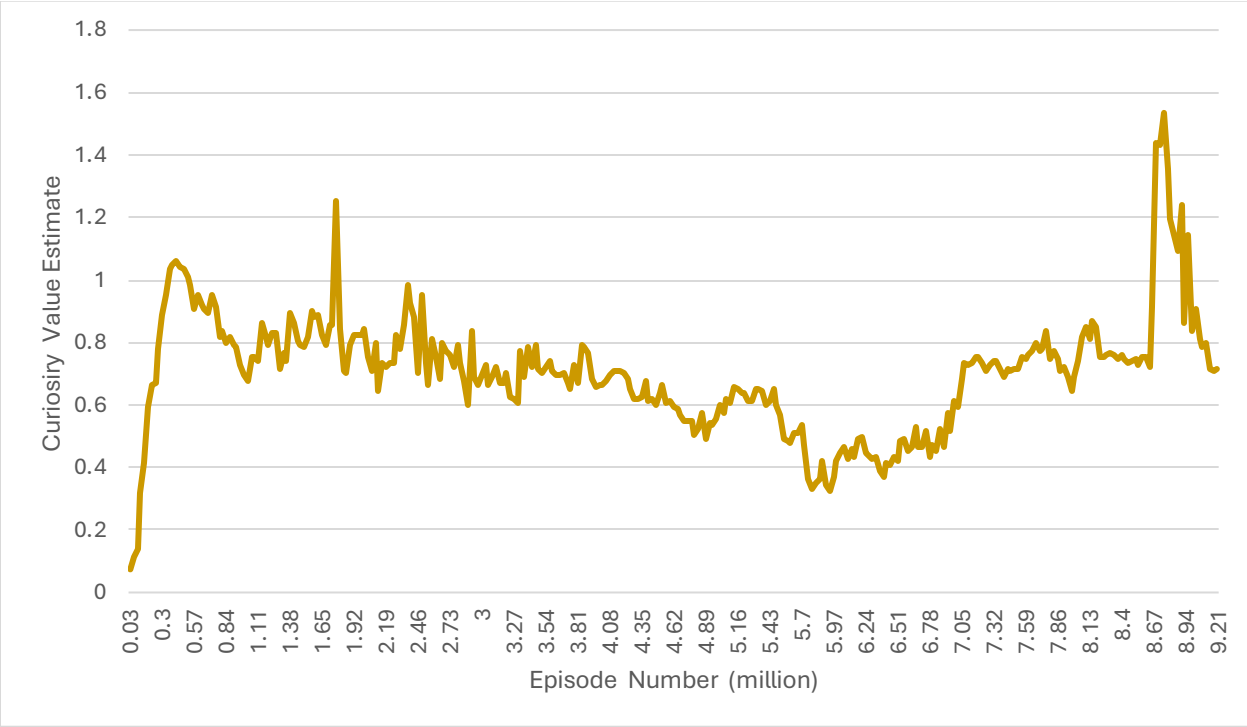


Figure 48: Curiosity value estimate for mixed RL model.

4.3.6. GAIL reward for mixed RL model

This section presents the findings for the application of the mixed RL model in mobile crane path planning. The results obtained after training are presented in Figure 49. The GAIL rewards figure 49indicates GAIL's impact on the rewards collection process, which can provide insight into the discriminator training process and its efficiency for training. During the first portion of the training, the agent is heavily reliant on the GAIL portion to develop a policy; the demonstrations are then used to accumulate the majority of the initial rewards, as can be seen in Figure 49, where the agent obtained a total of 76.63 points in episode 0.87 million. The 76.63 points of rewards obtained through GAIL are evened out with the cumulative penalties obtained through the RL portion, and a total reward of −90.42 is obtained. The fact that the cumulative rewards are negative despite the significant contribution of GAIL to the training is notable, considering that the RL model lags behind the GAIL model in the initial training portions. In the next 2.5 million episodes, the GAIL-based rewards decrease in value from 76.63 to 3.38 points, which the training can explain gained

through the RL model within the same period. The period between 0.87 and 3.4 million episodes can be explained by the increased reliance of the agent on the RL model as the primary reward-collecting tool. The reliance on the GAIL algorithm continues to drop in the last 5 million episodes until it reaches an all-time low of 0.88 points in episode 7.62 million. The pattern in Figure 48 indicates that the agent initially focuses on replicating the demonstrations as a primary approach to collect rewards once the discriminator and the agent are trained efficiently. The agent becomes less reliant on the provided demonstration and becomes more reliant on the collected information from the RL model.

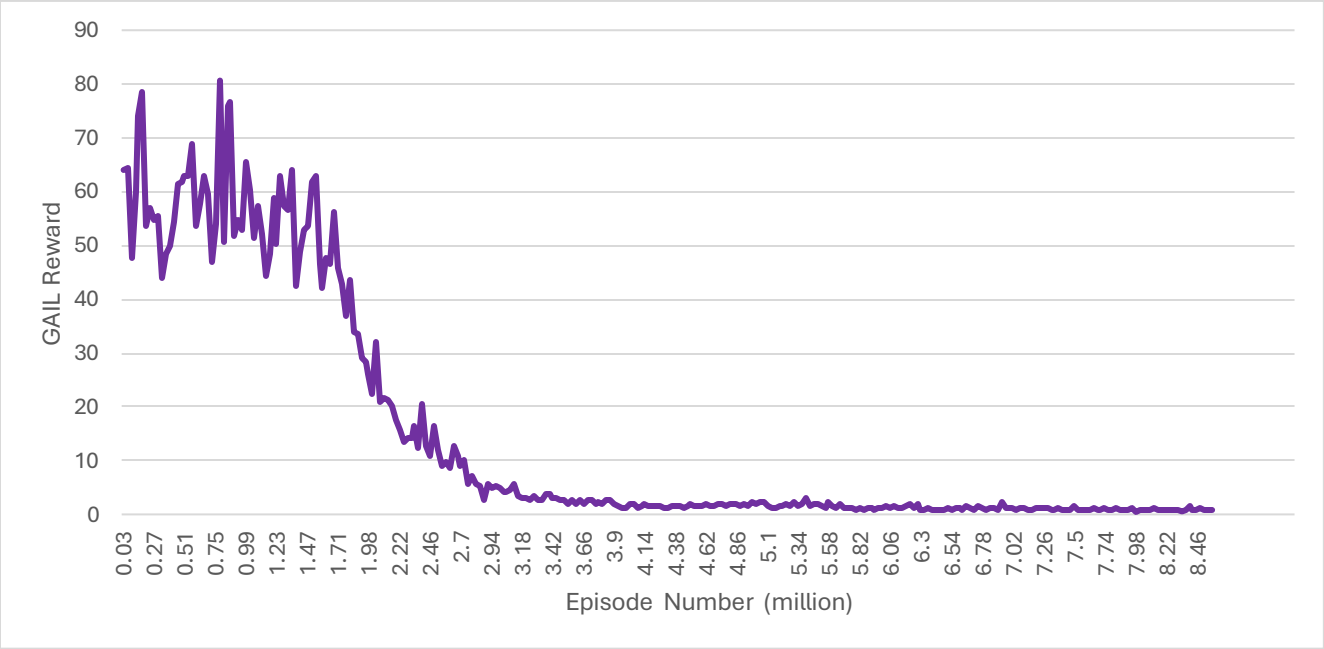


Figure 49: GAIL rewards for mixed RL model.

4.4.7. GAIL value estimate

To gauge the effectiveness of the GAIL parameter on the mixed RL model, the agent makes a set of predictions based on the rewards that could be obtained from the GAIL algorithm. The GAIL estimate values can be divided into two phases (see Figure 50). The first phase is an increased GAIL rewards estimate, where the first 0.24 million represent an overall positive trend of 0.74, which suggests the GAIL model would obtain a large number of rewards from the GAIL algorithm which coincides with the GAIL rewards, the agent continues to rely on the GAIL algorithm to improve its performance up until episode 2.7 million. The second phase is the phase where the agent begins to rely less on the GAIL model and more on the RL algorithm, where the value estimate starts to

decrease significantly from 0.48 to an all-time low of 0.062 in episode 8.37 million, which correlates with the data presented in the cumulative rewards and the GAIL rewards sections.

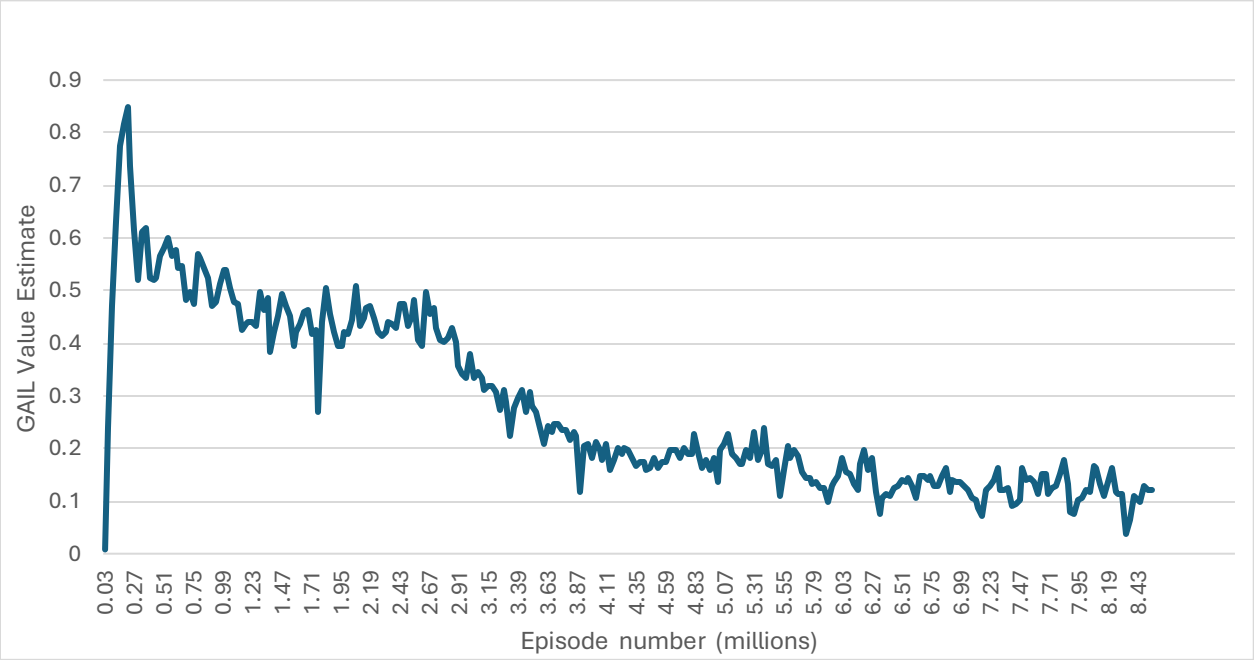


Figure 50: GAIL value estimate for mixed RL approach.

4.5. Curriculum learning approach

4.5.1. Overview

The CL approach is a branch of RL, where the agent is expected to train on a given task in small and more straightforward portions; after the agent manages to complete the subtasks efficiently, the difficulty level is increased, and a new, more difficult task is unlocked to train the agent, In order to understand the curriculum used to train the agent, please refer to section 2.2.2, where the overall curriculum is described in detail. Besides the progressive nature of the CL approach, it is identical to the pure RL approach; thus, the hyperparameters used for the pure RL are used in the CL one.

4.5.2. Cumulative rewards

This section presents the findings for the implementation of the CL approach to mobile crane path planning. The approach involved optimization of hyperparameters as discussed in section 3.2.2. Figure 51 and Table 10 show the cumulative rewards attained per episode, giving insights into the training process. It is worth noting that the agent is assigned nine subtasks of varying difficulty; the agent's efficiency in completing the tasks is discussed below. Across the initial 0.51 million episodes, the agent was capable of achieving significant results in terms of the first task, with a maximum reward of 232.22 points; the rewards also displayed some initial fluctuations due to the initial exploration but managed to develop a valid policy within 0.51 million episodes. After completing subtask 1, the agent unlocks subtask 2; since the agent must start the training for a different task, the rewards drop significantly in episode 0.54 million to 189.52 points. However, the agent quickly finished the second task with 0.09 million episodes, accumulating 16% improvements or the equivalence of 226.935 points, effectively achieving subtask number two in episode 0.63 million.

Subtask 3 was initiated in Episode 0.66 million, and, similar to in the case of subtask 2, there was an initial loss of rewards to a low of 185.62 points. However, the agent continued to struggle to achieve the task for an additional 0.03 million episodes, receiving the lowest reward in subtask 3 with a total of 168.34. The next 0.09 million episodes produced significantly better results, with the agent obtaining the best reward for subtask 3 in episode 0.78 million, with a total reward of 239.392, or the equivalent of a 22% improvement in 0.12 million episodes.

Subtask 4 was initiated in Episode 0.81 million, and, similar to in the case of subtask 2, there was an initial loss of rewards to a low of 180.098 points. However, the agent managed to improve its performance in the next 0.15 million episodes. The agent obtained the best reward for subtask 4 in episode 0.96 million, with a total reward of 242.71, or a 25% improvement in 0.15 million episodes.

Subtask 5 was initiated in Episode 0.99 million. However, this subtask was more challenging for the agent to achieve, and the results fluctuated accordingly. Besides the initial results dip of 202.646 points, the agent struggled for an additional 0.12 million episodes, reaching a low of 169.26 in episode 1.11 million. The next 0.24 million episodes displayed significant improvements

in terms of rewards, with a total reward of 241.187 points or the equivalent of 16% improvement in 0.36 million episodes. It is worth noting that during this phase, the agent achieved the task successfully. However, it did not master the task or stabilize its learning process; the consequences of this are reflected in the subsequent subtasks.

In episode 1.38 million, subtask 6 is initiated. This subtask requires only 0.03 million episodes to terminate with a final reward of 226.456; however, this is a negative improvement of −3% relative to episode 1.38 million. In this subtask, the agent managed to complete the task quickly due to its similarity to the previous subtask. However, the agent is far from mastering this task, and the consequences of that are seen in subtask 7.

Table 10: Summary of cumulative rewards for curriculum learning model.

Subtask	Start episode (m)	End episode (m)	Start reward	End reward	Lowest reward (LR)	Episode number for LR	Improve-ment rate
Subtask 1	0.00	0.51	−141.98	232.22	−161.15	0.15	N/A
Subtask 2	0.54	0.63	189.52	226.94	189.52	0.54	16%
Subtask 3	0.66	0.78	185.62	239.39	168.34	0.69	22%
Subtask 4	0.81	0.96	180.98	242.71	180.10	0.81	25%
Subtask 5	0.99	1.35	202.46	241.19	169.26	1.11	16%
Subtask 6	1.38	1.41	232.52	226.46	226.46	1.41	−3%
Subtask 7	1.41	1.86	226.46	237.78	62.69	1.47	5%
Subtask 8	1.89	4.47	51.92	232.06	−27.56	2.04	78%
Subtask 9	4.50	4.53	234.70	251.42	234.70	4.50	7%
Remaining tasks	4.53	10.00	251.42	286.88	226.10	6.15	12%

Subtask 7 required a total of 0.45 million episodes, which is longer than any previous subtasks besides the first. In this subtask, the agent realized that the lift strategy used in previous subtasks is suboptimal for the new geometrical configuration present in subtask 7. Initially, the agent starts

training following the same approach as previous episodes. To that extent, a significant loss of rewards can be seen in episode 1.47 million, where the cumulative rewards collected are 62.685 points, a 160-point decrease from episode 1.38 million. In the following 0.39 million episodes, the agent began developing a new policy that improved the rewards collection process until reaching 237.78 points in episode 1.86 million, equivalent to a 5% improvement from episode 1.38 million.

Subtask 8 also presented a significant challenge for the agent, where the rewards dropped instantly in episode 1.89 million to 51.92 points. The agent struggled significantly until it reached an all-time low of −27.564 points in episode 2.04 million, which suggests that the agent was completely lost in the initial 0.15 million episodes of subtask 8. The agent then decided to change the lifting approach for a second time to fit the new geometric configuration and managed to improve the rewards, where a reward of 232.09 was achieved by episode 4.47 million, marking an improvement rate of 78% in 2.58 million episodes. It is also worth noting that subtask 8 was the most challenging for the agent and took the most significant number of episodes compared to any other subtasks.

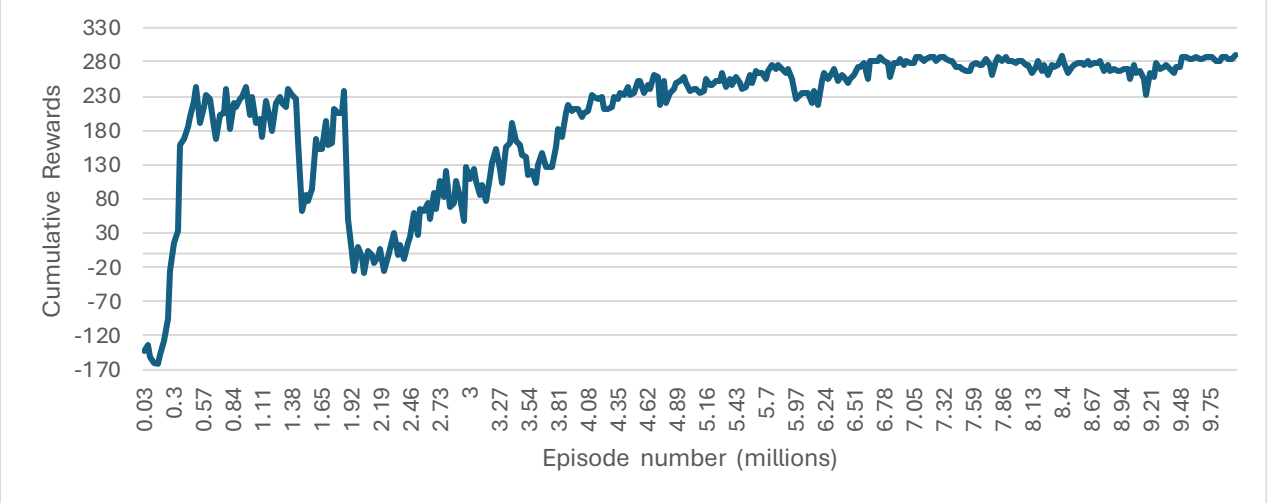


Figure 51: Cumulative rewards for curriculum learning approach.

The ninth and final subtasks were short since the agent spent a long-time training in the previous subtask, and both final subtasks are similar in geometrical disposition. The rewards improved to 251.42 points by episode 4.53 million.

The rest of the training followed the same pattern as that of the pure RL approach, where the agent initially exploited the results and then began the final optimization phase. The best solution obtained through the CL approach was a reward of 287.293 points (in episode 9.78 million). The training is then terminated by episode 10 million since the CL approach proved its effectiveness, and both the pure RL and the CL approach were compared based on 10 million episodes. It is worth noting that the pure RL approach performed better, as described in previous sections. A graphical comparison between the two approaches is provided in Figure 52.

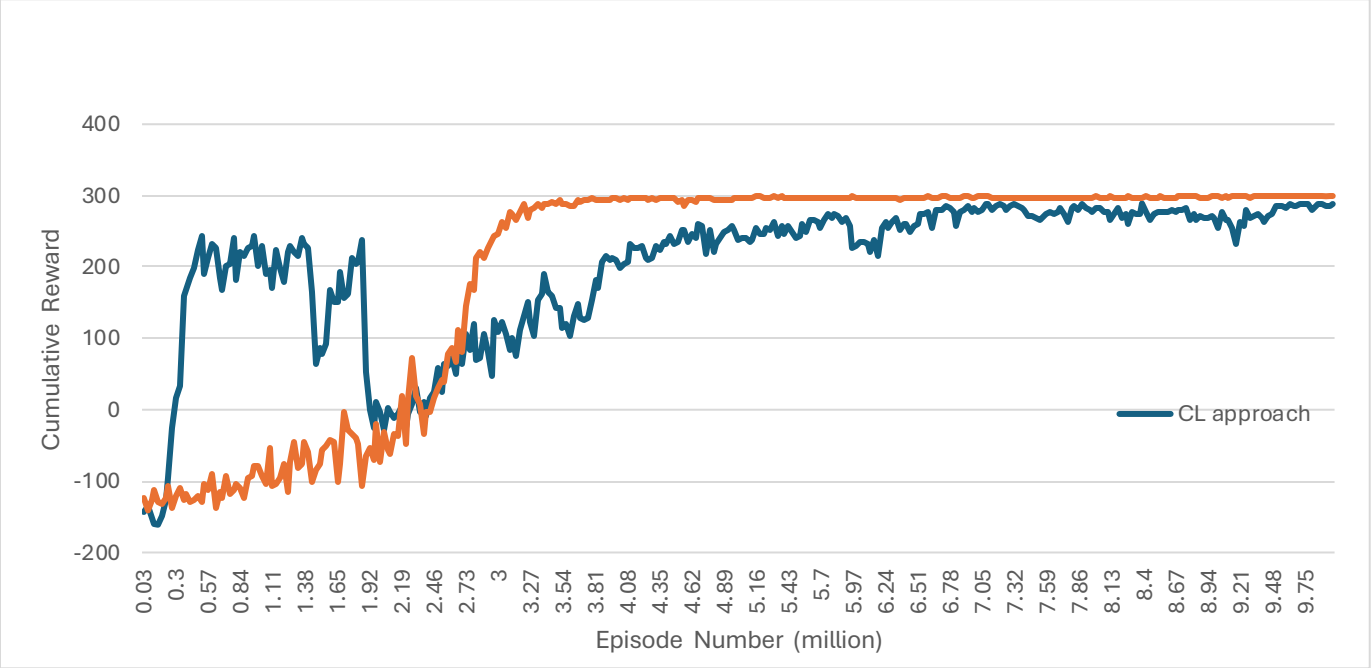


Figure 52: Cumulative rewards—curriculum learning versus pure reinforcement learning.

4.5.3. Episode length in CL model

This section provides an in-depth analysis of the episode length criterion within the context of the CL model used for crane path planning. The main objective of the agent in terms of episode length is to minimize it. Due to the nature of the CL model, during the agent's learning of eight separate subtasks, there is a significant variation in episode length among the different subtasks.

The results of the CL model are presented in Figure 52 and also summarized in Table 10. As can be seen, across the initial 0.51 million episodes, the agent was capable of achieving significant results in terms of the first task, with a minimum episode length of 2,413.3 s; the episode length also displayed some initial fluctuations due to the initial exploration but managed to develop a valid policy within 0.51 million episodes. After finishing subtask 1, the agent unlocks subtask 2. Since the agent must start the training for a different task, the episode length slightly increased in episode 0.54 million to 2,416.05 s. However, the agent quickly finished the second task with 0.09 million episodes, accumulating 17% improvements or the equivalence of 2,071.5 s, effectively achieving subtask number two in episode 0.63 million.

Subtask 3 was initiated in Episode 0.66 million; unlike in the case of subtask 2, here the agent immediately began decreasing the episode length, obtaining a maximum episode length of 1,749.55 s, or a 28% improvement in 0.12 million episodes. Subtask 4 was initiated in episode 0.81 million, and, similar to in subtask 3, the agent managed to decrease the episode length further to reach 1,319.08 s in episode 0.96 million, with a 33% improvement in 0.15 million episodes.

Subtask 5 was initiated in Episode 0.99 million. However, this subtask was more challenging for the agent to achieve, and the results fluctuated accordingly. Notwithstanding the initial results increase of 1,334.115 s, the agent continued to struggle for an additional 0.12 million episodes, reaching a high of 1,487.9 s in episode 1.11 million. The next 0.24 million episode displayed significant improvements in terms of episode length with a score of 1,170.575 s or the equivalent of 14% improvement in 0.36 million episodes. It is worth noting that during this phase, the agent achieved the task successfully. However, it did not master the task or stabilize its learning process; the consequences of this are reflected in the subsequent subtasks. In episode 1.38 million, subtask 6 was initiated; this subtask required only 0.03 million episodes to terminate, with a final episode length of 1,203.925 s. However, there were no improvements in episode 1.38 million. In this subtask, the agent managed to complete the task quickly due to its similarity to the previous subtask. However, the agent is far from mastering this task, and the consequences of that are seen in subtask 7.

Subtask 7 required a total of 0.45 million episodes, which is longer than any previous subtasks besides the first. In this subtask, the agent realized that the lift strategy used in previous subtasks is suboptimal for the new geometrical configuration present in subtask 7. Initially, the agent starts training following the same approach as previous episodes. To that extent, a significant loss of

episode length can be seen in episode 1.56 million, where the agent needed an average of 2,850.2 s to finish an episode. In the following 0.3 million episodes, the agent began developing a new policy, which improved the episode length until reaching 1,466.4 s in episode 1.86 million, equivalent to a −18% improvement from episode 1.38 million.

Subtask 8 also presented a significant challenge for the agent, where the episode length increased instantly in episode 1.89 million to 2,147.405 s. The agent struggled significantly until it reached an all-time high of 3,675 s in episode 2.52 million, which suggests that the agent was completely lost in the initial 0.63 million episodes of subtask 8. The agent then decided to change the lifting approach for a second time to fit the new geometric configuration and improved the episode length with a score of 1,596.79 s, achieved by episode 4.47 million, marking an improvement rate of 34% in 2.58 million episodes. It is also worth noting that subtask 8 was the most challenging for the agent and took the largest number of episodes compared to any other subtasks. The ninth and final subtasks were short since the agent took a long-time training in the previous subtask, and both final subtasks are similar in geometrical disposition. The episode length only improved the episode initially but then increased the episode length to 1,055 s.

The rest of the training followed the same pattern as that of the pure RL approach, where the agent initially exploited the results and then began the final optimization phase. The best solution obtained through the CL approach was an episode length of 547.35 s (in episode 9.78 million). The training is then terminated by episode 10 million since the CL approach proved its effectiveness, and both the pure RL and the CL approach were compared based on 10 million episodes. It is worth noting that the pure RL approach performed better in terms of episode length, as described in section 4.23.

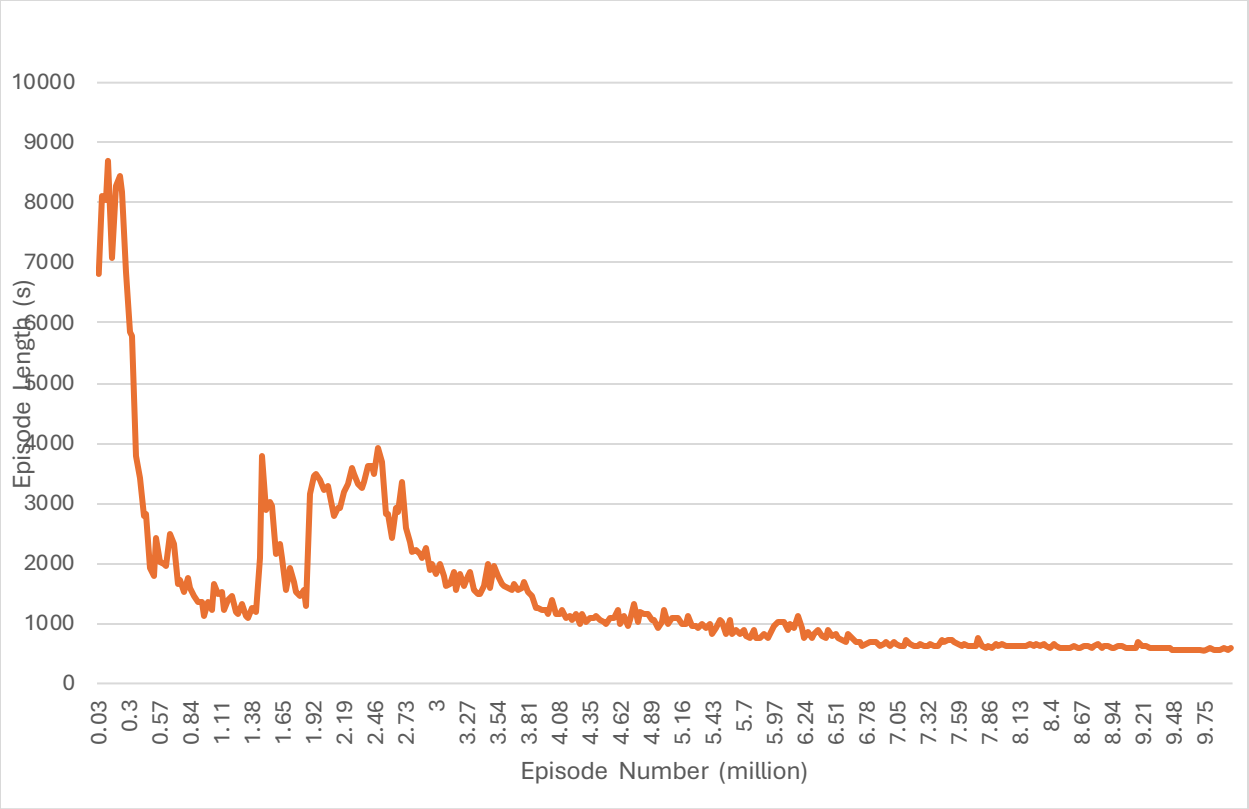


Figure 53: Episode length for CL model.

Table 11: Episode length summary for curriculum learning model.

Subtask	Start episode (mil)	End episode (mil)	Start Episode length(sec)	End Episode length(sec)	Longest episode length (LEL)	Improvement rate
Subtask 1	0.00	0.51	8,160.00	2,413.32	8.16	N/A
Subtask 2	0.54	0.63	2,416.05	2,071.50	2,416.05	17%
Subtask 3	0.66	0.78	2,237.00	1,749.55	2,237.00	28%
Subtask 4	0.81	0.96	1,757.07	1,319.08	1,757.07	33%
Subtask 5	0.99	1.35	1,334.12	1,170.58	1,487.92	14%
Subtask 6	1.38	1.41	1,208.41	1,203.93	1,208.41	0%

Subtask 7	1.41	1.86	1,203.93	1466.40	2,850.21	−18%
Subtask 8	1.89	4.47	2,147.41	1058.90	3,678.50	34%
Subtask 9	4.50	4.53	1,037.42	1,057.58	1,057.57	−2%
remaining tasks	4.53	10.00	1,057.58	569.19	1,156.75	86%

4.5.4. Extrinsic value estimate

The crane agent uses the EVE decision-making tool to forecast and assess the quality of its actions. Its primary objective is to maximize cumulative rewards while maintaining a balance between short-term and long-term decision-making processes, which is crucial for developing successful policies to accomplish tasks. The EVE process results are depicted in Figure 54 and also summarized in Table 12.

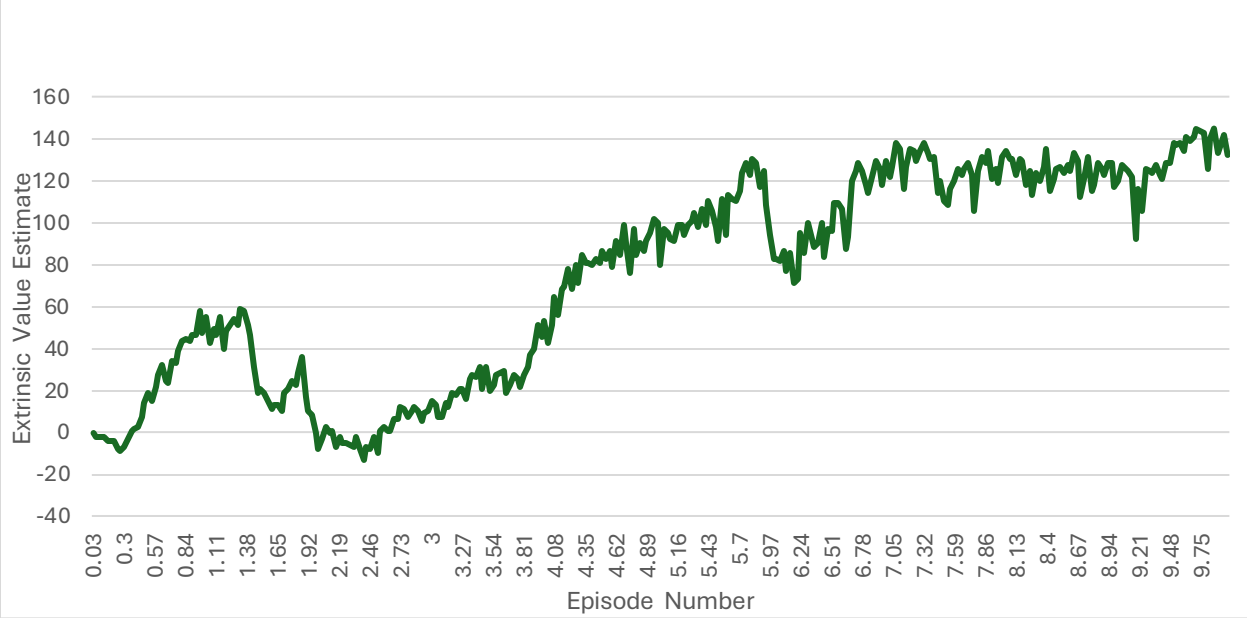


Figure 54: Extrinsic reward estimate in curriculum learning model.

The agent followed the same trends seen in the CL model's cumulative rewards section, where the agent initially overestimated the rewards obtained in the first 0.3 million episodes, after which the agent became better at predicting the rewards obtained. The improved predictions on the part of the agent can be seen in Figure 54, where both the estimates and the cumulative rewards are compared. It is also worth noting that the EVE is always conservative in its estimates compared to

the cumulative rewards graph, which can be attributed to various factors, including approximation errors, uncertainty, and the inherent trade-offs between exploitation and exploration.

Table 12: Extrinsic value estimate summary for curriculum learning model.

Subtask	Start episode (mil)	End episode (m)	Start episode EVE	End episode EVE	Lowest EVE	Improvement rate
Subtask 1	0.00	0.51	0.00	11.59	−7.16	N/A
Subtask 2	0.54	0.63	12.90	25.19	12.90	49%
Subtask 3	0.66	0.78	24.76	33.40	24.26	26%
Subtask 4	0.81	0.96	37.59	49.61	37.59	24%
Subtask 5	0.99	1.35	48.64	55.48	46.27	12%
Subtask 6	1.38	1.41	53.84	50.93	50.93	−6%
Subtask 7	1.41	1.86	50.93	28.85	12.76	−77%
Subtask 8	1.89	4.47	23.92	80.37	−8.94	70%
Subtask 9	4.50	4.53	82.79	82.78	82.78	0%
remaining tasks	4.53	10.00	82.78	136.06	76.53	39%

4.5.5. Curiosity rewards and curiosity value estimate for CL model

Like the pure RL model, the CL model uses curiosity for exploration. The results for both the curiosity rewards and the curiosity value estimate are presented in Figures 55 and 56, respectively. The behaviour of the curiosity parameter in the CL model follows similar trends seen in the pure RL model, where the agent relies heavily on curiosity to discover the environment and the action space extensively. Once the agent improves its rewards, it begins to rely less on the curiosity parameters. In the case of the CL model, Figure 55 shows that the fluctuation is related to the agent's increased reliance on curiosity at the start of each subtask, followed by a decrease until either the subtask is finished or the overall training is complete.

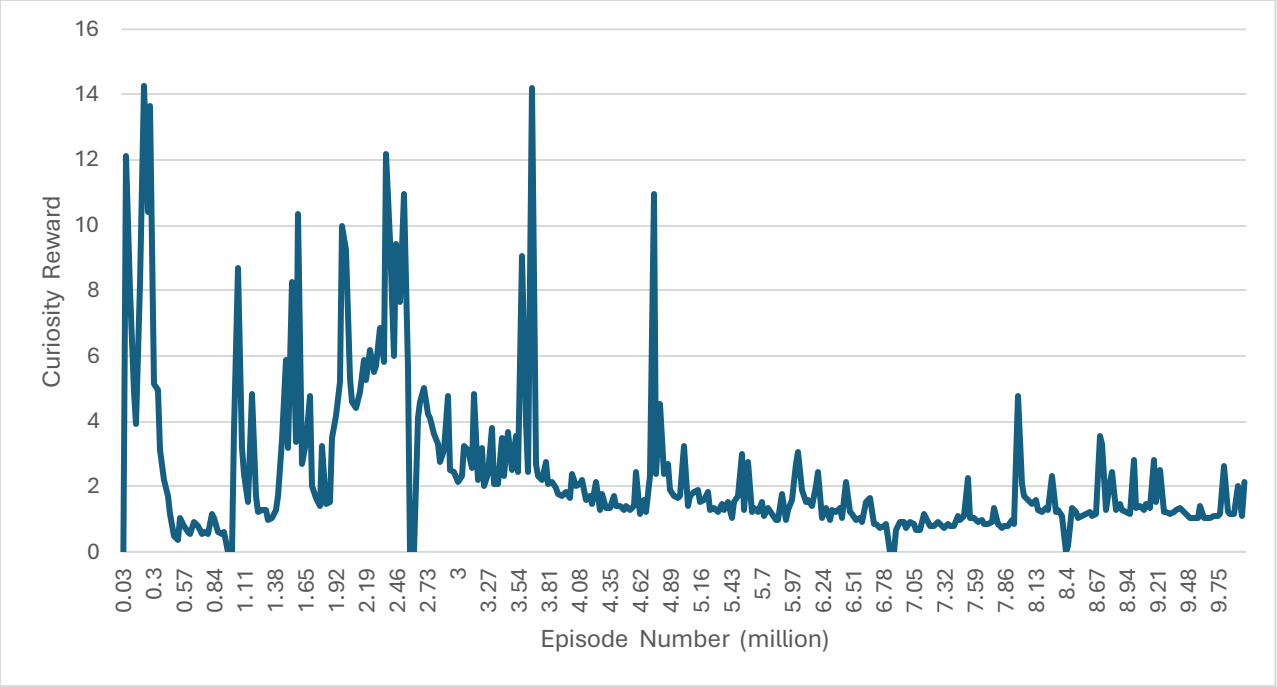


Figure 55: Curiosity rewards for curriculum learning approach.

The curiosity value estimate, on the other hand, is less affected by the subtasks since the CVE follows a more conservative approach. Thus, it allows for a more stable set of predictions, as can be seen in Figure 56. This stability allows the agent to acquire better rewards from the curiosity model.

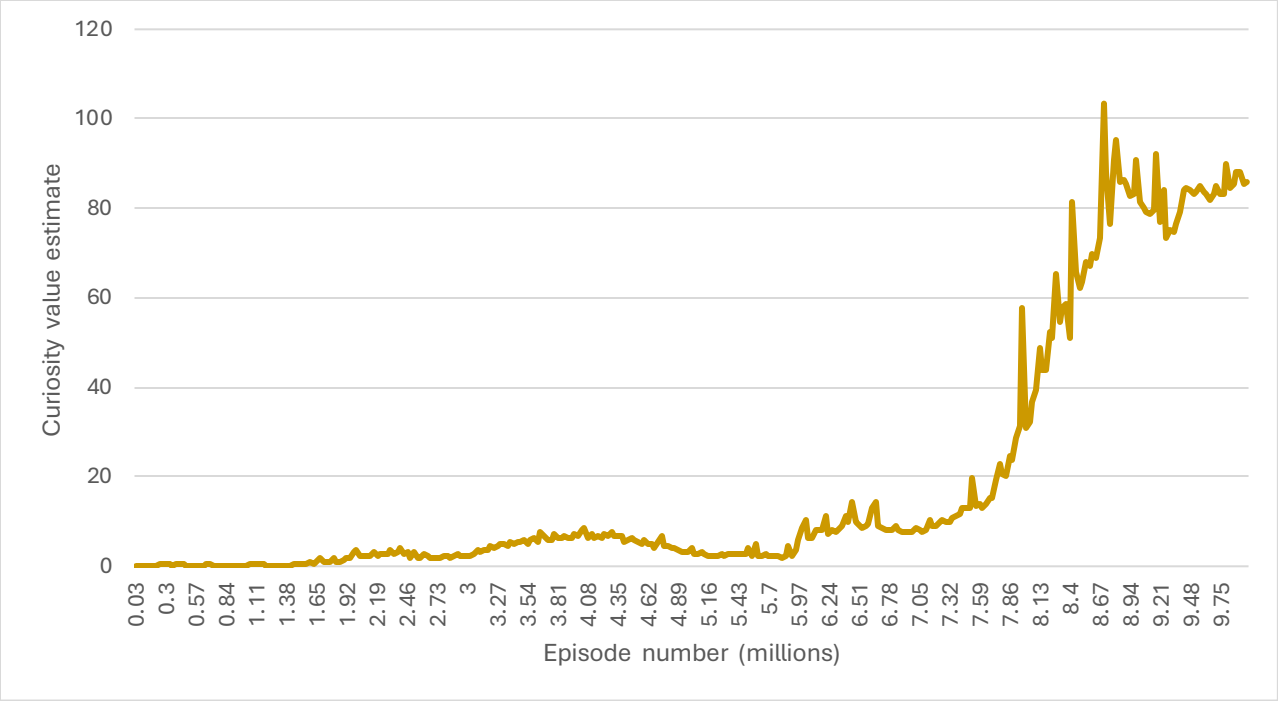


Figure 56: Curiosity value estimate for CL model.

5. CONCLUSION AND FUTURE WORK

5.1. General conclusion

This chapter summarizes the research contributions while discussing future work and the limitations of the current methodology.

This work presents a novel approach to solving the path planning problem for mobile cranes. The developed approach employs reinforcement learning (RL) alongside virtual reality (VR) and game engines to generate a detailed, realistic simulation of crane path planning. Three types of learning are used to train an agent to perform the lifting task: pure RL, imitation learning, and curriculum learning (CL). Additionally, the integration of different RL approaches to optimize the results obtained through each approach individually is explored. The importance of this work has to do with its impact on lift efficiency, safety, and lifting time. In terms of safety, the developed approach accounts for the complex dynamics of construction sites and mitigates the risk of incidents by simulating all possible scenarios and training the crane agent to take the right actions in each scenario. In terms of efficiency, the developed approach can be expected to significantly reduce planning, lifting, air time, and lift complexity. It is shown to be capable of generating optimal paths for lift planning, ensuring the shortest and most efficient route. In terms of energy consumption, the proposed tool minimizes lifting time and thereby decreases fuel consumption.

5.2. Research contributions

The research contributions can be summarized as follows:

- Integration of VR Simulations and RL: This study pioneers integrating VR simulations with RL techniques specifically tailored for crane operations. This innovative approach provides a deeper understanding of optimal solutions within complex and dynamic construction environments.
- Incorporation of Advanced Machine Learning Techniques: This work is the first to incorporate RL, CL, and Imitation Learning techniques within VR simulations and game engines for crane operations and path planning. This unique integration enhances the research's practicality and effectiveness in addressing complex challenges such as optimal path planning and adaptation to dynamic construction environments.

- Precise solutions considering environmental factors: The methodology, incorporating 3D components, identifies precise solutions considering time constraints, complex movements, and realistic scenarios. This ensures practical and reliable lift paths for real-world construction settings.
- Enhanced exploration and adaptation with RL: The methodology enhances exploration within construction sites through RL, particularly in automated path planning for mobile cranes. This adaptive decision-making capability is essential for coping with dynamic site conditions and frequent changes in construction projects.
- Integrating VR simulations and RL facilitates a thorough evaluation of alternatives and leads to selecting the most optimal solutions for crane operations and path planning. This, in turn, significantly boosts efficiency and safety, thereby reducing errors and delays in construction projects.
- Filling the gap in automated path planning: The methodology addresses the gap in fully autonomous path-planning methods for mobile cranes and dynamic construction settings. It develops a framework that adapts to changes in the built environment during construction, contributing to better solutions for complex and dynamic building projects.

These contributions advance the state-of-the-art in crane operations and path planning methodologies, offering practical solutions to improve efficiency, reliability, and adaptability in construction site operations.

5.3. Research limitations

This study was subject to some limitations that point to opportunities for further improvement in future work.

- One of the limitations of this research is the need for more consideration for weather conditions such as wind, rain, and snow, which can significantly affect crane operations and path planning in real-world construction sites. Future studies should incorporate weather-related simulations to better understand their effects on crane performance and safety.
- While the work is extensively evaluated in a virtual environment, further comparisons with real-life scenarios are necessary. Real-world construction sites present challenges

and complexities that virtual simulations may only partially capture. Conducting experiments and validations in actual construction settings can provide valuable insights into the practical applicability of the proposed methodologies.

- Current research primarily focuses on mobile cranes, neglecting other types commonly used in construction activities, such as tower or overhead cranes, which may have distinct operational considerations and challenges. Future studies should include a broader range of crane types to better understand crane operations and path-planning strategies.

5.4. Future work

This work marks an initial step towards creating a completely autonomous crane that can perform lifts without human intervention. The main objective of future work will be to develop a digital twin of a mobile crane and construction site. This digital twin will mirror changes occurring at the physical construction site in the virtual environment, allowing the agent to adjust accordingly during lift operations. Several areas of research and development can be pursued to improve autonomous crane operations:

1. Incorporating Weather Conditions Simulation: Future research should focus on integrating weather condition simulations (such as wind, rain, and snow) into the virtual environment. Understanding the impact of adverse weather on crane operations is critical to developing robust and resilient autonomous lifting strategies.

2. Validation in Real-Life Scenarios: Further comparisons and validations in actual construction sites are necessary. Field trials and experiments with autonomous cranes will provide valuable data and insights into the performance and practicality of the developed methodologies.

3. Extension to Various Crane Types: Expanding the scope of research to include various types of cranes beyond mobile cranes is essential. Towers, overhead cranes, and specialized lifting equipment pose unique challenges and require tailored autonomous operation strategies. Investigating these crane types will contribute to a more comprehensive understanding of autonomous crane operations.

4. Development of Adaptive Digital Twin: Enhancing the digital twin capabilities to create a dynamic and adaptive environment is a promising area for future research. Integrating real-time data from construction sites into the digital twin and enabling the agent to adapt to changes

dynamically will improve the accuracy and reliability of autonomous lifting operations. By addressing these future research directions, the advancement towards fully autonomous crane operations will be accelerated, paving the way for safer, more efficient, and more adaptable construction practices.

REFERENCES

Aghajamali, K., Nekouvaght Tak, A., Taghaddos, H., Mousaei, A., Behzadipour, S., & Hermann, U. (2023). Planning of Mobile Crane Walking Operations in Congested Industrial Construction Sites. Journal of Construction Engineering and Management, 149(7). https://doi.org/10.1061/jcemd4.coeng-13109

Ali, G. M., Asce, S. M., Ahmed Bouferguene, ;, Al-Hussein, M., & Asce, M. (2023). Crane Mat Layout Optimization Based on Agent-Based Greedy and Reinforcement-Learning Approach. https://doi.org/10.1061/JCEMD4

Ali, M. S. A. D., Babu, N. R., & Varghese, K. (2005). Collision Free Path Planning of Cooperative Crane Manipulators Using Genetic Algorithm. Journal of Computing in Civil Engineering, 19(2), 182–193. https://doi.org/10.1061/(ASCE)0887-3801(2005)19:2(182)

Boutouhami, K., Bouferguene, A., Lemouchi, R., Assaf, M., AL-Hussein, M., & Kosa, J. (2023). Hybrid Approaches For Handling Mobile Crane Location Problems In Construction Sites. 2023 Winter Simulation Conference (WSC), 2722–2733. https://doi.org/10.1109/WSC60868.2023.10407242

BuHamdan, S., Alwisy, A., & Bouferguene, A. (2020). Explore the application of reinforced learning to support decision making during the design phase in the construction industry. Procedia Manufacturing, 42, 181–187. https://doi.org/10.1016/j.promfg.2020.02.068

Cai, P., Cai, Y., Chandrasekaran, I., & Zheng, J. (2016). Parallel genetic algorithm based automatic path planning for crane lifting in complex environments. In Automation in Construction (Vol. 62, pp. 133–147). Elsevier. https://doi.org/10.1016/j.autcon.2015.09.007

Canadian Standards Association. (2019). Design of steel structures.

Dutta, S., Cai, Y., Huang, L., & Zheng, J. (2020). Automatic re-planning of lifting paths for robotized tower cranes in dynamic BIM environments. Automation in Construction, 110, 102998. https://doi.org/https://doi.org/10.1016/j.autcon.2019.102998

ElNimr, A., Fagiar, M., & Mohamed, Y. (2016). Two-way integration of 3D visualization and discrete event simulation for modeling mobile crane movement under dynamically

changing site layout. Automation in Construction, 68, 235–248. https://doi.org/10.1016/j.autcon.2016.05.013

Government of Canada. (2020). National Building Code.

Hayashi, K., & Ohsaki, M. (2020). Reinforcement Learning and Graph Embedding for Binary Truss Topology Optimization Under Stress and Displacement Constraints. Frontiers in Built Environment, 6. https://doi.org/10.3389/fbuil.2020.00059

Hu, S., Fang, Y., & Bai, Y. (2021a). Automation and optimization in crane lift planning: A critical review. Advanced Engineering Informatics, 49, 101346. https://doi.org/https://doi.org/10.1016/j.aei.2021.101346

Hu, S., Fang, Y., & Bai, Y. (2021b). Automation and optimization in crane lift planning: A critical review. Advanced Engineering Informatics, 49, 101346. https://doi.org/https://doi.org/10.1016/j.aei.2021.101346

Jeong, J.-H., & Jo, H. (2021). Deep reinforcement learning for automated design of reinforced concrete structures. Computer-Aided Civil and Infrastructure Engineering, 36(12), 1508–1529. https://doi.org/https://doi.org/10.1111/mice.12773

Kayhani, N., Taghaddos, H., Mousaei, A., Behzadipour, S., & Hermann, U. (2021a). Heavy mobile crane lift path planning in congested modular industrial plants using a robotics approach. Automation in Construction, 122. https://doi.org/10.1016/j.autcon.2020.103508

Lemouchi, R., Boutouhami, K., Bouferguene, A., & Al-Hussein, M. (2023, May). Reinforcement learning for design.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). Continuous control with deep reinforcement learning. http://arxiv.org/abs/1509.02971

Mousaei, A., Taghaddos, H., Nekouvaght Tak, A., Behzadipour, S., & Hermann, U. (2021). Optimized Mobile Crane Path Planning in Discretized Polar Space. Journal of Construction Engineering and Management, 147(5). https://doi.org/10.1061/(asce)co.1943-7862.0002033

Pathak, D., Agrawal, P., Efros, A. A., & Darrell, T. (2017). Curiosity-driven Exploration by Self-supervised Prediction. http://arxiv.org/abs/1705.05363

Reddy, H. R., & Varghese, K. (2002). Automated path planning for mobile crane lifts. Computer-Aided Civil and Infrastructure Engineering, 17(6), 439–448. https://doi.org/10.1111/0885-9507.00005

Shahnavaz, F., Taghaddos, H., Najafabadi, R. S., & Hermann, U. (2020). Multi crane lift simulation using Building Information Modeling. Automation in Construction, 118. https://doi.org/10.1016/j.autcon.2020.103305

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., Van Den Driessche, G., Graepel, T., & Hassabis, D. (2017). Mastering the game of Go without human knowledge. Nature, 550(7676), 354–359. https://doi.org/10.1038/nature24270

Sivakumar, P. L., Varghese, K., & Babu, N. R. (2003). Automated Path Planning of Cooperative Crane Lifts Using Heuristic Search. Journal of Computing in Civil Engineering, 17(3), 197–207. https://doi.org/10.1061/(ASCE)0887-3801(2003)17:3(197)

Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction, 2nd ed. In Reinforcement learning: An introduction, 2nd ed. The MIT Press.

Tak, A. N., Taghaddos, H., Mousaei, A., Bolourani, A., & Hermann, U. (2021). BIM-based 4D mobile crane simulation and onsite operation management. Automation in Construction, 128, 103766. https://doi.org/https://doi.org/10.1016/j.autcon.2021.103766

Tian, J., Luo, S., Wang, X., Hu, J., & Yin, J. (2021). Crane Lifting Optimization and Construction Monitoring in Steel Bridge Construction Project Based on BIM and UAV. Advances in Civil Engineering, 2021. https://doi.org/10.1155/2021/5512229

Wang, X., Zhang, Y. Y., Wu, D., & Gao, S. (2011). Collision-Free Path Planning For Mobile Cranes Based On Ant Colony Algorithm. Key Engineering Materials, 467–469, 1108–1115. https://doi.org/10.4028/www.scientific.net/KEM.467-469.1108

Zhou, Y., Zhang, E., Guo, H., & Candidate, P. D. (2020). Lifting Path Planning of Mobile Cranes Based on RRT Algorithm.

Zhou, Y., Zhang, E., Guo, H., Fang, Y., & Li, H. (2021). Lifting path planning of mobile cranes based on an improved RRT algorithm. Advanced Engineering Informatics, 50, 101376. https://doi.org/https://doi.org/10.1016/j.aei.2021.101376