Proceedings of the 2023 Winter Simulation Conference
C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.

MULTI-AGENT PROXIMAL POLICY OPTIMIZATION FOR A DEADLOCK CAPABLE TRANSPORT SYSTEM IN A SIMULATION-BASED LEARNING ENVIRONMENT

Marcel Müller
Tobias Reggelin
Hartmut Zadek

Institute of Logistics and Material Handling Systems
Otto von Guericke University Magdeburg
Universitätsplatz 2
Magdeburg, 39106, GERMANY

Lorena S. Reyes-Rubiano

Data & Business Analytics
RWTH Aachen University
Kackertstraße 7
Aachen, 52072, GERMANY

ABSTRACT

In this paper, we explore the potential of multi-agent reinforcement learning (MARL) for managing the driving behavior of autonomous guided vehicles (AGVs) in production logistics environments with single-lane tracks, where deadlocks pose a significant challenge. We build upon previous work and adopt a MARL approach using the Proximal Policy Optimization (PPO) algorithm. We conduct a thorough hyperparameter search and investigate the impact of varying numbers of agents on the performance of the AGVs. Our results demonstrate the effectiveness of the MARL approach in addressing deadlocks and coordinating AGV behavior, as well as the scalability of the learned policy to different numbers of agents. The Bayesian optimization process and increased iteration count contribute to improved performance and more stable learning curves.

1 PROBLEM AND MOTIVATION

The increasing utilization of automated systems in production and logistics leads to greater complexity in planning and controlling these systems. This complexity can make resource usage within the system more opaque and dynamic, which, in turn, poses challenges in detecting circular dependencies in resource demands during planning. Failure to detect these dependencies results in more frequent occurrences of deadlocks in automated material flow systems, potentially causing the entire logistics system to halt through chain reactions. Addressing these challenges is crucial for ensuring the resilience of logistics systems, making it an essential topic in the context of resilient systems.

There are several types of deadlocks (Tanenbaum and Bos 2022). This paper focuses on resource deadlocks, which occur when at least two processes wait for each other in a circular reference and cannot proceed to the next step due to infeasible conditions (Coffman et al. 1971; Tanenbaum and Bos 2022). Deadlocks can arise even when not all resources in a system are utilized (Coffman et al. 1971).

Simulation models can help in detecting deadlocks and developing appropriate resolution strategies. Through numerous random experiments, even highly improbable scenarios can be discovered, which are not readily apparent or analytically determinable in complex logistics systems (Müller et al. 2019). Reservation systems, one of the approaches to address deadlocks, require sufficient information about the current situation in a logistics system to guarantee accurate scheduling of reservation slots (Lienert and Fottner 2017).

Centralized control mechanisms, such as problem-specific control rules and reservation systems, exhibit poor scalability for larger systems. With problem-specific control rules, achieving comprehensive coverage of all deadlock situations in large systems is challenging, as rare occurrences may not be detected in advance during planning. In contrast, reservation systems diverge further from reality in larger systems due to their deterministic approach, as exceptional disturbance events become more likely.

The limitations of existing deadlock solutions in terms of scalability and adaptability to structural changes motivate us to explore machine learning approaches. We hypothesize that trained agents, which tackle decision problems relevant to deadlock emergence, can offer superior flexibility and scalability (Müller et al. 2022). This approach could potentially render the categorization of strategy approaches for deadlock handling (prevention, avoidance, and detection and resolution) obsolete. In light of this, we pose several research questions:

- Can a multi-agent Proximal Policy Optimization (PPO) approach effectively manage deadlocks in a simulated logistics environment?
- How do varying hyperparameters impact the performance of the PPO algorithm in this context?
- Can the proposed approach offer scalability with varying numbers of agents?

To address these challenges, we propose a multi-agent PPO approach within a simulation-based learning environment. By leveraging the potential of machine learning techniques and simulation environments, this work aims to develop resilient logistics systems capable of effectively handling deadlocks, thus enhancing the overall efficiency and adaptability of modern logistics operations.

2 LITERATURE

2.1 Reinforcement Learning

Reinforcement learning (RL) is a branch of machine learning that focuses on training agents to make decisions by interacting with an environment to maximize cumulative rewards. Over the past few years, RL has gained considerable attention due to its success in solving complex decision-making problems in various domains, such as robotics, finance, and healthcare (Kober et al. 2013; Charpentier et al. 2021; Yu et al. 2021). In RL, an agent learns a policy that maps states to actions by optimizing a reward function through trial and error. The basic building block of RL is the Markov Decision Process (MDP), which is a mathematical framework that represents an agent's decision-making process in a stochastic environment (Sutton and Barto 2018).

One significant breakthrough in RL has been the integration of deep learning techniques with RL algorithms, known as deep reinforcement learning (DRL). DRL has shown remarkable performance in complex tasks, such as learning to play games like Go, Atari, and Poker at a superhuman level (Mnih et al. 2013; Silver et al. 2016; Brown et al. 2020).

Proximal Policy Optimization (PPO) is a prominent DRL algorithm that has gained popularity for its stability, sample efficiency, and capability to learn complex policies (Schulman et al. 2017). PPO addresses the problem of large policy updates in policy gradient methods by incorporating a clipping mechanism within the objective function. Traditional policy gradient methods can sometimes suffer from large policy updates, which can lead to unstable learning and reduced performance. PPO mitigates this issue by introducing a clipped objective function that penalizes excessively large policy updates. As a result, the learning process becomes more stable and efficient, ensuring a more consistent improvement in the agent's performance throughout the training process. This clipping mechanism helps strike a balance between exploration and exploitation, allowing the agent to learn an effective policy without experiencing drastic fluctuations in its behavior.

Single-agent reinforcement learning approaches may not be sufficient when applied to problems that involve multiple interacting agents (Müller et al. 2022). In such scenarios, agents need to learn not only to optimize their individual policies but also to coordinate effectively with other agents in the system. This necessity motivates the exploration of multi-agent reinforcement learning (MARL) techniques, which extends the principles of RL to the multi-agent setting.

2.2 Multi-Agent Systems

Multi-agent systems (MAS) are a research area that focuses on understanding and designing systems where multiple autonomous agents interact with one another to achieve specific goals (Ferber and Weiss 1999). MAS have attracted significant interest due to their potential for improving efficiency, flexibility, and adaptability in a wide range of applications, including logistics, transportation, and manufacturing (Burmeister et al. 1997; Balaji and Srinivasan 2010; Lee and Kim 2008). In the context of logistics, multi-agent systems can involve various types of agents, such as Automated Guided Vehicles (AGVs), drones, or even human workers, that collaborate to achieve efficient material flow and transportation within a facility.

One example of a multi-agent logistics system is a setting where multiple AGVs operate concurrently to transport goods between different locations within a facility, such as production sites, warehouses, and distribution centers. The effective management of such systems requires real-time coordination among AGVs, deadlock handling, and collision avoidance (Yan et al. 2017). This paper focuses on the challenges and optimization of multi-agent logistics systems involving AGVs, but the principles and techniques discussed can also be extended to other types of agents in logistics applications.

There has been a growing interest in applying RL to multi-agent systems, leading to the development of various multi-agent RL (MARL) algorithms (Busoniu et al. 2008; Zhang et al. 2021) and frameworks (Liang et al. 2017; Samvelyan et al. 2019; Papoudakis et al. 2021; Terry et al. 2021). There are several dimensions to how MARL algorithms can be categorized. A distinction is often made according to the centralization of the learning process and the level of communication and coordination between the agents (Papoudakis et al. 2021):

- Independent Learning: In independent learning, each agent learns its own policy without any explicit communication or coordination with other agents. The agents may still observe other agents' actions and incorporate them into their decision-making process. Examples of independent learning algorithms are Independent Q-Learning (IQL) (Tan 1993) and Independent Proximal Policy Optimization (IPPO).
- Centralized Learning: In centralized learning, agents share a common policy or value function, and learning is performed in a centralized manner. Centralized learning can potentially lead to better coordination among agents but may suffer from scalability issues as the number of agents increases. Examples of centralized learning algorithms include QMIX (Rashid et al. 2020), Multi-Agent Deep Deterministic Policy Gradient (MADDPG) (Lowe et al. 2017), and Multi-Agent PPO (MAPPO) (Yu et al. 2022).

2.3 Deadlock Handling with Machine Learning

Deadlocks are situations in which multiple agents are stuck in a cycle of waiting for one another to release a resource, leading to a halt in system operation (Coffman et al. 1971). In multi-agent logistics systems, deadlocks can occur due to resource contention among AGVs, making deadlock handling a crucial challenge (Fanti 2002). Traditional deadlock prevention and resolution techniques involve the use of scheduling algorithms, Petri nets, or supervisory control (Ezpeleta et al. 1995; Roszkowska 2004; Lienert and Fottner 2017). These methods often require extensive domain knowledge and can be computationally expensive in large-scale systems.

While traditional deadlock prevention and resolution techniques have been extensively studied, the investigation of deadlocks in the context of machine learning remains limited (Bouderba and Moussa 2019; Bouton et al. 2019; Kujirai and Yokota 2019; Reijnen et al. 2020). A comprehensive examination of how deadlocks affect the learning behavior of machine learning algorithms and the extent to which AI agents can learn to handle deadlocks has yet to be fully explored. Notably, Sørensen et al. (2020) have touched upon this topic by implementing a reinforcement learning approach with a dueling deep Q-Network architecture for an airport baggage system, augmented with a deadlock avoidance algorithm. Their AI agent succeeded in reducing the number of deadlocks but did not completely eliminate them. As the volume of baggage increased, deadlocks occurred more frequently, causing episodes to restart in the learning environment. This highlights the need for further research into how machine learning algorithms can effectively handle deadlocks in multi-agent logistics systems.

3 METHODOLOGY

3.1 Conceptual Model

The objective of this research is to design and train a neural network capable of effectively managing the driving behavior of AGVs in a production logistics environment, where single lane tracks present challenges for the AGVs such as deadlocks. In this paper, we build upon our previous preliminary work (Müller et al. 2022), where we investigated the challenges faced by neural networks in addressing deadlock situations. The present study extends this work by adopting a multi-agent reinforcement learning approach. We have chosen MARL for several reasons: its ability to capture the complex interactions among multiple autonomous agents, its potential to learn efficient and coordinated policies, and its suitability for adapting to dynamic environments. We have modified the learning environment to be more conducive to deadlocks and intricate driving maneuvers. By incorporating such challenges, we aim to demonstrate the effectiveness of the MARL approach in addressing deadlock situations and efficiently coordinating AGV behavior in complex logistics systems. The considered learning environment features a dead end in the system leading to the sink, necessitating either AGVs to wait until the dead end is unoccupied or to reverse to provide space for other AGVs to leave the dead end. Figure 1 shows the conceptual model of the considered logistics system with typical possible deadlock situations in the system.

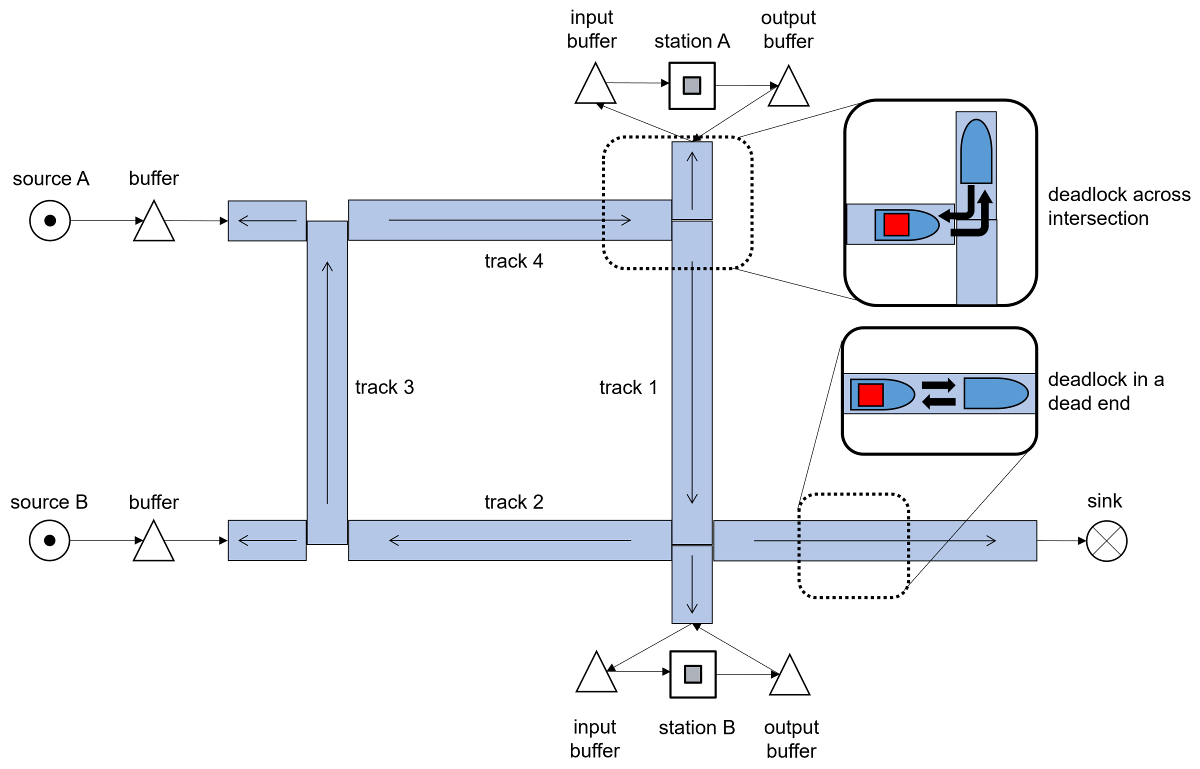

Figure 1: Conceptual model of the considered logistic system with single lane tracks.

The logistics system in question comprises two sources, source A and source B, which produce product types A and B, respectively. Two processing units are available, with each dedicated to a specific product type. The products from both processing units share a common sink. The system components are interconnected through single lane bidirectional tracks, with buffers positioned after the sources, and before and after the processing units. The agents within this model are the AGVs, with the number of agents ranging from one to four. The forward direction of the AGVs is shown by the arrows on the tracks in Figure 1. A turning maneuver is not possible, but the AGVs can drive backwards at any time. The sources generate products intended for the respective processing units. AGVs must traverse the single lane tracks while avoiding collisions and deadlocks. The routing algorithm employed in this context gives priority to output buffers of processing units over input buffers of sources. The multi-agent PPO algorithm applied to this model, incorporating the following state/observation space, action space, and reward function:

Table 1: Summary of observation space, action space, and reward function.

Observation space	
General	ID of the considered AGV/agent status of stations A and B number of items in buffers of sources A and B
For each agent *i*	*x* and *y* positions of agent *i* current speed of agent *i* target destination of agent *i* emptiness status of agent *i* remaining route length to target destination of agent *i*
Action space	
Driving behavior	0 (hold), 1 (forwards), 2 (backwards)
Reward function	
Delivering product	$+1$ for delivering a finished product to the sink
Picking up item	$+0.1$ for picking up an item from the source
Placing in unit	$+0.1$ for placing it in the correct processing unit
Stations not working	-0.01 for every agent per second/timestep for every station not working
Collision	-1 for each collision (for the oncoming AGV)

The reward function used for the multi-agent PPO algorithm is constructed to incentivize the behavior that promotes smooth operation of the logistics system and discourages situations that hinder the system's effectiveness. The reward function operates on the principle of reinforcement learning, where the agents (AGVs) learn by interacting with their environment and receiving feedback in the form of rewards or penalties. When an AGV successfully delivers a finished product to the sink, it receives a reward of $+1$. This motivates the AGVs to complete their primary task of transporting the finished products from the processing units to the sink. An additional reward of $+0.1$ is awarded when an AGV picks up an item from the source and another $+0.1$ when it places the item in the correct processing unit. These rewards are designed to encourage efficient pick-up and correct placement of items, which are critical steps towards successful delivery to the sink. To ensure the continuity of the system operation, penalties are introduced for undesirable circumstances. If a station is not working, a penalty of -0.01 is assigned to each agent for every second/timestep. This negative reinforcement acts as a deterrent for the AGVs, prompting them to act in ways that prevent the stations from halting, such as avoiding collisions or deadlocks which might result in system disruption. Finally, a strong penalty of -1 is assigned for each collision (for the oncoming AGV). This high negative reward acts as a powerful disincentive for actions leading to collisions, thereby promoting safe and careful navigation of the AGVs on the single-lane tracks. Through this balance of rewards and penalties, the multi-agent PPO algorithm guides the AGVs to learn optimal behavior that maximizes the throughput of the logistics system while avoiding disruptions. The values of the rewards and penalties are set to reflect the relative importance of each action or situation to the overall goal of efficient and smooth operation of the logistics system.

3.2 Technical Implementation

The simulation model was implemented with Tecnomatix Plant Simulation version 2201. Figure 2 shows a screenshot of the simulation model during the simulation of two agents. Each agent is randomly generated at the start of the simulation to a random position of the track with the same ID as the agent, that means for example agent 2 is always generated on track 2.

Plant Simulation leverages its integrated programming language, SimTalk 2.0, to facilitate the implementation of functions directly within the simulation model. The core function in our simulation model, "StepPython," is invoked by an external Python script through a Component Object Model (COM) interface. Upon starting the simulation, "StepPython" executes the "SimulationStop" function after one second of simulation time. The status of the simulation, specifically the event controller, is monitored by the Python script, which resumes execution when the simulation is halted. This process corresponds to a single time step in our reinforcement learning loop, enabling a seamless integration of the learning algorithm with the simulation environment.

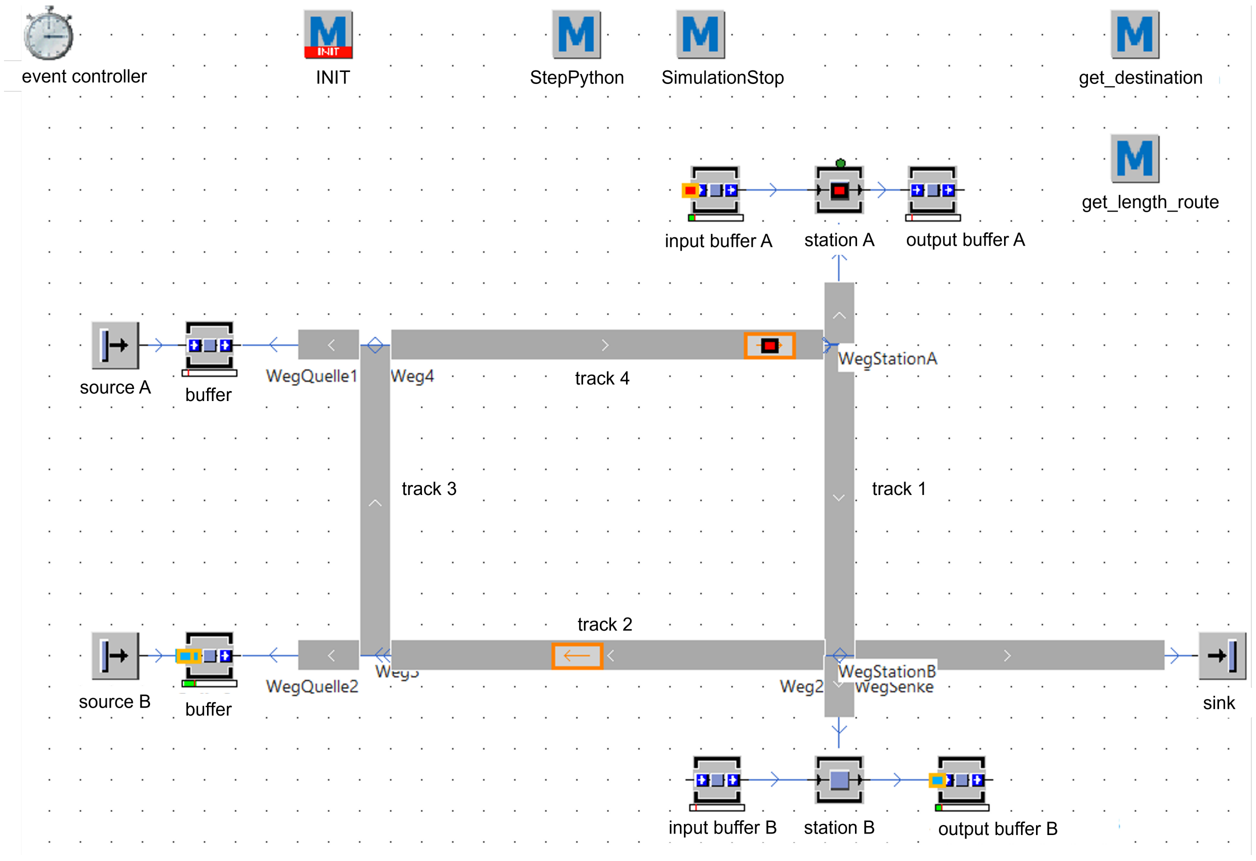

Figure 2: Simulation model during the simulation of two agents.

The technical implementation of the Python script, which sets up the agents and translates the simulation model into a readable learning environment for the agents, consisted of using Ray 3.0.0 with RLlib to manage our multi-agent reinforcement learning experiments, a custom environment adhering to the Gymnasium standards (previously OpenAI Gym) to describe our simulation model, and Weights & Biases to track experiment statistics. The custom environment was designed to interface with the Plant Simulation model through SimTalk 2.0 functions, and it implements the Gymnasium interface methods, such as reset and step. The reset method initializes the simulation by resetting the step counter, the simulation model, and the random seed for the simulation. The step method executes actions for each agent, updates the simulation, retrieves new observations, and calculates rewards for all agents. We used the Proximal Policy Optimization (PPO) algorithm with the same policy for every agent. This also means that the agents have learned the same policy at the same time. The PPO algorithm was tuned using Ray's Tune library, with a custom configuration, stopping criteria, and callbacks for Weights & Biases integration. The tuning process included hyperparameter search and evaluation. The custom environment's observation space was constructed by collecting various simulation data, such as the states of different stations, buffer levels, vehicle positions, speeds, destinations, and route lengths. No specific preprocessing or feature engineering was applied to the observations or rewards before using them in the learning algorithm. The training procedure varied across different experiments, with 250 to 1,000 iterations and 1,000 time steps per episode. However, Ray divided the episodes into smaller ones, approximately 500 time steps per episode.

3.3 Hyperparameter Search

In our experiment, we implemented an artificial neural network (ANN) using a fully connected feedforward architecture to model the agent's policy in a multi-agent environment with one to four agents operating concurrently. The ANN comprised two hidden layers, each consisting of 256 neurons, with the hyperbolic tangent (tanh) activation function employed in the hidden layers. The experiment was configured to use the Stochastic Sampling exploration strategy and the training process utilized two worker nodes, each with one central processing unit (CPU). Evaluation of the agent's performance was carried out using a separate set of episodes in parallel with the training process. Continuing from the implementation of the artificial neural network and the training process, we further conducted a thorough search for the optimal hyperparameters of the PPO algorithm. The primary goal was to maximize the performance of the agent in the multi-agent environment with concurrent operation of one to four agents. We employed a Bayesian optimization for exploring the hyperparameter space. Inspired by the article from AurelianTactics (2018), we selected a broad range of hyperparameters for the PPO algorithm, covering various aspects such as learning rate, clipping parameter, and regularization terms. From our point of view, this was the best starting point, since we did not yet know anything about the learning behavior of the PPO in this environment. The number of agents for the hyperparameter search was set at two so that collisions and deadlocks could still occur, but compared to three or four agents, there was still more freedom of movement and avoidance. Table 2 presents the search space for each hyperparameter in both Bayesian optimization runs, as well as the initial point for the second run and the parameters of the BayesOptSearch algorithm. The initial point for the second run was determined by choosing the run with the highest mean reward over 100 episodes. The table aims to provide a comprehensive overview of the hyperparameter exploration process and highlights the differences in the search strategy between the two runs. By performing these optimization runs, we aimed to identify a set of hyperparameters that would enable the PPO algorithm to initiate learning effectively within the learning environment.

4 RESULTS

During the first Bayesian optimization run, a satisfactory solution was identified within the random sampling phase. The subsequent 175 samples, generated through Bayesian optimization, yielded no substantial improvements. As a result, our focus shifted to the outcomes of the second run, which was more finely tuned. In this run, an iteration stopper was set at 500, and a trial plateau stopper was employed for samples exhibiting an episode mean reward with a standard deviation below 0.2 over a span of 100 iterations to prematurely terminate unpromising runs.

Table 2: Hyperparameters for Bayesian optimization runs.

Hyperparameter	**Run 1**	**Run 2**	**Init. point for run 2**
number of samples	225	43	-
random search samples	50	0	-
minibatch size	512	512	-
epoch range	20	20	-
clipping range	[0.1, 0.3]	[0.1, 0.3]	0.1749
learning rate α	[5e-6, 0.003]	[5e-6, 0.001]	0.000179
KL initialization range	[0.3, 1]	[0.3, 1]	0.7191
KL target range	[0.003, 0.03]	[0.003, 0.03]	0.007212
discount factor γ	[0.8, 0.9997]	[0.8, 0.9997]	0.9462
GAE parameter λ	[0.9, 1]	[0.9, 1]	0.9156
value function coefficient	[0.5, 1]	[0.5, 1]	0.9331
entropy coefficient	[0, 0.01]	[0.001, 0.01]	0.009507
BayesOptSearch Parameters			
kind	UCB	UCB	-
κ	2.5	0.5	-
ξ	0.0	0.0	-

Figure 3 illustrates the distribution of hyperparameters in the second Bayesian optimization run. The optimal sample yielded a mean reward of -16.918, whereas the least favorable sample resulted in a mean reward of -26.7. A considerable number of samples continue to exhibit inaction throughout the episode to circumvent collisions, consequently incurring penalties for non-operational stations. This behavior results in mean reward values around -20 for the two-agent scenario. Our findings indicate that employing a smaller learning rate and a reduced lambda value yielded improved outcomes. The clipping and Kullback-Leibler (KL) initialization range emerged as significant hyperparameters, and a gamma value around 0.95 proved advantageous.

In Figure 4, the mean reward of Bayesian optimization run 2 is depicted, with colors representing the maximum reward achieved. As the second run progresses, we observe that even slight variations in parameters enable surpassing the best solution obtained from run 1.

Although no substantial leaps in mean reward improvement are evident, the rolling average displays an upward trend, signifying enhanced performance. This observation suggests that the Bayesian optimization demonstrates a marked improvement during the second run compared to the first. Following the completion of the two Bayesian optimization runs, we aimed to investigate if superior solutions could be achieved with a higher iteration count of 2,500 instead of 500, and whether the identified hyperparameters were applicable to varying numbers of agents. To address these questions, we conducted a series of experiments using the optimal hyperparameter configuration from the second Bayesian optimization run. We varied the number of agents from one to four, and for each configuration, we generated three samples to examine the variance of the learning curves associated with the same configuration. Figure 5 presents the outcomes of the aforementioned experiments. As the number of agents increases, a decline in the mean reward is observed, primarily attributable to the negative reward each agent incurs for no stations with working status. The standard deviation growths alongside the increasing agent count. An anomaly is observed when only one agent is present, as the most substantial variance in mean reward transpires in this scenario. We attribute this to the heightened difficulty in exploration for a single agent, making it more challenging to discover the beneficial behavior to transport goods from the source to the stations. It also becomes clear how quickly the agents learn to avoid collisions, while it is more difficult to generate throughput consistently.

As no collisions occur with a single agent, it is also not difficult for the agent to escape the behavior of remaining stationary to avoid collisions. The steepest learning curve for more than one agent is observed

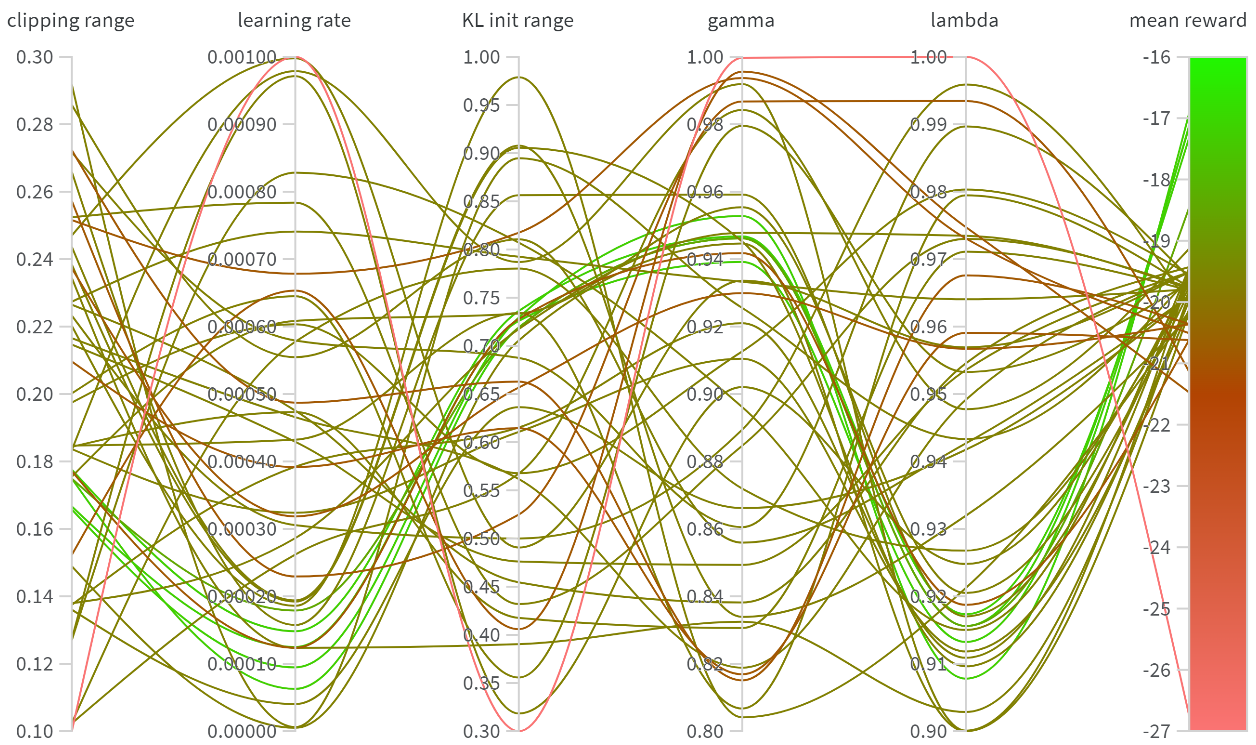

Figure 3: Hyperparameter ranges and impact on the mean reward for Bayesian optimization run 2.

in the case of two agents. With three agents, a learning effect is still discernible. When four agents are present, the mean reward remains relatively constant, even though rewards of up to -26 have been achieved in some episodes. These rewards, due to the high risk of collision, have not been consistently attainable thus far.

5 CONCLUSION AND OUTLOOK

This paper addresses the posed research questions demonstrating the potential of multi-agent reinforcement learning using Proximal Policy Optimization in managing deadlock situations and efficiently coordinating AGV behavior in complex logistics systems. The response to our first research question has been positive as the MARL approach, through a rigorous hyperparameter search, succeeded in developing an AGV policy adept at navigating single-lane tracks while avoiding collisions and deadlocks. Regarding the impact of varying hyperparameters, our second research question, our results reveal that even slight variations can lead to substantial improvements in the AGVs' performance. This underlines the significance of the hyperparameter optimization process, which allowed us to identify a set of hyperparameters that were applicable to varying numbers of agents. Our third research question investigated the scalability of our approach. The experiments conducted with varying numbers of agents demonstrated that the same policy could be effectively applied to different numbers of AGVs, exhibiting the scalability of the proposed approach. While our findings present a promising outlook for the application of MARL in AGV coordination within production logistics environments, further research is required to investigate the robustness of the learned policies in diverse and more complex scenarios. An extension of the hyperparameter search and an adjustment of the network structures could bring even better results. Future work could involve exploring additional reward functions and investigating the impact of different learning algorithms on the performance of the AGVs. The integration of more advanced techniques such as communication protocols between

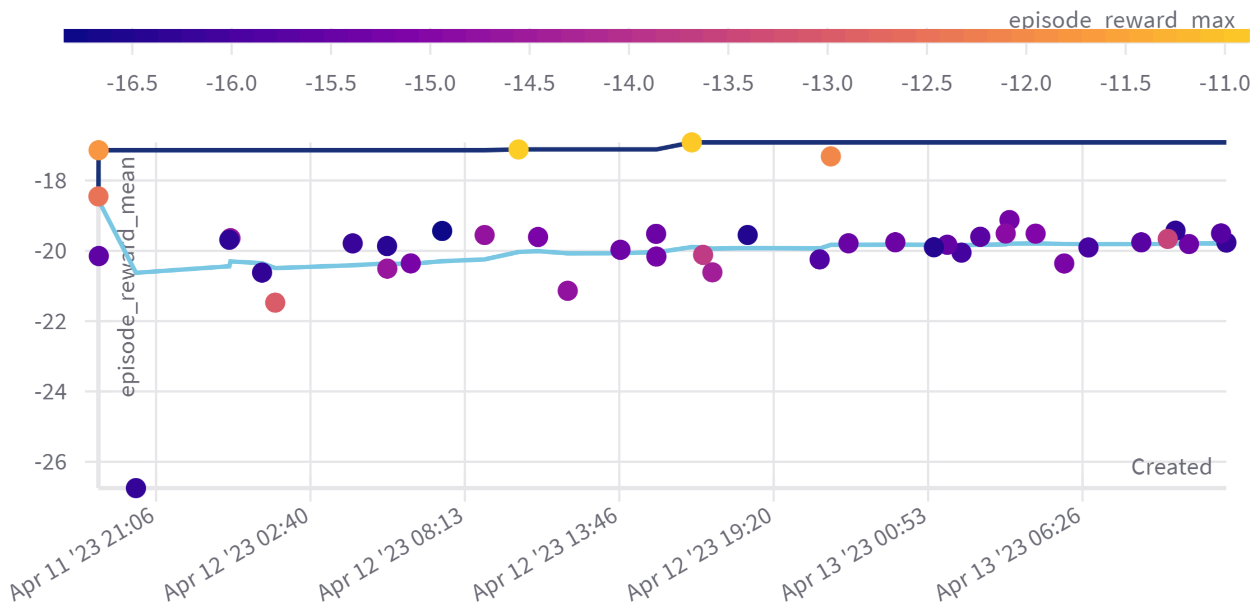

Figure 4: Scatter plot of mean episode reward and maximum reward of the samples of Bayesian optimization run 2.

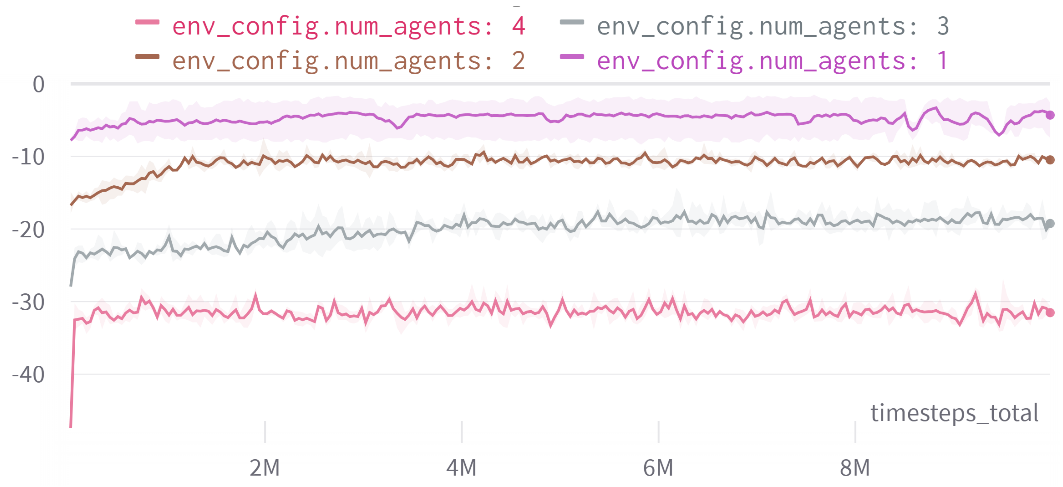

Figure 5: Learning curves during training process grouped by number of agents.

agents or employing a centralized critic in a decentralized setup could be investigated to further enhance AGV coordination and deadlock resolution in production logistics environments.

REFERENCES

AurelianTactics 2018. “PPO Hyperparameters and Ranges”. https://medium.com/aureliantactics/ppo-hyperparameters-and-ranges-6fc2d29bccbe, accessed 21.04.2023.

Balaji, P., and D. Srinivasan. 2010. “Multi-Agent System in Urban Traffic Signal Control”. *IEEE Computational Intelligence Magazine* 5(4):43–51.

Bouderba, S. I., and N. Moussa. 2019. “Reinforcement Learning (Q-LEARNING) Traffic Light Controller within Intersection Traffic System”. In *Proceedings of the 4th International Conference on Big Data and Internet of Things*, edited by

M. Lazaar, C. Duvallet, M. Al Achhab, O. Mahboub, and H. Silkan, 1–6. New York City, New York: Association for Computing Machinery.

Bouton, M., A. Nakhaei, K. Fujimura, and M. J. Kochenderfer. 2019. "Cooperation-Aware Reinforcement Learning for Merging in Dense Traffic". In *Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference*, 3441–3447. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Brown, N., A. Bakhtin, A. Lerer, and Q. Gong. 2020. "Combining Deep Reinforcement Learning and Search for Imperfect-Information Games". *Advances in Neural Information Processing Systems* 33:17057–17069.

Burmeister, B., A. Haddadi, and G. Matylis. 1997. "Application of Multi-Agent Systems in Traffic and Transportation". *IEE Proceedings-Software* 144(1):51–60.

Busoniu, L., R. Babuska, and B. De Schutter. 2008. "A Comprehensive Survey of Multiagent Reinforcement Learning". *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38(2):156–172.

Charpentier, A., R. Elie, and C. Remlinger. 2021. "Reinforcement Learning in Economics and Finance". *Computational Economics* 62:425–462.

Coffman, E. G., M. Elphick, and A. Shoshani. 1971. "System Deadlocks". *ACM Computing Surveys* 3(2):67–78.

Ezpeleta, J., J. M. Colom, and J. Martinez. 1995. "A Petri Net Based Deadlock Prevention Policy for Flexible Manufacturing Systems". *IEEE Transactions on Robotics and Automation* 11(2):173–184.

Fanti, M. P. 2002. "Event-based Controller to Avoid Deadlock and Collisions in Zone-Control AGVS". *International Journal of Production Research* 40(6):1453–1478.

Ferber, J., and G. Weiss. 1999. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, Volume 1. Boston, Massachusetts: Addison-Wesley.

Kober, J., J. A. Bagnell, and J. Peters. 2013. "Reinforcement Learning in Robotics: A Survey". *The International Journal of Robotics Research* 32(11):1238–1274.

Kujirai, T., and T. Yokota. 2019. "Breaking Deadlocks in Multi-agent Reinforcement Learning with Sparse Interaction". In *Proceedings of the 16th Pacific Rim International Conference on Artificial Intelligence*, edited by A. C. Nayak and A. Sharma, Volume 11670 of *Lecture Notes in Computer Science*, 746–759. Cham, Switzerland: Springer Nature Switzerland AG.

Lee, J.-H., and C.-O. Kim. 2008. "Multi-Agent Systems Applications in Manufacturing Systems and Supply Chain Management: A Review Paper". *International Journal of Production Research* 46(1):233–265.

Liang, E., R. Liaw, R. Nishihara, P. Moritz, R. Fox, J. Gonzalez, K. Goldberg, and I. Stoica. 2017. "Ray rllib: A Composable and Scalable Reinforcement Learning Library". *arXiv preprint arXiv:1712.09381* 85.

Lienert, T., and J. Fottner. 2017. "No More Deadlocks – Applying the Time Window Routing Method to Shuttle Systems". In *Proceedings of the 31st European Conference on Modelling and Simulation*, edited by Z. Z. Paprika, P. Horák, K. Váradi, P. T. Zwierczyk, Á. Vidovics-Dancs, and J. P. Rádics, 169–175. Budapest, Hungary: European Council for Modelling and Simulation.

Lowe, R., Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch. 2017. "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments". *Advances in Neural Information Processing Systems* 30.

Mnih, V., K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. 2013. "Playing Atari with Deep Reinforcement Learning". *arXiv preprint arXiv:1312.5602*.

Müller, M., T. Reggelin, I. Kutsenko, H. Zadek, and L. S. Reyes-Rubiano. 2022. "Towards Deadlock Handling with Machine Learning in a Simulation-Based Learning Environment". In *Proceedings of the 2022 Winter Simulation Conference*, edited by B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C. G. Corlu, L. H. Lee, E. P. Chew, T. Roeder, and P. Lendermann, 1485–1496. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Müller, M., S. Schmidt, and T. Reggelin. 2019. "Deadlock and Collision Handling for Automated Rail-Based Storage and Retrieval Units". In *Proceedings of the 2019 Winter Simulation Conference*, edited by N. Mustafee, K.-H. G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, 1591–1601. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Papoudakis, G., F. Christianos, L. Schäfer, and S. V. Albrecht. 2021. "Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks". In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, edited by J. Vanschoren and S. Yeung, Volume 1. Virtual: Curran.

Rashid, T., M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson. 2020. "Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning". *The Journal of Machine Learning Research* 21(1):7234–7284.

Reijnen, R., Y. Zhang, W. Nuijten, C. Senaras, and M. Goldak-Altgassen. 2020. "Combining Deep Reinforcement Learning with Search Heuristics for Solving Multi-Agent Path Finding in Segment-based Layouts". In *Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence*, 2647–2654. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Roszkowska, E. 2004. "Supervisory Control for Deadlock Avoidance in Compound Processes". *IEEE Transactions on Systems, Man, and Cybernetics-part A: Systems and Humans* 34(1):52–64.

Samvelyan, M., T. Rashid, C. S. de Witt, G. Farquhar, N. Nardelli, T. G. J. Rudner, C.-M. Hung, P. H. S. Torr, J. Foerster, and S. Whiteson. 2019. "The StarCraft Multi-Agent Challenge". *CoRR* abs/1902.04043.

Schulman, J., F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. 2017. "Proximal Policy Optimization Algorithms". *arXiv preprint arXiv:1707.06347*.

Silver, D., A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. 2016. "Mastering the Game of Go with Deep Neural Networks and Tree Search". *Nature* 529(7587):484–489.

Sørensen, R. A., M. Nielsen, and H. Karstoft. 2020. "Routing in Congested Baggage Handling Systems Using Deep Reinforcement Learning". *Integrated Computer-Aided Engineering* 27(2):139–152.

Sutton, R. S., and A. G. Barto. 2018. *Reinforcement Learning: An Introduction*. Cambridge, Massachusetts: MIT press.

Tan, M. 1993. "Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents". In *Proceedings of the Tenth International Conference on Machine Learning*, 330–337. San Francisco, CA: Morgan Kaufmann Publishers.

Tanenbaum, A. S., and H. Bos. 2022. *Modern Operating Systems*. 5$^{\text{th}}$ ed. Hoboken, New Jersey: Pearson Education, Inc.

Terry, J., B. Black, N. Grammel, M. Jayakumar, A. Hari, R. Sullivan, L. S. Santos, C. Dieffendahl, C. Horsch, R. Perez-Vicente et al. 2021. "Pettingzoo: Gym for Multi-Agent Reinforcement Learning". *Advances in Neural Information Processing Systems* 34:15032–15043.

Yan, X., C. Zhang, and M. Qi. 2017. "Multi-AGVs Collision-Avoidance and Deadlock-Control for Item-to-Human Automated Warehouse". In *Proceedings of the 2017 International Conference on Industrial Engineering, Management Science and Application*. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Yu, C., J. Liu, S. Nemati, and G. Yin. 2021. "Reinforcement Learning in Healthcare: A Survey". *ACM Computing Surveys (CSUR)* 55(1):1–36.

Yu, C., A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu. 2022. "The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games". *Advances in Neural Information Processing Systems* 35:24611–24624.

Zhang, K., Z. Yang, and T. Başar. 2021. "Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms". *Handbook of Reinforcement Learning and Control*:321–384.

AUTHOR BIOGRAPHIES

MARCEL MÜLLER is a research fellow at the Otto von Guericke University Magdeburg. He earned his master degree in Industrial Engineering for Logistics at the Otto von Guericke University Magdeburg. His research interests include modeling and simulation of logistics systems, multi agent reinforcement learning and handling of deadlocks. His email address is marcel1.mueller@ovgu.de. His website is https://www.ilm.ovgu.de/mueller.

LORENA S. REYES-RUBIANO is a full-time lecturer at the Universidad de La Sabana, Chia-Colombia. She works also as a researcher at the RWTH Aachen University, Aachen, Germany. She has a Ph.D. degree in mathematics and statistics from the Public University of Navarre (2019). Her research interests are urban logistics, humanitarian logistics, and multi-objective algorithms. Her email address is lorena.reyes1@unisabana.edu.co.

TOBIAS REGGELIN is a project manager, researcher and lecturer at the Otto von Guericke University Magdeburg and the Fraunhofer Institute for Factory Operation and Automation IFF Magdeburg. His main research and work interests include modeling and simulation of production and logistics systems and developing and applying logistics management games. Tobias Reggelin received a doctoral degree in engineering from the Otto von Guericke University Magdeburg. Furthermore, he holds a master's degree in Engineering Management from Rose-Hulman Institute of Technology in Terre Haute, IN and a diploma degree in Industrial Engineering in Logistics from the Otto von Guericke University Magdeburg. His email address is tobias.reggelin@ovgu.de.

HARTMUT ZADEK received the call to Otto von Guericke University Magdeburg and has been head of the Chair of Logistics in the Institute of Logistics and Material Flow Engineering in the Faculty of Mechanical Engineering since January 1, 2008. He is also a lecturer at the University of Dortmund and a lecturer at the Hamburg School of Logistics. He is a member of the Association of German Industrial Engineers and the German Logistics Association (BVL). He has headed the Sustainable Production Logistics working group for BVL since June 2009. He is the editor of various specialist books, author of numerous specialist publications, sought-after speaker at specialist conferences in Germany and abroad, moderator of specialist congresses and seminars and leader of strategy workshops. In January 2010 he was appointed to the Saxony-Anhalt Logistics Advisory Board by the Minister for Regional Development and Transport. His email address is zadek@ovgu.de.