DOI: 10.22032/dbt.57796

Simulation in Produktion und Logistik 2023
Bergmann, Feldkamp, Souren und Straßburger (Hrsg.)
Universitätsverlag Ilmenau, Ilmenau 2023
DOI (Tagungsband): 10.22032/dbt.57476

Simulation-Based Resolution of Deadlocks in Automated Guided Vehicles using Deep Reinforcement Learning

Simulationsgestützte Lösung von Deadlocks bei fahrerlosen Transportsystemen mit Hilfe von Deep Reinforcement Learning

Mustafa Jelibaghu, Michael Eley, Alexander Palatnik
Technische Hochschule Aschaffenburg, Aschaffenburg (Germany),
{mustafa.jelibaghu, michael.eley, alexander.palatnik}@th-ab.de

Abstract: This paper discusses the use of deep reinforcement learning to resolve deadlocks in material flow systems with automated guided vehicles (AGVs). The paper proposes a strategy for dealing with deadlocks based on a single Agent reinforcement learning approach (SARL). The agent will find the optimal solution strategy in real time. The proposed approach is evaluated using a material flow simulation for a real use case in industry. The effectiveness in reducing the occurrence of deadlocks as well as the number of collisions in the system is demonstrated. This study highlights the potential of deep reinforcement learning for improving the performance and efficiency of material flow systems with AGVs.

1 Introduction

Automated guided vehicles (AGVs) are fleets of autonomous vehicles that transport material or products in warehouses or production environments. AGVs are increasingly used in the logistics industry to automate the internal material flow, as they are more efficient, safer and cost-effective than manual transport systems.

In the process, the vehicles have to carry out the transport orders in compliance with safety and quality standards achieving the required throughput. The planning process for AGV-fleets includes dispatching, scheduling and routing.

- Dispatching refers to the allocation of orders to individual AGVs. Here it is decided which vehicle should take over which order. Dispatching must take into account the available resources, such as the number of AGVs, their capacity and availability, as well as the location and requirements of the orders.
- Scheduling is about determining the processing sequence and transportation times of the orders. The main objective is to optimize the throughput of the material flow while securing due dates.

- Routing is about determining the optimal driving route for each individual vehicle in order to complete the jobs assigned to the vehicle within the given time constraints. Deadlock can occur when the vehicles are following the shortest path calculated by the algorithm and end up in a situation where they are unable to pass each other. This is usually done using an algorithm such as Dijkstra (Javaid 2013), which is based on a directed graph. The directed graph represents the nodes (vehicles and other resources) and edges (connections between the resources) and the algorithm calculates the shortest path between the nodes.

The planning of dispatching, scheduling and routing is crucial for the productivity and efficiency of the internal material flow. Particular attention must be paid to detecting and avoiding deadlocks during routing. Such a deadlock occurs when two or more AGVs block each other and cannot longer continue their journey because they are waiting for each other to release a resource (see Figure 1). This leads to orders not being processed at the planned times or the material flow even coming to a standstill. Deadlocks occur specifically when the problems outlined above are solved one after the other in the specified order. To resolve deadlocks, there are various strategies such as using waiting or changing the route of the individual vehicles (re-routing) (e.g., (Xu et al. 2014), (Hussain et al. 2015)). However, these strategies are not always effective, and resolving one deadlock can lead to another deadlock.

In the classic planning process, these problems are considered one after the other in the specified order. Despite careful planning, so-called deadlocks can occur that can significantly impair the productivity and efficiency of the logistics processes.

In our scenarios we use the power of machine learning approach. Reinforcement Learning (RL) is a possible approach to solving deadlocks in warehouses. RL is a machine learning technique in which an agent makes decisions in an environment through trial-and-error learning in order to maximize a reward. The Deep Q-Network (DQN) Algorithm is a special form of RL based on deep learning and uses the Q-learning method to learn an optimal decision strategy.

The article is structured as follows. The relevant literature is discussed in the section 2. Section 3 describes the simulation environment used. An approach to avoiding deadlocks based on Markov Decision Process (MDP) and DQN is presented in the section 4. The two concluding sections 5 and 6 summarize test results and give an outlook on possible extensions.

2 Literature Review

There are many problem-specific approaches in the literature for deadlock handling. The problem is not only discussed in context of logistics but also in computer science. In addition to the sequential solution of scheduling and routing, approaches are also discussed in the literature in which the two subproblems are solved simultaneously, e.g., (Wang et al. 2019; Vivaldini et al. 2015; Li et al. 2019). In practice, however, the problem often arises with the solutions generated in such wise that the plans determined in this way cannot be implemented as planned due to disruptions in the operational process, and deadlocks therefore arise at other points.

In order to resolve deadlocks, there are various strategies, such as using waiting or changing the route of the individual vehicles (re-routing), e.g., (Xu et al. 2014; Sumanas et al. 2022). However, these strategies are not always effective, and

resolving one deadlock can lead to another deadlock. Deadlocks can be detected by using graph theory. Here, the transport flow is modelled as a directed graph and searched to detect deadlocks. Graph theory is an effective method for detecting deadlocks and is discussed extensively in the literature, e.g., (Yao et al. 2016; Ni et al. 2009; Wang et al. 2013).

The use of machine learning algorithms to prevent deadlocks in a simulation-based learning environment is explored in (Müller et al. 2022). He chooses a RL approach that uses a DQN algorithm to learn a policy for detect and resolution deadlocks. For future research he proposes second approach "Multi-Agent Reinforcement Learning (MARL)" because the first approach in (Müller et al. 2022) with increasing the number of AGVs brings with it an exponential possible action that an AGV can make in the environment (see Figure 5).

In (Kuhnle 2020), MARL is used. MARL is a popular technique for coordinating agents in complex environments. In the context of an AGV system, MARL can be used to prevent deadlocks by enabling the agents to learn and adapt their behavior based on the system's dynamics and the actions of other agents. To apply MARL to an AGV system, each AGV is modelled as an agent, and the agents are trained to cooperate to avoid deadlocks. MARL is suitable for problems where multiple agents need to collaborate or compete to achieve a common goal, while DQN is suitable for problems where a single agent needs to learn how to take optimal actions in a given environment. Both approaches have their own advantages and limitations, and the choice of approach depends on the specific problem and the requirements of the application.

It is true that Artificial Intelligence (AI) approaches can help solve deadlocks in warehouses more effectively than classic approaches. By using machine learning and other AI techniques, complex algorithms can be created that enable fast and accurate analysis of the warehouse structure and processes. AI systems can also be continually optimized to respond to changes in storage conditions or requirements. When adapting AI approaches to real layouts, however, it must be noted that the examples presented in the literature cannot always be directly transferred to reality. The topology of the warehouse can vary from company to company, and warehouse structures can also change over time within the same company. For this reason, it is important that AI approaches are adapted to the specific needs and requirements of a given warehouse. In order to close the gap between the examples presented in the literature and the requirements in reality, detailed analyses must be carried out to understand the specific conditions and requirements of a bearing. This can be done through the use of data analysis tools and machine learning models applied to real warehouse data. In this way, AI models can be created that are specifically tailored to the needs of the warehouse and can therefore deliver the best results.

3 Simulation Model

As part of the research project, a real application for an AGV system was considered and modelled in Plant Simulation (see Figure 1). The application is a high-bay warehouse with several aisles that the AGVs can only enter and exit from one side (dead ends). There are three AGVs available that have the task of moving pallets from the goods receipt, where the orders are created automatically and assigned to the

AGVs (well known as dispatching), to the high rack. At the beginning the AGVs are located at the park station.

In Plant Simulation, the different strategies for detecting, avoiding and resolving deadlocks will be implemented and compared in terms of their performance. A deadlock situation is shown in Figure 1. The deadlock occurs because the AGV01 currently located on the STR02 wants to enter aisle02. At the same time, the AGV02 wants to leave aisle02.

The simulation model is a digital twin of the logistics system, allowing for experimentation with different scenarios and the optimization of system performance. The model can simulate the movement of materials through the system and track the performance of different components, such as AGVs and conveyor systems. This can help identify potential bottlenecks and areas for improvement.

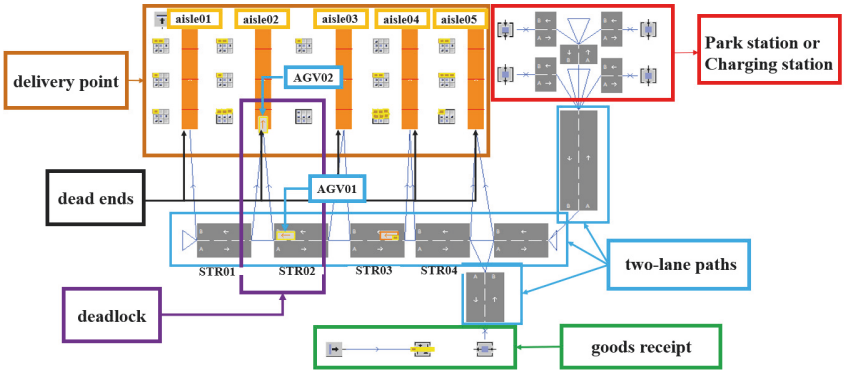

Figure 1: Illustration of a deadlock with three AGVs at the beginning of dead ends

Compared to other approaches in the literature, the complexity of our concept is very high. The number of dead ends and AGVs in the simulation model are realistic. Figure 1 shows that five dead ends and two-lane paths are examined more closely.

4 Deadlock Resolution based on Deep Reinforcement Learning Agent

Reinforcement learning is a branch of machine learning that involves an agent learning to interact with an environment to maximize a reward signal (see Figure 2). The goal is for the agent to learn an optimal policy, which maps states to actions, that maximizes the expected cumulative reward over time.

Formally, reinforcement learning can be described using the Markov Decision Process (MDP) (Sutton and Barto 2005) framework, which consists of the following components:

- State space (S): a set of possible states that the agent can be in.
- Action space (A): a set of possible actions that the agent can take.
- Transition function (T): a function that specifies the probability of moving to a new state given the current state and action.

- Reward function (R): a function that specifies the immediate reward received by the agent for taking a particular action in a particular state.
- Discount factor (γ): a value between 0 and 1 that determines the importance of future rewards relative to immediate rewards.

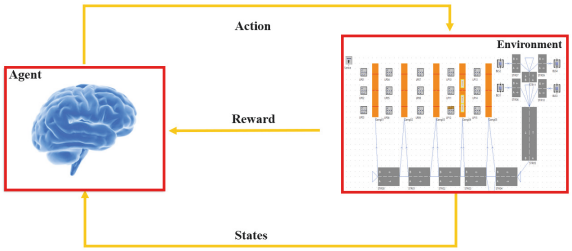

Figure 2: Agent/Environment Interaction

At each time step, the agent observes the current state of the environment, selects an action according to its policy, and receives a reward from the environment. The goal of the agent is to learn a policy that maximizes the expected cumulative reward over time. This can be done using a variety of reinforcement learning algorithms, such as Q-learning, SARSA, and DQN, which use different methods to estimate the optimal policy, e.g., (Müller et al. 2022; Sutton and Barto 2005).

Deep Q-Network is a type of reinforcement learning algorithm that uses a neural network to approximate the Q-value function. The Q-value function represents the expected reward for taking a particular action in a given state and is used to guide the decision-making process in a dynamic environment. In our case we consider the DQN algorithm. Because the Agent can learn from experience, handle complex environments, find optimal policies, and adapt to changing environments.

The DQN algorithm consists of two key components: the neural network and the experience replay memory. The neural network is used to approximate the Q-value function and takes the state of the environment as input and outputs the expected Q-values for each action. The experience replay memory is used to store and randomly sample previous experiences, which helps to stabilize the training process and prevent overfitting.

During training, the DQN agent interacts with the environment by selecting actions based on the current state and the Q-values predicted by the neural network. The agent then receives a reward and observes the new state, which is added to the experience replay memory. The neural network is then updated using a combination of the Bellman equation and stochastic gradient descent, which helps to adjust the Q-values based on the observed rewards and states.

The Bellman equation (1) is a key component of the DQN algorithm and is used to update the Q-values based on the observed rewards and states. The Bellman equation can be written as:

$$Q(s,a) = E\left[R + \gamma \, max\left(Q(s',a')\right)\right] \tag{1}$$

where $Q(s,a)$ is the Q-value for state s and action a, R is the reward received for taking action a in state s, γ is the discount factor, s' is the next state, and a' is the action taken in the next state. The term $max\big(Q(s',a')\big)$ represents the maximum Q-value for the next state s', and is used to estimate the expected future reward.

The DQN algorithm uses a variation of the Bellman equation, known as the Q-learning update rule, to adjust the Q-values during training. The Q-learning update rule can be written as:

$$Q(s,a) = Q(s,a) + \alpha\,(R + \gamma\, max\big(Q(s',a')\big) - Q(s,a)) \tag{2}$$

In equation (2):

- $Q(s,a)$ is the Q-value for state s and action a.
- α is the learning rate, which determines how much the Q-values are updated in each iteration.
- R is the immediate reward for taking action a in state s.
- γ is the discount factor, which determines how much weight to give to future rewards.
- $max\big(Q(s',a')\big)$ is the maximum Q-value over all actions a' in the next state s'.
- $Q(s',a')$ is the Q-value for the next state s' and action a'.

The equation (2) represents the update rule for the Q-values, which are updated after every action taken by the agent in the environment. The goal of the DQN algorithm is to learn the optimal Q-values for all state-action pairs, so that the agent can choose the action that maximizes its expected future reward in any given state.

Overall, the DQN algorithm is a powerful and flexible tool for reinforcement learning, and has been used successfully in a variety of applications, including robotics, game playing, and autonomous driving, e.g., (Lee and Yusuf 2022; Mnih et al. 2013; Friji et al. 2020).

In our simulation model in Figure 1, states s of the AGVs are sent to agents as a matrix input. These possible states are position of the AGVs, velocity of the AGVs, and the position of the destination of the order. These parameters will form our states.

The second important input parameter for an agent is reward or punishment. R is defined as reward. Arriving at its destination, AGV will be awarded with highest reward. The unnecessary steps or movements are seen as punishment. The collisions are defined as punishment. This means that the AGVs try to avoid collisions as much as possible. After the agent has received the following inputs (such as the states of the AGVs and the associated rewards), the agent will perform a random action a based on the Epsilon Greedy algorithm. Possible actions a are defined as moving forward, moving backward, and stop. Based on these actions, the agent will develop its Q-value function.

The Epsilon Greedy algorithm is a simple way to balance exploration and exploitation in the DQN algorithm. During the training phase, the DQN agent must explore the environment to find the optimal policy, but at the same time, it must exploit the information it has learned to maximize the expected reward. The Epsilon Greedy algorithm achieves this balance by allowing the agent to choose the best action with high probability (exploitation), but also randomly select a non-optimal action with low probability (exploration), e.g., (Sutton and Barto 2005; Mignon and Luis de Azevedo da Rocha 2017).

5 Results

The results of the simulation model developed in Plant Simulation and Python were obtained by running a series of experiments to evaluate the effectiveness of the DQN agent in resolving deadlocks in the AGV system. The connection between Plant Simulation and Python was established via the COM interface. The experiments were conducted using a simulated environment that closely mimics real-world AGV systems, including the presence of obstacles, varying task priorities, and traffic congestion.

In the Figure 3, a small number of episodes has been chosen to see if Agent adapts well to the environment. Two scenarios were analysed. The first scenario is to put collision avoidance at the dead end and transfer point. It can be seen in Figure 3 that the collisions in the bearing decrease with the increase in the number of episodes. This means that the agent controls its behaviour and makes good decisions regarding collision avoidance.

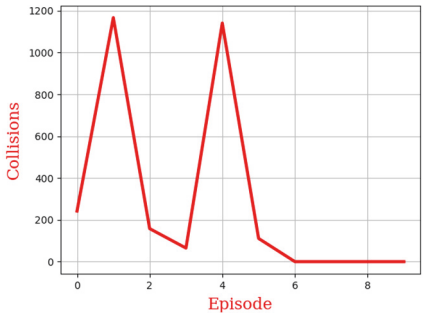

Figure 3: Collisions per episode for three AGVs.

In the second scenario (see Figure 4), the focus will be on using the Agent to teach AGVs how to behave in a storage environment to avoid or resolve deadlocks. The AGVs are trained through trial and error to learn the best actions to take in different situations, with the ultimate goal of avoiding or solving deadlocks that can occur when multiple AGVs are moving at the same time.

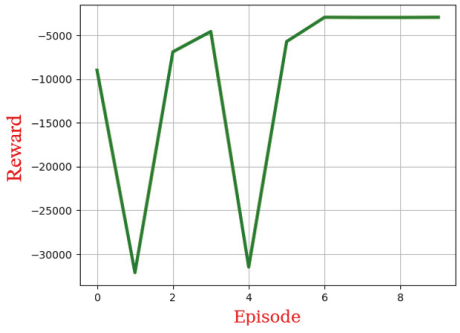

Figure 4: Cumulative rewards per episode for three AGVs.

These illustrations (see Figure 3, Figure 4) show the impact of the DQN agent on the performance of the system and demonstrate its ability to detect and resolve deadlocks more effectively than other Conventional algorithms, e.g., (Yoo et al. 2005; Nguyen and Le 2015).

During the development of the model, it was found that the developed model has a disadvantage with the increase the number of AGVs (see Figure 5). It can be seen very clearly that as the number of AGVs increases, the number of possible actions in the environment increases exponentially.

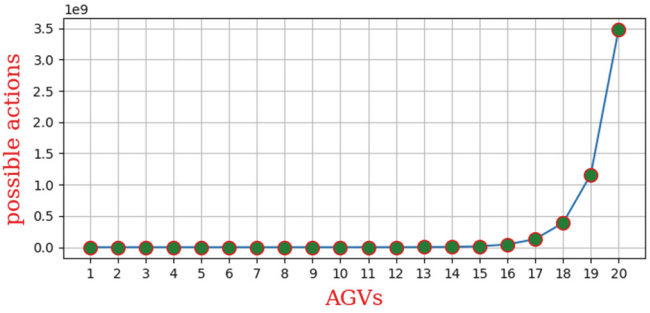

Figure 5: Increase possible actions when increasing the number of AGVs

In conclusion, the simulation model developed in Plant Simulation and Python provides a powerful tool for evaluating the effectiveness of different algorithms in resolving deadlocks in AGV systems. The illustrations generated from the results of the simulation model provides valuable insights into the performance of the system and the potential benefits of using reinforcement learning techniques in the optimization of decision-making processes in complex dynamic environments.

6 Outlook

In fact, the use of neural networks can provide effective solutions to the complex problems associated with material handling systems using AGVs. The machine learning approaches, especially reinforcement learning, allow the analysis of large systems with multiple resources and movements, whereby deadlocks can be detected and avoided before they occur. In addition, reinforcement learning models can learn from past experience and predict optimal solution strategies in real time, reducing the risk of system downtime and increasing overall system efficiency. These methods also provide a more dynamic and adaptive approach to managing material handling systems, allowing the system to respond quickly to changes in demand and capacity. In summary, the integration of neural networks offers a promising avenue to develop effective solutions to detect, avoid and resolve deadlocks in AGV-based material handling systems. Multi Agent Reinforcement Learning has shown to be a powerful technique for solving complex decision-making problems in dynamic environments. In the case of AGV systems, multi-agent reinforcement learning can be particularly useful for detecting deadlocks and resolving them more effectively than other algorithms such as DQN.

Acknowledgement

The research was funded by the Bavarian State Ministry for Economic Affairs, Regional Development and Energy as part of the R&D program "Information and communication technology" of the Free State of Bavaria, project title: KAnIS: Cooperative Autonomous Intralogistics Systems. Project partner: Linde Material Handling GmbH, Aschaffenburg.

References

Friji, H.; Ghazzai, H.; Besbes, H.; Massoud, Y.: A DQN-based autonomous car-following framework using RGB-D frames. IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT) 12-16 December 2020 Dubai (2020), pp. 1-6. https://doi.org/10.1109/GCAIoT51063.2020.9345899.

Hussain, S.; Kumar, S.; Janardh, G.: Deadlock avoidance and re-routing of automated guided vehicles in flexible manufacturing systems. International Journal of Advances in Production and Mechanical Engineering (IJAPME) 1 (2015) 4.

Javaid, A.: Understanding Dijkstra's algorithm. SSRN Electronic Journal (2013). https://doi.org/ 10.2139/ssrn.2340905.

Kuhnle, A.: Adaptive order dispatching based on reinforcement learning application in a complex job shop in the semiconductor industry. Ph.D Thesis, Karlsruher Institut für Technologie, Karlsruhe (Germany). 2020. https://doi.org/10.5445/IR/1000127740.

Lee, M.; Yusuf, H.: Mobile robot navigation using deep reinforcement learning. Processes 10 (2022) 12. https://doi.org/10.3390/pr10122748.

Li, X.; Zhang, C.; Yang, W.; Qi, M.: Multi-AGVs conflict free routing and dynamic dispatching strategies for automated warehouse. In: Kim, K.; Kim, H. (Eds.) Mobile and Wireless Technology 2018. ICMWT 2018. Lecture Notes in Electrical Engineering 513 (2019). https://doi.org/10.1007/978-981-13-1059-1_26.

Mignon, A.; Luis de Azevedo da Rocha, R.: An adaptive implementation of epsilon-greedy in reinforcement learning. Procedia Computer Science 109 (2017), pp. 1146-1151, https://doi.org/10.1016/j.procs.2017.05.431.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M.: Playing Atari with deep reinforcement learning. 2013. https://doi.org/10.48550/arXiv.1312.5602.

Müller, M.; Reggelin, T.; Kutsenko, I.; Zadek, H.; Reyes-Ruiano, L.: Towards deadlock handling with machine learning in a simulation based learning environment. Proceedings of the 2022 Winter Simulation Conference 11-14 December 2022 Singapore, 2022, pp. 1485-1496. https://doi.org/10.1109/WSC57314.2022.10015270.

Nguyen, H.; Le, V.: Detection and avoidance deadlock for resource allocation in heterogeneous distributed platforms. International Journal of Critical Infrastructures 6 (2015) 2.

Ni, Q.; Sun, W.; Ma, S.: Deadlock detection based on resource allocation graph. Fifth International Conference on Information Assurance and Security, Xi'an, China, 2009, pp. 135-138, https://doi.org/10.1109/IAS.2009.64.

Sumanas, M.; Petronis, A.; Bucins, V.; Bučinskas, V.; Dzedzickis, A.; Virzonis, D.; Morkvenaite, V.: Deep q-learning in robotics: improvement of accuracy and repeatability. Sensors 2022 MDPI 22 (2022) 10. https://doi.org/10.3390/s22103911.

Sutton, R.; Barto, A.: Reinforcement learning: an introduction. IEEE Transactions on Neural Networks 16 (2005), pp. 285-286. https://doi.org/ 10.1109/TNN.1998.712192.

Vivaldini, K.; Rocha, L.; Becker, M.; Moreria, M.: Comprehensive review of the dispatching, scheduling and routing of AGVs. Moreira, Matos, A.; Veiga, A.; G. (Eds.) CONTROLO'2014 – Proceedings of the 11th Portuguese Conference on Automatic Control. Lecture Notes in Electrical Engineering 321 (2015). https://doi.org/10.1007/978-3-319-10380-8_48.

Wang, X.; Wang, Q.; Guo, Y.; Lu, J.: Deadlock detection based on parallel graph theory algorithm. Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013), 2013, https://doi.org/10.2991/iccsee.2013.180.

Wang, Y.; Li, D.; Ouyang, B.; Wu, D.: Artificial Intelligence empowered multi-AGVs in manufacturing systems. https://arxiv.org/abs/1909.03373, accessed January 13th, 2023.

Xu, J.; Zheng, Z.; Lyu, M.: CGA-based deadlock solving strategies towards vehicle sensing systems. EURASIP Journal on Wireless Communications and Networking 2014 (2014). https://doi.org/10.1186/1687-1499-2014-214.

Yao, B.; Yin, J.; Wu, W.: Deadlock avoidance based on graph theory. International Journal of u- and e- Service, Science and Technology 9 (2016), pp. 353-362. https://doi.org/10.14257/ijunesst.2016.9.2.34.

Yoo, J.; Sim, E.; Cao, C.; Park, J.: An algorithm for deadlock avoidance in an AGV system. The International Journal of Advanced Manufacturing Technology 26 (2005), pp. 659-668. https://doi.org/10.1007/s00170-003-2020-4.