Proceedings of the 2019 Winter Simulation Conference
N. Mustafee, K.-H.G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, eds.

TASK SELECTION BY AUTONOMOUS MOBILE ROBOTS IN A WAREHOUSE USING DEEP REINFORCEMENT LEARNING

Maojia P. Li
Prashant Sankaran
Michael E. Kuhl

Rochester Institute of Technology
Industrial & Systems Engineering Department
Rochester, NY 14623, USA

Raymond Ptucha
Amlan Ganguly
Andres Kwasinski

Rochester Institute of Technology
Computer Engineering Department
Rochester, NY 14623, USA

ABSTRACT

We introduce a deep Q-network (DQN) based model that addresses the dispatching and routing problems for autonomous mobile robots. The DQN model is trained to dispatch a small fleet of robots to perform material handling tasks in a virtual, as well as, in an actual warehouse environment. Specifically, the DQN model is trained to dispatch an available robot to the closest task that will avoid or minimize encounters with other robots. Based on a discrete event simulation experiment, the DQN model outperforms the shortest travel distance rule in terms of avoiding traffic conflicts, improving the makespan for completing a set of tasks, and reducing the mean time in system for tasks.

1 INTRODUCTION

In recent years, autonomous mobile robots (AMRs) have been widely implemented in the areas of transportation, surveillance, and inspection. These systems often require a team of AMRs working together to achieve a desired outcome. The operation and collaboration of AMRs in a constrained environment is a challenging problem, and there are many open questions remaining. In this paper, we address the task selection and routing problems for a team of material handling AMRs in a warehouse environment. In particular, the warehouse system that we consider is one in which a fleet of autonomous mobile robots (autonomous forklifts) need to work together to complete storage and retrieval tasks of unit loads (pallets) as efficiently as possible. Further, due to relatively narrow warehouse aisles the travel of an AMR may be impeded if other AMRs are encountered (e.g., passing, following, performing retrieval task, etc.) en route to its pick-up/drop-off location. Thus, our goal is to design a method with the goal of avoiding traffic and completing the tasks as efficiently as possible.

In material handling, task selection problems are typically addressed either using an optimization model to generate delivery schedules for a specified time horizon or a rule-based dispatching algorithm to make real-time decisions. However, the warehouse environment is highly stochastic. This uncertainty arises from multiple sources including variability in customer orders, arrival times of trucks, material handling times, and packaging time, among others. Thus, constructing a schedule for AMRs that will maintain optimality over a long horizon is extremely difficult in this dynamic environment. Given this, dispatching is often more favorable (Le-Anh and De Koster 2006). Dispatching algorithms attempt make favorable decisions for the available AMR based on real-time information, such as travel distance, task waiting time, queue size, etc. Although these decisions may not be optimal in the long run, this greedy heuristic approach can lead to a series of good dispatching decisions.

Therefore, for AMRs in a warehouse environment we propose a novel dispatching approach. In particular, we develop a methodology that utilizes a deep reinforcement learning framework called Deep

Q-Network (DQN) (Mnih et al. 2015). The model aims to reduce the overall mission time (makespan) to complete a list of tasks using a fleet of AMRs. The DQN model is designed to avoid vehicle traffic on the dispatching level while taking into account the travel distance to the task.

The remainder of the paper is organized as follows. In Section 2 we summarize related work in the areas of dispatching and deep reinforcement learning. We present the DQN-based task selection and routing methodology in Section 3. In Section 4, we present the results of a set of simulation experiments and compare the performance of the DQN model to the shortest travel distance (STD) dispatching rule. Finally, in Section 5, we present our conclusions.

2 RELATED WORK

2.1 Dispatching and Routing of Material Handling Robots

Scheduling and dispatching are two popular approaches to solve the task selection problem. A scheduling model combined with a routing algorithm decides when, where, and how a robot should act to perform delivery tasks (Fazlollahtabar and Saidi-Mehrabad 2015). Le-Anh and De Koster (2006) classify material handling robot scheduling into online and offline models. In the offline case, all transportation requests are known in advance so that delivery schedules can be pre-determined before the start of production (Corréa et al. 2007; Deroussi et al. 2008; Miyamoto and Inoue 2016). In practice, environments are often stochastic due to job arrival time, travel time, loading and unloading time fluctuations, and vehicle and machine breakdowns. An online scheduling model updates delivery schedules based on the real-time information of the system (Nishi et al. 2011; Nishi and Tanaka 2012; Umar et al. 2015). The major challenges of online scheduling are reformulating the optimization model based on online information and solving the model in real-time (Fazlollahtabar and Saidi-Mehrabad 2015).

A dispatching algorithm makes real-time delivery decisions. Egbelu and Tanchoco (1984) find that the performance of a material handling system is mainly governed by vehicle-initiated dispatching problems. A vehicle-initiated problem describes the scenario when a vehicle becomes available and needs to determine its next task. The decision is often made based on some current system attributes (queue size, buffer capacity, machine utilization, etc.) or task properties (distance, waiting time, due date, etc.) (Klei and Kim 1996). Most recent studies use a combination of both to make dispatching decisions (Jeong and Randhawa 2001; Yang et al. 2007; Guan and Dai 2009). Researchers find these multi-attribute dispatching algorithms outperform single-attribute dispatching rules under certain conditions.

Although a number of researchers consider dispatching algorithms, only a few researchers have considered the dispatching and routing problems simultaneously. Desaulniers et al. (2003) formulate a methodology where the objective is to minimize the travel distance while avoiding multiple vehicles utilizing the same guide-path segments. In addition, Wu and Zhou (2007) formulate an optimization model to find the closest destination for the dispatched vehicle as well as the route to approach to the destination that avoids vehicle blocking and deadlock situations.

2.2 Deep Reinforcement Learning

Reinforcement learning (RL) algorithms have been implemented to dispatch AMRs in several studies. Makar et al. (2001) implement a hierarchical reinforcement learning model to dispatch multiple material handling robots. Their model enables multiple vehicles working collaboratively in a warehouse environment. Li et al. (2002) train RL and memory-based RL algorithms to make option and action level decisions, which reflect dispatching and routing problems, respectively. Proper and Tadepalli (2006) present an after state version H-learning (AVH) model to dispatch product delivery agents. The model resolves the scale-up problem in table-based reinforcement learning algorithms.

The DRL algorithm has been successfully applied in many fields of robotics, such as localization, robot manipulation, and mobile robot navigation. It combines deep neural network (NN) with RL algorithms. The most popular DRL model is DQN — originally introduced by Mnih et al. (2015) to play Atari 2600

video games. The model learns, from raw pixels, policies to play various video games at human-level performances. The video game is formulated into a Markov decision process. The current frame of the game is observed as the current state of the environment, s_t. An agent in the environment makes an action, a_t that transits the environment to the next state, s_{t+1}. The agent receives a reward r_t based on the outcome of the transition. The objective is to maximize the discounted cumulative reward R_t at any time,

$$R_t = \sum_{k=0}^{T-t} \gamma^k r_{t+k},$$

where $\gamma \in [0,1]$ is the discount factor that encourages the agent to get a reward as soon as possible and T is the length of an episode. The state transformation, action, and reward information are saved in the DQN memory for training purposes. After gradually updating the weights in the NN, the DQN learns to predict the cumulative rewards, $Q(s_t, a_t; \theta_i)$ (also referred as state-action value or Q-value) for each action at any state, through forward propagation using online weights, θ_i. The weight parameters are copied and updated after every τ steps, and the offline weights, θ_{i-1}, are used to compute the target Q-value:

$$Q(s_t, a_t; \theta_{i-1}) = \mathbb{E}\left[r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_{i-1}) \right]. \tag{1}$$

Thus the loss function of Q-value estimate can be expressed as,

$$L(\theta_i) = \left[\left(r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_{i-1} \right) - Q(s_t, a_t; \theta_i) \right]^2. \tag{2}$$

The success of the DQN model had drawn researchers' attention, and many extensions of DQN have been proposed to enhance stability and training speed. Wang et al. (2016) present a dueling DQN model that has two streams of computation sharing the convolutional layers. Schaul et al. (2016) propose a prioritized replay model that encourages the DQN to replay experiences with higher TD errors. Van Hasselt et al. (2016) introduce a double DQN model to mitigate the overestimate issue in q-learning. In the double DQN model, the optimal action of the next state and the corresponding Q-value are computed using online and offline weights, respectively. Mnih et al. (2016) present an n-step DQN that uses the next n-step rewards in the memory to compute the target Q-value more accurately. Rather than estimating the expected return from each action, Bellemare et al. (2017) develop a distributional DQN that predicts the distribution of the return. Fortunato et al. (2018) introduce a noisy DQN that uses noisy weights to control the exploration and exploitation stages of the model. Finally, Hessel et al. (2018) combine all six extensions discussed above to establish a rainbow DQN.

3 METHODOLOGY

3.1 DQN-based Dispatching System

In this section, we introduce a DQN-based dispatching system. An overview of the dispatching system framework is depicted in Figure 1. Once an agent becomes available, it will send a request to the dispatching system. The dispatching system collects the current location of the dispatching agent, current routes of active agents, and locations of awaiting tasks. The active agents are AMRs currently executing delivery tasks. Based on this information, the current state of the environment, s_t, is summarized to a 110×110 image in a virtual environment. The empty space, obstacles, active agents, dispatching agent, and possible task locations are represented by different pixel values. The DQN determines an action, a_t, for the dispatching agent, which is to move to one of the neighboring pixels to the north, south, west, or east of the current pixel. As the active agents' locations are also updated based on their routes, the virtual environment will transit to the next state s_{t+1} and receive a reward r_t based on a_t. The transition (s_t, a_t, r_t, s_{t+1}) is saved in

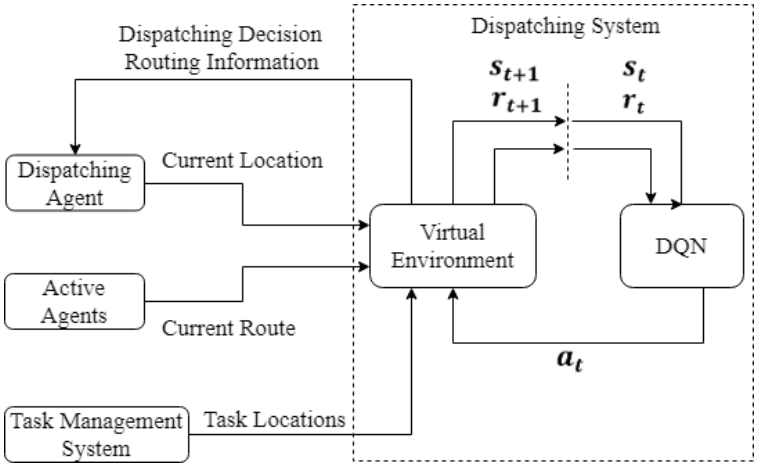


Figure 1: Framework of DQN-based dispatching system.

the DQN memory for training. The process repeats until the dispatching vehicle reaches a task location. This location reflects the dispatching decision from the DQN. The DQN also provides high-level routing information (waypoints) to the agent based on the path used in the virtual environment to avoid traffic.

The DQN model implemented in this study is based on the rainbow DQN (Hessel et al. 2018). Based on a preliminary experiment, we observe that double DQN, dueling DQN, distributional DQN, and prioritized replay enhance DQN's performance, while n-step DQN and noisy DQN increase the training time without improving the performance. In this case, we implement a DQN model that only combines the first four extensions. In a dueling DQN, the NN consists of two streams of computations sharing convolution encoders. The Q-value is determined by state value $V(s_t;\xi,\eta)$ and advantage value $A(s_t,a_t;\xi,\psi)$, such that

$$Q(s_t,a_t) = V(s_t;\xi,\eta) + A(s_t,a_t;\xi,\psi) - \frac{\sum_a A(s_t,a;\xi,\psi)}{|A|}.$$

where ξ, η and ψ are the weights for the convolution encoders, fully connected layers in V stream, and fully connected layers in A stream, respectively. The output passes through a softmax layer to obtain the normalized Q-value.

When computing the target Q-value, the double DQN utilizes online weights of the NN to compute the optimal action of the next state, a^*_{t+1}, by

$$a^*_{t+1} = \underset{a_{t+1}}{\operatorname{argmax}}\, Q(s_{t+1},a_{t+1};\theta_i).$$

The offline weights determine the Q-value that corresponds to the optimal decision, $Q(s_{t+1},a^*_{t+1};\theta_{i-1})$. The double learning technique reduces the bias introduced by the *max* function in (1) and (2). Rather than computing the expected return, our DQN predicts the distribution of return. Thus, the output from the NN has a dimension of $N_{action} \times N_{atom}$, where N_{action} is the number of possible actions and N_{atom} is the number of atoms in a discrete distribution. The loss of an estimate is the KL-divergence between the predicted and target distribution, $D_{KL}(Q^{'}||Q)$. The target distribution is expressed as,

$$Q^{'} \equiv \Phi_z(r_t + \gamma z, Q(s_{t+1},a^*_{t+1};\theta_{i-1})),$$

where Φ_z is a L2-projection of the output Q-distribution from the NN to a fixed support z. The NN consists of five convolution layers and two fully connected (FC) layers. The convolutional (Conv) layers extract the high level content information from the image while the FC layers learn a policy for decision making. The FCa1 and FCa2 are advantage value computations, and the FCv1 and FCv2 are state value streams. A summary of the dimensions of the DQN are shown in Table 1.

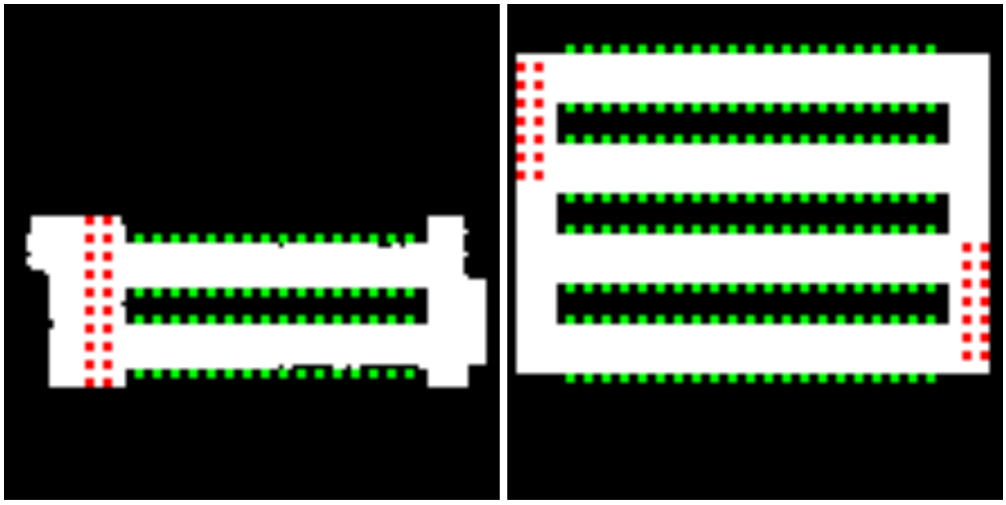
Table 1: Neural network architecture.

Layer	Dimension	
Input	$110 \times 100 \times 4$	
Conv1	16, $6 \times 6 \times 4$, stride=2	
Conv2	32, $5 \times 5 \times 16$, stride=2	
Conv3	64, $5 \times 5 \times 21$, stride=2	
Conv4	64, $3 \times 3 \times 64$, stride=1	
Conv5	64, $3 \times 3 \times 64$, stride=1	
FCa1, FCv1	3135×1408	3136×1408
FCa2, FCv2	$1408 \times N_{action} N_{atom}$	1408×1

3.2 DQN Training

The DQN-based dispatching system is trained in the two simulation environments shown in Figure 2. The environment on the left (S1) is generated by a robot equipped with LiDAR using simultaneous localization and mapping in a real warehouse. We summarize the environment with a 110×110 image, where each pixel represents a 1 ft. by 1 ft. area. The environment on the right (S2) is a virtual environment. A task's pickup location can either be on a shelf (shelf location) or on the floor near the docking area (dock location). All possible shelf and dock locations are shown in Figure 2. At the beginning of a training episode, the locations of the dispatching and active agents are randomly initialized at any of the empty aisle positions. Each active agent is randomly assigned to a task and the shortest route is computed using an A-star algorithm. The training model also randomly generates 1 to 15 pickup locations, which can be either at a shelf or dock location. The dispatching agents are manipulated by the DQN model until each of them reaches a pickup location.

When the DQN moves the dispatching agent to a pixel occupied by an active agent, the episode is terminated. If such a decision is implemented, the two AMRs will encounter each other in the real world. Another termination condition is when DQN fails to move the dispatching agent to a pickup location within 100 steps. The agent receives a -0.05 reward when moving to an empty space, a 2.5 reward when reaching



(a) (b)

Figure 2: Simulation environments for (a) S1, and (b) S2. The black and white pixels represents obstacles and empty space, respectively. All shelf and dock locations are represented by green and red squares, respectively.

a destination, and a -2.5 reward when encountering an active agent. If the DQN directs the agent to an obstacle, the agent will remain at the current location and receive a -0.5 reward. Based on our experiments, such a reward system yields good performance.

The DQN model is trained for 100,000 episodes in each environment. The exploration and exploitation phases of DQN is controlled through an $\varepsilon - greedy$ approach. When DQN is deciding an action, it randomly generates a value between 0 and 1. If the value is larger than the current ε, the DQN makes a decision that maximizes the return. Otherwise, the DQN randomly chooses an action. The randomness encourages DQN to explore unseen states at the beginning of training. The ε has an initial value of 1 and linearly decreases to 0 in 1,000,000 training steps. Figure 3 shows the training results using 3 agents in S2 for percentage of time that the agent successfully reaches a pickup location (success rate), DQN directs an agent to an obstacle (obstacle rate), agent encounters (conflict rate) and passes (traffic rate) an active agent over 1000 episodes. The DQN agent learns to maximize the cumulative reward by finding a close destination that does not cause a conflict.

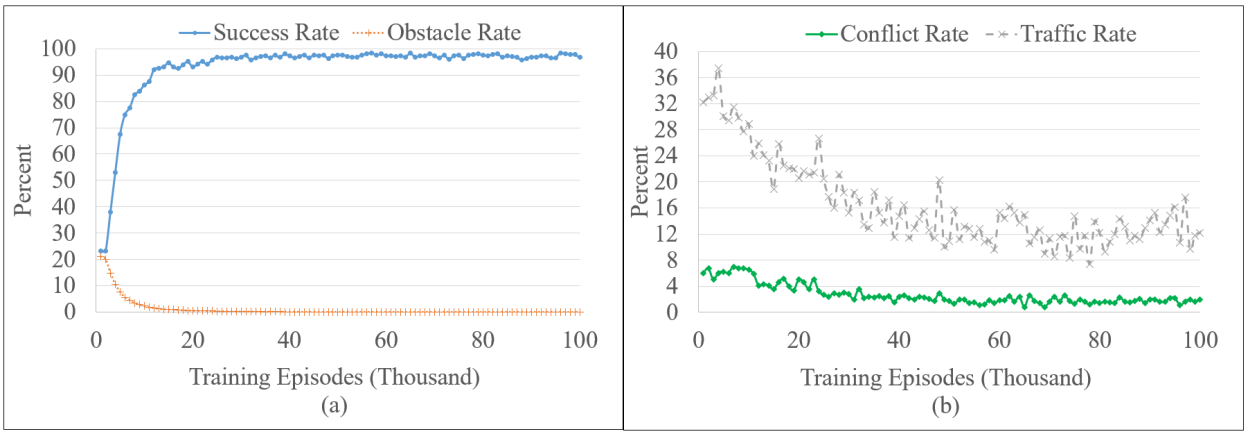


Figure 3: Training performance of DQN with 3 agents in S2.

4 SIMULATION EXPERIMENT

After training the model, the DQN model is compared with the shortest travel distance (STD) rule in S1 with 2 agents (S1A2), S2 with 3 agents (S2A3) and S2 with 4 agents (S2A4). The performance measures include the mean flow time (MFT) of parts, makespan, total travel distance for agents, and number of times two or more agents are closer than 4 ft. to each other (traffic). When multiple agents get closer than 4 ft., all agents will stay at the current location for 15 time steps. To simplify the traffic logic in the simulation model, we assume agents can tackle any types of conflict within 15 time steps. Such an assumption may not always hold true in a real system and may underestimate the impact of traffic. An agent normally moves at 4 ft. per time step. The loading and unloading process both take 15 time steps. At the beginning of an replication, the system randomly generates 20 tasks. A task can be either delivering from a shelf location to a dock location, or vice versa. The replication ends when all tasks are completed. The initial location of each agent is also randomly determined.

Table 2 compares the performances of DQN with STD in terms of traffic. In all three scenarios, DQN reduces the occurrence of traffic. Although the traffic density in S1A2 and S2A4 are very similar (one agent per aisle and two agents per dock on average), the most significant percentage improvement is in S2A4, where 28.81% of traffic can be avoided using DQN. Having an additional agent in S2 increases the percentage difference between STD and DQN. Table 3 compares the total travel distance for agents. The trade-off of using DQN is the increase in travel distance. In order to reduce on average 8.50 traffic occurrences in S2A4, the total travel distance of DQN increased by 77.94 ft. (3.20%). Having an additional

agent in S2 increases the percentage difference between DQN and STD in travel distance. The most significant percentage increase is 4.08% in S1A2.

The net effect of reducing traffic but increasing distance is captured by makespan shown in Table 4. The DQN model outperforms STD rule in all scenarios. The scenario with a higher traffic density receives greater benefit from the DQN model. The most significant absolute and percentage improvement occurs in S2A4, where the overall mission time is reduced by 34.00 time steps (9.86%). Table 5 shows that the DQN model also reduces the mean flow time of parts across all scenarios. When using DQN in S2A4, the MFT is decreased by 12.54% when compared to STD.

Table 2: Mean and standard deviation for traffic.

Scenario	Traffic			
	STD Unit	DQN Unit	Diff. Unit	Diff. %
S1A2	25.83 (10.47)	20.25 (9.19)	5.59 (9.81)	21.62
S2A3	21.48 (8.05)	17.41 (7.93)	4.07 (9.29)	18.95
S2A4	29.50 (9.98)	21.00 (9.43)	8.50 (12.06)	28.81

Table 3: Mean and standard deviation for distance.

Scenario	Distance			
	STD ft.	DQN ft.	Diff. ft.	Diff. %
S1A2	1771.16 (153.04)	1843.41 (154.52)	-72.25 (50.13)	-4.08
S2A3	2430.74 (177.02)	2488.68 (165.87)	-57.94 (85.46)	-2.38
S2A4	2434.25 (201.05)	2512.16 (202.99)	-77.94 (100.41)	-3.20

Table 4: Mean and standard deviation for makespan.

Scenario	Makespan			
	STD time step	DQN time step	Diff. time step	Diff. %
S1A2	575.14 (92.78)	544.27 (70.73)	30.86 (58.03)	5.37
S2A3	413.98 (58.22)	397.45 (56.41)	16.53 (37.38)	3.99
S2A4	344.76 (56.77)	310.76 (50.12)	34.00 (39.72)	9.86

Table 5: Mean and standard deviation for MFT.

Scenario	MFT			
	STD time step	DQN time step	Diff. time step	Diff. %
S1A2	374.34 (78.25)	347.61 (66.06)	26.73 (52.63)	7.14
S2A3	244.51 (58.48)	227.29 (58.10)	17.22 (55.65)	7.04
S2A4	200.70 (57.73)	174.92 (54.14)	25.08 (60.21)	12.54

5 CONCLUSION

In this study, we introduce a deep Q-network based dispatching algorithm that addresses the dispatching and routing problems for AMRs. The DQN model manipulates the dispatching agent in a virtual environment and outputs the dispatching destination and high level routing information. The DQN model is tested in a real and a virtual warehouse environment through discrete event simulation. When compare to the STD rule, the DQN model reduces the occurrences of traffic congestion but increases the travel distance. The net effect reduces the makespan to complete a fixed number of delivery tasks and the mean flow time of tasks. In general, a system with higher traffic density tends to benefit more from the DQN model.

ACKNOWLEDGMENTS

This research is sponsored in part by a grant form Toyota Material Handling of North America and Raymond Corporation.

REFERENCES

Bellemare, M. G., W. Dabney, and R. Munos. 2017. "A Distributional Perspective on Reinforcement Learning". In *Proceedings of the* 34^{th} *International Conference on Machine Learning*, edited by D. Precup and Y. W. Teh, August 6^{th}–11^{th}, Sydney, Australia, 449–468.

Corréa, A. I., A. Langevin, and L.-M. Rousseau. 2007. "Scheduling and Routing of Automated Guided Vehicles: a Hybrid Approach". *Computers & Operations Research* 34(6):1688–1707.

Deroussi, L., M. Gourgand, and N. Tchernev. 2008. "A Simple Metaheuristic Approach to the Simultaneous Scheduling of Machines and Automated Guided Vehicles". *International Journal of Production Research* 46(8):2143–2164.

Desaulniers, G., A. Langevin, D. Riopel, and B. Villeneuve. 2003. "Dispatching and Conflict-free Routing of Automated Guided Vehicles: An Exact Approach". *International Journal of Flexible Manufacturing Systems* 15(4):309–331.

Egbelu, P. J., and J. M. Tanchoco. 1984. "Characterization of Automatic Guided Vehicle Dispatching Rules". *The International Journal of Production Research* 22(3):359–374.

Fazlollahtabar, H., and M. Saidi-Mehrabad. 2015. "Methodologies to Optimize Automated Guided Vehicle Scheduling and Routing Problems: a Review Study". *Journal of Intelligent & Robotic Systems* 77(3-4):525–545.

Fortunato, M., M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg. 2018. "Noisy Networks for Exploration". In 6^{th} *International Conference on Learning Representations*, April 30^{th}– May 3^{rd}, Vancouver, Canada, 1801–1810.

Guan, X., and X. Dai. 2009. "Deadlock-free Multi-attribute Dispatching Method for AGV Systems". *The International Journal of Advanced Manufacturing Technology* 45(5-6):603.

Hessel, M., J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. 2018. "Rainbow: Combining Improvements in Deep Reinforcement Learning". In 32^{nd} *AAAI Conference on Artificial Intelligence*, February 2^{nd}–7^{th}, New Orleans, Louisiana.

Jeong, B. H., and S. U. Randhawa. 2001. "A Multi-attribute Dispatching Rule for Automated Guided Vehicle Systems". *International Journal of Production Research* 39(13):2817–2832.

Klei, C., and J. Kim. 1996. "AGV Dispatching". *International Journal of Production Research* 34(1):95–110.

Le-Anh, T., and M. De Koster. 2006. "A Review of Design and Control of Automated Guided Vehicle Systems". *European Journal of Operational Research* 171(1):1–23.

Li, X., T. Geng, Y. Yang, and X. Xu. 2002. "Multiagent AGVs Dispatching System Using Multilevel Decisions Method". In *Proceedings of the 2002 American Control Conference,*, May 8^{th}–10^{th}, Anchorage, Alaska, 1135–1136.

Makar, R., S. Mahadevan, and M. Ghavamzadeh. 2001. "Hierarchical Multi-agent Reinforcement Learning". In *Proceedings of the 5^{th} International Conference on Autonomous Agents*, May 28^{th}–June 1^{st}, Montreal, Canada, 246–253.

Miyamoto, T., and K. Inoue. 2016. "Local and Random Searches for Dispatch and Conflict-free Routing Problem of Capacitated AGV Systems". *Computers & Industrial Engineering* 91(1):1–9.

Mnih, V., A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. 2016. "Asynchronous Methods for Deep Reinforcement Learning". In *International Conference on Machine Learning*, edited by M. F. Balcan and K. Q. Weinberger, June 19^{th}–24^{th}, New York, New York, 1928–1937.

Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, B. Charles, S. Amir, A. Loannis, K. Helen, K. Dharshan, W. Daan, L. Shane, and H. Demis. 2015. "Human-level Control Through Deep Reinforcement Learning". *Nature* 518(7):529–533.

Nishi, T., Y. Hiranaka, and I. E. Grossmann. 2011. "A Bilevel Decomposition Algorithm for Simultaneous Production Scheduling and Conflict-free Routing for Automated Guided Vehicles". *Computers & Operations Research* 38(5):876–888.

Nishi, T., and Y. Tanaka. 2012. "Petri Net Decomposition Approach for Dispatching and Conflict-free Routing of Bidirectional Automated Guided Vehicle Systems". *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 42(5):1230–1243.

Proper, S., and P. Tadepalli. 2006. "Scaling Model-based Average-reward Reinforcement Learning for Product Delivery". In *European Conference on Machine Learning*, September 18^{th}–22^{nd}, Berlin, Germany, 735–742.

Schaul, T., J. Quan, I. Antonoglou, and D. Silver. 2016. "Prioritized Experience Replay". In *Proceedings of the International Conference on Learning Representations*, May 2^{nd}–4^{th}, San Juan, Puerto Rico, 245–250.

Umar, U. A., M. Ariffin, N. Ismail, and S. Tang. 2015. "Hybrid Multiobjective Genetic Algorithms for Integrated Dynamic Scheduling and Routing of Jobs and Automated-guided Vehicle (AGV) in Flexible Manufacturing Systems (FMS) Environment". *The International Journal of Advanced Manufacturing Technology* 81(9-12):2123–2141.

Van Hasselt, H., A. Guez, and D. Silver. 2016. "Deep Reinforcement Learning with Double Q-learning". In 30^{th} *AAAI Conference on Artificial Intelligence*, February 12^{th}–17^{th}, Phoenix, Arizona, 2094–2100.

Wang, Z., T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas. 2016. "Dueling Network Architectures for Deep Reinforcement Learning". In *Proceedings of the 33^{rd} International Conference on Machine Learning*, June 19^{th}–24^{th}, New York, New York, 1995–2003.

Wu, N., and M. Zhou. 2007. "Shortest Routing of Bidirectional Automated Guided Vehicles Avoiding Deadlock and Blocking". *IEEE/ASME Transactions on Mechatronics* 12(1):63–72.

Yang, T., Y. Kuo, and C. Cho. 2007. "A Genetic Algorithms Simulation Approach for the Multi-attribute Combinatorial Dispatching Decision Problem". *European Journal of Operational Research* 176(3):1859–1873.

AUTHOR BIOGRAPHIES

MAOJIA P. LI is a Ph.D. student in Engineering at Rochester Institute of Technology. His email address is mxl8487@rit.edu.

PRASHANT SANKARAN is an Ph.D. student in Engineering at Rochester Institute of Technology. His research interests include simulation, optimization, and machine learning in supply chain and manufacturing systems. His email address is pxs8917@rit.edu.

MICHAEL E. KUHL is a Professor in the Industrial and Systems Engineering Department at Rochester Institute of Technology. His research interests include modeling and simulation of stochastic arrival processes, and the application of simulation to autonomous material handling, healthcare, and manufacturing systems. He is a member of the WSC Board of Director representing the INFORMS Simulation Society. He has also served WSC as Proceedings Editor (2005), Program Chair (2013), and Mobile App Chair (2014-2019). His email address is Michael.Kuhl@rit.edu.

RAYMOND PTUCHA is an Associate Professor in the Computer Engineering Department at Rochester Institute of Technology. His research interests include machine learning, computer vision, robotics, graph processing and signal processing, all with an emphasis with deep learning. His the chair of the local IEEE Signal Processing Society. His email address is rwpeec@rit.edu.

AMLAN GANGULY is an Associate Professor in the Computer Engineering Department at Rochester Institute of Technology. His research interests are in energy-efficient interconnection architectures for multicore chips and multichip systems using novel technologies such as wireless and photonic interconnects and data center networks. His email address is axgeec@rit.edu.

ANDRES KWASINSKI is a Professor in the Computer Engineering Department at Rochester Institute of Technology. He is Chief Editor for the IEEE SigPort and Area Editor for the IEEE Signal Processing Magazine. His research interests include cognitive radio and networking, multimedia communications and networking, and Internet-of-Things. His email address is axkeec@rit.edu.