The Graph Neural Network–Based Dynamic Routing Algorithm for Overhead Hoist Transport Vehicles in Semiconductor Fabrication Plants

Jaeho Lee and Young Jae Jang

***Abstract*— Automated material handling systems (AMHS) play a critical role in semiconductor fabrication plants (fabs). The primary type of AMHS used in fabs is the overhead hoist transport (OHT) system, which transports lots between processing machines. A modern large-scale fab may operate thousands of OHT vehicles and thus often experiences OHT vehicle congestion. This paper proposes a reinforcement learning-based dynamic routing algorithm to address the OHT vehicle congestion problem. We develop a graph neural network–based predictive model to determine in advance the situation on an OHT vehicle's succeeding track. This predictive model enables the algorithm to recognize the traffic volume regardless of the data distribution and the track topology. We show via simulation that this novel algorithm reduces the mean OHT vehicle travel time in a controlled case. In future work, we will conduct simulation verification and then apply our model to a commercial OHT management system to study its real-world in-fab performance.**

***Keywords*:** Automated material handling system, overhead hoist transport, dynamic routing, graph neural network

I. INTRODUCTION

The process of forming modern semiconductor wafers consists of numerous steps with re-entrance performed by hundreds of machines [1]. Bundles of 25 wafers are loaded into a unit called a front-opening unified pod (FOUP) for transport via an overhead hoist transport (OHT) system to the machine that performs each step. Thus, an OHT system forms the backbone of an automated material handling system (AMHS) in a fab, as shown in Figure 1.

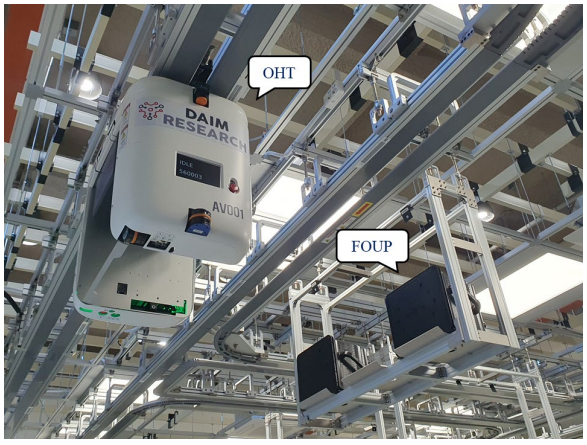


Figure 1. Components of an overhead transport system

The authors thank the Ministry of Education of the Republic of Korea and the National Research Foundation of Korea for their financial support (NRF-2021R1A2C3008172)

Dr. Jaeho Lee is a graduate research assistant at the Korea Advanced Institute of Science and Technology (KAIST), Republic of Korea.

Prof. Young Jae Jang is the Associate Professor of Industrial and Systems Engineering at the Korea Advanced Institute of Science and Technology (KAIST), Republic of Korea. He is also the founder and CEO of DAIM Research Corporation.

The details of OHT systems, their performance measures, and operational aspects can be found in [2]. Due to recent increases in production, most large-scale semiconductor fabs operate thousands of OHT vehicles on a track simultaneously. Such large-scale systems may suffer from severe OHT vehicle congestion, especially as many OHT routes overlap and thus OHT vehicles often become concentrated in certain locations. This congestion increases OHT vehicle travel time, thereby highly increasing the overall cycle time of a product. In worst-case situations, a deadlock occurs, which means that the passage of OHT vehicles is halted. An intuitive way to solve this problem is for OHT vehicles to be able to recognize congestion before encountering it, and thus select a route to use that avoids congestion. In this study, Q routing is used as a route-selection method in the context of congestion. Q routing is a reinforcement learning algorithm that receives a reward based on the edge travel time described in [2]. However, a disadvantage of Q routing is its delayed adaptation in updating a Q value; that is, it updates a Q value after an OHT vehicle has completed traveling an edge. In other words, the succeeding edge travel time is not reflected in the Q value at the moment of route decision-making.

In this study, we solve this delayed adaptation problem by developing a predictive model for edge travel time, which determines the predicted edge travel time as a Q value before an OHT vehicle travels an edge. This enables a process denoted active Q routing, whereby an OHT vehicle is routed to a non-congested edge. The predictive model is based on a graphical neural network (GNN), which ensures that the adjacency status of the track and related information is effectively embedded in the OHT system. In comparison to other models that use the entire set of data from a map, our GNN-based model uses only surrounding edge data and thus has greatly improved learning speed and computational time.

II. LITERATURE REVIEW

A. Pickup and delivery problem with time windows

The vehicle routing problem (VRP) is a conventional study to find the optimal trajectory that minimizes the total distance traveled. In particular, a pickup and delivery problem with time windows (PDPTW) deals with transporting objects from origin to destination like AMHS. Exact algorithm based on branch-and-price, such as [3], [4], and [5], achieve the near-optimal solution efficiently. In addition, many studies propose the meta-heuristic approach to solve PDPTW [6], [7]. An adaptive large neighborhood search heuristic [8] solves various instances of PDPTW with high utility and robustness. Existing PDPTW studies have focused on computational time to overcome NP-hardness. However, in practice, the physical movement

of material handling robots arises congestion and deadlock problems [9].

B. Routing algorithm for the AGV system

The automated guided vehicle (AGV) is a typical material handling robot that moves along a set of predetermined paths on the floor. Studies for the AGV routing algorithm consider physical movement to avoid collision and deadlock in reality [10]. Petri net-based AGV routing algorithms address collision by using state restrict concept [11], [12]. Studies that adding occupation time constraints makes implicit interconnection between AGVs to solve the routing problem [13], [14], and [15]. Jiaoyang, *et al.* [16] proposed multi-agent routing without collisions for warehouses. On the other hand, the increased number of AGVs and map complexity leads to many studies on learning-based algorithms. Reinforcement learning [17] is one of the main methods for the general routing problem. Fundamental Q value-based approach with proper modeling are found in [18], [19]. Moreover, Binyu, *et al.* [20] proposed a dynamic routing model to an arbitrary environment, and Guillaume, *et al.* [21] design the multi-agent reinforcement learning algorithm for robot routing. The graph structure represents the topography, therefore, graph-based AGV routing algorithm have been studied [22], [23].

C. Routing algorithm for the OHT system

The OHT track is a directed graph $G(N, E)$ where N is a set of nodes and E is a set of edges. A unique characteristic of the OHT track compared to other AMHS is a one-way layout and bay structure [24]. Therefore, serious congestion may occur with a traffic volume increase in the central loop that connects bays. Petri-net has also been applied to the OHT routing algorithm to avoid congestion [25], [26]. In addition, the well-known k shortest path algorithm is applied to the OHT routing problem [27]. Moreover, there is a routing algorithm using markov decision process modeling [28] and a practical method using the congestion monitoring system (CMS) [29]. Bartlett, *et al.* [30] proposed the dynamic Dijkstra algorithm that updates the edge cost dynamically, and solved up to the 250 OHTs scale problem. Hwang and Jang [2] proposed the reinforcement learning-based dynamic routing algorithm called Q routing. There exists a Q-learning-based routing algorithm for dynamically changing networks [31], [32]. Hwang and Jang define the Q(λ) learning framework for OHT routing and solve the real fab-sized system in practical computational time.

III. METHOD

A. Route Selection Method: Q Routing

Q routing is a reinforcement learning algorithm that learns the traffic volume of the route. We define Q value $Q[(d, i), j]$ as the estimated remaining travel time for an OHT vehicle traveling from the current node i to the destination node d, where the OHT vehicle chooses its next node j. Therefore, the Q value can be represented as the level of congestion on the succeeding route of an OHT vehicle. Note that (d, i) is a state of the current vehicle, and $j \in A(d, i)$ is an action where $A(d, i)$ is a set of possible succeeding nodes. Each OHT vehicle updates the Q value with $t(i, j)$, the travel time from the current node i to the next node j, whenever it completes traveling an edge. A Q value is dynamically updated with the edge travel time via basic Q(λ) learning. However, before the edge travel time is used as the immediate reward, it is subjected to reward shaping [33] for stability. This is represented by equation (1), in which $\varphi(d, i)$ is the potential function that is the deterministic shortest travel time from the node i to the destination node d. Equation (2) is the one-step temporal difference error. The Q value of the edge is updated with this $\delta(t)$ and by use of the eligibility trace concept.

$$R[(d, i), j] = t(i, j) + \rho\{\varphi(d, j) - \varphi(d, i)\} \quad (1)$$

$$\delta(t) = R[(d, i), j] + \gamma \min Q[(d, j), k] - Q[(d, i), j] \quad (2)$$

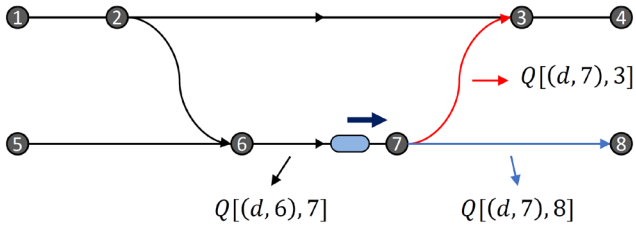
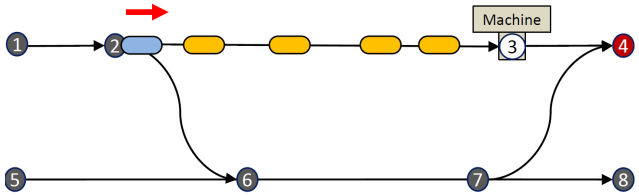


Figure 2. Example of the route selection

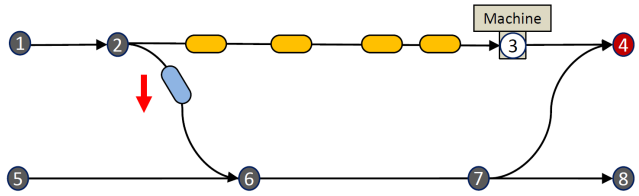
Figure 2 shows an example of a traveling OHT vehicle with a given Q value. When an OHT vehicle arrives at node 7, Q[(*d*, 6), 7] is updated with the edge travel time. This delayed adaptation is denoted as a post update. In addition, when an OHT vehicle arrives at a branching node, such as node 7, it selects its next destination node based on the Boltzmann softmax action policy between succeeding Q values. Since the post update approach can cause problems, we develop an active update approach to update the Q value with edge travel time before an OHT vehicle travels a given edge. The edge travel time $t(i, j)$ is replaced by the predicted edge travel time in equation (1). Figure 3 shows a typical problem caused by the use of a post update approach. Assume that the destination node of the blue OHT vehicle is node 4 and there is a job to be processed on node 3, thus there will be severe congestion on edge (2, 3). In the post update approach, the blue OHT vehicle cannot recognize the congestion on edge (2, 3). This is because no OHT vehicle on edge (2, 3) has completed traveling the edge, and the congestion on this edge is not yet reflected in its Q-value. Thus, the blue OHT vehicle may choose node 3 as its next node. In contrast, our active update approach reflects the predicted edge travel time of edge (2, 3) before the blue OHT vehicle begins traveling edge (2, 3). Therefore, the blue OHT vehicle recognizes the Q value of the edge and consequently detours to node 6.

B. Description of the GNN predictive model

Our novel predictive model for edge travel time is based on the graph neural network (GNN), and the overall network structure is illustrated in Figure 4. We define the machine learning task to be regression, which is a typical form of supervised learning. The predictive model is divided into two parts: a message passing part based on high-level representation extraction, and a regression part. The message passing framework is a spatial-based convolutional GNN that learns a high-level representation of each node in a graph, meaning a representation of remote nodes that are the same distance from the target node. The information from a given level can reach the target node by aggregating the information for the neighboring nodes of each node. This aggregation can be performed using various



(a) Post update, the blue OHT vehicle increases congestion level



(b) Active update, the blue OHT vehicle takes detour

Figure 3. Delayed adaptation problem of the post update approach

forms of the adjacent matrix. Our predictive model is based in particular on GraphSAGE, a general inductive framework [34]. The readout function makes graph representation and the final value is extracted by a fully connected regression network. The target value predicted by the model should not be the scale of the travel time; rather, it should be the additional travel time caused by congestion. Therefore, we define the true label Δt^* as the difference between the actual travel time (with congestion) and ideal travel time (with no congestion).

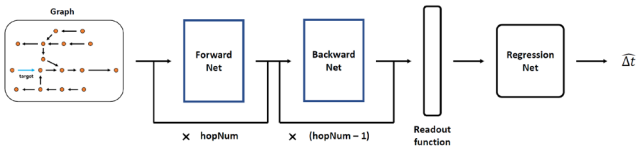


Figure 4. Summary of the predictive model

C. *Data extraction and preprocessing*

The process of preparing the training data is divided into two parts. The first part is the extraction of data from the AutoMod simulation; the second part is the preprocessing of this extracted data to generate PyTorch geometric graph data. Figure 5 shows the overall process of data preparation.

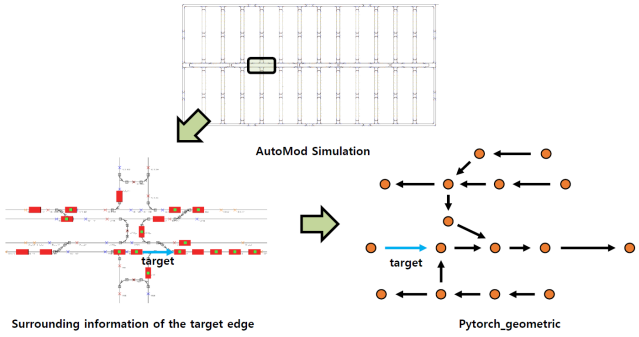


Figure 5. Extraction of training data

Note that graph data are formed by extracting only the surrounding edge data of the target edge to be predicted. This is crucial for a routing algorithm suitable for application to large-scale OHT systems. That is, if the number of the training data is extremely high, the GNN learning time would be prohibitively long and the GNN may exhibit instability and fail to converge. Moreover, as the scale of a map increases, the number of edges increases exponentially, meaning it is impossible to perform learning using the entire data. Figure 6 describes the difference between the entire set of data and the target edge surrounding data, which illustrates the importance of our selection of the graph data structure.

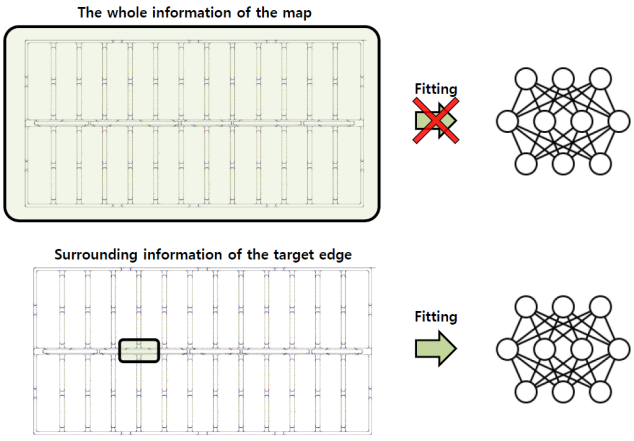


Figure 6. Difference between whole and surrounding information

In the PyTorch geometric conversion process, the model deletes edges that do not affect the target edge from graphs collected as far as a given distance. This reduces the size of the data, which increases its efficiency and clarifies its meaning.

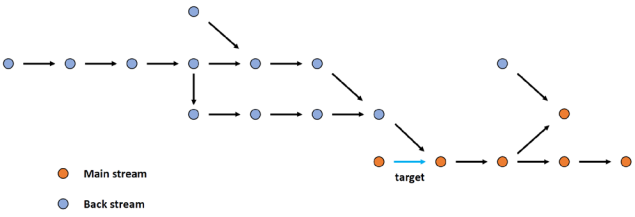


Figure 7. Example of the main stream and a back stream

In addition, we define the main stream and back stream of the graph, as exemplified in Figure 7. The main stream denotes edges that can be reached from the target edge without any direction change, whereas the back stream denotes other edges that are not in the main stream. We assume that edges in different streams have different effects on the target edge. Therefore, as described in Figure 4, our model first applies forward message passing to the back stream, and then backward message passing to the main stream.

D. *Additional methods to increase performance*

We apply the properly modified graph attention network (GAT) [35] structure to the model. The major difference between the OHT problem and general GNN tasks is the existence of the target edge. Moreover, we want to know the influence of each edge on the target edge, not the influence of each edge's neighbors. The score function follows the GAT score structure but uses a softmax for all edges rather than just for the neighbors of each edge.

On the other hand, the distribution of the collected training data is skewed, therefore, the prioritized experience replay (PER) concept is used [36]. PER is a method that greatly improves the performance of the Deep Q Network (DQN) and is used in many reinforcement learning approaches [37]. Figure 8 compares the use of PER for supervised learning and a DQN. We save all of the training data to the replay buffer and sample the mini-batch with priorities. We define the last-seen error for each training data as each priority. Because supervised learning does not have an environment and does not store new samples

through actions, the model updates the priority of all replay buffer data at certain cycles.

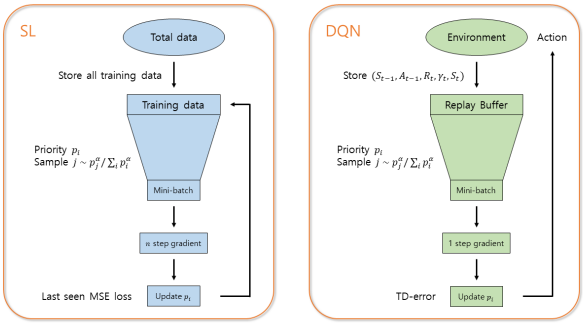


Figure 8. Prioritized experience replay for supervised learning

IV. Numerical Experiments

A. Model without PER using a uniform dataset

We cut the skewed training data which has the maximum value of 212.21s to make a uniform dataset before applying described PER method. The key performance indicators (KPIs) for predictions are the r2 score and mean squared error (MSE). The best model for a uniform dataset has the r2 score of 0.879 and the MSE of 35.172s^2 . We use AutoMod simulation to test the performance of active Q routing. The testbed is a controlled small case with 25 OHT vehicles and 5 machines that is easy to control the traffic volume for each edge. We define the KPI as the mean transport time, i.e., the average time an OHT vehicle takes to travel, after FOUP loading, to the next machine in an OHT system. As the learning progresses, the predictive accuracy of our model gradually increases; thus, based on the validation set r2 score, models are stored at 0.05s intervals from 0s to 0.8s. Then, each model is tested in an AutoMod simulation. Figure 9 depicts the results of this simulation testing, demonstrating that the active update approach outperforms the post update approach if the prediction accuracy is high enough. However, this predictive model has poor extrapolation because we cut the extremely few high travel time data to make training data uniform.

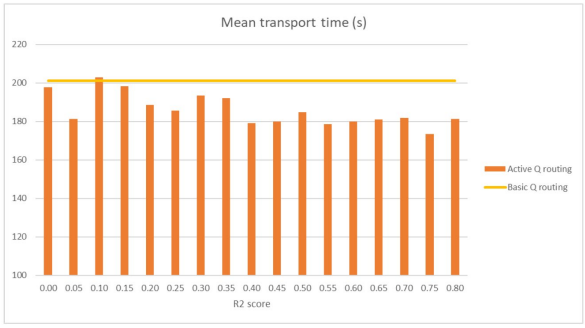


Figure 9. Mean travel time from active update approach according to r2 score

B. Model with PER using a cut entire dataset

We, therefore, expand the range of the dataset to the maximum value of 212.21s, and thus PER is the essential method and is used for supervised learning (as described in the Methods section). As there are extremely few high travel time data points for each class, these cannot be divided into a training set and a test set. Thus, they are all put into a training set. Figure 10 and 11 compare the performance when PER is and is not applied. The PER model generates accurate predictions, even for high travel time data points, but the no-PER model exhibits a lack of learning, as shown by its very insufficient number of high

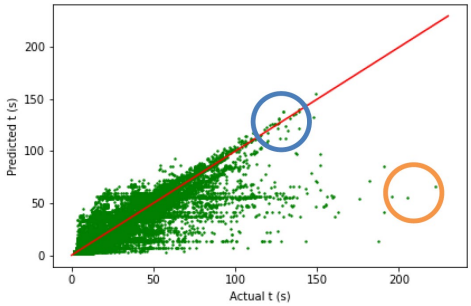


Figure 10. Scatter plot without prioritized experience replay

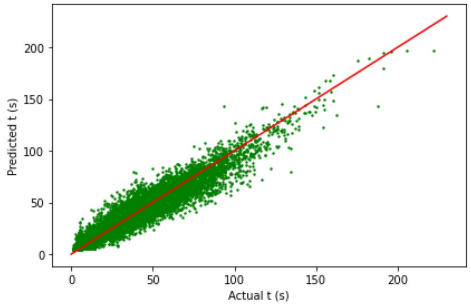


Figure 11. Scatter plot with prioritized experience replay

travel time samples. Based on the training data, the final model has the r2 score of 0.909 and the MSE of 62.861s^2 . We also test this PER model in the AutoMod simulation, which generates the mean travel time shown in Figure 12. The active Q routing with PER model outperforms the dynamic Dijkstra and post Q routing. Also, it outperforms the active Q routing with the no-PER model as prediction accuracy increases for all range data.

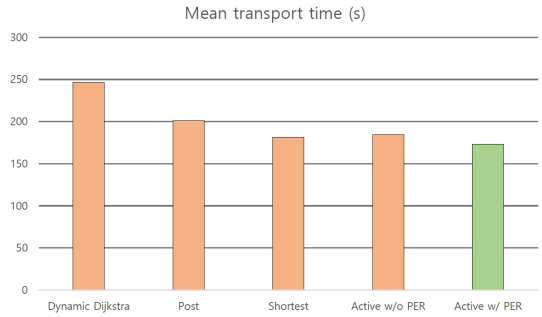


Figure 12. Test of prioritized experience replay model in AutoMod

V. Conclusion

This study develops a GNN-based dynamic routing algorithm to assist OHT vehicles to avoid congested routes in an OHT system. The resulting model has high predictive accuracy when applied to a small map, but whether this model reduces the actual travel time of OHT vehicles in a fab remains to be confirmed. In future work, we will apply our model to a large map to examine its statistical performance over several replications. We will also present mathematical logic on the impact of volatile rewards generated by deep learning models on reinforcement learning performance. After the simulation verification has been completed, our model will be incorporated into a commercial OHT management system and its performance examined in an industrial setting.

REFERENCES

[1] J. Kim, G. Yu, and Y. J. Jang, "Semiconductor fab layout design analysis with 300-mm fab data:"is minimum distance-based layout design best for semiconductor fab design?"," Computers & Industrial Engineering, vol. 99, pp. 330–346, 2016.

[2] I. Hwang and Y. J. Jang, "Q (λ) learning-based dynamic route guidance algorithm for overhead hoist transport systems in semiconductor fabs," International Journal of Production Research, vol. 58, no. 4, pp. 1199– 1221, 2020.

[3] Y. Dumas, J. Desrosiers, and F. Soumis, "The pickup and delivery problem with time windows," European journal of operational research, vol. 54, no. 1, pp. 7–22, 1991.

[4] R. Baldacci, E. Bartolini, and A. Mingozzi, "An exact algorithm for the pickup and delivery problem with time windows," Operations research, vol. 59, no. 2, pp. 414–426, 2011.

[5] R. Masson, S. Ropke, F. Lehu´ed´e, and O. P´eton, "A branch-and-cut-and-price approach for the pickup and delivery problem with shuttle routes," European Journal of Operational Research, vol. 236, no. 3, pp. 849–862, 2014.

[6] W. P. Nanry and J. W. Barnes, "Solving the pickup and delivery problem with time windows using reactive tabu search," Transportation Research Part B: Methodological, vol. 34, no. 2, pp. 107–121, 2000.

[7] H.-F. Wang and Y.-Y. Chen, "A genetic algorithm for the simultaneous delivery and pickup problems with time window," Computers & industrial engineering, vol. 62, no. 1, pp. 84–95, 2012.

[8] S. Ropke and D. Pisinger, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows," Transportation science, vol. 40, no. 4, pp. 455–472, 2006.

[9] T. Le-Anh and M. De Koster, "A review of design and control of automated guided vehicle systems," European Journal of Operational Research, vol. 171, no. 1, pp. 1–23, 2006.

[10] M. De Ryck, M. Versteyhe, and F. Debrouwere, "Automated guided vehicle systems, state-of-the-art control algorithms and techniques," Journal of Manufacturing Systems, vol. 54, pp. 152–173, 2020.

[11] T. Nishi and R. Maeno, "Petri net decomposition approach to optimization of route planning problems for agv systems," IEEE Transactions on Automation Science and Engineering, vol. 7, no. 3, pp. 523–537, 2010.

[12] T. Nishi and Y. Tanaka, "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, vol. 42, no. 5, pp. 1230–1243, 2012.

[13] S. Rajotia, K. Shanker, and J. Batra, "A semi-dynamic time window constrained routeing strategy in an agv system," International Journal of Production Research, vol. 36, no. 1, pp. 35–50, 1998.

[14] N. Smolic-Rocak, S. Bogdan, Z. Kovacic, and T. Petrovic, "Time windows based dynamic routing in multi-agv systems," IEEE Transactions on Automation Science and Engineering, vol. 7, no. 1, pp. 151–155, 2009.

[15] T. J. Chen, Y. Sun, W. Dai, W. Tao, and S. Liu, "On the shortest and conflict-free path planning of multi-agv system based on dijkstra algorithm and the dynamic time-window method," in Advanced Materials Research, vol. 645. Trans Tech Publ, 2013, pp. 267–271.

[16] J. Li, A. Tinka, S. Kiesel, J. W. Durham, T. S. Kumar, and S. Koenig, "Lifelong multi-agent path finding in large-scale warehouses," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 13, 2021, pp. 11 272–11 281.

[17] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction. MIT press, 2018.

[18] S. Yu, J. Zhou, B. Li, S. Mabu, and K. Hirasawa, "Q value-based dynamic programming with sarsa learning for real time route guidance in large scale road networks," in The 2012 International Joint Conference on Neural Networks (IJCNN). IEEE, 2012, pp. 1–7.

[19] T. Xue, P. Zeng, and H. Yu, "A reinforcement learning method for multi-agv scheduling in manufacturing," in 2018 IEEE International Conference on Industrial Technology (ICIT). IEEE, 2018, pp. 1557–1561.

[20] B. Wang, Z. Liu, Q. Li, and A. Prorok, "Mobile robot path planning in dynamic environments through globally guided reinforcement learning," IEEE Robotics and Automation Letters, vol. 5, no. 4, pp. 6932–6939, 2020.

[21] G. Sartoretti, J. Kerr, Y. Shi, G. Wagner, T. S. Kumar, S. Koenig, and H. Choset, "Primal: Pathfinding via reinforcement and imitation multi-agent learning," IEEE Robotics and Automation Letters, vol. 4, no. 3, pp. 2378–2385, 2019.

[22] J. Yu and S. M. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in Twenty-Seventh AAAI Conference on Artificial Intelligence, 2013.

[23] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph neural networks for decentralized multi-robot path planning," in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020, pp. 11 785–11 792.

[24] J. C. Chen, R.-D. Dai, and C.-W. Chen, "A practical fab design procedure for wafer fabrication plants," International journal of production research, vol. 46, no. 10, pp. 2565–2588, 2008.

[25] D.-Y. Liao, M.-D. Jeng, and M. Zhou, "Application of petri nets and lagrangian relaxation to scheduling automatic material-handling vehicles in 300-mm semiconductor manufacturing," IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 37, no. 4, pp. 504–516, 2007.

[26] R. Nakamura, K. Sawada, S. Shin, K. Kumagai, and H. Yoneda, "Model reformulation for conflict-free routing problems using petri net and deterministic finite automaton," Artificial Life and Robotics, vol. 20, no. 3, pp. 262–269, 2015.

[27] J.-W. Yang, H.-C. Cheng, T.-C. Chiang, and L.-C. Fu, "Multiobjective lot scheduling and dynamic oht routing in a 300-mm wafer fab," in 2008 IEEE international conference on systems, man and cybernetics. IEEE,2008, pp. 1608–1613.

[28] H.-W. Huang, C.-H. Lu, and L.-C. Fu, "Lot dispatching and scheduling integrating oht traffic information in the 300mm wafer fab," in 2007 IEEE International Conference on Automation Science and Engineering. IEEE, 2007, pp. 495–500.

[29] S. Lee, J. Lee, and B. Na, "Practical routing algorithm using a congestion monitoring system in semiconductor manufacturing," IEEE Transactions on Semiconductor Manufacturing, vol. 31, no. 4, pp. 475–485, 2018.

[30] K. Bartlett, J. Lee, S. Ahmed, G. Nemhauser, J. Sokol, and B. Na, "Congestion-aware dynamic routing in automated material handling systems," Computers & Industrial Engineering, vol. 70, pp. 176–182, 2014.

[31] J. Boyan and M. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," Advances in neural information processing systems, vol. 6, 1993.

[32] S. Choi and D.-Y. Yeung, "Predictive q-routing: A memory-based reinforcement learning approach to adaptive traffic control," Advances in Neural Information Processing Systems, vol. 8, 1995.

[33] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in Icml, vol. 99, 1999, pp. 278–287.

[34] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," Advances in neural information processing systems, vol. 30, 2017.

[35] P. Veli˘ckovi´c, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," arXiv preprint arXiv:1710.10903, 2017.

[36] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," arXiv preprint arXiv:1511.05952, 2015.

[37] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in Thirty-second AAAI conference on artificial intelligence, 2018