Proceedings of the 2021 Winter Simulation Conference
S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo, and M. Loper, eds.

A TRAFFIC AVOIDANCE PATH PLANNING METHOD FOR AUTONOMOUS VEHICLES IN A WAREHOUSE ENVIRONMENT

Sriparvathi Shaji Bhattathiri
Maojia P. Li

Kate Gleason College of Engineering
Rochester Institute of Technology
81 Lomb Memorial Drive
Rochester, NY 14623, USA

Michael E. Kuhl

Department of Industrial and Systems Engineering
Rochester Institute of Technology
81 Lomb Memorial Drive
Rochester, NY 14623, USA

ABSTRACT

Autonomous mobile robots are widely used today in supply chain, manufacturing, and service systems. Major challenges in these systems are dispatching and path planning. Centralized systems typically assume the knowledge of location, planned path, status of all mobile robots in the system. This paper presents a simulation-based decentralized path planning method that has a traffic prediction and avoidance component. The method is applied to autonomous mobile robots in a warehouse environment. Given only the starting, pick-up, and drop-off locations of the other robots in the system, we utilize a deep-learning algorithm to predict heavy traffic zones with the goal of minimizing travel time. We conduct a Monte Carlo simulation analysis to demonstrate the capabilities of the method.

1 INTRODUCTION

In recent years, the use of autonomous mobile robots (AMRs) has become more common in warehouses for material handling purposes. As the prevalence of AMRs increases, there are new challenges to consider. For example, as the number of mobile robots increases the complexity of the path planning goes up and the computational intensity and costs increase. Further, in a large warehouse, it is nearly impossible to always know, with certainty, the exact location of each mobile robot at all points in time which is a typical requirement for centralized path planning methods. These challenges have pushed researchers to embrace decentralized path planning.

Decentralized path planning involves the robot planning its own path without knowing with certainty the paths to be traveled by other robots, taking into consideration performance goals such as minimizing travel distance, travel time, etc. When the robots encounter each other en route to their destination, local navigation policies are typically used to resolve conflicts much like the traffic rules followed by vehicles on the road. Therefore, just like on the road there can be traffic and congestion which can increase the time to the destination. There are also safety concerns associated with multiple mobile robots choosing to traverse through a particular area in the warehouse at the same time. Although there are policies to deal with encountering other vehicles, there is an increased risk of collisions or other accidents. To address these issues, hybrid approaches have been proposed that incorporate a traffic prediction component that is centralized and a path planning component that is decentralized.

This paper proposes a decentralized system where decentralized robots are assigned tasks by a centralized system. The central system provides the robot information such as pick-up and drop-off locations of the active robots in the system, we utilize a deep-learning algorithm to predict heavy traffic zones and plan a path for a newly dispatched robot that will minimize expected travel time.

978-1-6654-3311-2/21/$31.00 ©2021 IEEE

The remainder of the paper is organized as follows. Section 2 provides an overview of the state of the art techniques used in path planning and traffic prediction. In section 3, the methodology of the path planning system is discussed as well as the implementation. In section 4, we present the results of a Monte Carlo simulation analysis. Finally, in section 5, we discuss the conclusions and future work.

2 Related Work

Contemporary research is prolific in techniques concerning mobile robots both pertaining to single mobile robots, as well as multi mobile robot systems. Where multiple mobile robots are concerned, the problem of efficient communication is a prominent issue. Using decentralized systems where AMRs are completely independent of each other is possible but resource conflicts are a chief problem. A common type of resource conflict is shared space or a common way-point that more than one mobile robot needs to access. To deal with this problem, motion planning or task planning can be used (Yan, Jouandeau, and Cherif 2013). The relatively young field of cooperative multi-agent path planning includes a wide variety of concepts including time-dependent planning, planning in the non-deterministic domain, and automated planning (Torreno, Onaindia, Komenda, and Štolba 2017). Optimized path planning for multi-mobile agent systems is computationally intense even when simplifying the problem and using a planar grid. Further, even after using these simplifying techniques, there arises a need for local adaptation or local planning (Salzman and Stern 2020).

Decentralized path planning has been used with mobile robots, for both guided mobile robots as well as self-driving vehicles. An approach to avoid traffic is to deploy a conflict prediction system with a traditional path planning algorithm, such as A*. Liu et al. (2020) provides an example of how decentralized path planning is used along with traffic prediction at the path segment level. The authors show A* with traffic prediction outperforms naïve A* in terms of avoiding conflicts. However, as the traffic prediction requires central information and computation, the system is unpractical on a completely decentralized AMR system.

Another common practice employed by decentralized systems is to communicate with other mobile units to obtain conflict-free paths. In Rothfuß et al. (2019) A* is used to find the shortest path where the mobile robot units cooperatively negotiate conflict-free paths using an iterative process that results in a deterministic solution. Qingbiao Li et al. (2020) propose using convolutional neural networks to acquire local features, along with graph neural networks to communicate these features to other agents in a decentralized system. In addition, Liu et al. (2020) propose the use of a reinforcement learning algorithm for decentralized path planning in a dynamic environment. These approaches require local communications among robots, which makes them more useful for local navigation and conflict resolving than for path planning.

While most algorithms try to avoid any conflict between two mobile robots, due to unforeseen circumstances, the mobile robots may come in close proximity to each other, and the preferred behavior of mobile robots varies. In some cases, the path planning algorithm takes care of these issues and in others, motion control is used (Krä, Vogt, Spannagl, and Schilp 2020). A common practice is to use traffic rules similar to those used in street traffic. There is a stronger commercial interest in traffic management policies which has lead to the development of neural networks that learn weights based on an entire traffic management team (Chung, Rebhuhn, Yates, Hollinger, and Tumer 2019).

The analysis of the current literature has inspired the idea of a simple traffic prediction model that uses machine learning techniques to learn areas of heavier traffic so they can be avoided. Our objective is to develop a traffic prediction system that only requires the minimum level of global information to work with the path planning algorithm. The goal is to construct a path between the robot's current location and destination that minimizes the probability of encountering other agents given their initial locations and destinations. A* is a benchmark algorithm and a common choice for path planning. In addition, while there are many machine learning algorithms available, neural networks form the backbone of the sudden boom in artificial intelligence and have shown promise in the literature pertaining to decentralized path planning. As such, the methods we develop will build on these concepts.

3 WAREHOUSE SYSTEM DESCRIPTION AND LOGICAL REPRESENTATION

The target application of the decentralized AMR dispatching and path planning system is a typical warehouse. In this section, we describe a representative warehouse that we use to illustrate the methodology and for experimentation. In addition, we discuss the approach for determining AMR travel paths within the warehouse.

3.1 Warehouse Description

The type of warehouse system under consideration is a moderately sized warehouse with multiple autonomous mobile robots used for material handling purposes. In our experiment, we consider a 200,000 sq. ft warehouse with 64 dock locations and 5,769 rack locations. The aisles and highways are bi-directional allowing a maximum of 2 AMRs to pass by each other. There are six AMRs (for example, autonomous fork trucks) that travel throughout the warehouse. These AMRs comprise the decentralized mobile robot system. In each trip, the AMRs have to go from the starting point to a particular pick-up location to retrieve a unit load and then to a drop-off location. The goal is to find the fastest path considering potential delays due to traffic.

To represent the warehouse, the warehouse layout is converted into a grid map with 100x100 cells. Figure 1 shows a depiction of the warehouse layout with dock/staging locations on the left side and rows of rack storage locations. An AMR occupies one cell at a time and can move to one of the neighboring cells to the north, south, east, or west during each time step. We define the intercept among aisles and highways as an intersection. A segment of an aisle or highway bounded by two intersections is referred to as a zone. There are 1,044 zones in the system. When more than one AMR moves in the same zone, all AMRs in the zone slow down to avoid accidents or to resolve any conflicts. Assuming there are m AMRs in a zone, the number of time steps to move from one cell to the neighbor becomes m.

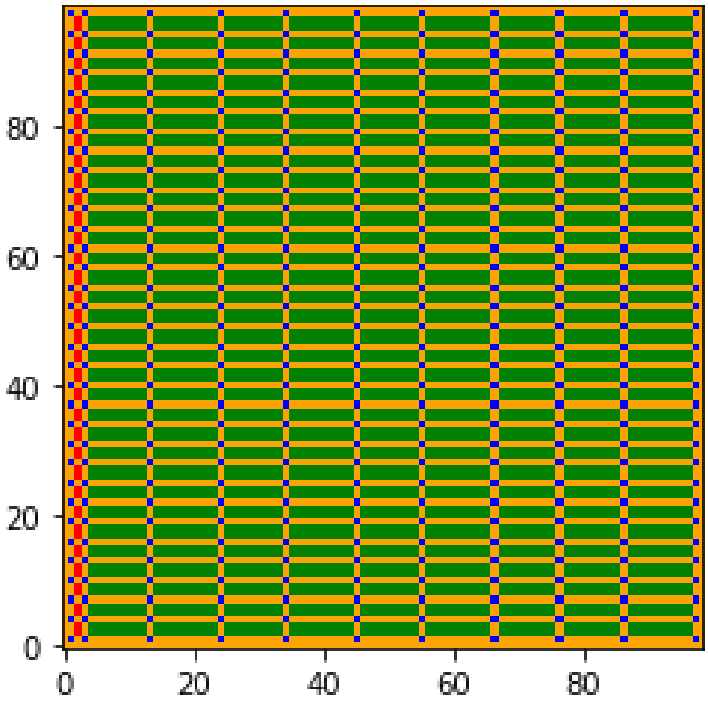


Figure 1: A graphical representation of the warehouse grid map. Intersections are blue, other traversable cells are yellow, dock/staging locations are red, and storage racks are green.

In the simulation environment, the warehouse is represented as a matrix, where 0 represents the free (traversable) path cells, -1 represents the outside walls, 100 represents the racks, 200 represents the docks, and 50 represents the intersections. Figure 2 shows a section of the warehouse represented as a matrix. For path planning, the matrix is converted to an occupancy grid where obstacles are represented by 1s and the free paths are represented by 0s. Here the dock and the rack locations are considered obstacles, and cannot be crossed. All non-zero values in the matrix in Figure 2 are translated to a value of 1 in the occupancy grid.

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	50	0	50	0	0	0	0	0	0	0	0	0
-1	0	200	0	100	100	100	100	100	100	100	100	100
-1	0	200	0	100	100	100	100	100	100	100	100	100
-1	50	0	50	0	0	0	0	0	0	0	0	0
-1	0	200	0	100	100	100	100	100	100	100	100	100
-1	0	200	0	100	100	100	100	100	100	100	100	100
-1	50	0	50	0	0	0	0	0	0	0	0	0
-1	0	200	0	100	100	100	100	100	100	100	100	100
-1	0	200	0	100	100	100	100	100	100	100	100	100
-1	50	0	50	0	0	0	0	0	0	0	0	0
-1	0	200	0	100	100	100	100	100	100	100	100	100
-1	0	200	0	100	100	100	100	100	100	100	100	100
-1	50	0	50	0	0	0	0	0	0	0	0	0
-1	0	200	0	100	100	100	100	100	100	100	100	100
-1	0	200	0	100	100	100	100	100	100	100	100	100

Figure 2: Matrix representing a warehouse using 0 for free paths, 50 for intersections, 100 for racks, and 200 for dock locations. This matrix represents the upper left corner of the warehouse in Figure 1.

3.2 AMR Path Representation

To represent the routes AMRs take to travel through the warehouse, the local path planning algorithm implemented on the grid map is A*. A* is a graph traversal algorithm commonly used in mobile robot path planning. The set of cells in the system is denoted by S. A* evaluates the cost of using each traversable cell $s \in S$ to construct a path. To minimize the path length, the cells are evaluated using the F-cost. Given the initial location, s_0 and destination $s*$, the F-cost of any intermediate cell s is determined by,

$$f(s) = g(s_0, s) + h(s, s^*)$$

where $g(s_0, s)$ is the lowest cost found so far to move from the initial location to s; $h(s, s^*)$ is the heuristic, which estimates the lowest cost to move from s to destination. The two most common heuristics are Manhattan distance and Heuristic distance. Provided the heuristic does not overestimate the actual distance, A* will find the shortest path. Figure 3 shows an example of the visited cells and path computed by A* for an AMR in the warehouse.

The traffic avoidance system introduced in our work requires a low level of global information. When an AMR finishes a delivery task, it sends a dispatching request and its current location to the task management system. The task management system sends back a new assignment as well as other active agents' starting, pick-up, and drop-off locations. Figure 4a shows an example of the predicted paths for five active AMRs. The blue paths represent the cells that would be traversed by one or more active AMRs. Figure 4b shows the the cells that would be traversed by two or more active AMRs. Likewise, similar plots could be constructed for cells traversed by 3 or more active AMRs, and so on. Based on this information, the dispatched agent first predicts the high-traffic zones with a deep learning model and then computes a path with A* considering high-traffic zones to be non-traversable. The approach is to find the path that will result in the minimum travel time.

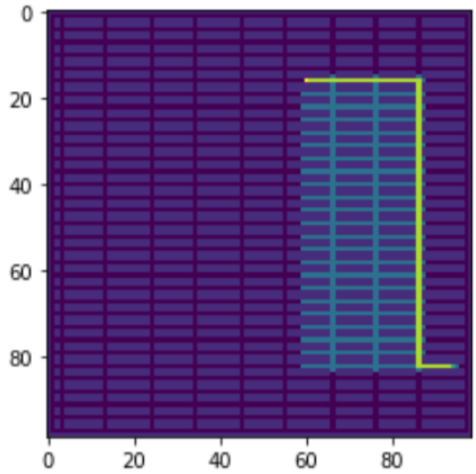
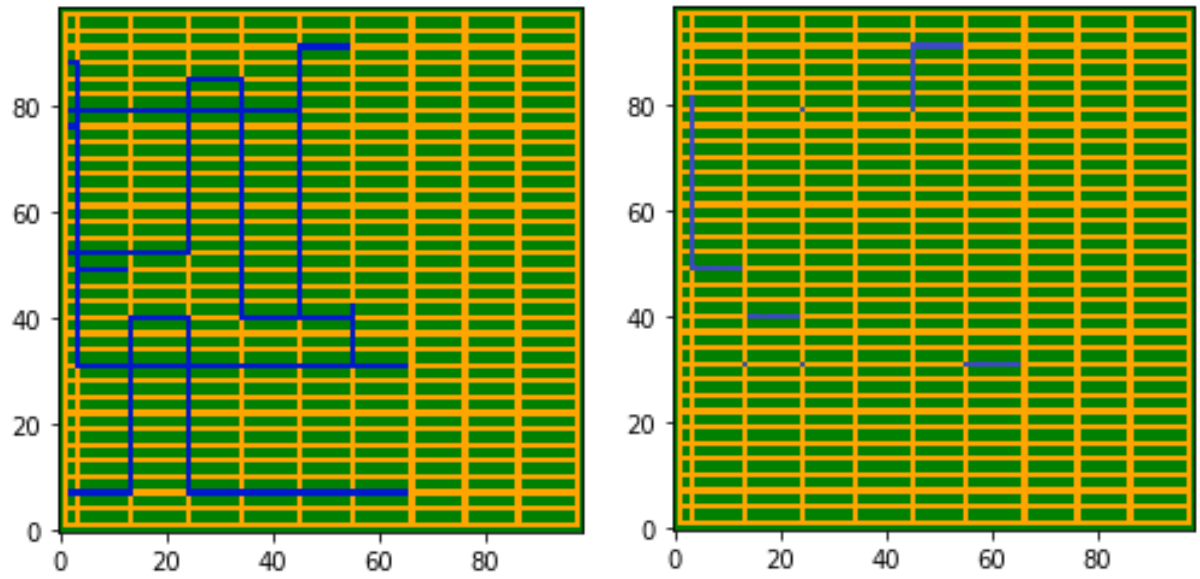


Figure 3: Example of AMR path planning. The dark blue spaces represent the free path, light blue represents the obstacles, sky blue represents the explored paths, and yellow is the shortest path obtained using A*.



(a) (b)

Figure 4: Example routes generated using A* (in blue) where (a) shows all cells traversed by one or more AMRs; and (b) shows all cells traversed by two or more AMRs.

4 METHODOLOGY

To be able to predict the traffic, a neural network model is designed which takes the initial location of the AMR, unit load pick-up location, and unit load drop-off destination as the inputs and gives the zones that are likely to be occupied as the output. Rather than predicting one zone at a time, we introduce a novel neural network architecture shown in Figure 5, in which we predict the occupancy of all zones via a single computation. Such a problem is often referred to as a multi-label classification problem. Given the number of zones N, the input of neural network $x \in \mathbb{R}^{2N}$ is a one-shot encoding matrix with initial location and destination information. Each neuron in the output layer corresponds to a zone, where the value of a

neuron represents the predicted probability of occupancy. To train such a model the data is synthesized by randomly picking start points, racks, and docking stations, and then using A* to predict the paths and assigning the zones values 1 and 0 depending on whether the zone is traversed or not.

Various machine learning algorithms including support vector machines, k- and random forest (Kumari and Srivastava 2017) were explored, but they showed zero training accuracy. Then neural networks were tested. The deeper neural networks did not converge, so a shallow network was employed. The neural network has 4-layeres and the activation function of the final layer is sigmoid function which is commonly used for multi-label classification problems. The number of neurons in each layer have been varied along with the hyperparameters including the number of epochs, and batch size to obtain optimal performance. These hyper-parameters impact how the network is trained, and need to be determined before the optimization of the weights and biases that occur during training. The neural network architecture that gives the best training and testing accuracy has the architecture shown in Figure 5.

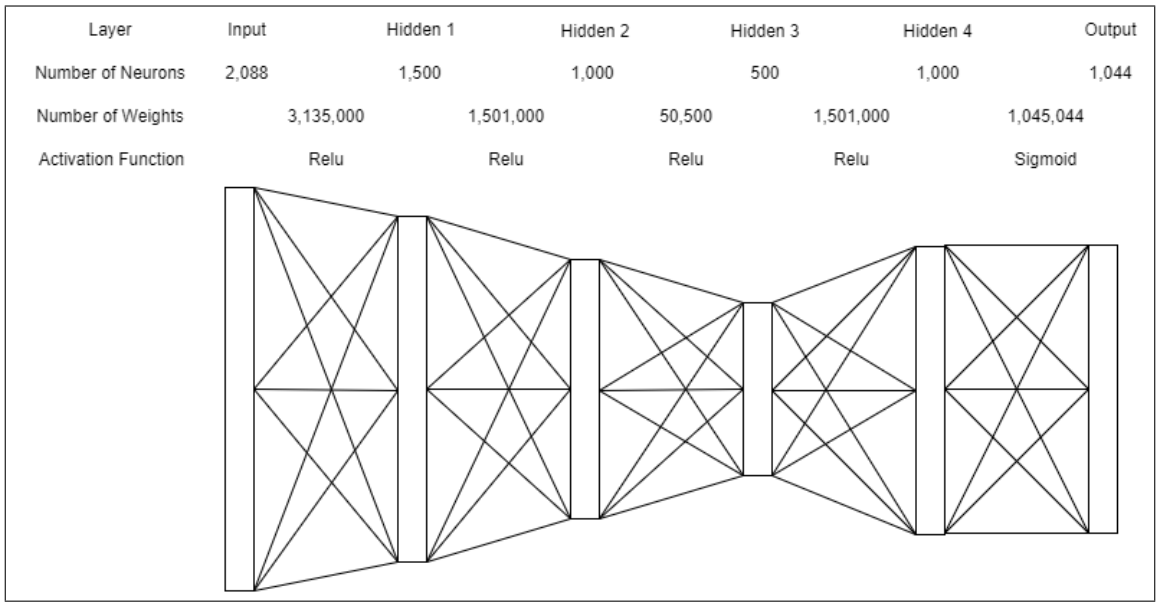


Figure 5: Neural network architecture.

The goal of the neural network is to minimize the Hinge loss,

$$H_i = [0, 1 - y_i z(x_i)], \tag{1}$$

for each zone i, where $z(x_i)$ denotes the output of neural network and $y_i \in \{0,1\}$ represents ground truth such that $y_i = 1$ if the zone is occupied and 0 otherwise. When moving from one location to another, an AMR uses a limited number of zones. This leads to unbalanced training data, in which the majority of the zones are not occupied. To overcome this issue, we use a weighted loss function expressed by

$$l_i = \frac{N}{N_i} H_i, \tag{2}$$

where N_i is the number of zones with the sample label as zone i. Another technique implemented in our neural network to address the unbalanced data is initializing the bias by,

$$b = log(N_1/N_0), \tag{3}$$

where N_1 represents the number of zones occupied while N_0 is the number of zones not occupied.

The performance of occupancy classification is measured by accuracy, precision, and recall, as follows:

$$Accuracy = (TP+TN)/(TP+TN+FP+FN) \tag{4}$$

$$Precision = TP/(TP+FP) \tag{5}$$

$$Recall = TP/(TP+FN) \tag{6}$$

where a true positive (TP) prediction indicates the neural network correctly predicts a zone will be occupied. A true negative (TN) indicates the neural network correctly predicts a zone will not be occupied. A false positive (FP) means the neural network makes a wrong prediction about a zone that will be occupied. Finally, a false negative (FN) means the neural network makes a wrong prediction about a zone that will not be occupied. In addition, another common performance measure, Matthew's Correlation Coefficient (MCC), for multi-label classification is utilized which is calculated by,

$$MCC = (TP*TN) - (FP*FN)/\sqrt{TP+FP)*(TP+FN)*(TN+FN)}. \tag{7}$$

5 EXPERIMENTATION AND RESULTS

In this section, we present experimentation for training and testing the neural network followed by a Monte Carlo simulation experiment to investigate the capabilities of the traffic avoidance method to find a path that will minimize travel time.

5.1 Neural Network Training Experiment

The neural network in Figure 5 is trained to predict the traffic level of each zone in the warehouse in Figure 1. In each training sample, the starting, pickup, and dropoff locations of each active agent are randomly generated. We use A* to compute the path of each agent and determine the total number of agents occupying each zone in the system. Given the starting, pickup, and dropoff locations of all agents, the neural network is trained to predict the occupancy of each zone by minimizing the Hinge loss in (1).

The neural network is trained for 100 epochs with a batch size of 32. Each epoch consists of 15,000 training samples. An early stop technique is deployed to stop the training when the loss is not improved in the last 5 epochs. Various metrics are used to determine if neural network should be updated from the last epoch. For instance, if the metrics are accuracy, precision, and recall, the neural network is only updated if the last epoch of training improves all 3 metrics.

The preliminary experiments show traditional neural network have a low precision and recall in this problem because of the unbalanced data. More than 95% of training samples are negative as those zones will not be occupied by any active agent. The neural network tends to make negative predictions unless it is very confident the zone will be occupied. Therefore, two machine learning techniques to mitigate this issue, output bias initialization and sample weights, are evaluated.

For output bias, as the majority of training samples show the zones are not occupied, the bias term in the neural network are initialized such that it is more likely to predict the zones will be occupied. It is a common practice to have output bias equal to the ratio between the minority class and the total observations. The performance of neural network on the training data and testing data with output bias are shown in Tables 1 and 2. The output bias show some improvement in precision on the training data. However, the low precision and recall but high accuracy on the testing data indicate the neural network still tends to make negative predictions.

The second method employed to improve performance is to give higher weights to those samples of the minority class. In this problem, we give a higher penalty if the model fails to detect a zone being

occupied compare to the scenario where the model predicts an empty zone to be occupied. Tables 3 and 4 show the performance of using sample weights on the training and testing data. The sample weights significantly improve the recall on both training and testing data. For the testing metrics of accuracy the values are balanced with an accuracy of 0.60 and recall of 0.75.

Note that, the technique of using output bias, weighted loss etc. are some of the solutions to deal with unbalanced classes and improved the performance of the model but, the problem of unbalanced classes is still an open problem with no one perfect solution. It is an active area of research that must be explored further (Showrov, Dubey, Hasib, and Shameem 2021).

Table 1: Neural network training performance with output bias.

Metrics	**Accuracy**	**Precision**	**Recall**
Accuracy	0.97	0.48	0.00
Precision, Recall	0.97	0.39	0.00
Recall	0.97	0.00	0.00
Accuracy, Hamming Loss	0.97	0.52	0.00
Accuracy, Precision, Recall	0.97	0.21	0.34

Table 2: Neural network testing performance with output bias.

Metrics	**Accuracy**	**Precision**	**Recall**
Accuracy	0.97	0.00	0.00
Precision, Recall	0.97	0.02	0.00
Recall	0.97	0.00	0.00
Accuracy, Hamming Loss	0.04	0.52	0.00
Accuracy, Precision, Recall	0.02	0.21	0.00

Table 3: Neural network training performance with sample weights.

Metrics	**Accuracy**	**Precision**	**Recall**
Precision, Recall	0.49	0.40	0.84
Precision	0.97	0.00	0.00
Accuracy, Hamming Loss	0.66	0.05	0.10
Accuracy, Matthew's Correlation Coefficient	0.97	0.00	0.00

Table 4: Neural network testing performance with sample weights.

Metrics	**Accuracy**	**Precision**	**Recall**
Accuracy	0.60	0.05	0.75
Precision, Recall	0.45	0.00	0.84
Precision	1.00	0.00	0.00
Accuracy, Hamming Loss	0.66	0.05	0.62
Accuracy, Matthew's Correlation Coefficient	0.97	0.00	0.00

5.2 Simulation Experiment

The Traffic Avoidance A* method is compared with traditional A* in the simulated warehouse in Figure 1 with five active agents. The Traffic Avoidance A* starts by predicting the number of agents occupying each zone using the neural network. Give the predicted traffic level, traffic avoidance A* exhaustively computes paths that consider a zone occupied by more than A agents as an obstacle, where A is a constant from 1 to 6. When $A = 6$, the traffic avoidance A* generates the same path as traditional A*, as there are only 5 active agents in the system. Then, traffic avoidance A* estimates the total travel time of each paths considering both traffic distance and potential delay caused by traffic. Finally, traffic avoidance A* pick the path with the smallest travel time.

Table 5 compares the performance of A* and traffic avoidance A* in the simulated warehouse across 100 trials. When traffic avoidance is introduced, the number of time the dispatched agent uses the same zone as an active agent when they are both executing current tasks is reduced. By reducing the number of conflicts, traffic avoidance A* reduces the average travel time from 229.97 to 18053, even though the traditional A* has a smaller average travel distance. However, of the 100 trials, 39 instances resulted in the alternative traffic avoidance path being taken that was different than the A* path. In these 39 cases, the average time was reduced by 126.76 with a standard deviation of 70.35. So, although the traffic avoidance approach resulted in a shorter time path in less than half of the trials, in the cases where an alternative path was found, the time savings were significant.

Table 5: Comparison between traditional A* and A* with traffic avoidance in terms of the average (std. deviation) travel time and travel distance.

Algorithm	**Travel Time**	**Travel Distance**
A*	229.97 (77.60)	160.25 (46.96)
Traffic Avoidance + A*	180.53 (59.81)	214.34 (104.26)

6 CONCLUSION AND FUTURE WORK

This paper presents the preliminary results obtained during the simulation of the warehouse and traffic prediction. The classification problem is a multi-label unbalanced classes problem and hence requires further investigation to improve the precision, recall, and accuracy to get better predictions. Future work would include dividing the warehouse into sectors and having different traffic prediction models for each sector, an attempt to create a less biased data set and using graphical inputs to the machine learning model to better learn the spatial relationships between zones. The solution to this traffic prediction problem can have many advantages by providing the mobile robots with alternate paths. In the experiments, all agents are assumed to use traditional A* except the active agent. In the future work, experiments will be conducted where all agents will use the traffic prediction algorithms in conjunction with A*.

REFERENCES

Chung, J. J., C. Rebhuhn, C. Yates, G. A. Hollinger, and K. Tumer. 2019. "A Multiagent Framework for Learning Dynamic Traffic Management Strategies". *Autonomous Robots* 43(6):1375–1391.

Krä, M., L. Vogt, V. Spannagl, and J. Schilp. 2020. "Multi-Agent Path Planning: Comparison of Different Behaviors in the Case of Collisions". In *Annals of Scientific Society for Assembly, Handling and Industrial Robotics*, 217–227. Springer Vieweg, Berlin, Heidelberg.

Kumari, R., and S. K. Srivastava. 2017. "Machine Learning: A Review on Binary Classification". *International Journal of Computer Applications* 160(7):11–15.

Li, Q., F. Gama, A. Ribeiro, and A. Prorok. 2020. "Graph Neural Networks for Decentralized Path Planning". In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*, 1901–1903. Richland, South Carolina: International Foundation for Autonomous Agents and Multiagent Systems.

Liu, Z., B. Chen, H. Zhou, G. Koushik, M. Hebert, and D. Zhao. 2020. "MAPPER: Multi-Agent Path Planning with Evolutionary Reinforcement Learning in Mixed Dynamic Environments". *arXiv preprint arXiv:2007.15724*.

Liu, Z., H. Wang, H. Wei, M. Liu, and Y.-H. Liu. 2020. "Prediction, Planning, and Coordination of Thousand-Warehousing-Robot Networks with Motion and Communication Uncertainties". *IEEE Transactions on Automation Science and Engineering*:1–13.

Rothfuß, S., R. Prezdnyakov, M. Flad, and S. Hohmann. 2019. "Decentralized Path Planning for Cooperating Autonomous Mobile Units". *Forschung im Ingenieurwesen* 83(2):137–147.

Salzman, O., and R. Stern. 2020. "Research Challenges and Opportunities in Multi-Agent Path Finding and Multi-Agent Pickup and Delivery Problems". In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*, 1711–1715. Richland, South Carolina: International Foundation for Autonomous Agents and Multiagent Systems.

Showrov, M. I. H., V. K. Dubey, K. M. Hasib, and M. A. Shameem. 2021. "News Classification from Microblogging Dataset using Supervised Learning". In *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, 53–57. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Torreno, A., E. Onaindia, A. Komenda, and M. Štolba. 2017. "Cooperative Multi-Agent Planning: A Survey". *ACM Computing Surveys (CSUR)* 50(6):1–32.

Yan, Z., N. Jouandeau, and A. A. Cherif. 2013. "A Survey and Analysis of Multi-Robot Coordination". *International Journal of Advanced Robotic Systems* 10(12):1–18.

AUTHOR BIOGRAPHIES

SRIPARVATHI SHAJI BHATTATHIRI is a Ph.D. student in Engineering at Rochester Institute of Technology. Her research interests include simulation modeling and analysis, machine learning, and deep learning. Her email address is ssb6096@rit.edu.

MAOJIA P. LI is a Ph.D. student in Engineering at Rochester Institute of Technology. His research interests include simulation modeling and analysis, machine learning, and deep learning. His email address is mxl8487@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 Directors representing the INFORMS Simulation Society. He has also served WSC as Proceedings Editor (2005), Program Chair (2013), and Mobile App Chair (2014-2019, 2021). His email address is Michael.Kuhl@rit.edu.