CAMETA: Conflict-Aware Multi-Agent Estimated Time of Arrival Prediction for Mobile Robots

Jonas le Fevre Sejersen [ID] and Erdal Kayacan [ID]

Abstract— **This study presents the conflict-aware multi-agent estimated time of arrival (CAMETA) framework, a novel approach for predicting the arrival times of multiple agents in unstructured environments without predefined road infrastructure. The CAMETA framework consists of three components: a path planning layer generating potential path suggestions, a multi-agent ETA prediction layer predicting the arrival times for all agents based on the paths, and lastly, a path selection layer that calculates the accumulated cost and selects the best path. The novelty of the CAMETA framework lies in the heterogeneous map representation and the heterogeneous graph neural network architecture. As a result of the proposed novel structure, CAMETA improves the generalization capability compared to the state-of-the-art methods that rely on structured road infrastructure and historical data. The simulation results demonstrate the efficiency and efficacy of the multi-agent ETA prediction layer, with a mean average percentage error improvement of 29.5% and 44% when compared to a traditional path planning method (A^*) which does not consider conflicts. The performance of the CAMETA framework shows significant improvements in terms of robustness to noise and conflicts as well as determining proficient routes compared to state-of-the-art multi-agent path planners.**

I. INTRODUCTION

Multi-agent path finding (MAPF) is the problem of generating valid paths for multiple agents while avoiding conflicts. This problem is highly relevant in many real-world applications, such as logistics, transportation, and robotics, where multiple agents must operate in a shared environment. MAPF is a challenging problem due to the need to find paths that avoid conflicts while minimizing the overall travel time for all agents. Many state-of-the-art MAPF solvers [1, 2, 3] employ various techniques to find a set of conflict-free paths on graphs representing the environment and the agents. However, a common limitation of these solvers is that they tend to generate tightly planned and coordinated paths. Therefore, the agents are expected to follow the exact path prescribed by the solver, which can lead to problems when applied to real-world systems with imperfect plan execution and uncertainties in the environment.

This work introduces a conflict-aware multi-agent estimated time of arrival (CAMETA) for indoor autonomous mobile robot (AMR) applications that operate in time-constrained scenarios. The proposed framework is a three-layered framework that is deployed on each agent. The

J. Fevre is with the Artificial Intelligence in Robotics Laboratory (AiR Lab), Department of Electrical and Computer Engineering, Aarhus University, 8000 Aarhus C, Denmark {jonas.le.fevre} at ece.au.dk E. Kayacan is with the Automatic Control Group, Department of Electrical Engineering and Information Technology, Paderborn University, Paderborn, Germany. {erdal.kayacan} at uni-paderborn.de
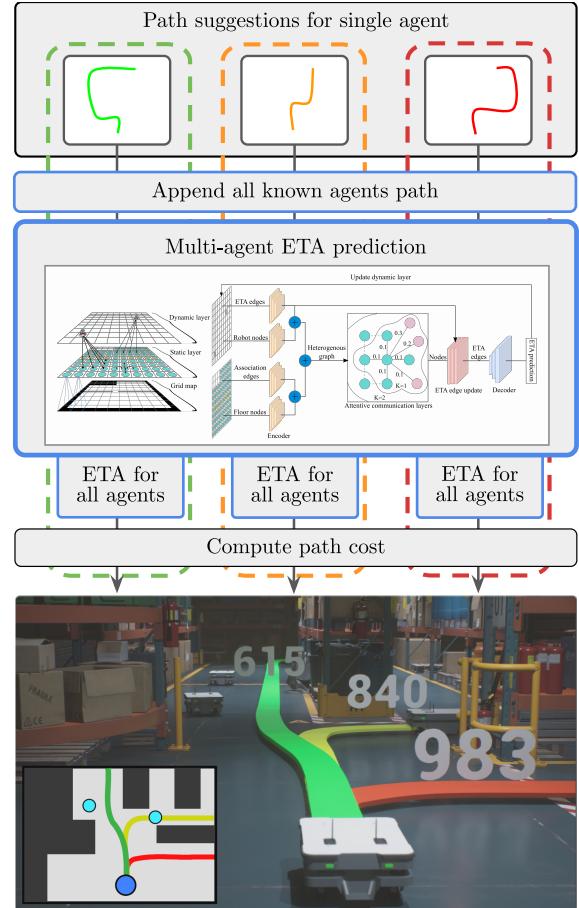
Fig. 1: The conflict-aware multi-agent planner framework is depicted in three layers. The first layer generates multiple path suggestions for a single agent. The second layer, enclosed in blue borders, produces an estimated time of arrival prediction for all agents. Finally, the third layer computes the overall cost for each suggested path, using the information generated in the previous layer.

layers consist of a path planning layer, which generates route suggestions for the deployed agent, a multi-agent estimated time of arrival (ETA) prediction layer, which forecasts the ETA of all agents in the system given one of the suggested paths, and a path selection layer that minimizes the overall travel time by reducing the total number of conflicts in the system. In our problem definition, some agents are required to arrive at their destination quicker than others, which is a common scenario in logistics applications for airports and

warehouses. The proposed framework is illustrated in Fig. 1.

The main focus of this work is on the development of the multi-agent ETA prediction layer and its effectiveness in forecasting the ETA for each agent. The performance and accuracy of the proposed graph neural network (GNN) model and the multi-agent ETA prediction layer are evaluated by comparing them to a naive method. In this context, the term *"naive"* refers to path planning that does not consider the imperfect execution of plans and conflicts that may arise along the intended path. This comparison provides valuable insights into the performance and accuracy improvements achieved by utilizing the GNN model for ETA prediction, which explicitly considers the complexities associated with imperfect execution and conflicts. The comparison provides an understanding of the added value of the proposed model in accurately forecasting ETA and addressing real-world challenges in multi-agent systems. Furthermore, the proposed model exhibits flexibility and generalizability by effectively adapting to different robot densities. Given the potential variations in the number of robots due to seasonal fluctuations, demand changes, or business requirements, the model's ability to seamlessly accommodate shifts in robot densities without requiring retraining is of significant value. This characteristic empowers industries to easily add or remove robots as needed, thereby enhancing operational efficiency and adaptability in dynamic environments. To assess the generalizability and overall performance of the proposed framework, an experiment is conducted. This experiment aims to test how different trained prediction models scale during inference time, thereby demonstrating the framework's ability to handle varying robot densities.

In addition, an experiment is conducted to compare the overall performance of the proposed framework, including its ability to handle imperfect plan execution, with state-of-the-art MAPF planners [1, 2]. This comparative analysis serves to provide valuable insights into the effectiveness and competitiveness of the proposed framework in addressing MAPF challenges, particularly in environments where plan execution may be imperfect. Furthermore, to ensure a comprehensive evaluation, this experiment incorporates the presence of noise, which further tests the robustness and adaptability of the proposed framework under realistic conditions. The inclusion of noise allows for a more realistic assessment of the framework's performance in the presence of uncertainties and deviations from ideal plan execution.

The contributions of this study are the followings:
- A conflict-aware global planner is designed to optimize overall system flow while considering time constraints for all robots in industrial scenarios.
- A heterogeneous graph representation is proposed to model the interaction between agents and potential bottleneck areas occurring within the map.
- Finally, A novel GNN architecture is proposed for multi-robot ETA prediction in heterogeneous graphs supporting scaling to different robot densities during inference time.

The rest of this paper is organized as follows. Section II presents the problem formulation. Section III reviews the recent developments in MAPF and ETA prediction. Section IV provides the details of the proposed framework. Section V presents the experiment setup followed by the results in various grid world environments in Section VI. Finally, some conclusions are drawn from this study in Section VII.

II. Problem Formulation

This section presents the formal problem formulation addressed in this paper. The problem under consideration shares similarities with the MAPF problem [4], which involves determining conflict-free paths for multiple agents on a graph to reach their respective destinations. In the MAPF problem, time is divided into discrete timesteps, during which agents execute atomic actions synchronously, such as moving to adjacent nodes or remaining in their current locations. However, the problem addressed in this paper deviates from the traditional MAPF problem in two key aspects. Firstly, the objective is not solely focused on finding conflict-free paths for all agents but on minimizing overall travel time by reducing total conflicts and avoiding congested areas. Secondly, a time constraint is introduced for each agent, requiring them to reach their goals before a specific deadline. This introduces the concept of priority, where higher-priority agents are assigned shorter paths. The types of conflicts are explained in [4] and describes what kind of movement patterns are not allowed to be performed and are therefore considered conflicts. The ones considered in this paper are the following four conflicts: vertex conflicts, edge conflicts, swapping conflicts, and cycle conflicts. Vertex conflicts occur when agents occupy the same position at the same time. Edge conflicts occur when agents travel along or across the same edge. Swapping conflicts occur when two agents exchange positions and cycle conflicts occur when multiple agents form a cyclic movement pattern. The notion of conflicts used in MAPF differs from the conflicts used in the proposed work. A conflict will be defined as when an agent must alter its global path to avoid any of the four aforementioned movement patterns defined by MAPF. The environment is represented as a discrete occupancy grid map, which can be viewed as a graph according to the definition in MAPF. However, accurately modeling noise in such a setting can be challenging, as a full action would need to be performed at each time step, resulting in noise having a significant impact within a single time step. To address this challenge and simplify the incorporation of noise, the movement of the agents is modeled to operate at full velocity, and as such, the inclusion of noise in the simulation would not result in increased speed. On the contrary, the presence of noise would slow down the movement of the agents, causing them to force a wait action. Furthermore, it is assumed that each agent has the capability of peer-to-peer communication to resolve local conflicts using a local planner. Additionally, agents have a global connection to a state database containing each agent's committed plans and their current locations.

The formal problem formulation presented in this section sets the stage for developing effective algorithms and strate-

gies to address the specific challenges of minimizing travel time, incorporating time constraints and priority, handling different types of conflicts, considering noise and imperfect plan execution.

III. RELATED WORK

A. Traditional MAPF solutions

Global path planning algorithms for the MAPF problem are a class of solvers that first perform all computations in a single continuous interval and return a plan for the agents to follow. These plans are generated before the agents begin to move, and the agents follow the plan without any additional computation. This means that the plan cannot contain parts where agents collide. Some global solvers, such as conflict-based search (CBS) [1], aim to find optimal solutions according to a predefined cost function. However, these methods may not be able to scale up to larger systems due to the exponential growth of the search space as the number of agents increases [5]. Other global algorithms, the hierarchical cooperative A* (HCA*) [3] and priority inheritance with backtracking (PIBT) [2], sacrifice optimality or completeness in order to reduce computation time by using a spatiotemporal reservation table to coordinate agents and avoid collisions. A major weakness in global path planning algorithms is that agents frequently have imperfect plan execution capabilities and cannot perfectly synchronize their motions, leading to frequent and time-consuming replanning [6]. This is addressed in [7], where a post-processing framework is proposed, using a simple temporal graph to establish a plan-execution scheduler that guarantees safe spacing between robots. The framework exploits the slack in time constraints to absorb some of the imperfect plan executions and prevent time-intensive replanning. The work of [8] extends the method presented in [9] for a multi-agent path planner called uncertainty M* that considers collision likelihood to construct the belief space for each agent. However, this does not guarantee to remove conflicts caused by imperfect plan execution, so a local path planning algorithm is commonly needed for solving these conflicts when they occur.

Local path planning algorithms are a class of solvers that compute partial solutions in real-time, allowing agents to adjust their plans as they execute them. A simple approach for local MAPF is local-repair A* (LRA*) [10], which plans paths for each agent while ignoring other agents. Once the agents start moving, conflicts are resolved locally by constructing detours or repairs for some agents. Another notable local MAPF solver is the windowed hierarchical cooperative A* (WHCA*) algorithm [3], which is a local variant of the HCA* algorithm. WHCA* uses a spatiotemporal reservation table to coordinate agents, but only reserves limited paths, splitting the problem into smaller sections. As agents follow the partially-reserved paths, a new cycle begins from their current locations. In the traditional WHCA*, a different ordering of agents is used in each cycle to allow a balanced distribution of the reservation table. An extension of the WHCA* is proposed in [11], where a priority is computed based on minimizing future conflicts, improving the success rate, and lowering the computation time. Learning-based methods for local planning are also showing promising results. In [12], an end-to-end local planning method is introduced, using reinforcement learning to generate safe pathing in dense environments. [13] introduces the use of GNN for local communication and imitation learning for learning the conflict resolution of CBS in multi-agent environments.

B. ETA prediction and spatio-temporal sequence forecasting

In the field of spatio-temporal sequence forecasting (STSF), [14, 15] investigated two major learning strategies: iterated multi-step (IMS) estimation and direct multi-step (DMS) estimation. The IMS strategy trains a one-step-ahead forecasting model and iteratively uses its generated samples to produce multi-step-ahead forecasts. This strategy offers simplicity in training and flexibility in generating predictions of any length. [16] demonstrated improved forecasting accuracy by incorporating graph structure into the IMS approach.

However, the IMS strategy suffers from the issue of accumulated forecasting errors between the training and testing phases [17]. To address this disparity, [15] introduced DMS estimation directly minimizes the long-term prediction error by training distinct models for each forecasting horizon. This approach avoids error accumulation and can support multiple internal models for different horizons. Additionally, recursive application of the one-step-ahead forecaster is employed to construct multi-step-ahead forecasts, decoupling model size from the number of forecasting steps.

Although DMS offers advantages over IMS, it comes with increased computational complexity [15]. Multiple models need to be stored and trained in multi-model DMS, while recursive DMS requires applying the one-step-ahead forecaster for multiple steps. These factors result in greater memory storage requirements and longer training times compared to the IMS method. On the other hand, [18] shows that the IMS training process lends itself to parallelization as each forecasting horizon can be trained independently.

Several related works have leveraged the DMS approach for spatio-temporal forecasting. For instance, [19] proposed a homogeneous spatio-temporal GNN method for predicting ETA by combining recursive DMS and multi-model DMS. In [20], a congestion-sensitive graph structure was introduced to model traffic congestion propagation, along with a route-aware graph transformer layer to capture interactions between spatially distant but correlated road segments. Furthermore, [21] proposed a novel heterogeneous graph structure that incorporates road features, historical data, and temporal information at different scales, utilizing temporal and graph convolutions for learning spatio-temporal representations.

However, the existing methods mentioned above primarily focus on road features and consider a single vehicle traversing the graph, neglecting other types of vehicles and the influence of driver route choices on traffic conditions. Consequently, these models cannot readily be extended to multi-vehicle scenarios.
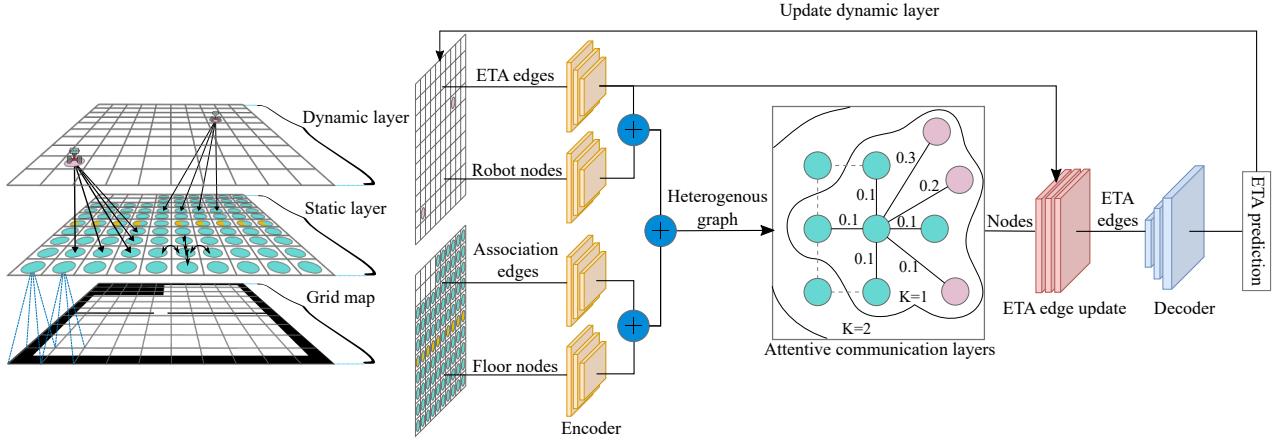
Fig. 2: Illustration of the proposed multi-agent ETA prediction layer. A simplified illustration of the three layers representing the indoor environment and the connection between the layers is shown on the left. The three layers consist of: a 2D occupancy grid map of the environment, a static layer containing all nodes and edges with static information, and a dynamic layer containing the time-variant features of the graph. The proposed GNN architecture for multi-agent ETA prediction is also shown.

IV. METHODOLOGY

A. Path planning

The path planning layer is similar in design to the LRA* [10], where other agents in the system are not considered, and resolving conflicts is left to the local planner. This allows for a decentralized global planning module, where each agent only needs to consider possible routes to its destination. While drastically improving scalability, it comes with the limitation of not being able to change the routes of other agents. Essentially the consideration of other agents is first introduced in the multi-agent ETA prediction layer. The global planner chosen for this framework is A^* [22], as it does not account for conflicts and can quickly compute different route suggestions. The proposed framework can use any off-the-shelf path-planning methods since conflicts are accounted for during the multi-agent ETA prediction layer. It is important to note that the GNN is trained based on the local and global path planning algorithm and, therefore, is still dependent on these. The GNN model learns to model the interactions between these two planners, including how much the global path-planning algorithm considers conflicts in its route suggestions and how effectively the local path-planning algorithm resolves conflicts.

It is important to note that the local planner is not directly part of the proposed pipeline but is used during execution to solve unexpected conflicts. The local planner used for training is the WHCA* [3] planner with a priority-based system. The robot with the highest priority maintains its path while others adapt their paths. Each robot declares its next five moves in a local area, and local replanning is done based on priority. Agents require local peer-to-peer communication and priority-based bidding to align the reservation of space.

B. Multi-agent estimated time of arrival prediction

The multi-agent ETA prediction layer is the second layer of the system. The objective of the layer is to anticipate the impact of the proposed path on the other agents within the system and to estimate the additional time added by these effects. It takes as input the suggested routes from the path planning layer, as well as the planned paths of all other agents, and a 2D occupancy grid map of the environment in order to predict the ETA for each agent. The multi-agent ETA prediction layer is composed of two main steps: It reconstructs the data into a spatio-temporal heterogeneous graph representation and then passes the graph through a recurrent spatio-temporal GNN model in order to make the ETA predictions.

1) Graph representation: To predict the ETA of multiple agents in an indoor environment, a spatio-temporal heterogeneous graph representation is used to capture the environment's structure and the interactions between routes. The graph is divided into two layers: a static layer and a dynamic layer. The interaction between the layers is depicted in Fig. 2.

The static layer of the graph only needs to be constructed once, as its features and connections remain unchanged. The static layer contains nodes of type *floor* with static features of the environment, such as walls, restricted areas, and open spaces. The edges in the static layer are of type *association* and represent the spatial links between nodes. The spatial characteristics of the environment are obtained from a 2D occupancy grid map, which is divided into $N \times N$ tiles. A node of type *floor* is constructed for each tile created from the given map, and the node features are the spatial information within the tile. In order to lower the number of floor nodes, all nodes containing only occupied space are removed. In the case of a solid line separating a tile, multiple *floor* nodes are

stacked to represent each side of the patch, as traffic on one side does not directly impact the other. These stacked nodes are depicted as multi-colored nodes in Fig. 2. Association edges contain no features and are created when a direct path between two adjacent nodes or a self-loop is established.

The dynamic layer consists of nodes and edges with temporal features, where the features are subject to change or removal in the recurrent stages of the GNN model. Nodes in this layer are of type *robot*, and the temporal feature of each node is the robot's priority, computed based on the buffer time between the current time estimation and the constraint time of the robot task. If two robots have equal priority, the priority is rounded to the nearest integer and appended with the robot's internal ID as decimals to make it unique to avoid deadlocks. Edges in the dynamic layer are of type *eta* and are created from a *robot* node to all *floor* nodes in the robot's planned trajectory. All outgoing *eta* edges from a *robot* node represent the planned path for the corresponding robot. The temporal features of the *eta* edges include the estimated duration at each *floor* node, the estimated arrival time, and the timestamp of the edge. The timestamp reflects the order in which the GNN processes the edges. All nodes contain a self-loop edge, which has no features, to ensure that the node's own features are combined with those of its neighbors during the message-passing phase.

Combining the dynamic and static layer, the spatio-temporal heterogeneous graph is defined as follows:

$$G = (V, E, T_v, T_e, X_{v(t)}, X_{e(t)}) \tag{1}$$

where V is the set of nodes, E is the set of edges, T_v represents the node types belonging to the set $\{\text{robot}, \text{floor}\}$, T_e represents the edge types belonging to the set $\{\text{eta}, \text{association}\}$, $X_{v(t)}$ is the set of node-features at a given time t, and $X_{e(t)}$ is the set of edge-features at a given time t.

2) Model architecture: The architecture of the GNN model is created accordingly to the encode-process-decode format illustrated in Fig. 2. First, each node and edge in the heterogeneous graph is sent to the encoding layer. An encoder is trained for each type of node and edge in the graph to extract specific type features into a 64-dimensional feature vector. The encoded nodes and edges are combined into the heterogeneous graph and passed to the attentive communication layers for message passing.

In the message passing module, we utilize the heterogeneous edge-enhanced graph attention (HEAT) operator [23]. This operator enhances the graph attention network (GAT) [24] by incorporating type-specific transformations of nodes and edges features during message passing. Similar to GAT, HEAT allows running several independent attention heads to stabilize the self-attention mechanism. The features from each attention head are concatenated as the updated output feature. The output of the HEAT operator is the updated node features from the neighborhood $K = 1$ and the number of attention heads $H = 3$ concatenated. The updated node features are passed to the edge updater module along with the encoded *eta* edges. In the edge updater module, the *eta*

edges are concatenated with the corresponding nodes and sent through a linear layer.

The decoder receives the *eta* edges with timestamp T and converts the eta feature vectors to ETA predictions. For DMS methods, the naive time of arrival of all *eta* edges in the dynamic layer is updated based on the ETA prediction to reflect the new predicted state of all robots. For IMS methods, the actual arrival time is used instead of the ETA prediction. As this is a recurrent process, this is repeated for $T = T + 1$ until all *eta* edges have been decoded.

As in the first layer, the second layer does not rely on a central station to process any information. However, global information is requested from a centralized station during run-time in order to gather information about all other agents in the system. While not a completely decentralized solution, it does improve scale-ability in terms of distributing the computations to each agent.

C. Path selection and Validating time-constraints

The path selection layer is the third layer of the proposed framework. This layer utilizes the ETA to evaluate time constraints and calculate the cost of each suggested route. The layer is used to find a feasible route within a fixed time constraint and to validate the chosen route's impact on the rest of the system by utilizing the predicted ETA for all agents. Various approaches can be used to evaluate a time-constrained path since the metric to select the best path depends on the specific application scenarios. In this work, a path is considered invalid if one or more of the time constraints are not met. To select between the valid paths, a cost function is proposed that considers the trade-off between the buffer times on the time constraints and the predicted arrival time. The cost function is defined as follows:

$$cost = \sum_{i=1}^{N} \left(\max\left(TC\right) - \left(TC_i - \text{eta}_i\right) \right)^2 \tag{2}$$

where *TC* represents the set of all the time constraints, *eta* is the set of predicted ETA for all N robots. The cost function is designed to maximize the buffer time between the constraint time and the predicted arrival time. At the same time, it penalizes the paths with a shorter buffer time and provides more slack for the paths with a larger buffer time. This ensures that the path selection process balances the objective of meeting the time constraints with the goal of minimizing the overall time spent to complete all tasks.

V. EXPERIMENTAL SETUP

This section concisely describes the generated training and evaluation datasets and the error metrics used for evaluation and training parameters. The first set of experiments focuses on evaluating the accuracy of the ETA prediction model, as it is a critical factor for the success of the proposed framework. The second set of experiments compares the proposed framework with state-of-the-art global planners to assess its overall performance.
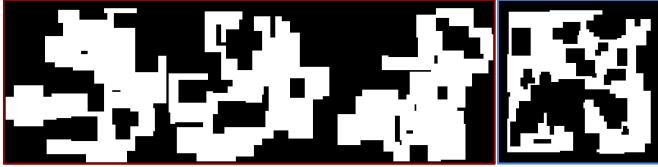
Fig. 3: The figure illustrates several examples of the maps utilized in training the multi-agent ETA prediction module (enclosed in red), and the map utilized for evaluating the overall performance of the system (enclosed in blue). The depiction of the black pixels indicates occupied spaces, while the white pixels signify unoccupied areas.

A. Simulation environment and noise

Global path planners are evaluated in terms of their robustness to conflicts and imperfections in plan execution due to noise. The experiment is designed to assess the impact of varying levels of imperfect plan execution noise, which are set to the following degrees: $\{0\%, 0.00001\%, 0.0001\%, 0.001\%, 0.01\%\}$. These levels correspond to the probability of a forced wait action resulting from accumulated noise.

We develop the ETA prediction module without considering the presence of noise to avoid any potential bias towards a pre-defined noise level. However, during the evaluation of the overall framework, noise was introduced to all methods to conduct a more comprehensive analysis.

The experiments utilize the WHCA* priority method as the local planner to resolve any conflicts that may arise during the simulation. This method was chosen for its simplicity, efficiency, and integration of priority selection into the algorithm. A constant time constraint is applied for all agents, prioritizing agents with longer paths.

B. Estimate time of arrival prediction

1) Training data: The GNN is trained on 5000 unique generated warehouse environments of the size of a maximum 100×100 meter. A few generated map examples are displayed in Fig. 3. The environments are populated with $N = \{250, 500, 1000\}$ robots, each with a corresponding goal point randomly distributed around the environment. The naive path is computed using the A^* algorithm, which is used as initial input to the GNN model. PIBT and CBS are not chosen as naive path planning methods for a couple of reasons. First, they modify the path of all robots and not just a single robot. Second, as the training simulation does not consider any noise or imperfect plan execution, this is the perfect scenario for these methods, resulting in no conflicts. After performing the simulation, the actual arrival time is recorded and used as the label.

2) Evaluation criteria: The loss function used during training is the mean average percentage error (MAPE). MAPE computes the percentage between predicted values $\hat{y}$ and ground truth y: $MAPE = \frac{1}{m}\sum_{i=1}^{m}|\hat{y}_i - y_i|/y_i$. It is the most popular metric for ETA prediction tasks, as the percentage penalizes the relative distance error, making

it robust against outliers compared to root mean squared error (RMSE), which strongly penalizes the outliers with big errors.

3) Training parameters: The training and evaluation are conducted using an NVIDIA GeForce RTX 3090Ti GPU with 24GB VRAM. A high amount of VRAM is proven essential when training as the space required for computing the gradient of long temporal graphs is very high. However, the gradient is not computed during inference time, so a large amount of VRAM is no longer required. Due to the extensive use of VRAM, only a batch size of one is used during training. We use the Adam optimizer, while the learning rate $\gamma = 0.001$ was scheduled to decay with 0.75 every 8th epoch. Each model has roughly trained 20 epochs, which with our hardware configuration, would take 36 to 48 hours. While training takes a long time to complete, the inference time is between 400 to 700 ms depending on the number of robots and the length of the paths.

C. Evaluation of the global path planners

The proposed method CAMETA is evaluated against other global path planning methods, including A^*, PIBT, and CBS with respect to their robustness in the presence of conflicts and plan execution noise. The A^* is chosen as a method that does not take into account the presence of other agents or conflicts in the paths being planned. PIBT is chosen as a suboptimal method that is designed to be computationally efficient and considers both all agents and conflicts. CBS is chosen as an optimal method. Although less computationally efficient than PIBT and cannot run in real-time, it can determine the optimal paths for all agents.

The environment chosen for the experiment is shown in Fig. 3 enclosed in blue. The map features both open spaces as well as corridors and potential bottlenecks. Each experiment will be conducted with 500 agents and repeated 100 times with different seeds of noise to eliminate a method being unlucky with the noise. However, the starting position and destination of robots will remain the same across all methods and seeds. The average *makespan*, which is the average time for all robots to finish, and the sum of cost (SOC) are presented in the results section.

VI. RESULTS

A. Estimate time of arrival prediction

1) Method comparisons: The first experiment compares the performance of the naive method, which does not use conflict-aware correction, the IMS method, and the DMS method. As shown in Table I, using conflict-aware correction with either IMS or DMS significantly reduces error. It can be seen that the DMS methods perform better than IMS methods due to the accumulated error nature in IMS, as explained in Section III-B. There is a disparity between the training and testing phases. As the density of robots rises and more conflict happens in the environment, the harder it is for the IMS method to adapt to incorrect predictions. The best trained IMS model improves the average MAPE by 29.5%

TABLE I: Comparison of the model and prediction method.

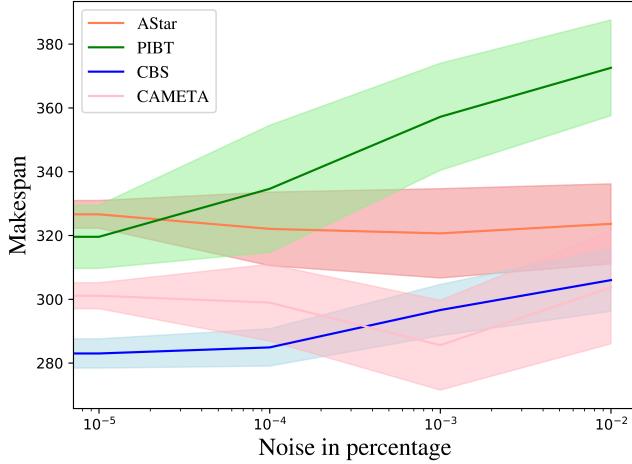| Training settings | | Test environments with: | | | | | | | | | | |
Methods	Trained with	250 robots			500 robots			1000 robots			Average		
		RMSE	MAPE[%]	MAE	RMSE	MAPE[%]	MAE	RMSE	MAPE[%]	MAE	RMSE	MAPE[%]	MAE
Naive (A^*)		8.06	8.36	3.59	13.45	12.79	6.80	42.29	26.08	24.83	21.27	15.74	11.74
IMS	250 robots	7.18	4.59	3.19	12.26	8.75	5.99	38.36	21.15	21.81	19.27	11.50	10.33
	500 robots	7.47	4.25	3.05	11.99	8.31	5.75	39.15	21.27	22.28	19.54	11.28	10.36
	1000 robots	7.36	4.53	3.15	11.70	8.31	5.66	37.79	20.46	21.30	18.95	11.1	10.04
DMS	250 robots	**6.14**	**3.72**	**2.52**	9.58	7.09	4.60	31.33	17.17	17.03	15.68	9.33	8.05
	500 robots	6.38	3.78	2.60	9.17	**6.91**	**4.44**	28.42	16.16	15.51	14.66	8.95	7.52
	1000 robots	6.37	4.03	2.70	**9.05**	7.02	4.45	**25.69**	**15.39**	**13.93**	**13.70**	**8.81**	**7.03**

Fig. 4: The plot shows the increase in the avg. makespan as noise is applied in various degrees. As the cost function prioritizes the agents with the tightest schedule, a stable makespan can be seen for CAMETA.
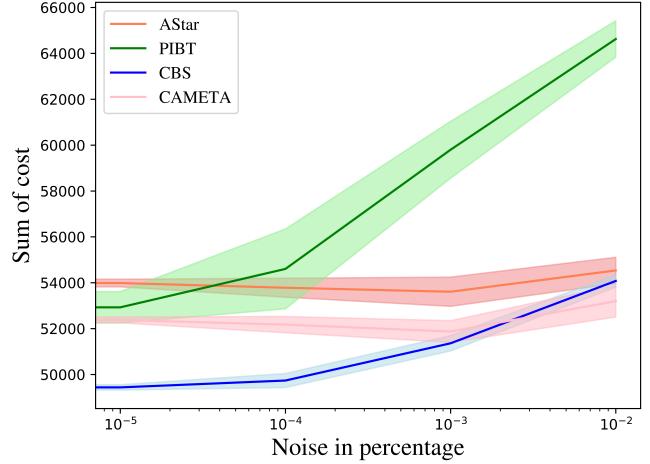

Fig. 5: The plot shows the increase in avg. sum of costs as noise is applied in various degrees. Both PIBT and CBS rises as the noise increases, while A^* and CAMETA is less affected by the noise.

compared to the naive method, while the best DMS model improves by 44%.

2) Generalization under different robot densities: The second experiment demonstrates the generalization of the models in environments of varying densities. As shown in Table I, three different models trained on a dataset containing 250, 500, and 1000 robots are compared at different densities. The results indicate that training in environments of high density generalizes well to environments of low density. This implies that none of the models are overfitting to any density-specific features in the graph representation.

The DMS models trained with the same number of robots as in the test scenario generally perform slightly better, as depicted in bold in Table I. However, in the 500 robot test environment, the 1000 robot DMS model outperforms the 500 robot DMS model in terms of RMSE. As more conflicts happen in higher-density environments, the 1000 robot DMS model is generally trained on paths much longer than the 500 robot DMS model. As a result, since RMSE is sensitive to longer paths as they accumulate more errors, as shown in experiment two, the 1000 robot DMS is better suited for outliers occurring within the 500 robot test dataset. The last column of Table I shows average results in all the test environments. The DMS model trained on 1000 robots has

proven to be the most generalized model.

B. Evaluation of the global path planners

The results of the path planners under the effect of noise are presented in Fig. 4 and Fig. 5. The CBS method yields the optimal solution at a noise level of 0%, which highlights the disparity between this method and the real-time counterparts. While the CBS algorithm demonstrates superior performance in terms of SOC compared to other methods, its computational time is extremely long, taking four hours to generate the optimal solution.

The CAMETA and PIBT methods yield comparable results at 0% noise, with CAMETA outperforming PIBT by utilizing the exploration of alternative paths with fewer conflicts. In contrast, PIBT only resolves conflicts based on the A^* algorithm, incorporating 'wait' actions and priority inheritance. The basic A^* method yields the poorest results at this noise level, as expected, due to the lack of conflict resolution in the planning. As the noise level increases, PIBT demonstrates a decrease in performance due to the coordinated nature of its planned paths, which leaves little room for error. The additional moves planned for conflict resolution do not align with the expected conflicts, leading to even more conflicts, resulting in a rapid increase in total moves required

to resolve everything as the noise level increases. This is reflected in Fig. 5 as a rapid increase in SOC for the PIBT method. The local planner, WHCA*, resolves conflicts as the noise level increases. However, due to the extra moves already planned for conflicts, conflicts arising from noise are harder to resolve. Both the basic A^* and CAMETA methods demonstrate a stable pattern as the noise level increases, as they rely on conflicts being resolved in real-time, making it easier for the local planner to correct. However, CAMETA exhibits better results in terms of total SOC and makespan, as it reduces the number of conflicts by allowing for longer routes and distributing traffic, resulting in noise not affecting the planned path as much.

Overall, the results illustrate that while CBS may provide the optimal solution in terms of SOC, it is not a feasible solution for real-time applications. On the other hand, CAMETA exhibits more promising and stable performance in terms of real-time path planning under noisy conditions.

VII. Conclusion and future work

In this work, we develop a framework to predict conflicts and ETA in multi-agent environments. We formulate our problem as a spatio-temporal graph focusing on edge prediction. The proposed methodology allows ETA prediction of all robots simultaneously, which was not possible by the previously published works. Through extensive simulation experiments, the proposed method demonstrates an increase in the accuracy of the predicted arrival time. It should be noted that our proposed framework does not solve the MAPF problem as it does not provide collision-free paths for each agent. Instead, we employ the use of ETA in order to minimize the number of conflicts that a local path-planning algorithm needs to resolve, resulting in a more resilient method that is better equipped to handle noise.

As future work, we aim to enhance the framework by incorporating a more sophisticated robot motion model and transitioning to a continuous space simulation to better reflect real-world dynamics. Additionally, we plan to extend the framework to include multiple types of robots in a heterogeneous graph representation, incorporating dynamics, constraints, and different types of robots as nodes. Furthermore, we will incorporate the path traversed within a floor tile as a feature attribute in the *eta* edges.

Acknowledgment

References

[1] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[2] K. Okumura, M. Machida, X. Défago, and Y. Tamura, "Priority inheritance with backtracking for iterative multi-agent path finding," *Artificial Intelligence*, p. 103752, 2022.

[3] D. Silver, "Cooperative pathfinding," in *Proceedings of the aaai conference on artificial intelligence and interactive digital entertainment*, vol. 1, no. 1, 2005, pp. 117–122.

[4] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. Kumar *et al.*, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 10, no. 1, 2019, pp. 151–158.

[5] J. Yu and S. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 27, no. 1, 2013, pp. 1443–1449.

[6] H. Ma, S. Koenig, N. Ayanian, L. Cohen, W. Hönig, T. Kumar, T. Uras, H. Xu, C. Tovey, and G. Sharon, "Overview: Generalizations of multi-agent path finding to real-world scenarios," *arXiv preprint arXiv:1702.05515*, 2017.

[7] W. Hönig, T. S. Kumar, L. Cohen, H. Ma, H. Xu, N. Ayanian, and S. Koenig, "Multi-agent path finding with kinematic constraints," in *Twenty-Sixth International Conference on Automated Planning and Scheduling*, 2016.

[8] G. Wagner and H. Choset, "Path planning for multiple agents under uncertainty," in *Twenty-Seventh International Conference on Automated Planning and Scheduling*, 2017.

[9] ——, "M*: A complete multirobot path planning algorithm with performance bounds," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 3260–3267.

[10] A. Zelinsky, "A mobile robot navigation exploration algorithm," *IEEE Transactions of Robotics and Automation*, vol. 8, no. 6, pp. 707–717, 1992.

[11] Z. Bnaya and A. Felner, "Conflict-oriented windowed hierarchical cooperative a," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3743–3748.

[12] H. I. Ugurlu, X. H. Pham, and E. Kayacan, "Sim-to-real deep reinforcement learning for safe end-to-end planning of aerial robots," *Robotics*, vol. 11, no. 5, 2022. [Online]. Available: https://www.mdpi.com/2218-6581/11/5/109

[13] Q. Li, W. Lin, Z. Liu, and A. Prorok, "Message-aware graph attention networks for large-scale multi-robot path planning," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5533–5540, 2021.

[14] D. R. Cox, "Prediction by exponentially weighted moving averages and related methods," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 23, no. 2, pp. 414–422, 1961.

[15] G. Chevillon, "Direct multi-step estimation and forecasting," *Journal of Economic Surveys*, vol. 21, no. 4, pp. 746–785, 2007.

[16] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *arXiv preprint arXiv:1707.01926*, 2017.

[17] X. Shi and D.-Y. Yeung, "Machine learning for spatiotemporal sequence forecasting: A survey," *arXiv preprint arXiv:1808.06865*, 2018.

[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[19] A. Derrow-Pinion, J. She, D. Wong, O. Lange, T. Hester, L. Perez, M. Nunkesser, S. Lee, X. Guo, B. Wiltshire *et al.*, "Eta prediction with graph neural networks in google maps," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 3767–3776.

[20] J. Huang, Z. Huang, X. Fang, S. Feng, X. Chen, J. Liu, H. Yuan, and H. Wang, "Dueta: Traffic congestion propagation pattern modeling via efficient graph learning for eta prediction at baidu maps," *arXiv preprint arXiv:2208.06979*, 2022.

[21] H. Hong, Y. Lin, X. Yang, Z. Li, K. Fu, Z. Wang, X. Qie, and J. Ye, "Heteta: heterogeneous information network embedding for estimating time of arrival," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 2444–2454.

[22] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968. [Online]. Available: https://doi.org/10.1109/tssc.1968.300136

[23] X. Mo, Z. Huang, Y. Xing, and C. Lv, "Multi-agent trajectory prediction with heterogeneous edge-enhanced graph attention network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 9554–9567, 2022.

[24] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *stat*, vol. 1050, p. 20, 2017.