2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

NLNS-MASPF for solving Multi-Agent scheduling and Path-Finding

Heemang Park[1], Kyuree Ahn[2] and Jinkyoo Park[1;2]

***Abstract*— In this work, we propose a novel method, NLNS-MASPF, to solve the Multi-Agent Scheduling and Pathfinding (MASPF) problem. The problem exhibits a bi-level structure, consisting of High-level Scheduling and Low-level Pathfinding. Our method applies a graph neural network in the high-level scheduling process and utilizes a MAPF solver with a schedule segmenting technique in the low-level pathfinding process. Through these approaches, NLNS-MASPF has experimentally demonstrated superior performance compared to the previous state-of-the-art MASPF algorithm, LNS-PBS, in solving the MASPF problem.**

I. Introduction

Effectively controlling multiple robots to operate harmoniously is of immense importance, a significance that has steadily increased across various industries. In logistics systems, optimizing and efficiently operating logistics robots responsible for delivery and transportation lies at the core of industrial growth. In real-life scenarios such as indoor maintenance, healthcare, and service environments, the ability of service robots to cooperate appropriately is also of paramount importance.

Among the research efforts aimed at effectively managing multi-robots, the field of Multi-Agent Path Finding (MAPF) has gained significant attention for developing methodologies to plan the paths of multiple robots. Researchers have been working on determining how to construct optimized and collision-free paths for multi-robots. Standard MAPF algorithms are designed to search for collision-free paths when there is a one-to-one correspondence between robots and tasks. For example, Cooperative A* [1], conflict-based search (CBS) [2], and extended CBS [3] aim to find the shortest paths for robots when robots and tasks are provided in a one-to-one matching scenario. On the other hand, Anonymous MAPF algorithms focus on constructing collision-free paths when robots are not assigned to specific tasks. For instance, algorithms like TAPF (Task Assignment for Path Finding) [4] and AMAPFwID-DOT (Anonymous Multi-Agent Path Finding with Incomplete and Dynamic Task Ordering) [5] focus on finding paths when the specific tasks for robots are not predefined.

While these methodologies are quite effective, they primarily deal with one-to-one mappings. In reality, many systems involve a large number of tasks that are not predefined. Consequently, the need to schedule task sequences and find collision-free paths becomes a simultaneous requirement. This complex problem is referred to as multi-agent scheduling and pathfinding (MASPF) [6].

[1]Department of Industrial and Systems Engineering, KAIST, Korea
[2]Omelet, Daejeon, Korea

The state-of-the-art algorithm for MASPF, LNS-PBS [7], tackles the challenge of scheduling tasks for multiple robots and tasks. LNS-PBS creates an initial schedule using heuristics, then generates multiple candidate schedules by modifying the schedule. Afterward, the MAPF solver calculates the sum of the true collision-free path cost for every candidate schedule, and selects the schedule with the lowest path cost. This improved schedule is repeated within the time budget until best schedule is chosen. In this step-by-step approach, the algorithm iteratively outputs the improved schedules from the initial schedule. Due to its bi-level structure, conducting scheduling and pathfinding sequentially allows for comprehensive computation and potentially results in optimal solutions. However, achieving good solutions may require multiple iterations, which can lead to a significant computational load.

Our research addresses a limitation of LNS-PBS through the use of neural large neighborhood search (NLNS), which necessitates multiple iterations and repeated executions of the MAPF solver to achieve substantial schedule improvement. To achieve this, we have developed a graph neural network (GNN) architecture designed to predict the path cost of the MAPF solver by employing an attention mechanism. We propose NLNS-MASPF, which eliminates the need for complicated and numerous executions of the MAPF solver during scheduling, as the neural network's forward process alone can efficiently predict the path cost of the MAPF problem. We experimentally validated that NLNS-MASPF could efficiently improve the low-level MAPF cost as well as the high-level scheduling performance, achieving 25% improved scheduling performance within the time budget compared to the starte-of-art MASPF solver.

The main contributions of our research are as follows:

- We designed a prediction model that can estimate the output of MAPF solver, allowing batched computation of MAPF cost which can be utilized in scheduling.
- We proposed a novel method for interpreting schedule cost through schedule decomposition method.
- We verified that the scheduling performance could increase through batched prediction of MAPF cost.

II. Problem Formulation

In this section, we formulate the multi-agent scheduling and pathfinding (MASPF) problem as a bi-level optimization problem, where a set of agents needs to complete a set of tasks while avoiding conflicts with each other. The objective is to find a schedule of all the agents and a set of collision-free paths for the agents that minimize the total travel cost given the optimized schedule. The MASPF is composed

of N robot agents and M tasks, where the set of robot and task are referred as $\mathcal{N}$ and $\mathcal{M}$, index of robot and task are referred as $n \in \mathcal{N}$ and $m \in \mathcal{M}$, and location of robot and task are referred as l_n and l_m, respectively. The problem consists of two hierarchical steps: **high-level scheduling**, which determines the task schedule for the robots, i.e., the sequence of task, and **low-level pathfinding**, which determines the shortest path for the robots to follow the task schedule without collisions. We follow the problem definition defined as in [6]. The bi-level formulation is as below:

$$\text{(High-level)} \quad \min_{\boldsymbol{s}} \quad c^*(\boldsymbol{s}) := \sum_{n=1}^{N} c_n^*(s_n; \boldsymbol{s}_{-n}) \tag{1}$$

$$\text{(Low-level)} \quad \text{s.t.} \quad c_n^*(s_n; \boldsymbol{s}_{-n}) = \text{Solver}(s_n, \boldsymbol{s}_{-n}), \forall n \tag{2}$$

where decision variable $\boldsymbol{s} = (s_1, \ldots, s_N)$ is the joint schedule of all the agents, s_n is the individual task schedule of robot n, $\boldsymbol{s}_{-n}$ is the joint task schedule other than n-th agent, $c_n^*(s_n; \boldsymbol{s}_{-n})$ represents the minimal path cost of robot n given joint tasks schedules, and $\boldsymbol{c}^*(\boldsymbol{s})$, the sum of the path costs of all robots, is the overall objective function that we want to minimize.

A. High-level Scheduling

The high-level scheduling problem, Eq 1, aims to find the optimal joint schedule of all the robot agents, $\mathbf{s}^*$, that minimizes the path-finding cost. Each high-level schedule consists of a robot and its corresponding task sequence. The schedule of n-th robot, s_n is defined as the sequence of robot/task indices:

$$s_n : [n, \tau_n^1, \ldots, \tau_n^{T_n}] \tag{3}$$

where $\tau_n^i \in \mathcal{M}$ denotes the i-th task sequence of robot n, and T_n indicates the number of tasks that n-th robot should visit.

B. Low-level Path Finding

In the low-level problem, the objective is to determine collision-free paths for each agent from their initial positions to their assigned tasks, while considering the paths of other agents and map information. Unlike traditional MAPF, which aims to find a path for all agents with a single goal location, MASPF deals with multiple goal locations, necessitating multiple instances of MAPF solving to find an optimal path with intermediate points. The low-level problem's goal is to minimize the total travel cost by finding paths with schedule. The result of the path planning is a path trajectory and the corresponding path cost $c^*(\boldsymbol{s})$.

III. NLNS-MASPF

We propose NLNS-MASPF, a novel methodology approach for effective multi-robot multi-task schedule planning while ensuring collision-free paths. NLNS-MASPF schedules the task orders of robots using a local search based on the LNS algorithm. NLNS-MASPF continuously refines the given initial schedule until it achieves low enough path cost. In this section, we elaborate on the step-by-step process of how the schedule is updated. Figure 1 depicts a single schedule update. Additionally, it is assumed that path cost is provided by a deep-learning based prediction model f_{MAPF} which estimates the path cost of the MAPF solver:

$$\hat{c}^*(\boldsymbol{s}) = f_{MAPF}(\boldsymbol{s}) \tag{4}$$

Algorithm 1 Pseudo Code of Scheduling Iteration

Input : initial schedule $\boldsymbol{s}$, time budget$B$
while $T > 0$ **do**
  $\mathbf{S}_c \leftarrow \emptyset$
  for $i = 1, \ldots, N_o$ **do**
    $\boldsymbol{s}' \leftarrow O(\boldsymbol{s})$
    $\mathbf{S}_c \leftarrow \{\boldsymbol{s}'\} \cup \mathbf{S}_c$
  end for
  $\hat{\mathbf{c}}^* : \left[\hat{c}^*(\boldsymbol{s}'_1), \ldots, \hat{c}(\boldsymbol{s}'_{N_o})\right] \leftarrow f_{MAPF}(\mathbf{S}_c)$
  $\boldsymbol{s}^* \leftarrow \arg\min_{\boldsymbol{s}'_i}(\hat{\mathbf{c}})$
  if $\hat{c}^*(\boldsymbol{s}^*) < \hat{c}^*(\boldsymbol{s})$ **then**
    $\boldsymbol{s} \leftarrow \boldsymbol{s}^*$
  end if
  $T- = operationtime$
end while

A. Schedule Modification

Scheduling procedure starts from establishing initial schedule $\boldsymbol{s}$. NLNS-MASPF utilized Hungarian algorithm to get appropriate initial schedule cheaply. Initial schedule $\boldsymbol{s}$ is generated by connecting task with the smallest Manhattan distance from each robot's starting point sequentially. After that, schedule modification is conducted. Schedule modification step proposes candidate schedule $\mathbf{s}'$ by modifying current schedule $\mathbf{s}$ by operator O. To be more specific, operator reinsert tasks into random positions in schedule $\boldsymbol{s}$ and produce s'. k is a hyperparameter which modulates the degree of change in schedule per operation. In our study, k is set to 3.

B. Schedule Selection

After generating N_o candidate schedules, the best schedule needs to be selected. NLNS-MASPF utilizes an efficient prediction model for this selection process. The predict model takes candidate schedule as an input and predicts the true path costs of given candidate schedule through a network forward process. This process can be carried out in a parallel and efficient manner by using batch data. By selecting the candidate schedule with the smallest path cost, we obtain an improved schedule denoted as $\boldsymbol{s}^*$. Repeating such single schedule update step within a predefined time budget B, NLNS-MASPF achieves the final improvement. The complete pseudocode describing the iterative process is provided in algorithm 1.

IV. Relational Transformer-based Cost Prediction

We designed a graph neural network for predicting path costs, which takes various map state features into account, including information on robots, tasks, obstacles, and the

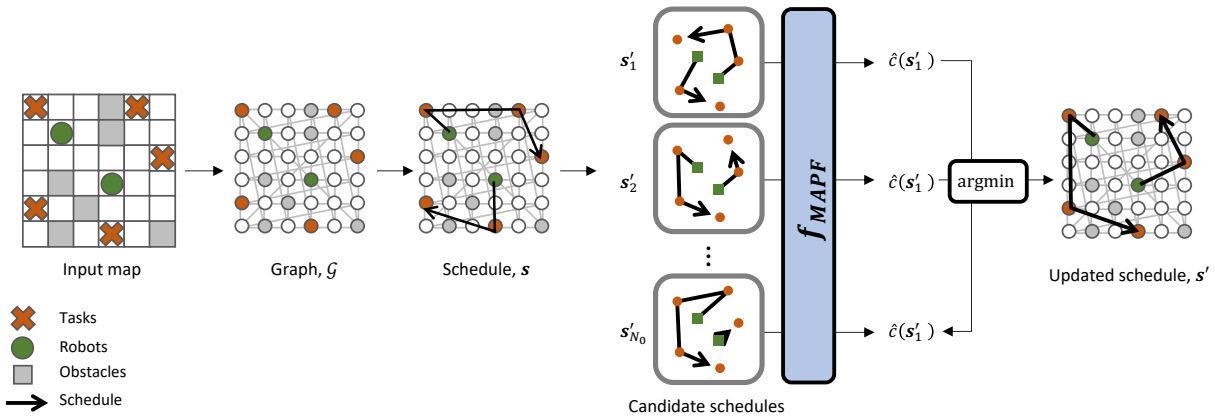

Fig. 1: The overall architecture of NLNS-MASPF involves representing the input map instance for which a schedule is desired as graph data. A search algorithm is then applied to find a candidate schedule that could be improved. f_{MAPF} function predicts the schedule cost of each candidate schedule, enabling the selection of the best candidate schedule.

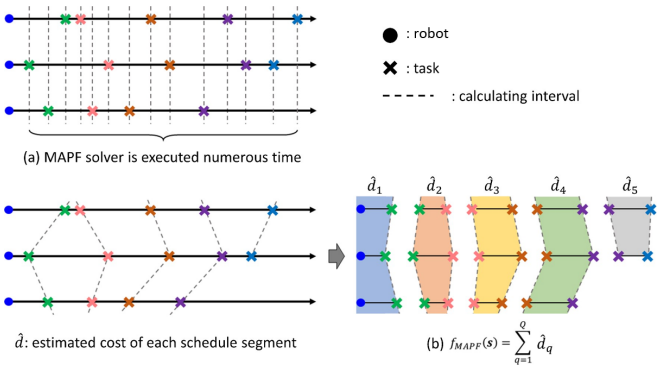

Fig. 2: Schedule Decomposition

robot-task schedule. This network creates unique path embeddings using these features. Path embeddings are tailored to capture information about the paths between nodes based on the distributions of robots, tasks, and obstacles. The neural network leverages these path embeddings to compute the sum of the tour lengths for the entire schedule.

Furthermore, MAPF solvers continually monitor the movement of robots over time to explore paths, offering precise results but exhibiting inefficiency and slowness. NLNS-MASPF takes a different approach by approximating the schedule's cost through division into several additive parts and summing the prediction results of each part. This approach allows the neural network to predict the path cost for the entire schedule within just a few forward network steps, eliminating the need to track the positions of robots over time. This not only enhances algorithm efficiency but also significantly improves speed.

A. Schedule Decomposition

In this section, we compare the schedule interpretation approach of previous MAPF research with our method, NLNS-MASPF. Figure 2 (a) illustrates the time-oriented cost calculation. When following this approach, the MAPF solver repeatedly calculates the paths of the robots at every time step. In other words, the solver continuously tracks the movement of the robots along their paths, recalculating the length of the paths multiple times throughout the process. Instead of that, our method factorizes the schedule into Q schedule segments divided by tasks and applied neural network to estimate path costs for each decomposed schedule. Estimated cost of schedule segment is $\hat{d}_q$. NLNS-MASPF sum entire $\hat{d}_q$ to attain total schedule cost. By computing the decomposed schedule cost in a compact network forward pass, it is possible to obtain a much faster yet substantial solution. Figure 2 (b) illustrates the decomposed cost estimation.

B. Graph Representation

We first define a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to effectively represent a MAPF instance. We define each grid cell as a node, $\mathcal{V} = \{v_i\}_{i=1}^{|X|\times|Y|}$, where $|X|$ and $|Y|$ indicates a grid size. The edges $e_{ij} \in \mathcal{E}$ represent the connectivity between grid cells. To process the graph using a GNN, we define a node feature as the concatenation of normalized node coordinates and node type, where node type represents whether the cell is one of {empty, obstacle, robot, task} where x_i and y_i are normalized coordinates of cell i. i.e.,

$$v_i = [x_i, y_i, \mathbb{1}_{empty}, \mathbb{1}_{obstacle}, \mathbb{1}_{robot}, \mathbb{1}_{task}] \quad \forall i \in |V| \quad (5)$$

C. GNN Prediction Model

In this section, we describe how the prediction model is structured and the process by which it predicts the path cost of a schedule. To effectively handle information distributed in the map about robots, tasks, and their schedules, the prediction model is constructed using a graph neural network. Our goal is for the neural network to correctly process information about paths between cells and infer path costs. To achieve this, we introduced the widely used attention mechanism from the transformer architecture. The core idea of this concept was proposed in Relational Transformer [8]. This research applied the attention mechanism to GNN, allowing the network to more effectively understand information about the graph's structure by treating node information as entity

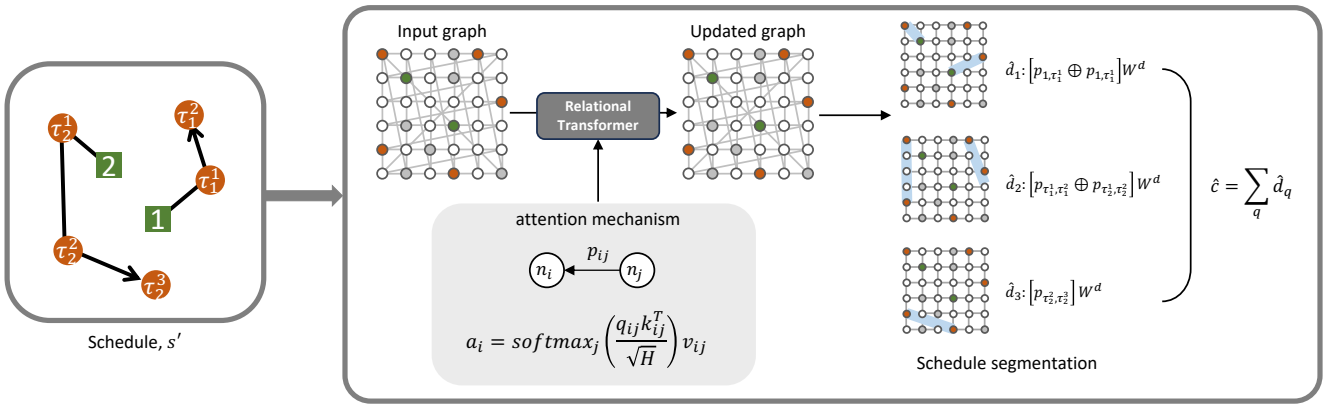

Fig. 3: Prediction Model Structure

features and edge information as relational features, enabling deeper insights into the graph's structure. Overall process where the prediction model takes each schedule as input and calculates the estimated path cost is illustrated in Figure 3.

In order to make path embeddings containing information between all nodes treated as source and destination nodes, each embedding with feature dimension F is transformed into query, key, and value vectors with hidden dimension H using separate linear layers:

$$k_{ij} = [n_j \oplus p_{ij}]W^K, \tag{6}$$

$$q_{ij} = [n_i \oplus p_{ij}]W^Q, \tag{7}$$

$$v_{ij} = [n_j \oplus p_{ij}]W^V \tag{8}$$

where $[n_i \oplus p_{ij}] \in \mathbb{R}^F$, $p_{ij} \in \mathbb{R}^E$, $W^Q, W^K, W^V \in \mathbb{R}^{F\times H}$, $k_{ij}, q_{ij}, v_{ij} \in \mathbb{R}^H$. n is a node embedding and p_{ij} is a path embedding from source node i to destination node j. Using the k_{ij}, q_{ij}, v_{ij}, the attention score of d a_i is computed. This process ensures that relational information about source-destination paths is effectively incorporated into the node embeddings, similar to how it works in the attention mechanism:

$$a_i = \text{softmax}_j\left(\frac{q_{ij}k_{ij}^T}{\sqrt{H}}\right)v_{ij} \tag{9}$$

Obtaining the attention vector guides node embeddings to effectively absorb relational information across the entire map. Subsequently, a standard message-passing scheme for edge updates is applied. The message m_{ij} between node i and node j is expressed as follows:

$$m_{ij} = \text{ReLU}([p_{ij} \oplus p_{ji} \oplus n_i \oplus n_j]W^m) \tag{10}$$

where $W^m \in \mathbb{R}^{(2\times F)\times H}$. Using the message m_{ij} and path embedding p_{ij}, an updated path embedding u_{ij} is generated as follows:

$$u_{ij} = \text{LayerNorm}(m_{ij}W^u + p_{ij}) \tag{11}$$

Finally, the path embedding is updated as follows:

$$p_{ij}^{updated} = \text{LayerNorm}(\text{ReLU}(u_{ij}W')W'' + p_{ij}) \tag{12}$$

where $W^u \in \mathbb{R}^{H\times E}$, $W' \in \mathbb{R}^{H\times E}$, $W'' \in \mathbb{R}^{H\times E}$. After our graph neural network generates path embeddings p_{ij} that contain information of the map and schedule, it outputs the predicted cost of each schedule segments. As mentioned before, we factorized schedule into max segment schedules. Predicted cost of total schedule $\mathbf{s}^*$ is sum of d_q:

$$\hat{d}_q^* = \bigoplus_{n=1}^{N} p_{s_{n[q]},s_{n[q+1]}}W^d \tag{13}$$

$$\hat{c}^*(\boldsymbol{s}) = \sum_{q=0}^{Q-1} \hat{d}_q^* \tag{14}$$

where $W^d \in \mathbb{R}^{H\times 1}$.

D. Training Neural Network

In this section, we will provide a detailed explanation of how we trained the neural network described in IV-C. To train our neural network, we followed a traditional supervised training process.

Data Collection: We gathered a dataset consisting of input data and corresponding label data. The input data includes maps and robot-task schedules. The purpose of this process is to train the GNN prediction model to predict how the true MAPF cost will be calculated when schedule data is given. The map is an 8x8 grid with various *obstacle ratio*, *N*, *M*.

when o represents the density of obstacles distributed on the map. For example, the first configuration corresponds to a scenario where there are 0% obstacle cells on the map, and there are 25 agents and 25 tasks, respectively. The label data consists of the true path costs calculated by the PBS [7] for each schedule.

Training Objective and Loss Function: The objective of the training is to minimize the difference between the model's predicted values and the ground truth values. To measure the distance between the predicted values and the ground truth, we used the $L1$ loss function defined as:

$$\mathcal{L} = \mathbb{E}_{\boldsymbol{s}\sim\mathcal{D}}\left[|\hat{c}^*(\boldsymbol{s}) - c^*(\boldsymbol{s})|\right] \tag{15}$$

V. EXPERIMENTS

Our experiments can be divided into two parts. First, we need to validate the prediction performance of the trained network. To solve the bi-level target problem, it is essential to accurately predict the low-level path cost. We aimed to experimentally demonstrate whether the network can predict high-accuracy values when given a schedule, allowing robots to move toward their designated tasks.

After verifying the network's performance in predicting low-level path costs, we examined the extent of performance improvement when applying it to high-level scheduling. The baseline algorithm used is LNS-PBS. LNS-PBS is a heuristic Large Neighborhood Search algorithm for high-level scheduling that does not incorporate neural networks. It operates by iteratively destroying and repairing the initial schedule within a predetermined time budget, ultimately leading to the schedule with the lowest path cost. We executed our model with the same time constraint as the baseline algorithm and compared the results. The hyperparameters of our model are as follows: The hidden dimension size of the neural network is 16. There are 8 layers stacked within the graph neural network. The node embedding aggregator uses the 'sum' operator, and the path embedding aggregator uses the 'max' operator.

A. Performance of low-level cost prediction

Experiment Setup We generated datasets using grid maps of size 8x8. The difficulty levels of these grid maps varied according to the density of obstacles. The difficulty level based on the density of obstacles ranges from 0% to 30%. For each map configuration, we set $(N, M) \in \{(5, 15), (5, 20), (5, 25), (5, 30)\}$. Figure 4 shows the prediction performance of 10,000 test samples in sparse, medium, and dense maps. For each subfigure, x-axis represents the model's predicted cost, while the y-axis represents the true cost. The closer our proposed model's predictions align with the label data, points will be plotted along a diagonal line.

Experiment Result The x-axis represents the predicted values of the GNN prediction model, and the y-axis represents the true values of the label data. Therefore, the higher the predictive accuracy of our model, the more the plot should resemble an upward-sloping diagonal line, similar to the graph of y=x. When examining the plots, it is evident that the prediction accuracy remains consistently stable regardless of the number of robots and tasks. This observation indicates that our proposed model exhibits robust performance concerning variations in the numbers of robots and tasks. The prediction accuracy in the dense map is relatively lower compared to the accuracy in other maps. However, it's important to emphasize that our target problem doesn't require obtaining the exact path cost values in every scenarios. The key lies in the ability of the model to compare the relative sizes of schedules in the subsequent high-level scheduling procedure and select the best schedule. From this perspective, the upward trend observed in the dense map case suggests that the model can correctly discern the relative magnitudes of schedules in challenging scenarios, which is crucial for the high-level scheduling process.

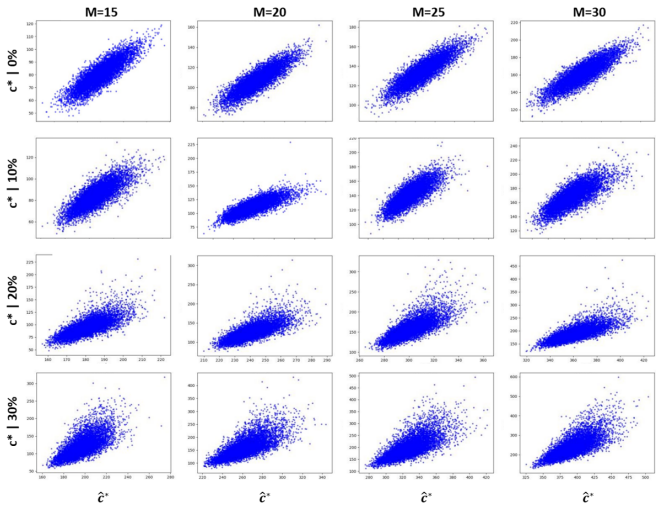

Fig. 4: Low-level cost prediction performance. In each subfigure, x-axis indicates predicted cost, $\hat{c}^*(\boldsymbol{s})$, and y-axis indicates the true cost, $c^*(\boldsymbol{s})$. 10,000 datapoints are plotted for each subfigure.

B. Performance of high-level scheduling

Experiment Setup We applied the NLNS-MAPSF in each iteration of scheduling improvement to verify how the prediction network could further improve schedule within time budget. We verified the scheduling cost in scenarios with $N = 5$ and $M \in \{15, 20, 25, 30\}$, with obstacle density of 0%, 10%, 20%, 30%.

Operator Variants The LNS algorithm explores the solution space by destroying and repairing the previous solution. In this process, we implemented various types of operators for destroying and repairing the solution to verify the robustness of our GNN prediction model. The process of selecting nodes to be destroyed by the operator was fixed to be random.

- **Random** operator modifies the solution by re-inserting the selected nodes at random positions.
- **Closest-first** operator modifies the solution by re-inserting the closest nodes to each other at random positions.
- **Fharest-first** operator modifies the solution by re-inserting the farthest nodes from each other at random positions.

Baseline Algorithms In the experiment, we compared NLNS-MASPF with three different baselines. The difference among these algorithms lies in how they select the best schedule from the candidate schedules as described in V-B. The descriptions of the baseline algorithms are as follows:

- LNS-PBS [7] calculates the true path costs for all candidate schedules using a solver and then selects the best one.
- LNS-PBS(r) randomly selects a schedule from the candidate schedules.

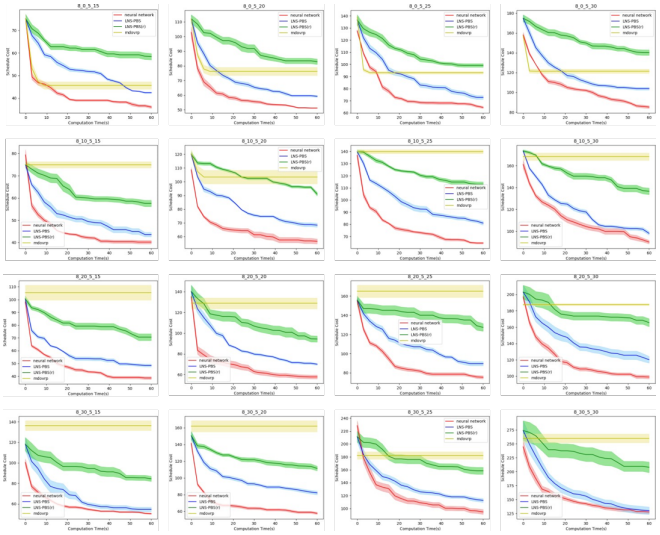

Fig. 5: High-level scheduling performance of **random** operator. Comparison of High-level scheduling performance between algorithms (8×8). In each subfigure, x-axis indicates the computation time budget and y-axis indicates the scheduling performance (cost) of each algorithms. Shaded region represents 95% confidence interval.

- MDOVRP is a combinatorial optimization solver for solving the multi-depot open vehicle routing problem. Due to the nature of the combinatorial optimization problem, it does not consider obstacles. When obstacle information is not considered, it finds a path that minimizes the total length.

High-level scheduling performance Figure 5 indicates the scheduling performance of all the algorithms. In figures, each column comprises subfigures with the same obstacle density, while each row corresponds to sparse, high, and medium map types. The shaded regions within each subfigure represent 95% confidence intervals, offering insights into obstacle distribution variability across different map categories and obstacle densities. During a time budget of 60 seconds, each algorithm is executed to determine how much the cost of the solution has decreased. Our algorithm, which schedules using the GNN prediction model, showed the fastest cost reduction in all scenarios. We experimentally validated that our GNN prediction model yields superior results compared to the conventional approach of verifying schedule path costs one by one using a solver. Further experinmental results on high-level insertion method can be found on Appendix D

C. Visualized multi-robot schedule

In this section, we qualitatively compared and evaluated the scheduling performance of our models, NLNS-MASPF and LNS-PBS. The experimental setup involved a scenario in which 5 robots on an 8x8 map should visit 15 tasks without collisions. Information regarding the matching and scheduling between robots and tasks was not provided at all. Each algorithm, LNS-PBS and NLNS-MASPF autonomously conducted scheduling with the objective of minimizing the sum of the path costs. Both algorithm performed scheduling for the same time duration (120s), and we compared the schedules generated by LNS-PBS and NLNS-MASPF by plotting their final results on the map. The schedule generated by LNS-PBS resulted in sum of path cost $\hat{c}^*(\boldsymbol{s})$, of 46. While the schedule generated by NLNS-MAPF resulted in a lower sum of path cost, which is 35, which is 25% improved schedule than that of PBS-LNS.

VI. Conclusion

In this study, we introduce an efficient approach for solving multi-agent scheduling and pathfinding (MASPF) problems by integrating a neural network-based technique. We formulated the problem with a bi-level structure, considering both task sequence scheduling and true path construction simultaneously. To address this, we developed a GNN model capable of effectively processing complex map state and path information. The trained GNN serves as a predictive model for generating higher-quality multi-robot schedules. Our experimental results demonstrated a significant performance enhancement of NLNS-MASPF compared to traditional heuristic-based scheduling methods. In future work, we aim to enhance the algorithm's capabilities by exploring advanced neural network architectures, optimizing training methodologies, and investigating real-world scenarios.

VII. Acknowledgement

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (2022-0-01032, Development of Collective Collaboration Intelligence Framework for Internet of Autonomous Things)

References

[1] David Silver. Cooperative pathfinding. In *Proceedings of the aaai conference on artificial intelligence and interactive digital entertainment*, volume 1, pages 117–122, 2005.

[2] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015.

[3] Jiaoyang Li, Wheeler Ruml, and Sven Koenig. Eecbs: A bounded-suboptimal search for multi-agent path finding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12353–12362, 2021.

[4] Hang Ma and Sven Koenig. Optimal target assignment and path finding for teams of agents. *arXiv preprint arXiv:1612.05693*, 2016.

[5] Gilad Fine, Dor Atzmon, and Noa Agmon. Anonymous multi-agent path finding with individual deadlines. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pages 869–877, 2023.

[6] Heemang Park Kyuree AHN and Jinkyoo Park. Learning to schedule for multi-agent pathfinding. In *IROS 2023*, 2023.

[7] Hang Ma, Daniel Harabor, Peter J Stuckey, Jiaoyang Li, and Sven Koenig. Searching with consistent prioritization for multi-agent path finding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 7643–7650, 2019.

[8] Cameron Diao and Ricky Loynd. Relational attention: Generalizing transformers for graph-structured tasks. *arXiv preprint arXiv:2210.05062*, 2022.

[9] Max Barer, Guni Sharon, Roni Stern, and Ariel Felner. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In *Proceedings of the International Symposium on Combinatorial Search*, volume 5, pages 19–27, 2014.

[10] Jiaoyang Li, Ariel Felner, Eli Boyarski, Hang Ma, and Sven Koenig. Improved heuristics for multi-agent path finding with conflict-based search. In *IJCAI*, volume 2019, pages 442–449, 2019.

[11] Wolfgang Hönig, Scott Kiesel, Andrew Tinka, Joseph Durham, and Nora Ayanian. Conflict-based search with optimal task assignment. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, 2018.

[12] Zhongqiang Ren, Sivakumar Rathinam, and Howie Choset. Conflict-based steiner search for multi-agent combinatorial path finding. In *Proceedings of Robotics: Science and Systems*, 2022.

[13] Merrill M Flood. The traveling-salesman problem. *Operations research*, 4(1):61–75, 1956.

[14] Paolo Toth and Daniele Vigo. An overview of vehicle routing problems. *The vehicle routing problem*, pages 1–26, 2002.

[15] Michael Polacek, Karl F Doerner, Richard F Hartl, and Vittorio Maniezzo. A variable neighborhood search for the capacitated arc routing problem with intermediate facilities. *Journal of Heuristics*, 14:405–423, 2008.

[16] David Pisinger and Stefan Ropke. Large neighborhood search. *Handbook of metaheuristics*, pages 99–127, 2019.

[17] Wouter Kool, Herke Van Hoof, and Max Welling. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*, 2018.

[18] Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning. *Advances in Neural Information Processing Systems*, 33:21188–21198, 2020.

[19] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30, 2017.

[20] Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takác. Reinforcement learning for solving the vehicle routing problem. *Advances in neural information processing systems*, 31, 2018.

[21] Yuhong Cao, Zhanhong Sun, and Guillaume Sartoretti. Dan: Decentralized attention-based neural network for the minmax multiple traveling salesman problem. *arXiv preprint arXiv:2109.04205*, 2021.

APPENDIX

A. Multi-Agent Path Finding

Multi-Agent Path Finding addresses the challenge of finding collision-free paths for multiple robots in an environment with designated start and goal locations. MAPF algorithms primarily focus on constructing collision-free paths. Traditional MAPF solvers can be categorized into two main types: optimal and bounded-suboptimal. Optimal solvers, such as CBS [2] and its variants, aim to find optimal solutions but may incur heavy computational overhead in complex scenarios. Bounded-suboptimal solvers, including ECBS [9], EECBS [3], and WDG [10], provide suboptimal yet reasonably practical solutions, offering computational advantages.

B. Multi-Agent Scheduling and Path Finding

Multi-Agent Scheduling and Path Finding (MASPF) problem is the primary focus of this research. While the previously described MAPF problem concentrated on finding collision-free paths within a predetermined schedule, the MASPF problem concurrently addresses task scheduling for multiple agents and the establishment of collision-free paths. Not many studies have tackled problems that consider both scheduling and pathfinding together, but there have been a few attempts in some research studies. CBS-TA [11] generates a search forest consisting of task-assignment sets and explore the search forest with tree search. Its scheduling performance relies on heuristic factors. CBSS [12] construct a high-level search forest of task assignments and utilize a TSP solver to traverse the search forest through Steiner search to find the optimal solution. Its performance heavily relies on heuristic techniques also. LNS-PBS considers the MASPF problem as the Multi-Goal Multi-Agent Pickup and Delivery (MG-MAPD) problem. Among current studies addressing the MASPF problem, LNS-PBS stands as one of the most scalable and computationally powerful algorithms, resolving the problem through a combination of Large-Neighborhood Search-based scheduling and MAPF solver. LNS-PBS is influenced by search power of the LNS algorithm.

C. Combinatorial Optimization

Combinatorial Optimization (CO) is a field of mathematics and computer science that focuses on finding the best solution from a finite set of possible solutions for problems characterized by combinatorial elements. In this context, our focus is on the application of combinatorial optimization in path planning and scheduling. Some well-known examples of combinatorial optimization problems include the Traveling Salesman Problem (TSP) [13] and the Vehicle Routing Problem (VRP) [14]. Several approaches employ improving methods to tackle these planning problems. Cross-Exchange [15] entails the exchange of edges between two distinct planned schedules, while Large Neighborhood Search [16] initiates with an arbitrary schedule as an initial solution and iteratively enhances it through schedule modifications. The algorithm we propose also performs scheduling based on the LNS method. Due to the nature of the LNS algorithm, improving the schedule requires repeatedly verifying the path lengths of numerous multi-agent schedules using a MAPF solver. Overcoming this computationally expensive drawback by harnessing the predictive performance of the graph neural network is the primary contribution of this research.

D. Neural Combinatorial Optimization

Neural Combinatorial Optimization (NCO) harnesses the capabilities of neural networks to tackle combinatorial optimization problems with increased efficiency and scalability. It primarily employs two approaches: constructive methods that gradually construct feasible solutions and improving methods that iteratively enhance initial solutions through local search processes. Previous research has successfully integrated neural networks into improving methods' strategies [17], [18], [19], [20], [21], leading to improved combinatorial optimization problem-solving capabilities. Inspired by these achievements, we employed a graph neural network structure to enhance the performance of LNS-PBS, thereby addressing multi-robot task scheduling with collision-free path requirements more effectively.

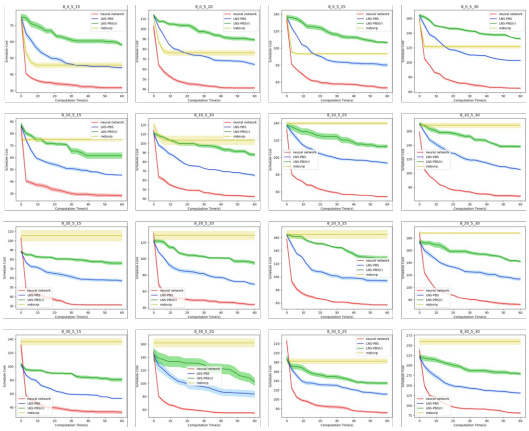

Fig. 6: High-level scheduling performance of **Closest-first** operator.

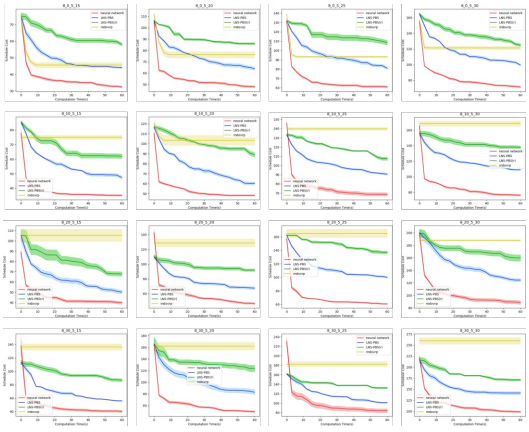

Fig. 7: High-level scheduling performance of **Farthest-first** operator.