

Multi-Agent Path Finding via Offline RL and LLM Collaboration

Merve Atasever, Matthew Hong, Mihir Nitin Kulkarni,

Qingpei Li, Jyotirmoy V. Deshmukh

Department of Computer Science, University of Southern California

Abstract

Multi-Agent Path Finding (MAPF) poses a significant and challenging problem critical for applications in robotics and logistics, particularly due to its combinatorial complexity and the partial observability inherent in realistic environments. Decentralized reinforcement learning methods commonly encounter two substantial difficulties: first, they often yield self-centered behaviors among agents, resulting in frequent collisions, and second, their reliance on complex communication modules leads to prolonged training times, sometimes spanning weeks. To address these challenges, we propose an efficient decentralized planning framework based on the Decision Transformer (DT), uniquely leveraging offline reinforcement learning to substantially reduce training durations from weeks to mere hours. Crucially, our approach effectively handles long-horizon credit assignment and significantly improves performance in scenarios with sparse and delayed rewards. Furthermore, to overcome adaptability limitations inherent in standard RL methods under dynamic environmental changes, we integrate a large language model (GPT-4o) to dynamically guide agent policies. Extensive experiments in both static and dynamically changing environments demonstrate that our DT-based approach, augmented briefly by GPT-4o, significantly enhances adaptability and performance.

Introduction

Multi-Agent Path Finding (MAPF) addresses the fundamental challenge of guiding multiple autonomous agents to their respective goals without collisions, a problem widely recognized for its computational complexity and practical importance in logistics, warehousing, and robotic swarm management. While centralized approaches offer solutions using global planning algorithms (e.g., CBS, M*, ODrM*, La-CAM), they assume full observability of an environment and lack robustness to real-time dynamic changes (Sharon et al. 2015; Hart, Nilsson, and Raphael 1968; Wagner and Choset 2011; Ferner, Wagner, and Choset 2013; Okumura 2023). This limitation severely restricts their applicability to realistic, partially observable environments where agents can only perceive their immediate surroundings.

In response, decentralized Multi-Agent Reinforcement Learning (MARL) methods have been developed to empower agents to independently navigate through local observations (Egorov and Shpilman 2022; Rashid et al. 2020; Sunehag et al. 2017; Tan 1993; Wang et al. 2020). However,

existing decentralized MARL approaches encounter significant difficulties: agents frequently develop self-centered behaviors leading to increased collisions, and the integration of complex inter-agent communication modules results in substantial computational overhead, which dramatically extends training periods to days or even weeks.

To overcome these persistent challenges, we propose a fundamentally different decentralized approach by leveraging an offline reinforcement learning setup. The Decision Transformer (DT) architecture inherently excels in modeling long-horizon dependencies, effectively addressing the credit assignment problem inherent in MAPF scenarios characterized by sparse and delayed rewards. By harnessing offline data, our DT-based agents drastically reduce training time requirements from weeks to a matter of hours without compromising performance.

Moreover, standard RL-based methods often falter when the environment undergoes dynamic alterations, which leads to agent indecision, oscillatory behaviors, or failure to adapt promptly. To address this critical issue, we pioneer the integration of a Large Language Model (LLM), specifically GPT-4o, to dynamically guide and adjust agent behaviors when encountering environmental changes. Unlike other communication-heavy methods, our integration of GPT-4o is concise and intermittent, providing short bursts of global awareness and strategic adaptability, which significantly improves agent performance in dynamic conditions (Achiam et al. 2023).

We rigorously validate our proposed framework across diverse experimental setups, including static scenarios and dynamically altered environments. In dynamic conditions, GPT-4o intervention ensures rapid real-time adjustments, directly navigating agents towards newly designated goals, thereby circumventing inefficient exploratory behaviors exhibited by purely DT-based agents. Our approach thus demonstrates a balanced and effective synergy between the fast, locally-informed policies of DT agents and the adaptive, globally-aware guidance provided by GPT-4o.

The contributions of our paper are:

- A novel decentralized offline reinforcement learning approach employing Decision Transformer to solve MAPF efficiently, significantly reducing training time from weeks to mere hours while maintaining robust performance.

- Effective management of the credit assignment challenge in long-horizon MAPF tasks, specifically addressing scenarios with delayed positive rewards at the end.
- The pioneering integration of GPT-4o for dynamic adaptation in MAPF, enabling significant performance improvements in environments subject to real-time changes.
- Comprehensive experimental validation in both static and dynamic scenarios, clearly demonstrating the advantages and practicality of our DT+LLM approach for responsive and adaptive multi-agent systems.

Related Work

A prominent benchmark is an approach called PRIMAL, where imitation learning (IL) is combined with reinforcement learning (RL), allowing agents to *imitate* centralized planner behaviors while being trained in a decentralized manner (Sartoretti et al. 2019). Prioritized Communication Learning method (PICO) and PRIMAL₂ are extensions of PRIMAL; whereas the first incorporates planning priorities and communication topologies into its learning pipeline to improve collision avoidance and boost collaborative behavior, the second works on lifelong version the problem (Li et al. 2022; Damani et al. 2021). On the other hand, Distributed Heuristic Learning with Communication (DHC) tries to achieve this objective by employing graph convolution for agent cooperation, (Ma, Luo, and Ma 2021). Here, each agent operates independently with heuristic guidance provided through potential shortest-path choices. Decision Causal Communication (DCC) enhances DHC by focusing on selective communication. Unlike other methods, DCC enables agents to choose relevant neighbors for communication, minimizing redundancy and overhead (Ma, Luo, and Pan 2021). SCRIMP introduces a differentiable transformer-based communication mechanism, which addresses the challenges posed by partial observations and enhances team-level cooperation (Wang et al. 2023). Finally, Confidence-based Auto-Curriculum for Team Update Stability (CACTUS) proposes a reverse curriculum approach that incrementally increases the potential distance between start and goal locations to learn effective policy (Phan et al. 2024).

On the other hand, LLMs have demonstrated the ability to understand and execute instructions for control and embodied tasks in robotics (Szot et al. 2023; Yu and Mooney 2022; Liang et al. 2023; Hu et al. 2023; Ahn et al. 2022), coding (Liu et al. 2023b; Chen et al. 2021b), strategic planning (Wu et al. 2023; Liu et al. 2023a), spatial reasoning (Wu et al. 2023), and even planning multi-agent collaboration tasks (Li et al. 2023; Zhang et al. 2023; Talebirad and Nadiri 2023). Interestingly, the direct application of LLM-based agents does not yield comparable results to current methods and underperforms the DT-based agents (Chen, Koenig, and Dilkina 2024).

Our experimental setup involves two scenarios: one with static environments and one with altered environments. In the first case, DT agents are given a fixed amount of time to finish the episode. GPT-4o is used if at least one of the DT

agents fails to complete the task within the allocated time. In the second scenario, we change the goal positions of some agents during the inference and include GPT-4o for only five timesteps after introducing the change in the environment, then we switch the agent policy back to the DT-based policy. Here, we remark that integrating an LLM alters the partial observability setting to full observability (for five timesteps) in dynamically changing environments. This framework allows for modifications and real-time adjustments, enhancing the adaptability of the agents as illustrated in Figure 2.

Preliminaries

Problem Setting

Our research focuses on a deterministic and partially observable environment where a team of agents operates in a grid world to complete given tasks. Each agent is assigned to move from a start point to an endpoint and can either move to neighboring cells or remain stationary. The goal is achieved by all agents when they complete their tasks while minimizing the total time taken and avoiding collisions with obstacles or other agents.

Multi-Agent Path Finding

Our setup constitutes a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) which is a framework used in multi-agent systems where multiple agents must make decisions based on their individual observations of the environment (Omidshafiei et al. 2017). The Dec-POMDP is defined by the tuple;

$$(I, S, A, T, \Omega, O, R, \gamma)$$

consisting of I : the set of agents, S : the set of states, $A = \times_{i \in I} A_i$: the set of joint actions, where A_i is the set of actions available to agent i , $T(s' | s, a)$: the state transition function that gives the probability of transitioning to state s' from state s after joint action a , $\Omega = \times_{i \in I} \Omega_i$: the set of joint observations, where Ω_i is the set of local observations of agent i , $O(o | s', a)$: the observation function that gives the probability of joint observation o given state s' and joint action a , $R(s, a)$: the reward function that gives the immediate reward received after taking joint action a in state s , and γ : the discount factor that determines the importance of future rewards.

In a shared environment, each agent behaves according to its policy $\pi^i(a^i | s) = P(A_t^i = a^i | S_t = s)$, which is the probability distribution over actions given states. At a time step t , the joint action $\vec{a}(t) = (a_1^1, \dots, a_t^N)$, where $N = |I|$, leads to a transition to a new state s_{t+1} according to the transition function T , and each agent receives a reward r_t^i according to the reward function R . We consider a finite system where each episode plays out for a given T timesteps.

Additionally, we define G to be the set of goal states $G \subseteq S$ and say that an agent i has reached its goal if $s_t^i \in G$ for some $t \leq T$. We consider an episode to terminate when all agents have reached their goals or at timestep T , whichever is sooner.

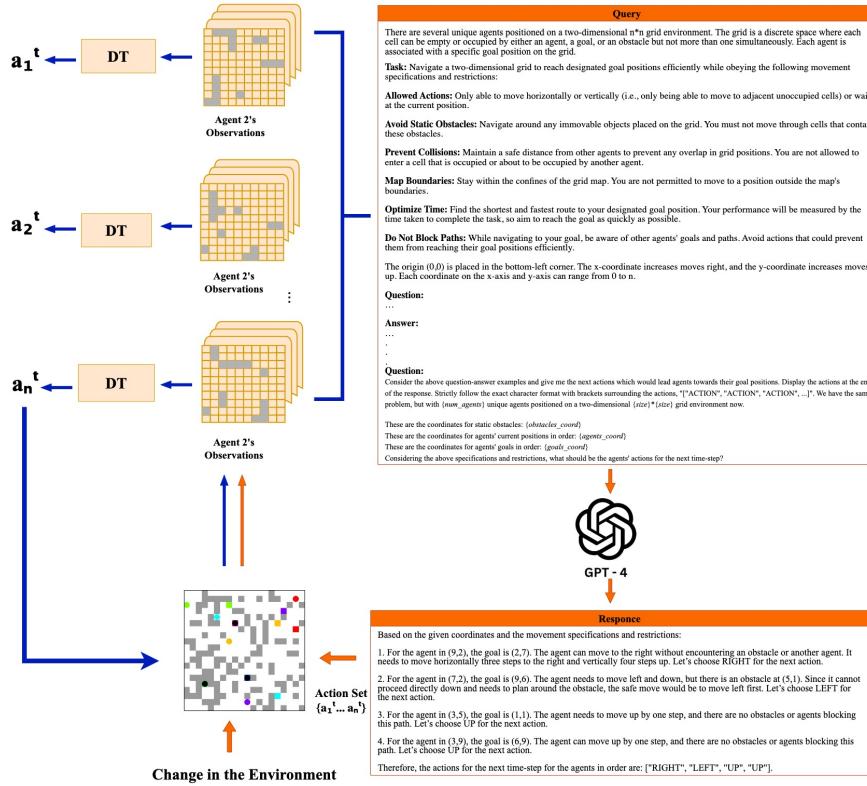


Figure 1: Architecture of our pipeline: blue and orange arrows represent the processes of DT and GPT-4o, respectively.

Table 1: Effort relative to other benchmarks in terms of training time, training episodes, and the number of parameters.

Benchmark	T. Time	T. Episodes	# of Params
PRIMAL	≈ 20 days	3.8M	13M
DCC	≈ 1 day	128K	1M
DT	≈ 3 hours	133K	1.3M

Decision Transformer

The Decision Transformer treats offline RL as a sequence modeling problem and learns a return-conditioned policy from an offline dataset (Chen et al. 2021a). It has provided a novel perspective to reinforcement learning, and several extensions of this concept have been introduced subsequently (Zheng, Zhang, and Grover 2022; Lee et al. 2022). In the architecture, embedded tokens of returns, states, and actions are fed into a decoder-only transformer to generate the next tokens autoregressively using a causal self-attention mask. In other words, the model learns the probability of the next token x_t conditioned on previous K tokens, $P_\theta(x_t|x_{t-K} < \dots < t)$, where K is a hyperparameter called *context length*. To achieve this, we consider sequences of the form:

$$\tau^i = (x_1^i, \dots, x_t^i, \dots, x_T^i) \quad \text{where} \quad x_t^i = (\hat{R}_t^i, o_t^i, a_t^i)$$

such that \hat{R}_t^i is *return-to-go* (rtg) representing the cumu-

lative rewards from the current time step until the end of the episode, o_t^i is the observation, and a_t^i is the action of agent i at time t .

We choose DT as the backbone for our method for three major reasons:

- It is an offline RL algorithm that significantly reduces training time, as it does not require online interaction with the environment during training.
- The transformer architecture effectively addresses the credit assignment problem in long-horizon MAPF scenarios with positive rewards given only at the end.
- DT performs well without extensive reward engineering by conditioning on the desired return.

Large Language Models

Building on the introduction of transformers in 2017, significant advancements in language models such as BERT, T5, the GPT series, Llama, and PaLM have extended the capabilities of LLMs and enabled their application to increasingly sophisticated tasks (Vaswani et al. 2017; Devlin et al. 2018; Raffel et al. 2020; Radford et al. 2018; Achiam et al. 2023; Touvron et al. 2023; Chowdhery et al. 2023). Notably, these models function as few-shot learners for downstream tasks through prompt engineering without requiring further training (Chang et al. 2024).

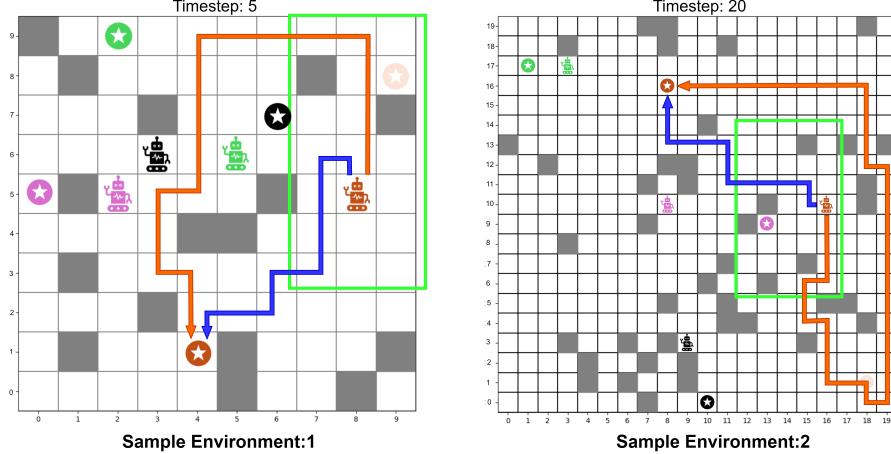


Figure 2: Illustration of GPT-4o’s assistance in the event of a goal change. In the first environment, the orange agent initially has its goal at (9,8), which is changed to (4,1) in the 5th time step. In the second environment, the orange agent’s goal is initially at (18,1), but is altered to (8,16) in the 20th time step. The orange arrows depict the path generated by the DT alone, while the blue arrows represent the path taken when decision-making is switched to GPT-4o for five time steps after the goal change, before returning to DT. Green rectangles highlight the five time steps during which GPT-4o and DT make decisions. Without GPT-4o’s assistance, DT agents initially explore the region around the previous goal before navigating to the new one. With GPT-4o’s assistance, however, the agents can directly navigate toward the new goal.

In-Context Learning: (ICL) is an approach utilized by LLMs to handle downstream tasks by conditioning on relevant input-output examples or demonstrations (Brown et al. 2020; Dong et al. 2022). Consequently, a pre-trained LLM model can tackle a wide range of tasks, from translation to question-answering, simply by modifying the examples in the prompt. This flexibility renders ICL a powerful tool for leveraging LLMs in new tasks.

Chain of Thought: (CoT) is an advanced demonstration design technique for prompt engineering used with LLMs to enhance their problem-solving abilities, particularly for tasks that require complex reasoning by structuring prompts to guide models through a step-by-step reasoning pathway (Wei et al. 2022). Although several advancements in prompt engineering have emerged such as SayCan, ReAct, ToT, and other variations of CoT, we have chosen to utilize CoT in our work due to its simplicity and effectiveness (Ahn et al. 2022; Yao et al. 2022, 2024; Wang et al. 2022).

Method

We first create an offline dataset consisting of expert-level trajectories utilized to train the Decision Transformer. The trained DT model is deployed to each agent to generate the next action at each time step. The DT-based agents then navigate towards their goals within the test environment for predetermined time steps. Subsequently, GPT-4o is integrated into the pipeline to assist agents that have yet to reach their goals. Environmental information, including the coordinates of static obstacles, the agents’ current positions, and their target destinations, is encoded into a query-prompt. The prompt is then provided to GPT-4o which generates the

next set of actions for one time-step. This iterative process continues for five time steps, after which the pipeline reverts to the DT policies to complete the episode. Detailed descriptions of the methodology and experimental results are presented in the following sections. The overall architecture of our pipeline is depicted in Figure 1.

Building Expert Trajectories & Training

To generate **expert-level** behavior, we collected trajectories using the ODrM* algorithm, a centralized classical MAPF solver frequently employed in the literature to create expert trajectories for imitation learning. The algorithm was executed on 80 randomly generated grid environments for each combination of varying parameters: (4, 16, 32, 64) agents, grid sizes of (10, 20, 40, 80), and obstacle densities of (0, 0.1, 0.2).

Each agent’s path constitutes a distinct trajectory in the training dataset. Given the paper’s emphasis on partially observable environments, agents are constrained to observing only their own fields of view (FOV), each of size 10 × 10. Observations are represented by four 2-dimensional arrays of shape (10, 10), encoding the following information about their local environments: positions of neighboring agents within the agent’s FOV, position of the agent’s own goal, positions of neighbors’ goals within the agent’s FOV, positions of obstacles within the agent’s FOV. Here, if the agent’s goal lies outside its field of view, the goal is projected onto the edge cell closest to it. If the goal falls within the agent’s FOV, it is displayed in the corresponding cell.

At each time step, agents have the option to either wait or move in one of four directions (N/E/S/W) while receiving rewards as outlined: -0.3 for moving, (0/ - 0.5) for wait-

Query Prompt
<p>There are several unique agents positioned on a two-dimensional $n \times n$ grid environment. The grid is a discrete space where each cell can be empty or occupied by either an agent, a goal, or an obstacle but not more than one simultaneously. Each agent is associated with a specific goal position on the grid.</p> <p>Task: Navigate a two-dimensional grid to reach designated goal positions efficiently while obeying the following movement specifications and restrictions:</p> <p>Allowed Actions: Only able to move horizontally or vertically (i.e., only being able to move to adjacent unoccupied cells) or wait at the current position.</p> <p>Avoid Static Obstacles: Navigate around any immovable objects placed on the grid. You must not move through cells that contain these obstacles.</p> <p>Prevent Collisions: Maintain a safe distance from other agents to prevent any overlap in grid positions. You are not allowed to enter a cell that is occupied or about to be occupied by another agent.</p> <p>Map Boundaries: Stay within the confines of the grid map. You are not permitted to move to a position outside the map's boundaries.</p> <p>Optimize Time: Find the shortest and fastest route to your designated goal position. Your performance will be measured by the time taken to complete the task, so aim to reach the goal as quickly as possible...</p> <p>Do Not Block Paths: While navigating to your goal, be aware of other agents' goals and paths. Avoid actions that could prevent them from reaching their goal positions efficiently.</p> <p>The origin (0,0) is placed in the bottom-left corner. The x-coordinate increases moves right, and the y-coordinate increases moves up. Each coordinate on the x-axis and y-axis can range from 0 to n.</p> <p>Question:...</p> <p>Answer:...</p> <hr/> <p>Question: Consider the above question-answer examples and give me the next actions which would lead agents towards their goal positions. Display the actions at the end of the response. Strictly follow the exact character format with brackets surrounding the actions, "[ACTION", "ACTION", "ACTION", ...]". We have the same problem, but with $\{num_agents\}$ unique agents positioned on a two-dimensional $\{size\} \times \{size\}$ grid environment now.</p> <p>These are the coordinates for static obstacles: $\{obstacles_coord\}$ These are the coordinates for agents' current positions in order: $\{agents_coord\}$ These are the coordinates for agents' goals in order: $\{goals_coord\}$ Considering the above specifications and restrictions, what should be the agents' actions for the next time-step?</p>

Figure 3: In the query prompt, the MAPF problem, environment, and constraints are first outlined, followed by task-specific in-context examples provided using Chain-of-Thought (CoT).

ing (on/off goal), -5 for collision, and $+20$ for reaching the goal. We also created a modified version of the dataset in which agents receive an extra reward of $+20$ upon successfully completing an episode, (i.e. all agents reach respective goals). However, the DT model trained on this modified dataset demonstrated inferior performance compared to the model trained on the original version.

Once the trajectories are collected, the dataset undergoes pre-processing to align with the input format required by the DT model. With the context length for the DT set to 50, we divided the trajectories into chunks of length 50. For chunks shorter than the specified context length, we applied zero-padding. The final dataset, composed of these chunks, was derived from a total of 133K episodes. For the most part, we retained the original architecture of the Decision Transformer except for replacing the linear layer with a convolutional encoder to process observations of shape (4, 10, 10).

Prompting GPT-4

We started with an initial prompt and iteratively refined it by incorporating corrective feedback from GPT-4o itself. While we experimented with prompt generation tools like AdalFlow¹, we found that our custom-designed prompts performed better overall. The prompt design that yielded the best performance has been integrated into our pipeline; the

prompt template is illustrated in Figure 3.

In the query prompt, we begin by outlining the problem, environment specifications, constraints, and the task. Following this, we provide task-specific in-context examples and pose a similar question generated by our pipeline. To construct these in-context examples, we analyzed environments where the DT model failed to find a path to a goal for at least one agent within T timesteps. Using both a simple and a challenging (failed) scenario, we curated sample question-answer pairs.

Experimental Results

Model training and experiments were performed on an NVIDIA Quadro RTX 5000 with 16 GB GPU memory.

Stationary Environments

The first part of the experiments are carried out by using *Random* and *Mazes* maps of the unified framework, Pogema, to be able to compare our method with other learning-based MAPF benchmarks (Skrynnik et al. 2022; Skrynnik et al.). The results in Figure-4 and Figure-5 are obtained by running tests on the maps consist of $\{17, 21\}$ -size grid environments with varying numbers of agents $\{8, 16, 24, 32, 48, 64\}$. The performance of the benchmarks is evaluated using two metrics: sum-of-costs (SoC, i.e. the sum of time steps across all agents to reach their respective goals) and cooperative success rate (CSR).

We draw the following conclusions from our experiments:

¹<https://github.com/SylphAI-Inc/AdalFlow>

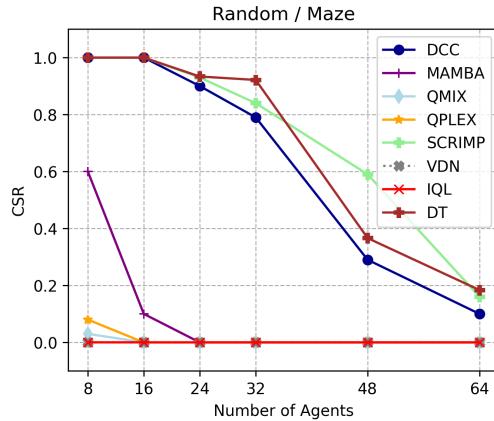


Figure 4: Performance of MAPF methods on Random and Mazes maps; higher CSR is better.

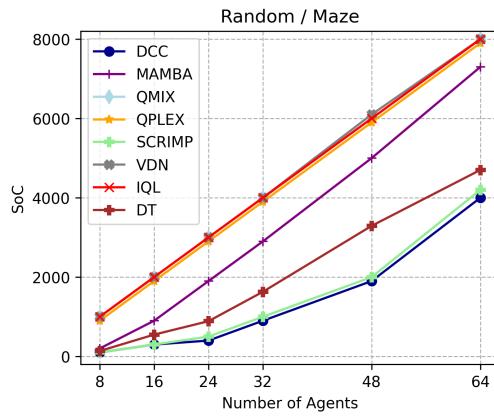


Figure 5: Performance of MAPF methods on Random and Mazes maps; lower SoC is better.

1. DT-based agents consistently outperform all baseline methods with higher success rates across all settings, except for the case with 48 agents. Notably, increasing the agent count to 64 yields the best overall performance.
2. SCRIMP demonstrates strong performance in Pogema’s environments; however, the original paper does not report results for larger environments. The published findings are limited to environment sizes of 10, 30, and 40, indicating that SCRIMP may be optimized primarily for smaller-scale settings.
3. DT agents tend to take slightly longer paths compared to DCC and SCRIMP, yet they achieve a lower sum of costs, reflecting more globally efficient navigation behaviors.

Dynamic Scenario Adaptation

The second part of our experiments are carried out in randomly generated $\{20, 40, 80\}$ -sized grid environments with varying numbers of agents $\{8, 16, 32, 64\}$, and obstacle densities $\{0, 0.1, 0.2\}$. The result, summarized in the Table 2, represent averages across 3 different obstacle density val-

ues and 60 environments (combinations of grid size, number of agents), resulting in a total of 720 test environments. We evaluated the performance of the benchmarks and our models using three key metrics: success rate (CSR or SR), makespan (MS), and collision rate (CR). The makespan refers to the duration of an episode, specifically the time taken for the last agent to reach its designated goal. Finally, the collision rate is computed as the number of collisions among agents in a successful episode divided by the episode’s duration. Collisions occurring in unsuccessful episodes are excluded from this calculation.

Based on the results presented in Table 2, we can draw the following conclusions:

1. The integration of GPT-4o reduces makespan across most environmental settings. This reduction is particularly significant when the new goal location is in the opposite direction to the agents’ prior trajectory. As illustrated in Figure-2, DT requires several timesteps to comprehend the goal change, often exploring areas near the previous goal location before adjusting its direction. In contrast, when the LLM-based suggestions are introduced concurrently with the dynamic goal change, agents immediately reorient towards their new goal location. This significantly reduces the makespan.
2. The success rates achieved by DT and LLM collaboration are equal to or surpass those of the DT alone in most environmental settings. Notably, when we alter the goal positions of half of the agents, the advantages of LLM guidance become particularly evident in complex environments characterized by larger sizes and a greater number of agents.
3. DT & GPT-4 collaboration surpasses DT in terms of collision rates, indicating that LLM integration results in safer behavior. This is particularly significant since collisions among agents in real-world scenarios, such as warehouses, can lead to substantial costs and safety hazards.

These findings demonstrate that LLM-assisted DT-based agents are highly effective for real-time adaptations and offer significant advantages in safety-critical environments.

Discussion

Offline vs. Online RL Approaches in MAPF We show that incorporating offline RL in MAPF (through the DT architecture) is an effective learning strategy that yields performance comparable to other learning-based methods that require online interaction with the environment during training. Table 1 highlights the significant reduction in required effort and our success in eliminating the necessity for real-time interaction with environments during training. Notably, our model does not experience the distributional shift issues that challenge offline RL algorithms when tested in new environments. By utilizing a dataset consisting of a broad range of samples from randomly generated grid environments, we mitigate the risk of distributional shift (analogous to (Tobin et al. 2017)).

Table 2: Impact of GPT-4o on dynamic environments: For 20, 40, 80 size environments, we modify the environment once at 15th, 30th, and 50th timesteps respectively and conduct our experiments according to two difficulty levels; altering the goals of .25 of the agents and .5 of the agents in the environments during inference. For makespan and collision rate, lower values (indicated by arrows) means better performance.

Env Size	# Agents	DT (1/4)			DT+GPT-4o (1/4)			DT (1/2)			DT+GPT-4o (1/2)		
		MS ↓	SR(%)	CR ↓	MS ↓	SR (%)	CR ↓	MS ↓	SR (%)	CR ↓	MS ↓	SR (%)	CR ↓
20x20	8	48.5	95	0.21	44.7	97	0.11	48.9	96	0.14	52.8	97	0.08
	16	68.5	85	0.48	68.1	88	0.51	71.5	85	0.49	74.2	95	0.37
	32	100.1	73	1.68	101.1	71	1.64	112.1	78	1.68	102.2	75	1.55
40x40	8	80.9	98	0.02	77.0	100	0.02	92.4	96	0.04	91.8	100	0.03
	16	97.2	98	0.15	92.4	98	0.14	104.4	92	0.13	97.1	98	0.12
	32	126.4	82	0.67	125.5	82	0.50	120.2	75	0.48	120.2	82	0.54
	64	162.9	60	2.03	150.7	61	1.82	172.7	60	2.18	161.5	64	1.82
80x80	8	133.0	94	0.01	128.5	94	0.00	147.9	93	0.02	145.4	92	0.01
	16	144.4	92	0.03	141.0	97	0.03	146.0	90	0.04	150.1	94	0.07
	32	155.5	82	0.18	158.0	81	0.17	171.7	78	0.16	166.3	85	0.24
	64	182.4	66	0.59	176.3	62	0.72	169.8	63	0.70	183.0	62	0.53

Conclusion

Our flexible framework introduces offline RL to MAPF into the literature and accommodates both static and dynamic environments, rapidly adapting to changes such as shifting goals. Although LLMs can occasionally hallucinate, i.e., yield outputs that deviate from factual accuracy or contextual relevance, our work seeks to harness the capabilities of LLMs within MAPF and points out contexts wherein the utilization of the models addresses specific challenges (Kambhampati 2024).

Limitations & Future Work In this paper, we opted for textual inputs because LLMs are still largely unexplored within the MAPF literature. Replicating this approach using visual inputs and Visual Language Models (VLMs) presents a promising direction for future research. Given the rapid developments, we believe that the integration of LLMs into MAPF methods is promising.

References

- Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Ahn, M.; Brohan, A.; Brown, N.; Chebotar, Y.; Cortes, O.; David, B.; Finn, C.; Fu, C.; Gopalakrishnan, K.; Hausman, K.; et al. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.
- Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.
- Chang, Y.; Wang, X.; Wang, J.; Wu, Y.; Yang, L.; Zhu, K.; Chen, H.; Yi, X.; Wang, C.; Wang, Y.; et al. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3): 1–45.
- Chen, L.; Lu, K.; Rajeswaran, A.; Lee, K.; Grover, A.; Laskin, M.; Abbeel, P.; Srinivas, A.; and Mordatch, I. 2021a. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34: 15084–15097.
- Chen, M.; Tworek, J.; Jun, H.; Yuan, Q.; Pinto, H. P. d. O.; Kaplan, J.; Edwards, H.; Burda, Y.; Joseph, N.; Brockman, G.; et al. 2021b. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- Chen, W.; Koenig, S.; and Dilkina, B. 2024. Why Solving Multi-agent Path Finding with Large Language Model has not Succeeded Yet.
- Chowdhery, A.; Narang, S.; Devlin, J.; Bosma, M.; Mishra, G.; Roberts, A.; Barham, P.; Chung, H. W.; Sutton, C.; Gehrmann, S.; et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240): 1–113.
- Damani, M.; Luo, Z.; Wenzel, E.; and Sartoretti, G. 2021. PRIMAL _2: Pathfinding via reinforcement and imitation multi-agent learning-lifelong. *IEEE Robotics and Automation Letters*, 6(2): 2666–2673.
- Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Dong, Q.; Li, L.; Dai, D.; Zheng, C.; Wu, Z.; Chang, B.; Sun, X.; Xu, J.; and Sui, Z. 2022. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*.
- Egorov, V.; and Shpilman, A. 2022. Scalable multi-agent model-based reinforcement learning. *arXiv preprint arXiv:2205.15023*.
- Ferner, C.; Wagner, G.; and Choset, H. 2013. ODrM* optimal multirobot path planning in low dimensional search spaces. In *2013 IEEE International Conference on Robotics and Automation*, 3854–3859. Karlsruhe, Germany.
- Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost

- paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2): 100–107.
- Hu, B.; Zhao, C.; Zhang, P.; Zhou, Z.; Yang, Y.; Xu, Z.; and Liu, B. 2023. Enabling intelligent interactions between an agent and an LLM: A reinforcement learning approach. *arXiv preprint arXiv:2306.03604*.
- Kambhampati, S. 2024. Can large language models reason and plan? *Annals of the New York Academy of Sciences*, 1534(1): 15–18.
- Lee, K.-H.; Nachum, O.; Yang, M. S.; Lee, L.; Freeman, D.; Guadarrama, S.; Fischer, I.; Xu, W.; Jang, E.; Michalewski, H.; et al. 2022. Multi-game decision transformers. *Advances in Neural Information Processing Systems*, 35: 27921–27936.
- Li, H.; Chong, Y. Q.; Stepputtis, S.; Campbell, J.; Hughes, D.; Lewis, M.; and Sycara, K. 2023. Theory of mind for multi-agent collaboration via large language models. *arXiv preprint arXiv:2310.10701*.
- Li, W.; Chen, H.; Jin, B.; Tan, W.; Zha, H.; and Wang, X. 2022. Multi-Agent Path Finding with Prioritized Communication Learning. *arXiv:2202.03634*.
- Liang, J.; Huang, W.; Xia, F.; Xu, P.; Hausman, K.; Ichter, B.; Florence, P.; and Zeng, A. 2023. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 9493–9500. IEEE.
- Liu, B.; Jiang, Y.; Zhang, X.; Liu, Q.; Zhang, S.; Biswas, J.; and Stone, P. 2023a. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*.
- Liu, X.; Yu, H.; Zhang, H.; Xu, Y.; Lei, X.; Lai, H.; Gu, Y.; Ding, H.; Men, K.; Yang, K.; et al. 2023b. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.
- Ma, Z.; Luo, Y.; and Ma, H. 2021. Distributed Heuristic Multi-Agent Path Finding with Communication. *arXiv:2106.11365*.
- Ma, Z.; Luo, Y.; and Pan, J. 2021. Learning Selective Communication for Multi-Agent Path Finding.
- Okumura, K. 2023. Lacam: Search-based algorithm for quick multi-agent pathfinding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 11655–11662.
- Omidshafiei, S.; Pazis, J.; Amato, C.; How, J. P.; and Vian, J. 2017. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *International Conference on Machine Learning*, 2681–2690. PMLR.
- Phan, T.; Driscoll, J.; Romberg, J.; and Koenig, S. 2024. Confidence-Based Curriculum Learning for Multi-Agent Path Finding. *arXiv preprint arXiv:2401.05860*.
- Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I.; et al. 2018. Improving language understanding by generative pre-training.
- Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140): 1–67.
- Rashid, T.; Samvelyan, M.; De Witt, C. S.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178): 1–51.
- Sartoretti, G.; Kerr, J.; Shi, Y.; Wagner, G.; Kumar, T. K. S.; Koenig, S.; and Choset, H. 2019. PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning. *IEEE Robotics and Automation Letters*, 2378–2385.
- Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 40–66.
- Skrynnik, A.; Andreychuk, A.; Borzilov, A.; Chernyavskiy, A.; Yakovlev, K.; and Panov, A. ???? Pogema: A benchmark platform for cooperative multi-agent navigation, 2024a. URL <https://arxiv.org/abs/2407.14931>.
- Skrynnik, A.; Andreychuk, A.; Yakovlev, K.; and Panov, A. I. 2022. POGEMA: partially observable grid environment for multiple agents. *arXiv preprint arXiv:2206.10944*.
- Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W. M.; Zambaldi, V.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J. Z.; Tuyls, K.; et al. 2017. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*.
- Szot, A.; Schwarzer, M.; Agrawal, H.; Mazoure, B.; Metcalf, R.; Talbott, W.; Mackraz, N.; Hjelm, R. D.; and Toshev, A. T. 2023. Large language models as generalizable policies for embodied tasks. In *The Twelfth International Conference on Learning Representations*.
- Talebirad, Y.; and Nadiri, A. 2023. Multi-agent collaboration: Harnessing the power of intelligent llm agents. *arXiv preprint arXiv:2306.03314*.
- Tan, M. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, 330–337.
- Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; and Abbeel, P. 2017. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. *CoRR*, abs/1703.06907.
- Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.
- Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.
- Wagner, G.; and Choset, H. 2011. M*: A complete multirobot path planning algorithm with performance bounds. In *2011 IEEE/RSJ international conference on intelligent robots and systems*, 3260–3267. IEEE.
- Wang, J.; Ren, Z.; Liu, T.; Yu, Y.; and Zhang, C. 2020. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062*.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; Narang, S.; Chowdhery, A.; and Zhou, D. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Wang, Y.; Xiang, B.; Huang, S.; and Sartoretti, G. 2023. SCRIMP: Scalable Communication for Reinforcement-and Imitation-Learning-Based Multi-Agent Pathfinding. *arXiv:2303.00605*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Wu, Y.; Tang, X.; Mitchell, T. M.; and Li, Y. 2023. Smartplay: A benchmark for llms as intelligent agents. *arXiv preprint arXiv:2310.01557*.

Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; and Narasimhan, K. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Yu, A.; and Mooney, R. J. 2022. Using both demonstrations and language instructions to efficiently learn robotic tasks. *arXiv preprint arXiv:2210.04476*.

Zhang, H.; Du, W.; Shan, J.; Zhou, Q.; Du, Y.; Tenenbaum, J. B.; Shu, T.; and Gan, C. 2023. Building cooperative embodied agents modularly with large language models. *arXiv preprint arXiv:2307.02485*.

Zheng, Q.; Zhang, A.; and Grover, A. 2022. Online decision transformer. In *international conference on machine learning*, 27042–27059. PMLR.