Safe Multi-Agent Navigation guided by Goal-Conditioned Safe Reinforcement Learning

Meng Feng*[1], Viraj Parimi*[1] and Brian Williams [1]

Abstract— **Safe navigation is essential for autonomous systems operating in hazardous environments. Traditional planning methods are effective for solving long-horizon tasks but depend on the availability of a graph representation with predefined distance metrics. In contrast, safe Reinforcement Learning (RL) is capable of learning complex behaviors without relying on manual heuristics but fails to solve long-horizon tasks, particularly in goal-conditioned and multi-agent scenarios.**

In this paper, we introduce a novel method that integrates the strengths of both planning and safe RL. Our method leverages goal-conditioned RL and safe RL to learn a goal-conditioned policy for navigation while concurrently estimating cumulative distance and safety levels using learned value functions via an automated self-training algorithm. By constructing a graph with states from the replay buffer, our method prunes unsafe edges and generates a waypoint-based plan that the agent then executes by following those waypoints sequentially until their goal locations are reached. This graph pruning and planning approach via the learned value functions allows our approach to flexibly balance the trade-off between faster and safer routes especially over extended horizons.

Utilizing this unified high-level graph and a shared low-level goal-conditioned safe RL policy, we extend this approach to address the multi-agent safe navigation problem. In particular, we leverage Conflict-Based Search (CBS) to create waypoint-based plans for multiple agents allowing for their safe navigation over extended horizons. This integration enhances the scalability of goal-conditioned safe RL in multi-agent scenarios, enabling efficient coordination among agents. Extensive benchmarking against state-of-the-art baselines demonstrates the effectiveness of our method in achieving distance goals safely for multiple agents in complex and hazardous environments. Our code will be released to support future research.

I. Introduction

In many real-world scenarios like search-and-rescue or environmental monitoring, autonomous agents must navigate to distant locations safely as repairs can be costly and time-consuming. This challenge becomes even more critical in multi-agent settings, such as drone swarms or robotic teams, where all agents must coordinate to reach their goals while managing risks throughout the entire mission. Ensuring safety in such complex environments is paramount, as failure to do so could compromise the entire operation. Recent advancements in Multi-Agent Path Finding (MAPF) [1], [2] demonstrate the ability of planners to efficiently generate path plans for multiple agents by decomposing a multi-agent problem into a series of single-agent problems. However, these approaches assume access to a structured graph representation of the environment, consisting of valid nodes and edges, annotated with utility information like safety levels and estimated travel times, limiting their applicability in scenarios where agents only have access to high-level observations, such as images.

In contrast, learning-based approaches such as deep reinforcement learning (RL) excel at learning state utilities from high-dimensional observations such as images. In particular, goal-conditioned RL [3], [4], [5] has been effective in enabling agents to achieve goals of interest in various domains, including navigation and manipulation. However, learning policies for distant goals remains challenging, as performance often degrades as the task horizon increases [6], [7] and requires significant reward shaping [8] or expert demonstrations [9], [10], which can limit the policy's ability to discover novel solutions. The challenge is compounded when agents must remain safe while navigating over long horizons. While constrained RL [11] has demonstrated the potential to incorporate safety constraints into policy learning, reaching a diverse set of distant goals under such constraints, particularly in multi-agent contexts, remains as an open research problem.

Prior research has tackled long-horizon problems using goal-conditioned RL combined with graph search techniques [12]. By constructing an intermediate graph representation of the environment from the replay buffer and using search-based methods to generate a waypoint-based plan, these methods decompose tasks into a series of shorter and easier goal-reaching tasks. However, existing approaches are limited to single-agent scenarios without considering any safety aspects. This paper extends these methods to address multi-agent, long-horizon, goal-conditioned tasks ensuring that the agents reach their goals safely.

Our approach leverages goal-conditioned RL and safe RL via a novel self-training algorithm to learn distance and safety predictors which allows us to dynamically construct an intermediate graph from the replay buffer. This enables greater planning flexibility that balances the trade-off between faster and safer routes depending on user-preference. By utilizing this intermediate graph representation, we extend the range of problems that MAPF approaches can solve, especially in challenging, image-based environments, effectively bridging the gap between planning and learning in multi-agent settings. Additionally, we empirically show that our approach can plan a series of safer sub-goals for multiple agents without requiring new training for each additional agent, thereby achieving scalability. In summary, we present

[1] Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 01239. Corresponding at {mfeng,vparimi,williams}@mit.edu. *These authors contributed equally to the paper.

This work was also supported by Defence Science and Technology Agency, Singapore

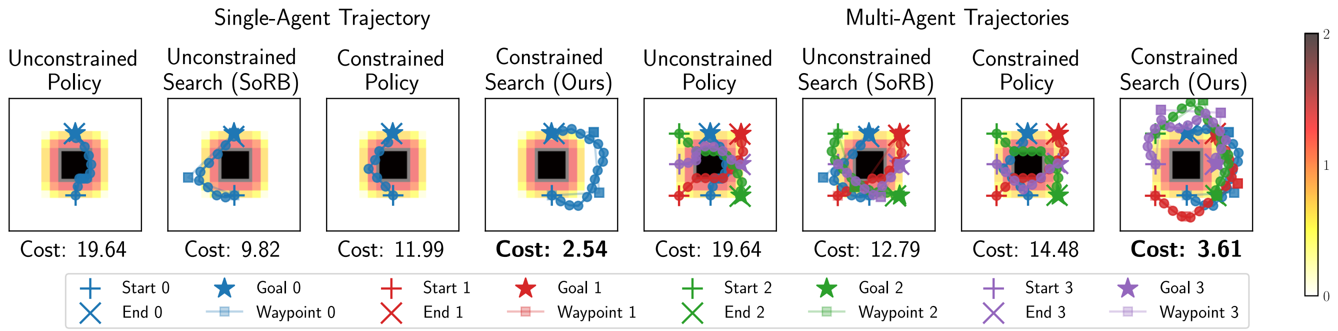

Fig. 1: Comparison of different approaches on the 2D navigation problem for a single agent (left) and four agents (right). Baselines tend to traverse high-cost regions, leading to suboptimal paths. In contrast, our approach finds a longer but lower-cost path by avoiding these regions. For both unconstrained and constrained policies, collision avoidance in the multi-agent case is ensured by having agents wait when another agent is nearby.

a scalable and safe approach for navigating multiple agents to their distance goals in sparse-reward environments, broadening the scope of MAPF through the integration of goal-conditioned safe RL.

To give an overview of the rest of the paper, first in Section II we define the relevant preliminary information to set the stage for our approach. In Section III, we describe our solution. Finally, Section IV presents our experimental results, where we compare our algorithm against state-of-the-art baselines.

II. Preliminaries

Goal-Conditioned Reinforcement Learning. In goal-conditioned RL (GCRL), an agent interacts with an environment modeled as an augmented Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s_{t+1}|s_t, a_t, s_g)$ is the transition probability, $R(s, a, s_g)$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. Unlike vanilla MDP, GCRL includes a goal $s_g \in \mathcal{S}$ in the problem formulation. The agent's objective is to learn a policy $\pi(a|s, s_g)$ that maximizes the expected cumulative reward, conditioned on both the current state s and the goal s_g. The reward function $R(s, a, s_g)$ is goal-dependent, reflecting how well the actions align with the desired goal. In practice, GCRL excels at learning short-horizon skills often with the help of reward shaping [8] and human demonstrations [9], [10] to enhance performance.

Bridge Planning and Single-Agent RL. To address long-horizon problems, [12] proposed Search on Replay Buffer (SoRB) that effectively integrates RL with single-agent planning. Specifically, SoRB defines $R(s, a, s_g) = -1$ and $\gamma = 1$ for the target environment. It then trains a goal-conditioned policy $\pi(a \mid s, s_g)$ and the associated value function $V(s, s_g)$ using standard off-policy RL algorithms [13]. Assuming that $\pi(a \mid s, s_g)$ is optimal, the value function $V(s, s_g)$ can be used to approximate the distance between a state s and a goal state s_g:

$$V(s, s_g) = -d_{\text{sp}}(s, s_g) \tag{1}$$

where d_{sp} is the shortest path distance from s to s_g, or equivalently, the expected number of steps to reach s_g from s under the optimal policy π.

SoRB then constructs a weighted, directed graph $\mathcal{G}$ from states that are either drawn from the replay buffer or randomly sampled form the environment state space. Each node in the graph corresponds to an observation and edges are added based on the predicted distance $V(s_1, s_2)$ between the node pairs (s_1, s_2). Node pairs $V(s_1, s_2) > \mathrm{d}_{\max}$ are excluded as they are considered potentially unreachable.

Finally, SoRB generates a waypoint-based plan by searching on this graph $\mathcal{G}$. The agent sequentially navigates to the individual waypoints before heading towards the final goal. This process is executed in either an open-loop or closed-loop manner. Notably, SoRB employs a greedy strategy and does not take the safety of the agent into account. However, when safety is factored in, longer and safer paths may be preferred over shorter yet riskier paths.

Constrained Reinforcement Learning . To capture the concept of risk and safety, we build our learning algorithm within the framework of Constrained RL (CRL). CRL solves a constrained Markov Decision Process (CMDP) [11] problem that augments MDP with extra constraints that restrict the set of allowable policies. CMDP defines an *auxiliary cost* function $C : S \times A \times S \rightarrow \mathbb{R}^+$ and a cost limit $\delta \in \mathbb{R}^+$. For conciseness, we will refer to auxiliary cost as *cost*. The cost function serves as a generic measure of safety. For example, regions of high environmental uncertainty may be deemed unsafe and induce higher costs. In this work, we do not require the cost function to carry any physical interpretation (e.g., uncertainty). Let $J_C(\pi) = \underset{\tau\sim\pi}{\mathbb{E}}\left[\sum_{t=0}^{\infty} \gamma^t C(s_t, a_t, s_{t+1})\right]$ denote the expected discounted *cost return* following policy π. The set of feasible stationary policies for a CMDP is then

$$\Pi_C \doteq \{\pi \in \Pi : J_C(\pi) \leq d\} \tag{2}$$

Standard off-policy methods can be adapted to solve CRL problems by including Lagrange multipliers to penalize unsafe actions that lead to the violation of cost constraints [14]. It is crucial to note that, in CRL, the cost function and cost limit is task-dependent and is not applicable to the goal-conditioned setting where each goal is akin to a distinct task. For a more detailed reference on CRL, we refer our readers to [15], [16], [17]. ,

Multi-Agent Path Finding (MAPF). Focusing on multi-agent settings, we explore the planning through the lens of a Multi-Agent Path Finding (MAPF) problem [1], [2]. A MAPF problem involves a weighted graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ and a set of agents $A = a_1, a_2, \ldots, a_k$, each with a unique start $s_i \in \mathcal{V}$ and a goal $g_i \in \mathcal{V}$. Agents can either wait or move to adjacent vertices along edges in $\mathcal{E}$, with each action incurring a penalty defined by $\mathcal{G}$. Once an agent reaches its goal, no further penalty is incurred.

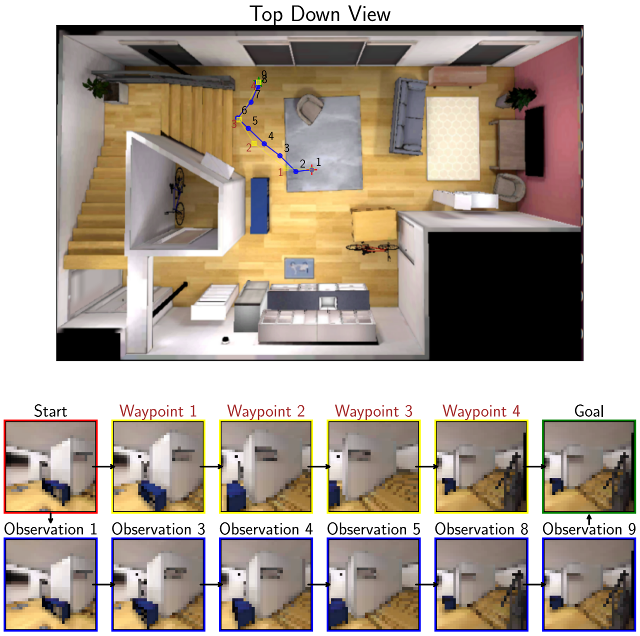

Fig. 2: **Safe Visual Navigation**: The top-down map illustrates the agent's path and waypoints according to our approach. The first row displays the forward-facing start, goal, and suggested waypoint images. Bottom row shows the agent's forward-facing observations during execution. The waypoints help the agent avoid the couch and blue furniture. Note that the top-down view is not available to the agent.

III. Approach

Goal-Conditioned Safe Reinforcement Learning. Inspired by SoRB [12], we would like to train an agent that can predict both distance and cost among state pairs. Meanwhile, the agent should also be capable of safely navigating amongst them. Since the value functions are conditioned on the associated policy, in order to obtain the distance and cost predictions following the shortest path, it is critical to first train an unconstrained agent that includes a goal-conditioned policy $\pi(s, a, s_g)$, a Q-function for cumulative reward $Q(s, a, s_g)$, and another Q-function for cumulative cost $Q_C(s, a, s_g)$. For the agent to learn safe navigation, we then fine-tune the unconstrained policy π by applying constrained RL methods which outputs a goal-conditioned safe policy $\pi_c(s, a, s_g)$ and its associated value functions. As previously noted, the cost limit from CRL is not directly applicable to the goal-conditioned setting. Therefore, we redefine the cost limit as a hyper-parameter, a soft constraint that penalizes unsafe actions without requiring strict compliance. However, we note that when the choice of cost limit is too low (penalty is too high), the resultant policy would focus too much on reducing costs rather than reaching the goals. In this work, we use the Lagrangian actor critic [14] thanks to its simplicity.

Self-Sampling and Training. To train a goal-conditioned agent, a set of goal states need to be sampled for the agent to explore and practice reaching them. The design of sampling strategy is crucial for successful training of a goal-conditioned agent. SoRB [12] accomplishes this by prioritizing nearby goal samples to improve the success rate of short-horizon navigation. However, it relies on an oracle with access to the ground-truth positions rather than the raw observations (e.g., images) to sample goals of desirable distances. This practice is unrealistic as such oracles often do not exist. Instead, we propose Algorithm 1, a self-sampling and training algorithm which, in essence, uses the agent's Q-functions to evaluate and select training samples with a diverse set of distances and cost targets. As the Q-functions become more accurate, the quality of the samples improve, which then generates better training samples to improve the policy.

Algorithm 1 Self-Sampling and Training of Goal-Conditioned Actor Critic

```
Inputs:  Environment E, Goal-Conditioned Policy $\pi$,
         Associated $Q$-functions for rewards or auxiliary
         costs $Q(s, a, s_g)$, Desirable sample target for
         rewards or costs $\tau$, Population size $N$,
         Number of training problems per batch $K$
Outputs: Updated $\pi(s, a, s_g)$, $Q(s, a, s_g)$
$\mathcal{P} \leftarrow$ Initialize an empty training set
for each batch do
    $\{(s_i, s_j)\}_N \leftarrow$ Randomly sample state pairs from E
    $\{v_{ij}\}_N \leftarrow Q(s_i, \pi(s_i, a, s_j), s_j)$
    $\{l_{ij}\}_N \leftarrow L^2$ distance to $\tau$
    Find $\{(s_i, s_j)\}$ corresponding to lowest $K$ $\{l_ij\}_N$
    Add the selected $\{(s_i, s_j)\}$ to the training set
    Train the agent with $\mathcal{P}$ until $\mathcal{P}$ is depleted
```

Categorical Cost Estimate. Training a robust goal-conditioned value function is challenging yet crucial for planning efficient waypoints [12]. Incorporating distributional RL [18] has shown to be an effective strategy in improving the quality of goal-conditioned value functions thereby enhancing the planning process. Therefore, we follow [18] to model the goal-conditioned value function for costs as a categorical distribution $Z_\pi \sim \text{Cat}(N, p_\pi(z \mid s, a, g))$ within a range of $[V^c_{\min}, V^c_{\max}]$, $V^c_{\min}, V^c_{\max} \in \mathbb{R}^+$, where $N \in \mathbb{N}$ is the number of evenly discrete classes, $p_\pi(z \mid s, a, g)$ is the

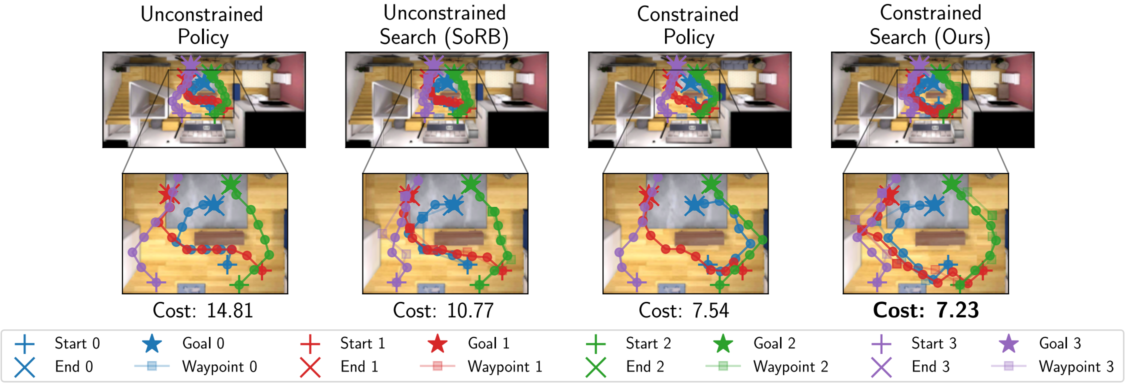

Fig. 3: Comparison of different approaches on the SC3 Staging 11 map for four agents shows that unconstrained baselines fail to avoid the furniture, whereas the constrained policy and our approach successfully navigates around it. Our approach goes further by re-routing the agents away from the furniture, ensuring the safest behavior. Note that for both unconstrained and constrained policies, inter-agent collision avoidance is ensured by having agents wait when another agent is nearby.

probability predicted by the cost value function conditioned on the current policy π. The class supports can be described by a set of atoms $\{z_i = V^c_{\text{min}} + i\Delta z : 0 \le i < N\}, \Delta z \triangleq (V^c_{\text{max}} - V^c_{\text{min}})/(N-1)$. Consequently, the expected value of the distribution is used estimate the goal-conditioned cost:

$$c_\pi(s_1, g) = \mathop{\mathbb{E}}_{p(z|s,a,g)}[z] \tag{3}$$

Since Z_π is a categorical distribution, standard Bellman update is not applicable. Instead, we follow the Categorical Algorithm [18] to update $Q_C(s, a, s_g)$.

Graph Construction from Replay Buffer. With a trained agent, we use the observations (e.g., images) in the replay buffer $\mathcal{B}$ to construct a weighted, directed graph $\mathcal{G}$. Each edge has a predicted distance and cost for the state pair (s_i, s_j):

$$d_{sp} \approx d_\pi \leftarrow Q(s, \pi(s_i, a, s_g), s_j) \tag{4}$$

$$c_{sp} \approx c_\pi \leftarrow Q_c(s, \pi(s_i, a, s_g), s_j) \tag{5}$$

$$\begin{aligned}
&\mathcal{G} \triangleq (\mathcal{V}, \mathcal{E}, \mathcal{W}_d, \mathcal{W}_c) \\
&\text{where } \mathcal{V} = \mathcal{B} \\
&\qquad \mathcal{E} = \mathcal{B} \times \mathcal{B} = \{e_{s_i \to s_j} | s_i, s_j \in \mathcal{B}\} \\
&\qquad \mathcal{W}_d\left(e_{s_i \to s_j}\right) = \begin{cases} d_\pi(s_i, s_j) & \text{if } d_\pi(s_i, s_j) < \mathrm{d}_{\text{max}} \\ \infty & \text{otherwise} \end{cases} \\
&\qquad \mathcal{W}_c\left(e_{s_i \to s_j}\right) = \begin{cases} c_\pi(s_i, s_j) & \text{if } c_\pi(s_i, s_j) < \mathrm{c}_{\text{max}} \\ \infty & \text{otherwise} \end{cases}
\end{aligned}$$

We exclude edges whose distance exceeds a maximum cutoff, d_{max}, or whose cost surpasses a maximum cutoff, c_{max}. Both d_{max} and c_{max} are treated as hyper-parameters. **Conflict-Based Search (CBS) for Multi-Agent Planning** [19] is a widely used MAPF algorithm that addresses the problem's exponential state space by decomposing it into a series of single-agent path planning problems which are then combined using efficient conflict-resolution strategies. CBS operates on two levels: the high-level search, which manages different constraints on agents' path plans, and the low-level search, where a space-time A* algorithm is used to find paths for each agent independently while satisfying the constraints imposed by the high-level search tree.

In CBS, constraints are typically defined in the form $\langle a_i, v, t\rangle$, indicating that agent a_i is prohibited from visiting vertex $v \in \mathcal{V}$ at time step t. The high level search constructs a *constraint tree* starting from a root node with no constraints. The tree expands in a best-first manner where each successor node inherits its parent's constraints and adds a new constraint for a single agent. The goal of CBS is to reach a node in the constraint tree where all paths of all agents are conflict-free. This two-level structure of CBS effectively manages the complexity of MAPF problems by isolating single-agent planning tasks and resolving conflicts incrementally.

High-Level Overview. In summary, our approach leverages goal-conditioned safe RL augmented with self-sampling and training algorithm 1 to build a graph from the replay buffer annotated with predicted distances and cost estimates using the Q-functions Q^π, Q^π_c of the *unconstrained* policy π. CBS then utilizes this intermediate graph to generate effective and safe sequential waypoints for the agents, which are followed using the trained low-level goal-conditioned *safe* RL policy π_c for enhanced bi-level safety.

IV. Experiments

Our experiments address the following questions to evaluate the effectiveness of our approach:

- **Q1:** Can our approach demonstrate safer agent behavior compared to other methods?
- **Q2:** Does our approach sacrifice its ability to reach distant goals when balancing the accumulated cost compared to other baselines?
- **Q3:** Does the suggested approach effectively demonstrate superior performance when scaled to larger number of agents compared to other methods?

For the single-agent experiments, we compare our approach against a goal-conditioned RL policy, SoRB and a

Problem Configurations		**Methods (Cumulative Cost (Success Rate))**			
Problem Type	Agents	Unconstrained Policy	Unconstrained Search (SoRB)	Constrained Policy	Constrained Search (Ours)
Easy	1	**0.46 ± 1.05 (100%)**	1.38 ± 2.62 (100%)	0.47 ± 1.07 (100%)	0.49 ± 1.06 (100%)
	5	N/A	6.39 ± 4.66 (100%)	N/A	**1.68 ± 1.21 (100%)**
	10	N/A	8.69 ± 4.05 (100%)	N/A	**2.24 ± 1.03 (100%)**
	20	N/A	10.51 ± 3.57 (100%)	N/A	**2.72 ± 0.80 (100%)**
Medium	1	1.58 ± 2.03 (100%)	2.30 ± 3.02 (100%)	**1.58 ± 2.01 (100%)**	1.59 ± 1.99 (100%)
	5	N/A	5.12 ± 3.65 (100%)	N/A	**3.09 ± 1.50 (100%)**
	10	N/A	7.27 ± 3.66 (98%)	N/A	**4.11 ± 1.48 (98%)**
	20	N/A	8.73 ± 3.73 (98%)	N/A	**4.78 ± 1.33 (98%)**
Hard	1	3.98 ± 4.40 (100%)	4.05 ± 4.19 (100%)	4.19 ± 4.23 (100%)	**3.09 ± 3.87 (100%)**
	5	N/A	8.58 ± 3.79 (100%)	N/A	**6.01 ± 1.96 (100%)**
	10	N/A	10.77 ± 3.51 (100%)	N/A	**7.24 ± 1.52 (100%)**
	20	N/A	11.96 ± 3.31 (100%)	N/A	**8.36 ± 1.58 (100%)**

TABLE I: **2D Navigation**: Comparison of accumulated cost across different approaches with varying problem difficulty and number of agents. Our approach is consistently able to plan successful and safest paths for multiple agents. Note that we extended SoRB by integrating ith with CBS to plan waypoints for multiple agents.

goal-conditioned safe RL policy. For the multi-agent experiments, we evaluated both off-policy and on-policy Multi-Agent RL (MARL) approaches, including MADDPG [20] and MAPPO [21]. However, these methods exhibited the same limitations as goal-conditioned RL approaches, ultimately failing to enable multiple agents to reach their goals. Hence, we compare our approach with an extended version of SoRB that utilises CBS to generate intermediate waypoints for multiple agents. Finally, we compare our approach with these baselines over two different tasks: a 2D navigation problem and a visual navigation problem that involves using images as observations.

Environments. We use a 2D navigation environment with Central Obstacle map, where the agent observations are their locations, represented as $s = (x, y) \in \mathbb{R}^2$. The agent's actions, $a = (dx, dy) \in [-1, 1]^2$, are added to the current position at each time step to update their location. For the visual navigation problems, we selected four distinct environments from the rich, interactive digital-twin ReplicaCAD dataset [22], integrated with Habitat-Sim [23], [24]. In these scenarios, the agent's observations consist of RGB panoramic images created by concatenating 32×32 first-person views from each of the four cardinal directions. The action space for the agent in the visual navigation problems is the same as in the 2D navigation scenario. For all experiments, we choose the cost limit $\delta = 10$, and we use a cost function that linearly decreases from the obstacle boundaries.

$$C(s) = \begin{cases} 2 - 2h/r & 0 \leq h \leq r \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where $h : \mathcal{S} \to \mathbb{R}_{\geq 0}$ is the distance to the boundary of the nearest obstacle and $r \in \mathbb{R}_{+}$ is the radius of influence of each obstacle. In our experiments, r is set to 10 for 2D navigation and 1 for visual navigation tasks respectively.

Figure 1 illustrates example trajectories generated by our approach and the baselines in single-agent and multi-agent settings in the 2D navigation task for a qualitative comparison. Figure 1 shows that, while the baseline methods often fail to generate safer paths when the sampled start and goal locations fall within high-cost regions, our approach successfully identifies a path that, while counter-intuitive from the perspective of the RL agent, aligns with human intuition. By navigating around the high-cost regions, our approach effectively minimizes the accumulated cost.

We now examine the effectiveness of our approach on the more challenging visual navigation problem as shown in Figure 2. Figures 3 present qualitative comparison with example trajectories generated by both the baselines and our approach. The results are consistent with those from the 2D navigation problem, demonstrating that our approach enables agents to re-route their paths away from obstacles, such as furniture in the room, leading to safer navigation behavior.

Our comprehensive quantitative experiments involved randomly generating problems with varying difficulty levels according to the distance and cost predictions from the trained value functions, placing start and goal positions closer together for easier tasks and further apart for harder ones to increase complexity. Each experiment consisted of 50 different problem instances for each difficulty level. In our multi-agent experiments, we compared our approach with an extended version of SoRB, varying the number of agents from 5 to 20 for the 2D navigation problem, and from 5 to 10 agents for the visual navigation problem.

On the simpler 2D navigation problem, Table I shows that our approach consistently generates safer paths with the lowest accumulated cost compared to the other baselines, both in single-agent and multi-agent settings. This safer behavior becomes more pronounced as the number of agents increases, with no loss in success rate.

For the challenging visual navigation problems, Table II demonstrates that our approach again outperforms the baselines by generating safer paths for the agents in both single and multi-agent settings. Notably, we observed a significant drop in success rate for SoRB with an increased number of agents, particularly in the SC2 Staging 08 map. This is likely due to longer re-routing times as agents avoid

Problem Configurations			**Methods** **(Cumulative Cost (Success Rate))**			
Map	Problem Type	Agents	Unconstrained Policy	Unconstrained Search (SoRB)	Constrained Policy	Constrained Search (Ours)
SC2 Staging 08	Easy	1	3.16 ± 9.40 (100%)	2.14 ± 2.22 (98%)	1.85 ± 1.80 (100%)	**1.28 ± 1.69 (100%)**
		5	N/A	4.32 ± 2.29 (98%)	N/A	**3.35 ± 1.48 (100%)**
		10	N/A	5.36 ± 2.60 (96%)	N/A	**4.02 ± 1.15 (100%)**
	Medium	1	4.98 ± 3.46 (98%)	5.15 ± 3.32 (98%)	4.77 ± 3.25 (100%)	**3.35 ± 3.23 (100%)**
		5	N/A	9.74 ± 11.13 (90%)	N/A	**6.18 ± 1.64 (98%)**
		10	N/A	14.23 ± 23.53 (84%)	N/A	**7.30 ± 1.69 (98%)**
	Hard	1	13.41 ± 4.00 (96%)	12.35 ± 5.10 (90%)	10.43 ± 3.07 (100%)	**9.62 ± 3.66 (100%)**
		5	N/A	21.23 ± 14.64 (84%)	N/A	**13.02 ± 2.38 (98%)**
		10	N/A	27.37 ± 19.05 (62%)	N/A	**13.87 ± 2.17 (92%)**
SC3 Staging 05	Easy	1	1.87 ± 1.65 (98%)	2.23 ± 1.85 (98%)	1.80 ± 1.58 (100%)	**1.25 ± 1.60 (100%)**
		5	N/A	3.71 ± 1.96 (98%)	N/A	**2.76 ± 1.36 (100%)**
		10	N/A	4.20 ± 1.73 (98%)	N/A	**3.32 ± 1.11 (100%)**
	Medium	1	5.06 ± 3.27 (100%)	4.87 ± 3.15 (100%)	4.38 ± 2.91 (100%)	**2.66 ± 2.32 (100%)**
		5	N/A	7.58 ± 2.16 (98%)	N/A	**5.88 ± 1.38 (100%)**
		10	N/A	8.83 ± 1.75 (98%)	N/A	**7.28 ± 1.53 (96%)**
	Hard	1	14.86 ± 4.9 (98%)	13.16 ± 4.24 (100%)	8.59 ± 2.21 (98%)	**6.47 ± 3.30 (98%)**
		5	N/A	17.79 ± 2.68 (96%)	N/A	**15.10 ± 2.19 (96%)**
		10	N/A	19.56 ± 2.66 (94%)	N/A	**16.71 ± 2.28 (94%)**
SC3 Staging 11	Easy	1	1.20 ± 1.56 (100%)	1.27 ± 1.35 (100%)	1.20 ± 1.55 (100%)	**0.78 ± 1.12 (100%)**
		5	N/A	2.32 ± 1.11 (100%)	N/A	**1.88 ± 0.90 (100%)**
		10	N/A	3.00 ± 1.07 (100%)	N/A	**2.54 ± 1.15 (100%)**
	Medium	1	3.66 ± 2.95 (98%)	4.19 ± 3.23 (98%)	3.40 ± 2.83 (98%)	**1.95 ± 2.24 (100%)**
		5	N/A	8.33 ± 2.37 (98%)	N/A	**5.93 ± 2.40 (96%)**
		10	N/A	10.23 ± 2.27 (96%)	N/A	**8.13 ± 2.18 (94%)**
	Hard	1	16.41 ± 6.50 (100%)	14.75 ± 5.43 (100%)	11.59 ± 3.60 (100%)	**10.93 ± 3.94 (100%)**
		5	N/A	18.67 ± 2.51 (96%)	N/A	**15.78 ± 2.16 (98%)**
		10	N/A	20.37 ± 2.41 (92%)	N/A	**16.94 ± 1.66 (92%)**
SC3 Staging 15	Easy	1	1.90 ± 1.76 (100%)	2.08 ± 1.73 (100%)	1.87 ± 1.77 (100%)	**1.47 ± 1.39 (100%)**
		5	N/A (100%)	3.82 ± 3.53 (100%)	N/A	**2.77 ± 1.32 (100%)**
		10	N/A	4.83 ± 3.68 (100%)	N/A	**3.63 ± 1.39 (100%)**
	Medium	1	3.76 ± 3.42 (100%)	4.16 ± 2.79 (100%)	3.23 ± 2.51 (100%)	**2.01 ± 1.75 (100%)**
		5	N/A	6.70 ± 1.81 (100%)	N/A	**5.18 ± 2.32 (100%)**
		10	N/A	10.21 ± 14.89 (96%)	N/A	**6.66 ± 1.59 (100%)**
	Hard	1	13.83 ± 4.35 (100%)	14.37 ± 11.06 (98%)	9.75 ± 2.74 (100%)	**7.61 ± 3.10 (100%)**
		5	N/A	19.23 ± 11.03 (88%)	N/A	**11.63 ± 2.70 (100%)**
		10	N/A	19.70 ± 3.73 (84%)	N/A	**12.85 ± 2.12 (98%)**

TABLE II: **Safe Visual Navigation**: Comparison of accumulated cost across different approaches with varying problem difficulty and number of agents.

each other, leading to violation of the time limits. However, as shown in Table II, our approach consistently maintains safer paths for the agents, with the safety margin improving as the number of agents increases.

Regarding success rate, our approach manages to maintain near-perfect scores, barring a slight drop in performance when scaling to larger numbers of agents. This behavior is expected, as with more agents, the traversable regions become narrower, making it more challenging for CBS to find collision-free paths. Consequently, agents may need to wait longer at some assigned waypoints, resulting in increased steps to reach their goals and a higher likelihood of exceeding time limits. Based on these results, we conclude that our approach not only plans safer paths for agents but does so with minimal loss in their ability to reach their goals. It also demonstrates consistent trends when scaling to larger numbers of agents, effectively addressing **Q1**, **Q2**, and **Q3**.

V. Conclusions

In conclusion, this work introduces a novel hierarchical framework that combines high-level multi-agent planning using MAPF approaches with low-level control through goal-conditioned safe RL policies. The framework leverages learned distance and cost critics to automatically generate and annotate dense graph networks from the replay buffer using self-sampling and training. At the high level, our approach utilizes CBS to plan efficient and safe waypoints for multiple agents, while at the low level, the goal-conditioned safe RL policy guides the agents to follow these waypoints toward distant goal-reaching tasks safely. This bi-level safety control ensures safe multi-agent behavior throughout execution, effectively tackling scenarios ranging from simple 2D navigation to more complex visual navigation challenges.

REFERENCES

[1] 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, 2019, pp. 151–158.

[2] 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.

[3] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and R. Hadsell, "Learning to navigate in complex environments," 2017.

[4] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 1312–1320. [Online]. Available: https://proceedings.mlr.press/v37/schaul15.html

[5] V. Pong, S. Gu, M. Dalal, and S. Levine, "Temporal difference models: Model-free deep rl for model-based control," 2020. [Online]. Available: https://arxiv.org/abs/1802.09081

[6] A. Levy, R. Platt, and K. Saenko, "Hierarchical reinforcement learning with hindsight," 2019.

[7] O. Nachum, S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," 2018.

[8] H.-T. L. Chiang, A. Faust, M. Fiser, and A. Francis, "Learning navigation behaviors end-to-end with autorl," 2019.

[9] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet, "Learning latent plans from play," 2019.

[10] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," 2018.

[11] E. Altman, *Constrained Markov decision processes*. Routledge, 2021.

[12] B. Eysenbach, R. R. Salakhutdinov, and S. Levine, "Search on the replay buffer: Bridging planning and reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[13] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2019. [Online]. Available: https://arxiv.org/abs/1509.02971

[14] A. Ray, J. Achiam, and D. Amodei, "Benchmarking Safe Exploration in Deep Reinforcement Learning," 2019.

[15] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

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

[18] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *International conference on machine learning*. PMLR, 2017, pp. 449–458.

[19] 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.

[20] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," 2020. [Online]. Available: https://arxiv.org/abs/1706.02275

[21] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of ppo in cooperative, multi-agent games," 2022. [Online]. Available: https://arxiv.org/abs/2103.01955

[22] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, A. Clarkson, M. Yan, B. Budge, Y. Yan, X. Pan, J. Yon, Y. Zou, K. Leon, N. Carter, J. Briales, T. Gillingham, E. Mueggler, L. Pesqueira, M. Savva, D. Batra, H. M. Strasdat, R. D. Nardi, M. Goesele, S. Lovegrove, and R. Newcombe, "The Replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019.

[23] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. Chaplot, O. Maksymets, A. Gokaslan, V. Vondrus, S. Dharur, F. Meier, W. Galuba, A. Chang, Z. Kira, V. Koltun, J. Malik, M. Savva, and D. Batra, "Habitat 2.0: Training home assistants to rearrange their habitat," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[24] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A Platform for Embodied AI Research," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.