Received 15 May 2023; revised 7 August 2023; accepted 29 August 2023. Date of publication 15 September 2023;
date of current version 9 October 2023. Recommended by Senior Editor Prof. W. Zhang.

Digital Object Identifier 10.1109/OJCSYS.2023.3316090

Constrained Environment Optimization for Prioritized Multi-Agent Navigation

ZHAN GAO (Student Member, IEEE), AND AMANDA PROROK (Member, IEEE)

(Intersection of Machine Learning With Control)

Department of Computer Science and Technology, University of Cambridge, CB3 0FD Cambridge, U.K.

CORRESPONDING AUTHOR: ZHAN GAO (e-mail: zg292@cam.ac.uk).

This work was supported by European Research Council (ERC) under Project 949940 (gAIa).
Preliminary results appeared in ICRA 2023 conference [DOI: 10.1109/ICRA48891.2023.10160813].

ABSTRACT Traditional approaches for multi-agent navigation consider the environment as a fixed constraint, despite the obvious influence of spatial constraints on agents' performance. Yet hand-designing conducive environments is inefficient and potentially expensive. The goal of this article is to consider the obstacle layout of the environment as a decision variable in a system-level optimization problem. In other words, we aim to find an automated solution that optimizes the obstacle layout to improve the performance of multi-agent navigation, under a variety of realistic constraints. Towards this end, we propose novel problems of *unprioritized* and *prioritized environment optimization*, where the former considers agents unbiasedly and the latter incorporates agent priorities into optimization. We show, through formal proofs, under which conditions the environment can change to guarantee completeness (i.e., all agents reach goals), and analyze the role of agent priorities in the environment optimization. We proceed to impose constraints on the environment optimization that correspond to real-world restrictions on obstacle changes, and formulate it mathematically as a constrained stochastic optimization problem. Since the relationship between agents, environment and performance is challenging to model, we leverage reinforcement learning to develop a model-free solution and a primal-dual mechanism to handle constraints. Distinct information processing architectures are integrated for various implementation scenarios, including online/offline optimization and discrete/continuous environment. Numerical results corroborate the theory and demonstrate the validity and adaptability of our approach.

INDEX TERMS Constrained optimization, environment optimization, multi-agent systems, navigation.

I. INTRODUCTION

Multi-agent systems present an attractive solution to spatially distributed tasks, wherein motion planning among moving agents and obstacles is one of the central problems. To date, the primal focus in multi-agent motion planning has been on developing effective, safe, and near-optimal navigation algorithms [2], [3], [4], [5], [6], [7]. These algorithms consider the *agents' environment as a fixed constraint*, where structures and obstacles must be circumnavigated. In this process, mobile agents engage in negotiations with one another for right-of-way, driven by local incentives to minimize individual delays. However, spatial constraints of the environment may result in dead-locks, live-locks and prioritization conflicts, even for state-of-the-art algorithms [8]. These insights highlight the impact of the environment on multi-agent navigation.

Not all environments elicit the same kinds of agent behaviors and individual navigation algorithms are susceptible to environmental artefacts; undesirable environments can lead to irresolution in path planning [9]. To deal with such bottlenecks, spatial structures (e.g., intersections, roundabouts) and markings (e.g., lanes) are developed to facilitate path de-confliction [10] but these concepts are based on legacy mobility paradigms, which ignore inter-agent communication, cooperation, and systems-level optimization. While it is possible to deal with the circumvention of dead-locks

© 2023 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

and live-locks through hand-designed environment templates, such hand-designing process is inefficient [11].

Reconfigurable and automated environments are emerging as a new trend in next generation buildings [1], [12], [13], [14], which incorporate mechatronic devices to establish interactive relations with agents. This makes it feasible to reconfigure the obstacle layout of the environment in a way that improves some objective measure of the multi-agent system, especially in structured settings where agents are expected to solve repetitive tasks (like warehouses, factories, restaurants, etc.). In tandem with that enabling technology, the goal of this article is to consider the obstacle layout of the environment as a *decision variable* in pursuit of the agents' incentives. We assume agents have a fixed navigation algorithm, and propose the problem of systematically *optimizing the obstacle layout of the environment to improve the navigation performance of the given multi-agent system*. Applications of environment optimization include warehouse logistics (e.g., finding the optimal positions of the shelves for cargo transportation [15]), search and rescue (e.g., clearing the best passages for trapped victims [16]), city planning (e.g., generating the optimal configuration of one-way routes [17]), and digital entertainment (e.g., building the best gaming scene for video games [18]). Environment optimization is independent of traditional approaches that focus on designing the agents' trajectory planner to interact with the environment, and can be applied as an add-on module to any trajectory planner in the same manner. Moreover, this allows us to facilitate multi-agent navigation even if the agents' actions cannot be directly controlled, e.g., when agents are human with inherent preferences and unknown decision-making processes, or physical robots with hardware and computation limitations. More in detail, our contributions are as follows:

i) We define novel problems of *unprioritized* and *prioritized* environment optimization, where the former considers agents unbiasedly and the latter accounts for agent priorities. We develop two solution variants, i.e., *offline* and *online* environment optimization, that adapt to different implementation scenarios.
ii) We conduct the completeness analysis for multi-agent navigation with environment optimization, which identifies the conditions under which all agents are guaranteed to reach their goals. We also analyze the effects of *agent priorities* on environment optimization, and show how environment "resources" can be negotitated to improve performance in an ordered scheme.
iii) We impose practical constraints on the environment optimization, and formulate it as a constrained stochastic optimization problem. We leverage reinforcement learning and a primal-dual mechanism to develop a model-free method, which allows us to integrate different information processing architectures (e.g., CNNs, GNNs) as a function of the problem setting.
iv) We evaluate the proposed framework in both discrete and continuous environment settings. The results corroborate theoretical findings and show the proposed approach can generalize beyond training instances, adapts to various optimization objectives and constraints, and allows for decentralized implementation.

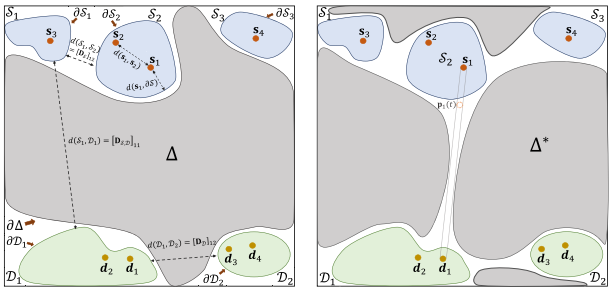
Related work: The problem of environment optimization is reminiscent of robot co-design [19], [20], [21], wherein sensing, computation, and control are jointly optimized. While this domain of robotics is *robot-centric* (i.e., does not consider the environment as an optimization criteria), there are a few works that, similarly to our approach, use reinforcement learning to co-optimize objectives [22], [23], [24]. A more closely related idea is exploited in [9], wherein the environment is adversarially optimized to fail the navigation tasks of state-of-the-art solutions. It conducts a worst-case analysis to shed light on directions in which more robust systems can be developed. On the contrary, we optimize the environment to *facilitate* multi-agent navigation tasks.

The role of the environment on motion planning has been previously explored. Specifically, [25], [26] emphasize the existence of congestion and deadlocks in undesirable environments and develop trajectory planning methods to escape from potential deadlocks. The work in [27] defines the concept of "well-formed" environments, in which the navigation tasks of all agents can be carried out successfully without collisions. In [28], Wu et al. show that the shape of the environment leads to distinct *path prospects* for different agents, and that this information can be shared among the agents to optimize and coordinate their mobility. Gur et al. in [29] generate adversarial environments and account for the latter information to develop resilient navigation algorithms. However, none of these works consider optimizing the environment to improve system-wide navigation performance.

Our problem is also related to the problem of environment design. The work in [30] design environments to influence a agent's decision and [31], [32] focus on boosting the interpretability of a robot's behavior for human, while both are formulated for a single-agent scenario. The works of Hauser [33], [34] consider removing obstacles from the environment to improve navigation performance of one agent. Bellusci et al. [35] extends a similar idea to the multi-agent setting, which searches for solutions by considering all possible environment configurations. However, this is limited in discrete environments and it may not be practical to remove obstacles in real-world applications. The problem of multi-agent pick-up and delivery also considers re-configuring the environment [36], [37], [38], [39]. However, these environment changes are pre-defined tasks for agents and the goal is to develop algorithms to complete these tasks, which are different problem settings. Moreover, these works consider discrete environment configurations that mismatch many practical scenarios.

II. PROBLEM FORMULATION

Let $\mathcal{E}$ be a 2-D environment described by a starting region $\mathcal{S}$, a destination region $\mathcal{D}$ and an obstacle region Δ without overlap, i.e., $\mathcal{S} \bigcap \mathcal{D} = \mathcal{S} \bigcap \Delta = \mathcal{D} \bigcap \Delta = \emptyset$. Consider a multi-agent system with n agents $\mathcal{A} = \{A_i\}_{i=1}^{n}$ distributed



(a) (b)

FIGURE 1. Example environment with 4 starting positions $\{\mathbf{s}_i\}_{i=1}^4$ and 4 destinations $\{\mathbf{d}_i\}_{i=1}^4$. There are $C_{\mathcal{S}} = 3$ self-connected starting components $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$ in the starting region $\mathcal{S} = \bigcup_{l=1}^3 \mathcal{S}_l$ (blue), $C_{\mathcal{D}} = 2$ self-connected destination components $\mathcal{D}_1, \mathcal{D}_2$ in the destination region $\mathcal{D} = \bigcup_{k=1}^2 \mathcal{D}_k$ (green), and the obstacle region Δ (grey). Agent A_1 is initialized at $\mathbf{s}_1$ in the starting component $\mathcal{S}_{A_1} = \mathcal{S}_2$ and moves towards its goal $\mathbf{d}_1$ in the destination component $\mathcal{D}_{A_1} = \mathcal{D}_1$. Analogous notation applies for the other agents A_2, A_3 and A_4. (a) Initial environment before optimization. It demonstrates notations used in this work for theoretical analysis. (b) Environment after prioritized optimization. It optimizes the obstacle region from Δ to Δ^*, which guarantees the existence of a valid solution and improves the solution performance based on agent priorities.

in $\mathcal{E}$. The agent bodies are contained within circles of radii $\{r_i\}_{i=1}^n$. Agents are initialized at positions $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_n]$ in $\mathcal{S}$ and deploy a given trajectory planner π_a to move towards destinations $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_n]$ in $\mathcal{D}$. Let $\boldsymbol{\rho} = [\rho_1, \ldots, \rho_n]^\top$ be a set of priorities that represent the importance of agents w.r.t. the navigation tasks, i.e., the larger the priority, the more important the agent; in other words, for $\rho_1 \geq \cdots \geq \rho_n$, agent A_1 has the highest priority.

Denote by $\partial\mathcal{S}$, $\partial\mathcal{D}$, $\partial\Delta$ the boundaries of the regions $\mathcal{S}$, $\mathcal{D}$, Δ and $B(\mathbf{s}, r)$ a closed disk centered at position $\mathbf{s}$ with a radius r. This allows us to represent each agent A_i as $B(\mathbf{s}_i, r_i)$ where $\mathbf{s}_i$ is the agent position and r_i is the agent radius. Define $d(\mathbf{s}_i, \mathbf{s}_j)$ as the closest distance between agents A_i, A_j, i.e., $d(\mathbf{s}_i, \mathbf{s}_j) = \min \|\mathbf{z}_i - \mathbf{z}_j\|_2$ for any $\mathbf{z}_i \in B(\mathbf{s}_i, r_i)$ and $\mathbf{z}_j \in B(\mathbf{s}_j, r_j)$, and $d(\mathbf{s}_i, \partial\mathcal{S})$ the closest distance between agent A_i and the starting region boundary $\partial\mathcal{S}$, i.e., $d(\mathbf{s}_i, \partial\mathcal{S}) = \min \|\mathbf{z}_i - \mathbf{z}_\mathcal{S}\|_2$ for any $\mathbf{z}_i \in B(\mathbf{s}_i, r_i)$ and $\mathbf{z}_\mathcal{S} \in \partial\mathcal{S}$. Analogous definitions apply for $\partial\mathcal{D}$ and $\partial\Delta$. Let $\mathbf{p}(t) : [0, \infty) \to \mathbb{R}^2$ be the trajectory representing the central position movement of an agent. The trajectory $\mathbf{p}_i$ of agent A_i is collision-free w.r.t. the obstacle region Δ if $d(\mathbf{p}_i(t), \partial\Delta) \geq 0$ for all $t \geq 0$. The trajectories $\mathbf{p}_i$, $\mathbf{p}_j$ of two agents A_i, A_j are collision-free with each other if $d(\mathbf{p}_i(t), \mathbf{p}_j(t)) \geq 0$ for all $t \geq 0$ – see Fig. 1 for demonstration. A valid solution for the trajectory planner π_a is defined as follows.

Definition 1: A valid solution of the trajectory planner π_a is a set of trajectories $\{\mathbf{p}_i(t)\}_{i=1}^n$ that satisfy

$$\mathbf{p}_i(0) = \mathbf{s}_i,\ \mathbf{p}_i(T) = \mathbf{d}_i \text{ for } i = 1, \ldots, n, \tag{1}$$

and

$$d\left(\mathbf{p}_i(t), \mathbf{p}_j(t)\right) \geq 0,\ d\left(\mathbf{p}_i(t), \partial\Delta\right) \geq 0 \tag{2}$$

for any $i, j = 1, \ldots, n$ and $t \in [0, T]$, where T is the maximal operation time for agents.

Definition 1 satisfies the navigation convergence with condition (1) and the collision avoidance with condition (2).

The performance of multi-agent navigation solutions depends not only on the agents' trajectory planner but also on their surrounding environment. A "well-formed" environment with an appropriate obstacle region yields good performance for a simple planner, while a "poorly-formed" environment may result in poor performance for an advanced planner. This insight implies an important role played by the environment in multi-agent navigation, and motivates the definition of the problem of environment optimization. The overarching goal is to optimize the obstacle region Δ in the environment to improve the performance of multi-agent navigation. The principle of environment optimization is that the obstacle region Δ can be controlled (i.e., changed) while its area $|\Delta|$ remains the same, i.e., $|\Delta| = |\widehat{\Delta}|$ where $\widehat{\Delta}$ is the changed obstacle region. That is, we do not remove or add any obstacle from our environment. We consider two variants: (1) *unprioritized environment optimization* and (2) *prioritized environment optimization*.

Problem 1 (Unprioritized Environment Optimization): Given an environment with an initial obstacle region Δ and a multi-agent system of n agents $\{A_i\}_{i=1}^n$ that are initialized at positions $\mathbf{S}$ and targeted towards destinations $\mathbf{D}$ with the trajectory planner π_a, optimize the obstacle region from Δ to Δ^* such that in the optimized environment, (i) there exists a valid solution $\{\mathbf{p}_i(t)\}_{i=1}^n$ for π_a [Def. 1] and (ii) the solution performance is maximized.

Problem 2 (Prioritized Environment Optimization): Given an environment with an initial obstacle regions Δ, a multi-agent system of n agents $\{A_i\}_{i=1}^n$ that are initialized at positions $\mathbf{S}$ and targeted towards destinations $\mathbf{D}$ with the trajectory planner π_a, agent priorities $\boldsymbol{\rho}$ that are ordered by the index $\rho_1 \geq \rho_2 \geq \cdots \geq \rho_n$, and a metric of interest $M(\cdot) : \mathbf{p}(t) \to \mathbb{R}$ that measures the navigation performance of agent trajectories,[1] optimize the obstacle region from Δ to Δ^* such that in the optimized environment, (i) there exists a valid solution $\{\mathbf{p}_i(t)\}_{i=1}^n$ for π_a [Def. 1] and (ii) the solution performance is improved according to agent priorities $\boldsymbol{\rho}$, i.e., the metric values of agents' trajectories are ordered by agent priorities $\boldsymbol{\rho}$ as

$$M(\mathbf{p}_1) \leq M(\mathbf{p}_2) \leq \cdots \leq M(\mathbf{p}_n). \tag{3}$$

Problem 1 considers agents with equal priority and optimizes the environment to improve their performance unbiasedly. Problem 2 accounts for agent priorities, which not only guarantees the existence of a valid solution in the optimized environment but also orders the solution performance based on agent priorities $\boldsymbol{\rho}$ [cf. (3)]. It reduces to Problem 1 when agent priorities are equal to each other, i.e., $\rho_1 = \cdots = \rho_n$. Problems 1 and 2 focus on optimizing the obstacle region of the environment to provide solution guarantees and to improve

[1] Without loss of generality, we assume the lower the value of $M(\cdot)$, the better the performance, e.g., corresponding to traveled distance and navigation time.

the solution performance. An underlying trajectory planner is assumed to exist, yet we do not prescribe how the trajectories should be computed – see Assumption 2 in Section III.

In Section III, we analyze the completeness for Problem 1 to show the effectiveness of environment optimization, i.e., all agents are able to carry out navigation tasks without collision with environment optimization. In Section IV, we conduct the completeness analysis for Problem 2 and characterize the effects of agent priorities on environment optimization. In Section V, we impose external constraints on the environment optimization corresponding to real-world restrictions on obstacle changes and formulate the problem mathematically as a constrained stochastic optimization problem. We transform the latter into the dual domain to tackle constraints and combine reinforcement learning with a primal-dual mechanism for solutions. Lastly, we evaluate the proposed approach numerically and corroborate theoretical findings in Section VI.

Remark 1: Problem 2 optimizes the obstacle region of the environment to guarantee that there exists a valid solution and the solution performance is ordered based on agent priorities [cf. (3)]. We remark that it is possible that an agent A_j finds a different trajectory $\widehat{\mathbf{p}}_j \neq \mathbf{p}_j$, which does not collide with the other trajectories $\{\mathbf{p}_i\}_{i\neq j}$ and satisfies $M(\widehat{\mathbf{p}}_j) < M(\mathbf{p}_j)$. However, the latter does not compromise the prioritized environment optimization proposed in Problem 2.

III. COMPLETENESS OF UNPRIORITIZED SYSTEM

In this section, we provide the completeness analysis of multi-agent navigation for unprioritized environment optimization [Problem 1]. Specifically, the multi-agent system may fail to find collision-free trajectories in an environment with an unsatisfactory obstacle region. Environment optimization overcomes this issue by modifying the obstacle region to guarantee the navigation success for all agents. In the following, we consider both *offline* environment optimization and *online* environment optimization.

A. OFFLINE ENVIRONMENT OPTIMIZATION

Offline environment optimization optimizes the obstacle region Δ based on the starting region $\mathcal{S}$ and the destination region $\mathcal{D}$, and completes the optimization before the agents start moving towards destinations. The optimized environment remains static during agent movement. The goal is to find an optimal obstacle region Δ^* that maximizes the navigation performance of the multi-agent system. Before proceeding, we need the following assumptions for completeness analysis.

Assumption 1: The initial positions $\{\mathbf{s}_i\}_{i=1}^n$ in $\mathcal{S}$ and the destinations $\{\mathbf{d}_i\}_{i=1}^n$ in $\mathcal{D}$ are distributed in a way such that the distance between either two agents or the agent and the region boundary is larger than the maximal agent size, i.e., $d(\mathbf{s}_i, \mathbf{s}_j) \geq 2\hat{r}$, $d(\mathbf{s}_i, \partial\mathcal{S}) \geq 2\hat{r}$ and $d(\mathbf{d}_i, \mathbf{d}_j) \geq 2\hat{r}$, $d(\mathbf{d}_i, \partial\mathcal{D}) \geq 2\hat{r}$ for $i, j = 1, \ldots, n$ with $\hat{r} = \max_{i=1,\ldots,n} r_i$.

Assumption 2: Using the trajectory planner π_a, each agent can compute its own optimal trajectory, leading it from start to goal, in the presence of the obstacle region Δ, and can pass that trajectory information on to other agents.

Assumption 1 indicates that the starting positions $\{\mathbf{s}_i\}_{i=1}^n$ (or goal positions $\{\mathbf{d}_i\}_{i=1}^n$) are not too close to each other, which is commonly satisfied in real-world navigation tasks. Assumption 2 allows each agent to find its own trajectory, if the latter exists, and disseminate the trajectory information among agents. For decentralized systems that require completeness, this is a necessary assumption (e.g., see [11], [27]). Information passing can be achieved through a variety of mechanisms, e.g., token-based [27], or communication protocol based [11]. With these preliminaries, we show the completeness of the offline environment optimization.

Theorem 1: Consider the multi-agent system in the environment $\mathcal{E}$ with starting, destination and obstacle regions $\mathcal{S}$, $\mathcal{D}$ and Δ satisfying Assumptions 1 and 2. Let $d_{\max}$ be the maximal distance between $\mathcal{S}$ and $\mathcal{D}$, i.e., $d_{\max} = \max_{\mathbf{z}_\mathcal{S}, \mathbf{z}_\mathcal{D}} \|\mathbf{z}_\mathcal{S} - \mathbf{z}_\mathcal{D}\|_2$ for any $\mathbf{z}_\mathcal{S} \in \mathcal{S}$, $\mathbf{z}_\mathcal{D} \in \mathcal{D}$. Then, if $\mathcal{E}$ satisfies

$$|\mathcal{E} \setminus (\Delta \cup \mathcal{S} \cup \mathcal{D})| \geq 2\mathrm{nd}_{\max}\hat{r} \tag{4}$$

where $\hat{r} = \max_{i=1,\ldots,n} r_i$ is the maximal radius of n agents and $|\cdot|$ represents the region area, the offline environment optimization guarantees that the navigation tasks of all agents can be carried out successfully without collision.

Proof: See Appendix A. ■

Theorem 1 states that the offline environment optimization can guarantee the success of all navigation tasks, if the area of the environment except the starting, destination and obstacle regions is larger than $2nd_{\max}\hat{r}$. This is a mild condition because it does not require any initial "well-formed" environment but only an obstacle-free area of minimal size [cf. (4)], which is common in real-world scenarios. The offline environment optimization depends on the starting region $\mathcal{S}$ and the destination region $\mathcal{D}$, and completes optimizing the obstacle region Δ *before* the agents start to move. This requires a computational overhead before each new navigation task. Moreover, we are interested in generalizing the problem s.t. $\mathcal{S}$ and $\mathcal{D}$ are allowed to be time-varying during deployments.

B. ONLINE ENVIRONMENT OPTIMIZATION

Online environment optimization changes the obstacle region Δ based on instantaneous states of the agents, e.g., positions, velocities and accelerations, after the agents start moving towards destinations. In other words, the environment is being optimized concurrently with agent movement. The goal is to find the optimal obstacle policy π_o that changes the obstacle region to maximize the navigation performance of the multi-agent system.

Specifically, define the starting region as the union of the starting positions $\mathcal{S} = \bigcup_{i=1,\ldots,n} B(\mathbf{s}_i, r_i)$ and the destination region as that of the destinations $\mathcal{D} = \bigcup_{i=1,\ldots,n} B(\mathbf{d}_i, r_i)$ s.t. $\mathcal{S} \bigcap \mathcal{D} = \emptyset$. Since the obstacle region Δ now changes continuously, we define the *capacity* of the online environment optimization as the maximal changing rate of the obstacle area $\dot{\Delta}$, i.e., the maximal obstacle area that can be changed

per time step. To proceed, we require an assumption for the environment with an empty obstacle region $\Delta = \emptyset$.

Assumption 3: For an environment $\mathcal{E}$ with no obstacle region $\Delta = \emptyset$, all navigation tasks can be carried out successfully without collision and the corresponding agent trajectories $\{\mathbf{p}_i\}_{i=1}^n$ are non-overlap.

Assumption 3 is mild because an environment with no obstacle region is the best scenario for the multi-agent navigation. For an environment with a non-empty obstacle region Δ, the following theorem shows the completeness of the online environment optimization.

Theorem 2: Consider the multi-agent system of n agents $\{A_i\}_{i=1}^n$ satisfying Assumption 3 with n collision-free trajectories $\{\mathbf{p}_i(t)\}_{i=1}^n$ for the environment with an empty obstacle region. Let $\{\mathbf{v}_i(t)\}_{i=1}^n$ be the velocities along $\{\mathbf{p}_i(t)\}_{i=1}^n$, respectively. For the environment $\mathcal{E}$ with a non-empty obstacle region $\Delta \subset \mathcal{E} \setminus (\mathcal{S} \cup \mathcal{D})$ and $\mathcal{E} \setminus (\Delta \cup \mathcal{S} \cup \mathcal{D}) \neq \emptyset$, if the capacity of the online environment optimization satisfies

$$\dot{\Delta} \geq 2n\hat{r}\|\hat{\mathbf{v}}\|_2 \tag{5}$$

where $\|\hat{\mathbf{v}}\|_2 = \max_{t\in[0,T]} \max_{i=1,\ldots,n} \|\mathbf{v}_i(t)\|_2$ is the maximal norm of the velocities and $\hat{r} = \max_{i=1,\ldots,n} r_i$ is the maximal agent radius, the navigation tasks of all agents can be carried out successfully without collision.

Proof: See Appendix B. ■

Theorem 2 states that the online environment optimization can guarantee the success of all navigation tasks as well as its offline counterpart, if the changing rate of the obstacle region is larger than $2n\hat{r}\|\hat{\mathbf{v}}\|_2$. The result is established under a mild condition on the obstacle changing rate [cf. (5)] rather than the initial obstacle-free area [cf. (4)]. This is due to the fact that the online environment optimization changes the obstacle region concurrently with agent movement. Hence, it improves navigation performance only if timely actions can be taken based on instantaneous system states. The completeness analysis in Theorems 1 and 2 demonstrates theoretically the effectiveness of the proposed environment optimization, in improving the performance of the multi-agent navigation.

IV. COMPLETENESS OF PRIORITIZED SYSTEM

Unprioritized environment optimization guarantees the success of navigation tasks in scenarios with sufficient "resources", i.e., a sufficiently large obstacle-free area [Thm. 1] and a sufficiently large obstacle changing rate [Thm. 2]. However, this may not be the case for scenarios with reduced resources. In the latter circumstances, the environment needs to be optimized with respect to the conflicts of interest among different agents, and allocates resources according to *priorities* in order to negotiate these conflicts.

With the formulation of prioritized environment optimization [Problem 2], we overcome this issue by incorporating agent priorities into environment optimization. That is, we put more emphasis on agents with higher priorities to guide the negotiation. In the sequel, we show that prioritized environment optimization is capable of guaranteeing the success of all navigation tasks with reduced resources by sacrificing the navigation performance of agents with lower priorities. Analogous to Section III, we analyze the completeness of offline and online prioritized environment optimization, respectively, which characterizes the explicit effects of agent priorities on the navigation performance.

A. OFFLINE PRIORITIZED ENVIRONMENT OPTIMIZATION

Offline environment optimization optimizes the obstacle region Δ before navigation and guarantees the success of all navigation tasks if $|\mathcal{E} \setminus (\Delta \cup \mathcal{S} \cup \mathcal{D})| \geq 2\mathrm{nd}_{\max}\hat{r}$ [Thm. 1]. We consider the reduced-resource scenario, where the initial obstacle-free area is smaller than $2\mathrm{nd}_{\max}\hat{r}$, i.e., $|\mathcal{E} \setminus (\Delta \cup \mathcal{S} \cup \mathcal{D})| < 2\mathrm{nd}_{\max}\hat{r}$. In this circumstance, offline prioritized environment optimization is able to maintain the success of all navigation tasks, at the cost of lower-priority agent performance.

We introduce additional notation as follows. Define the starting region $\mathcal{S}$ as *disconnected* if there exist two points in $\mathcal{S}$ that cannot be connected by a path inside $\mathcal{S}$, and *self-connected* otherwise. For a disconnected $\mathcal{S}$, let $\{\mathcal{S}_l\}_{l=1}^{C_\mathcal{S}}$ be the least number of self-connected components in $\mathcal{S}$ s.t. their union covers $\mathcal{S}$, i.e., $\bigcup_{l=1}^{C_\mathcal{S}} \mathcal{S}_l = \mathcal{S}$.[2] Assume that agent A_i is initialized in one of these component $\mathcal{S}_{A_i} \in \{\mathcal{S}_l\}_{l=1}^{C_\mathcal{S}}$ for $i = 1, \ldots, n$. Analogous definitions apply for the destination region $\mathcal{D}$. Let $d_{\max}(\mathcal{S}_{l_1}, \mathcal{S}_{l_2})$ be the maximal distance between the components $\mathcal{S}_{l_1}$ and $\mathcal{S}_{l_2}$ for $l_1 \neq l_2 \in \{1, \ldots, C_\mathcal{S}\}$, i.e., $d_{\max}(\mathcal{S}_{l_1}, \mathcal{S}_{l_2}) = \max \|\mathbf{z}_{\mathcal{S}_{l_1}} - \mathbf{z}_{\mathcal{S}_{l_2}}\|_2$ for any $\mathbf{z}_{\mathcal{S}_{l_1}} \in \mathcal{S}_{l_1}$ and $\mathbf{z}_{\mathcal{S}_{l_2}} \in \mathcal{S}_{l_2}$. Denote by $\mathbf{D}_\mathcal{S} \in \mathbb{R}^{C_\mathcal{S}\times C_\mathcal{S}}$ the matrix that collects the distances between the components $\{\mathcal{S}_l\}_{l=1}^{C_\mathcal{S}}$ with the (l_1, l_2)th entry $[\mathbf{D}_\mathcal{S}]_{l_1 l_2} = d_{\max}(\mathcal{S}_{l_1}, \mathcal{S}_{l_2})$ for $l_1, l_2 = 1, \ldots, C_\mathcal{S}$ and $[\mathbf{D}_\mathcal{S}]_{ll} = 0$ for $l = 1, \ldots, C_\mathcal{S}$, $\mathbf{D}_\mathcal{D} \in \mathbb{R}^{C_\mathcal{D}\times C_\mathcal{D}}$ the matrix that collects the distances between the components $\{\mathcal{D}_k\}_{k=1}^{C_\mathcal{D}}$, and $\mathbf{D}_{\mathcal{S},\mathcal{D}} \in \mathbb{R}^{C_\mathcal{S}\times C_\mathcal{D}}$ the matrix that collects the distances between $\{\mathcal{S}_l\}_{l=1}^{C_\mathcal{S}}$ and $\{\mathcal{D}_k\}_{k=1}^{C_\mathcal{D}}$ – see Fig. 1 for demonstration. For completeness analysis, we assume the following.

Assumption 4: The distance between the starting components $\{\mathcal{S}_l\}_{l=1}^{C_\mathcal{S}}$ (or destination components $\{\mathcal{D}_k\}_{k=1}^{C_\mathcal{D}}$) is significantly smaller than the distance between the starting components $\{\mathcal{S}_l\}_{l=1}^{C_\mathcal{S}}$ and the destination components $\{\mathcal{D}_k\}_{k=1}^{C_\mathcal{D}}$, i.e., for any entry $d_\mathcal{S}$ of $\mathbf{D}_\mathcal{S}$, entry $d_\mathcal{D}$ of $\mathbf{D}_\mathcal{D}$ and entry $d_{\mathcal{S},\mathcal{D}}$ of $\mathbf{D}_{\mathcal{S},\mathcal{D}}$, it holds that $d_\mathcal{S} + d_\mathcal{D} \leq d_{\mathcal{S},\mathcal{D}}$.

Assumption 4 is reasonable because the starting positions are typically close to each other but far away from the destinations in real-world applications. With these in place, we characterize the completeness with the offline prioritized environment optimization in the following theorem.

Theorem 3: Consider the multi-agent system of n agents $\{A_i\}_{i=1}^n$ in the environment $\mathcal{E}$ with the same settings as Theorem 1 and satisfying Assumption 4. Let $\boldsymbol{\rho}$ be the agent priorities ordered by the index $\rho_1 \geq \rho_2 \geq \cdots \geq \rho_n$, agent A_i be in the starting component $\mathcal{S}_{A_i} \in \{\mathcal{S}_l\}_{l=1}^{C_\mathcal{S}}$ and assigned to the destination component $\mathcal{D}_{A_i} \in \{\mathcal{D}_k\}_{k=1}^{C_\mathcal{D}}$ for $i = 1, \ldots, n$. Then,

[2] For a self-connected $\mathcal{S}$, we have $C_\mathcal{S} = 1$ and $\mathcal{S}_1 = \mathcal{S}$.

if $\mathcal{E}$ satisfies

$$|\mathcal{E}\setminus(\Delta\cup\mathcal{S}\cup\mathcal{D})|\geq 2d_{A_1}\hat{r}+\sum_{i=2}^{n}2(d_{A_i,\mathcal{S}}+d_{A_i,\mathcal{D}})\hat{r} \tag{6}$$

where d_{A_1} is the distance between $\mathcal{S}_{A_1}$ and $\mathcal{D}_{A_1}$, i.e., $d_{A_1}=[\mathbf{D}_{\mathcal{S},\mathcal{D}}]_{A_1A_1}$, $d_{A_i,\mathcal{S}}$ the minimal distance between $\mathcal{S}_{A_i}$ and $\{\mathcal{S}_{A_j}\}_{j=1}^{i-1}$, i.e., $d_{A_i,\mathcal{S}}=\min\{[\mathbf{D}_{\mathcal{S}}]_{A_iA_1},\ldots,[\mathbf{D}_{\mathcal{S}}]_{A_iA_{i-1}}\}$, $d_{A_i,\mathcal{D}}$ the minimal distance between $\mathcal{D}_{A_i}$ and $\{\mathcal{D}_{A_j}\}_{j=1}^{i-1}$, i.e., $d_{A_i,\mathcal{D}}=\min\{[\mathbf{D}_{\mathcal{D}}]_{A_iA_1},\ldots,[\mathbf{D}_{\mathcal{D}}]_{A_iA_{i-1}}\}$, the offline prioritized environment optimization guarantees that the navigation tasks of all agents can be carried out successfully without collision. Moreover, the traveled distance of agent A_i outside $\mathcal{S}$ and $\mathcal{D}$ is bounded by

$$d_{A_1}+\sum_{j=2}^{i}\left(d_{A_j,\mathcal{S}}+d_{A_j,\mathcal{D}}\right)=M_i \tag{7}$$

for $i=1,\ldots,n$, which increases with the decreasing of priority, i.e., $M_1\leq\cdots\leq M_n$ with the priorities $\rho_1\geq\cdots\geq\rho_n$.

Proof: See Appendix C. ■

Theorem 3 states that the offline prioritized environment optimization can guarantee the success of all navigation tasks and requires less obstacle-free area compared to its unprioritized counterpart, i.e., the lower bound in (6) is smaller than that in (4) [Asm. 4]. However, this improvement is obtained by sacrificing the navigation performance of the agents with lower priorities. That is, these agents are no longer moving along the shortest path and the traveled distance increases with the decreasing of agent priority [cf. (7)]. It corresponds to Problem 2 of prioritized environment optimization, where the metric $M(\cdot)$ is the traveled distance. The latter shows a trade-off between the navigation completeness of all agents and the individual performance of lower-priority agents.

Next, we consider scenarios with further reduced resources, i.e., we show the partial completeness when the obstacle-free area is smaller than that required in (6).

Corollary 1: Consider the same setting as in Theorem 3. If the environment $\mathcal{E}$ satisfies

$$\begin{aligned}&2d_{A_1}\hat{r}+\sum_{i=2}^{b}2\left(d_{A_i,\mathcal{S}}+d_{A_i,\mathcal{D}}\right)\hat{r}\\&\leq|\mathcal{E}\setminus(\Delta\cup\mathcal{S}\cup\mathcal{D})|<2d_{A_1}\hat{r}+\sum_{i=2}^{b+1}2\left(d_{A_i,\mathcal{S}}+d_{A_i,\mathcal{D}}\right)\hat{r}\end{aligned} \tag{8}$$

where b is an integer with $1\leq b<n$, the offline prioritized environment optimization guarantees that the navigation tasks of the agents with highest b priorities $\{A_i\}_{i=1}^{b}$ can be carried out successfully without collision.

For the rest of the agents $\{A_j\}_{j=b+1}^{n}$, if the starting position and the destination of agent A_j is within the same connected components in $\mathcal{S}$ and $\mathcal{D}$ as one of the agents $\{A_i\}_{i=1}^{b}$, i.e., $\mathbf{s}_j\in\mathcal{S}_{A_i}$ and $\mathbf{d}_j\in\mathcal{D}_{A_i}$ for any $i\in\{1,\ldots,b\}$, the navigation task of agent A_j can be carried out successfully without collision as well.

Proof: See Appendix D. ■

Corollary 1 demonstrates that the success of all navigation tasks may not be guaranteed if the obstacle-free area is further smaller than the lower bound in (6). In this case, the prioritized environment optimization guarantees preferentially the navigation tasks of the agents with higher priorities $\{A_i\}_{i=1}^{b}$, but overlooks the ones with lower priorities $\{A_j\}_{j=b+1}^{n}$. Moreover, the navigation tasks of lower-priority agents $\{A_j\}_{j=b+1}^{n}$ can be guaranteed only if their starting and goal positions are within the same connected components as one of the higher-priority agents $\{A_i\}_{i=1}^{b}$. This implies that lower-priority agents can benefit from higher-priority agents if they have "similar" navigation tasks.

B. ONLINE PRIORITIZED ENVIRONMENT OPTIMIZATION

Online environment optimization changes the obstacle region Δ during navigation and guarantees the success of all navigation tasks if $\dot{\Delta}\geq 2n\hat{r}\|\hat{\mathbf{v}}\|_2$ [Thm. 2]. We similarly consider the reduced-resource scenario, where the capacity of the obstacle region is smaller than $2n\hat{r}\|\hat{\mathbf{v}}\|_2$, i.e., $\dot{\Delta}<2n\hat{r}\|\hat{\mathbf{v}}\|_2$. In this circumstance, online prioritized environment optimization can guarantee the success of all navigation tasks, with a performance loss of lower-priority agents.

Theorem 4: Consider the multi-agent system of n agents $\{A_i\}_{i=1}^{n}$ in an environment $\mathcal{E}$ with the same settings as Theorem 2. Let $\boldsymbol{\rho}$ be the agent priorities ordered by the index $\rho_1\geq\rho_2\geq\cdots\geq\rho_n$ and the reduced capacity of environment optimization be

$$2b\hat{r}\|\hat{\mathbf{v}}\|_2\leq\dot{\Delta}<2(b+1)\hat{r}\|\hat{\mathbf{v}}\|_2 \tag{9}$$

where b is an integer with $1\leq b<n$. Then, online prioritized environment optimization guarantees that the navigation tasks of all agents can be carried out successfully without collision. Moreover, the navigation time of agent A_i can be bounded by

$$\frac{C_T}{\rho_i},\ \text{for}\ i=1,\ldots,n \tag{10}$$

that is proportional to the inverse of its priority, where C_T is a time constant determined by b.

Proof: See Appendix E. ■

Theorem 4 states that the online prioritized environment optimization can guarantee the success of all navigation tasks and requires a smaller obstacle changing rate compared to its unprioritized counterpart, i.e., the lower bound in (9) is less than that in (5). This improvement comes with the performance loss of lower-priority agents, i.e., the agents with higher priorities reach the destinations faster than the ones with lower priorities [cf. (10)]. We attribute this behavior to the fact that the online prioritized environment optimization changes the obstacle region concurrently to agent movement and is capable of continuously tuning the changing strategy based on the agent priorities during navigation. This corresponds to Problem 2 of prioritized environment optimization, where the metric $M(\cdot)$ is the navigation time.

We follow to consider scenarios where all agents are required to reach their destinations *within a required time*, and analyze the partial completeness of the online prioritized environment optimization in this setting.

Corollary 2: Consider the same setting as in Theorem 4. If all agents are required to reach their destinations within time $T_{\max}$, i.e., $T_i \leq T_{\max}$ for $i = 1, \ldots, n$, the online prioritized environment optimization guarantees the success of $n_p \leq n$ agents $\{A_i\}_{i=1}^{n_p}$ with highest priorities $\{\rho_i\}_{i=1}^{n_p}$, where n_p is determined by the changing rate of the obstacle region, i.e., b in (9), and the required time $T_{\max}$ [cf. (54)].

Proof: See Appendix F. ■

Corollary 2 demonstrates that the success of all navigation tasks may not be guaranteed if agents are required to reach destinations within a finite time $T_{\max}$. Online prioritized environment optimization allows preferentially a set of agents $\{A_i\}_{i=1}^{n_p}$ with higher priorities to reach destinations, while the rest of agents $\{A_i\}_{i=n_p+1}^{n}$ may fail navigation tasks. The number of successful agents n_p depends on the changing capacity of the obstacle region and the required time $T_{\max}$, i.e., the higher the changing rate and the required time, the larger the number of successful agents. Corollary 2 recovers Theorem 4 when the required time is larger than the maximal time in (10), i.e., $T_{\max} \geq C_T/\rho_n$.

Theorems 3 and 4 show the role played by agent priorities in the offline and online environment optimization with less resources, i.e., the obstacle-free area and the obstacle changing rate, than that required by Theorems 1 and 2. The prioritized environment optimization guides the resource allocation by emphasizing higher-priority agents. By doing so, it guarantees the success of all navigation tasks with reduced resources while sacrificing the navigation performance of lower-priority agents. Moreover, Corollaries 1 and 2 demonstrate the partial completeness of multi-agent navigation for scenarios with further reduced resources and added requirements (e.g., maximal time allowed). The prioritized environment optimization guarantees the success of higher-priority agents, but lower-priority agents may fail in these circumstances.

Remark 2: Theoretical analysis of offline and online environment optimization assumes that the obstacle region Δ can be controlled and re-shaped continuously as long as its area remains the same. The goal is to demonstrate the effectiveness of environment optimization in improving the performance of multi-agent navigation. However, it is worth mentioning that there may exist practical restrictions either on the obstacle region or on the way it can be controlled, in real-world applications. We account for the latter by imposing constraints when mathematically formulating the environment optimization problem in the next section.

V. METHODOLOGY

In this section, we formulate the prioritized environment optimization problem mathematically as a constrained stochastic optimization problem, where the imposed constraints correspond to resource limitations and physical restrictions on the environment optimization in real-world applications, and solve the latter by leveraging reinforcement learning with the primal-dual mechanism.

Specifically, consider the obstacle region Δ comprising m obstacles $\mathcal{O} = \{O_1, \ldots, O_m\}$, which can be of any shape and located at positions $\mathbf{O} = [\mathbf{o}_1, \ldots, \mathbf{o}_m]$. The obstacles can change positions to improve the navigation performance of the agents. Denote by $\mathbf{X}_o = [\mathbf{x}_{o1}, \ldots, \mathbf{x}_{om}]$ the obstacle states, $\mathbf{U}_o = [\mathbf{u}_{o1}, \ldots, \mathbf{u}_{om}]$ the obstacle actions, $\mathbf{X}_a = [\mathbf{x}_{a1}, \ldots, \mathbf{x}_{an}]$ the agent states and $\mathbf{U}_a = [\mathbf{u}_{a1}, \ldots, \mathbf{u}_{am}]$ the agent actions. For example, the states can be positions or velocities and the actions can be accelerations. Let $\pi_o(\mathbf{U}_o|\mathbf{X}_o, \mathbf{X}_a)$ be an optimization policy that controls the obstacles, a distribution over $\mathbf{U}_o$ conditioned on $\mathbf{X}_o$ and $\mathbf{X}_a$. The objective function $f(\mathbf{S}, \mathbf{D}, \pi_a, \boldsymbol{\rho}, \mathbf{O}, \pi_o)$ measures the performance of the multi-agent navigation task $(\mathbf{S}, \mathbf{D})$, given the trajectory planner π_a, the agent priorities $\boldsymbol{\rho}$, the obstacle positions $\mathbf{O}$ and the optimization policy π_o, while the cost functions $\{g_j(\mathbf{S}, \mathbf{D}, \pi_a, \boldsymbol{\rho}, \mathbf{O}, \pi_o)\}_{j=1}^{m}$ indicate the penalties of obstacle position changes, e.g., collision penalties of obstacle movements w.r.t. the other obstacles and agents. Moreover, a set of Q constraints are imposed on the obstacles corresponding to resource limitations and physical restrictions in real-world applications, which are represented by the constraint functions $\{h_q(\mathbf{X}_o, \mathbf{U}_o)\}_{q=1}^{Q}$. For example, the deviation distances of the obstacles from their initial positions or the moving velocities of the obstacles are bounded by some maximal constant. Since the multi-agent system is with random initialization, the objective, cost, constraint functions are random functions and an *expected* performance would be a more meaningful metric for performance evaluation.

The goal is, thus, to find an optimal obstacle policy π_o that maximizes the expected performance $\mathbb{E}[f(\mathbf{S}, \mathbf{D}, \pi_a, \boldsymbol{\rho}, \mathbf{O}, \pi_o)]$ regularized by the obstacle costs $\{\mathbb{E}[g_j(\mathbf{S}, \mathbf{D}, \pi_a, \boldsymbol{\rho}, \mathbf{O}, \pi_o)]\}_{j=1}^{m}$, while satisfying the imposed constraints. By introducing $\mathcal{U}_o$ as the action space of the obstacles, we formulate the prioritized environment optimization problem as a constrained stochastic optimization problem

$$\begin{aligned} \underset{\pi_o}{\operatorname{argmax}} & \mathbb{E}\left[f(\mathbf{S},\mathbf{D},\pi_a,\boldsymbol{\rho},\mathbf{O},\pi_o)\right] - \sum_{j=1}^{m} \beta_j \mathbb{E}\left[g_j(\mathbf{S},\mathbf{D},\pi_a,\boldsymbol{\rho},\mathbf{O},\pi_o)\right] \\ \text{s.t.} \quad & h_q(\mathbf{X}_o, \mathbf{U}_o) \leq 0, \ \text{for all } q = 1, \ldots, Q, \\ & \pi_o(\mathbf{U}_o|\mathbf{X}_o, \mathbf{X}_a) \in \mathcal{U}_o, \end{aligned} \tag{11}$$

where $\{\beta_j\}_{j=1}^{m}$ are regularization parameters. The objective function $f(\mathbf{S}, \mathbf{D}, \pi_a, \boldsymbol{\rho}, \mathbf{O}, \pi_o)$, the cost functions $\{g_j(\mathbf{S}, \mathbf{D}, \pi_a, \boldsymbol{\rho}, \mathbf{O}, \pi_o)\}_{j=1}^{m}$, the constraint functions $\{h_q(\mathbf{X}_o, \mathbf{U}_o)\}_{q=1}^{Q}$ and the action space $\mathcal{U}_o$ are not necessarily convex/non-convex depending on specific application scenarios. The problem is challenging due to four main reasons:

i) The closed-form expression of the objective function $f(\mathbf{S}, \mathbf{D}, \pi_a, \boldsymbol{\rho}, \mathbf{O}, \pi_o)$ is difficult to derive because the explicit relationship between agents, obstacles and navigation performance is difficult to model, precluding

the application of conventional model-based methods and leading to poor performance of heuristic methods.

ii) The imposed constraints $\{h_q(\mathbf{X}_o, \mathbf{U}_o)\}_{q=1}^{Q}$ restrict the space of feasible solutions and are difficult to address, leading to the failure of conventional unconstrained optimization algorithms.

iii) The policy $\pi_o(\mathbf{U}_o|\mathbf{X}_o, \mathbf{X}_a)$ is an arbitrary mapping from the state space to the action space, which can take any function form and is infinitely dimensional.

iv) The obstacle actions can be discrete or continuous and the action space $\mathcal{U}_o$ can be non-convex, resulting in further constraints on the feasible solution.

Due to the aforementioned challenges, we propose to solve problem (11) by leveraging reinforcement learning (RL) and the primal-dual mechanism. The former parameterizes the optimization policy with information processing architectures and allows us to train the architecture parameters in a model-free manner. The latter penalizes the constraints with dual variables, and updates the primal and dual variables alternatively while searching for feasible solutions.

A. REINFORCEMENT LEARNING

We reformulate problem (11) in the RL domain and start by defining a Markov decision process. At each step t, the obstacles $\mathcal{O}$ and the agents $\mathcal{A}$ observe the states $\mathbf{X}_o^{(t)}, \mathbf{X}_a^{(t)}$ and take the actions $\mathbf{U}_o^{(t)}, \mathbf{U}_a^{(t)}$ with the obstacle policy π_o and the trajectory planner π_a, respectively. The actions $\mathbf{U}_o^{(t)}, \mathbf{U}_a^{(t)}$ amend the states $\mathbf{X}_o^{(t)}, \mathbf{X}_a^{(t)}$ based on the transition probability function $P(\mathbf{X}_o^{(t+1)}, \mathbf{X}_a^{(t+1)}|\mathbf{X}_o^{(t)}, \mathbf{X}_a^{(t)}, \mathbf{U}_o^{(t)}, \mathbf{U}_a^{(t)})$, which is a distribution over the states conditioned on the previous states and the actions. Let $r_{ai}(\mathbf{X}_o^{(t)}, \mathbf{X}_a^{(t)}, \mathbf{U}_o^{(t)}, \mathbf{U}_a^{(t)})$ be the reward function of agent A_i, which represents its navigation performance at step t. The reward function of obstacle O_j comprises two components: (i) the global system reward and (ii) the local obstacle reward, i.e.,

$$r_{oj} = \frac{1}{n}\sum_{i=1}^{n} \rho_i r_{ai} + \beta_j r_{j,local}, \text{ for all } j = 1, \ldots, m \tag{12}$$

where ρ_i is the priority of agent A_i, β_j is the regularization parameter of obstacle O_j, and r_{ai}, $r_{j,local}$ are concise notations of $r_{ai}(\mathbf{X}_o^{(t)}, \mathbf{X}_a^{(t)}, \mathbf{U}_o^{(t)}, \mathbf{U}_a^{(t)})$, $r_{j,local}(\mathbf{X}_o^{(t)}, \mathbf{X}_a^{(t)}, \mathbf{U}_o^{(t)}, \mathbf{U}_a^{(t)})$. The first term is the team reward that represents the multi-agent system performance, which is shared over all obstacles. The second term is the individual reward that corresponds to the position change penalty of obstacle O_j, which may be different among obstacles, e.g., the collision penalty. This reward definition is novel that differs from common RL scenarios, which is a combination of a global reward with a local reward. The former is the main goal of all obstacles, while the latter is the individual cost of an obstacle. The priorities $\boldsymbol{\rho}$ weigh the agents' performance to put more emphasis on the agents with higher priorities, while the regularization parameters $\{\beta_j\}_{j=1}^{m} \in [0, 1]^m$ weigh the environment optimization penalty on the navigation performance. The total reward of the obstacles is defined as

$$r_o = \sum_{j=1}^{m} r_{oj} = \frac{m}{n}\sum_{i=1}^{n} \rho_i r_{ai} + \sum_{j=1}^{m} \beta_j r_{j,local}. \tag{13}$$

Let $\gamma \in [0, 1]$ be the discount factor accounting for the future rewards and the expected discounted reward can be represented as

$$\begin{aligned} R(\mathbf{S}, \mathbf{D}, \pi_a, \boldsymbol{\rho}, \mathbf{O}, \pi_o) &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_o^{(t)}\right] \\ &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \sum_{j=1}^{m}\sum_{i=1}^{n} \frac{\rho_i r_{ai}^{(t)}}{n}\right] + \sum_{j=1}^{m} \beta_j \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{j,local}^{(t)}\right] \end{aligned} \tag{14}$$

where $\mathbf{O}$, $\mathbf{S}$ and $\mathbf{D}$ are the initial positions and destinations that determine the initial states $\mathbf{X}_o^{(0)}$ and $\mathbf{X}_a^{(0)}$, and $\mathbb{E}[\cdot]$ is w.r.t. the action policy and the state transition probability. The obstacles are imposed with Q constraints at each step t, which are functions of the obstacle states $\mathbf{X}_o^{(t)}$ and actions $\mathbf{U}_o^{(t)}$ as $h_q(\mathbf{X}_o^{(t)}, \mathbf{U}_o^{(t)}) \leq 0$ for $q = 1, \ldots, Q$. By parameterizing the obstacle policy π_o with an information processing architecture $\boldsymbol{\Phi}(\mathbf{X}_o, \mathbf{X}_a, \boldsymbol{\theta})$ of parameters $\boldsymbol{\theta}$, we formulate the constrained reinforcement learning problem as

$$\begin{aligned} &\operatorname{argmax}_{\boldsymbol{\theta}} R(\mathbf{S}, \mathbf{D}, \pi_a, \boldsymbol{\rho}, \mathbf{O}, \boldsymbol{\theta}) \\ &\text{s.t. } h_q\left(\mathbf{X}_o^{(t)}, \mathbf{U}_o^{(t)}\right) \leq 0, \text{ for } q = 1, \ldots, Q, \ t = 0, 1, \ldots, \infty, \\ &\boldsymbol{\Phi}\left(\mathbf{X}_o^{(t)}, \mathbf{X}_a^{(t)}, \boldsymbol{\theta}\right) \in \mathcal{U}_o, \text{ for } t = 0, 1, \ldots, \infty. \end{aligned} \tag{15}$$

By comparing problem (11) with problem (15), we see equivalent representations of the objective, cost and constraint functions in the RL domain. The goal is to find the optimal parameters $\boldsymbol{\theta}^*$ that maximize the reward while satisfying the constraints at each step t.

B. PRIMAL-DUAL POLICY GRADIENT

Since there is no straightforward way to optimize $\boldsymbol{\theta}$ w.r.t. per-step hard constraints in problem (15), we define the reward of the constraint $h_q(\mathbf{X}_o^{(t)}, \mathbf{U}_o^{(t)}) \leq 0$ with an indicator function as

$$r_q^{(t)} = \mathbb{1}\left[h_q\left(\mathbf{X}_o^{(t)}, \mathbf{U}_o^{(t)}\right) \leq 0\right], \text{ for } q = 1, \ldots, Q \tag{16}$$

where $\mathbb{1}[h_q(\mathbf{X}_o^{(t)}, \mathbf{U}_o^{(t)}) \leq 0]$ is 1 if $h_q(\mathbf{X}_o^{(t)}, \mathbf{U}_o^{(t)}) \leq 0$ and 0 otherwise. It quantifies the constraint by rewarding success and penalizing failure at each step t. The cumulative reward of the constraint $h_q(\mathbf{X}_o^{(t)}, \mathbf{U}_o^{(t)}) \leq 0$ is

$$R_q(\mathbf{O}, \boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_q^{(t)}\right], \text{ for } q = 1, \ldots, Q \tag{17}$$

where γ is the discount factor. The preceding expression allows us to measure how well the constraints are satisfied in expectation and takes the same form as the expected discounted reward of the obstacles [cf. (14)]. We can then transform the constraints as

$$R_q(\mathbf{O}, \boldsymbol{\theta}) \geq \mathcal{C}_q, \text{ for } q = 1, \ldots, Q \tag{18}$$

where $\{\mathcal{C}_q\}_{q=1}^{Q}$ are constants that lower-bound the constraint rewards and control the constraint guarantees – see details in Section V-C. This yields an alternative of problem (15) as

$$\begin{aligned}\mathbb{P} := \underset{\boldsymbol{\theta}}{\text{argmax}}\ & R(\mathbf{S}, \mathbf{D}, \pi_a, \boldsymbol{\rho}, \mathbf{O}, \boldsymbol{\theta}) \\ \text{s.t. } & R_q(\mathbf{O}, \boldsymbol{\theta}) \geq \mathcal{C}_q, \text{ for } q = 1, \ldots, Q, \\ & \boldsymbol{\Phi}\left(\mathbf{X}_o^{(t)}, \mathbf{X}_a^{(t)}, \boldsymbol{\theta}\right) \in \mathcal{U}_o, \text{ for } t = 0, 1, \ldots, \infty. \end{aligned} \tag{19}$$

By introducing the dual variables $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_Q]^\top \in \mathbb{R}_+^Q$ that correspond to Q constraints, we formulate the Lagrangian of problem (19) as

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}) = R(\mathbf{S},\mathbf{D},\pi_a,\boldsymbol{\rho},\mathbf{O},\boldsymbol{\theta}) + \sum_{q=1}^{Q} \lambda_q \left(R_q(\mathbf{O}, \boldsymbol{\theta}) - \mathcal{C}_q\right) \tag{20}$$

which penalizes the objective with the constraint violation weighted by the dual variables $\boldsymbol{\lambda}$. The dual function is defined as the maximum of the Lagrangian $\mathcal{D}(\boldsymbol{\lambda}) := \max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda})$. Since $\mathcal{D}(\boldsymbol{\lambda}) \geq \mathbb{P}$ for any dual variables $\boldsymbol{\lambda}$, we define the dual problem as

$$\mathbb{D} := \min_{\boldsymbol{\lambda} \geq 0} \mathcal{D}(\boldsymbol{\lambda}) = \min_{\boldsymbol{\lambda} \geq 0} \max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}) \tag{21}$$

which computes the optimal dual variables that minimizes the dual function $\mathcal{D}(\boldsymbol{\lambda})$. The dual solution $\mathbb{D}$ in (21) can be considered as the best approximation of the primal solution $\mathbb{P}$ in (19). This translates a constrained maximization problem to an unconstrained min-max problem, which searches for a saddle point solution $(\boldsymbol{\theta}^*, \boldsymbol{\lambda}^*)$ that is maximal w.r.t. the primal variables $\boldsymbol{\theta}$ and minimal w.r.t. the dual variables $\boldsymbol{\lambda}$.

We propose to solve the dual problem (21) with the primal-dual policy gradient method, which updates $\boldsymbol{\theta}$ with policy gradient ascent and $\boldsymbol{\lambda}$ with gradient descent in an alternative manner. Specifically, it trains the model iteratively and each iteration consists of primal and dual steps.

Primal step: At iteration k, let $\boldsymbol{\theta}^{(k)}$ be the primal variables, $\boldsymbol{\lambda}^{(k)}$ the dual variables, $\mathbf{X}_o^{(k)}$ the obstacle states, and $\mathbf{X}_a^{(k)}$ the agent states. Given these system states, the obstacle policy $\boldsymbol{\Phi}(\mathbf{X}_o^{(k)}, \mathbf{X}_a^{(k)}, \boldsymbol{\theta}^{(k)})$ generates the obstacle actions $\mathbf{U}_o^{(k)}$ and the given trajectory planner π_a generates the agent actions $\mathbf{U}_a^{(k)}$. These actions $\mathbf{U}_o^{(k)}$, $\mathbf{U}_a^{(k)}$ change the states from $\mathbf{X}_o^{(k)}$, $\mathbf{X}_a^{(k)}$ to $\mathbf{X}_{o,1}^{(k)}$, $\mathbf{X}_{a,1}^{(k)}$ based on the transition probability function $P(\mathbf{X}_{o,1}^{(k)}, \mathbf{X}_{a,1}^{(k)} | \mathbf{X}_o^{(k)}, \mathbf{X}_a^{(k)}, \mathbf{U}_o^{(k)}, \mathbf{U}_a^{(k)})$, which feeds back the corresponding obstacle reward $r_o^{(k)}$ [cf. (13)] and the constraint rewards $\{r_q^{(k)}\}_{q=1}^{Q}$ [cf. (16)].

We follow the actor-critic method to define a value function $V(\mathbf{X}_o, \mathbf{X}_a, \boldsymbol{\omega})$ of parameters $\boldsymbol{\omega}$ that estimates the Lagrangian (20) initialized at the states $\mathbf{X}_o$, $\mathbf{X}_a$. Let $\boldsymbol{\omega}^{(k)}$ be the parameters of the value function at iteration k, and $V(\mathbf{X}_o^{(k)}, \mathbf{X}_a^{(k)}, \boldsymbol{\omega}^{(k)})$, $V(\mathbf{X}_{o,1}^{(k)}, \mathbf{X}_{a,1}^{(k)}, \boldsymbol{\omega}^{(k)})$ be the estimated values at $\mathbf{X}_o^{(k)}$, $\mathbf{X}_a^{(k)}$ and $\mathbf{X}_{o,1}^{(k)}$, $\mathbf{X}_{a,1}^{(k)}$. This allows us to compute the estimation error as

$$\delta_1^{(k)} = r_o^{(k)} + \sum_{q=1}^{Q} \lambda_q^{(k)} \left(r_q^{(k)} - (1-\gamma)\mathcal{C}_q\right)$$

Algorithm 1: Primal-Dual Policy Gradient Method.

1: **Input:** initial primal variables $\boldsymbol{\theta}^{(0)}$, initial dual variables $\boldsymbol{\lambda}^{(0)}$, initial value function parameters $\boldsymbol{\omega}^{(0)}$, trajectory planner π_a and transition probability function P
2: **for** $k = 0, 1, \ldots$ **do**
3: Compute value functions $V(\mathbf{X}_o^{(k)}, \mathbf{X}_a^{(k)}, \boldsymbol{\omega}^{(k)})$ and $V(\mathbf{X}_o^{(k+1)}, \mathbf{X}_a^{(k+1)}, \boldsymbol{\omega}^{(k)})$ to estimate the Lagrangian values of (20)
4: Compute the estimation error as in (22)
5: Update the primal variables $\boldsymbol{\theta}^{(k)}$ with policy gradient ascent as in (23) ψ times and set $\boldsymbol{\theta}^{(k+1)} := \boldsymbol{\theta}_{\psi}^{(k)}$ as the updated primal variables
6: Compute the expected constraint rewards as in (25)
7: Update the dual variables $\boldsymbol{\lambda}^{(k)}$ with gradient descent as in (24)
8: **end for**

$$+ \gamma V\left(\mathbf{X}_{o,1}^{(k)}, \mathbf{X}_{a,1}^{(k)}, \boldsymbol{\omega}^{(k)}\right) - V\left(\mathbf{X}_o^{(k)}, \mathbf{X}_a^{(k)}, \boldsymbol{\omega}^{(k)}\right) \tag{22}$$

which is used to update the parameters $\boldsymbol{\omega}^{(k)}$ of the value function. Then, we can update the primal variables $\boldsymbol{\theta}^{(k)}$ with policy gradient ascent as

$$\boldsymbol{\theta}_1^{(k)} = \boldsymbol{\theta}^{(k)} + \alpha \nabla_{\boldsymbol{\theta}} \log \pi_o\left(\mathbf{U}_o^{(k)} | \mathbf{X}_a^{(k)}, \mathbf{X}_o^{(k)}\right) \delta_1^{(k)} \tag{23}$$

where α is the step-size and $\pi_o(\mathbf{U}_o^{(k)} | \mathbf{X}_a^{(k)}, \mathbf{X}_o^{(k)})$ is the policy distribution of the obstacle action specified by the parameters $\boldsymbol{\theta}^{(k)}$. The primal update is model-free because (23) requires only the error value $\delta_1^{(k)}$ and the gradient of the policy distribution, but not the objective, cost and constraint function models. By performing the above procedure recursively $\psi \in \mathbb{Z}_+$ times, we obtain a sequence of primal variables $\{\boldsymbol{\theta}_1^{(k)}, \boldsymbol{\theta}_2^{(k)}, \ldots, \boldsymbol{\theta}_{\psi}^{(k)}\}$. Define $\boldsymbol{\theta}^{(k+1)} := \boldsymbol{\theta}_{\psi}^{(k)}$ as the new primal variables and step into the dual step.

Dual step: With the updated primal variables $\boldsymbol{\theta}^{(k+1)}$, we update the dual variables $\boldsymbol{\lambda}^{(k)}$ with gradient descent as

$$\lambda_q^{(k+1)} = \left[\lambda_q^{(k)} - \beta \left(R_q\left(\mathbf{O}, \boldsymbol{\theta}^{(k+1)}\right) - \mathcal{C}_q\right)\right]_+ \text{ for } q = 1, \ldots, Q \tag{24}$$

where β is the step-size and $[\cdot]_+$ is the non-negative operator corresponding to the positivity of dual variables. Since the constraint function values $\{h_q(\mathbf{X}_o^{(k)}, \mathbf{U}_o^{(k)})\}_{q=1}^{Q}$ are given, the cumulative constraint rewards $\{R_q(\mathbf{O}, \boldsymbol{\theta}^{(k)})\}_{q=1}^{Q}$ can be estimated by sampling $\mathcal{T}$ trajectories $\{\mathbf{X}_o^{(t),1}, \mathbf{U}_o^{(t),1}\}_t$, …, $\{\mathbf{X}_o^{(t),\mathcal{T}}, \mathbf{U}_o^{(t),\mathcal{T}}\}_t$, computing the respective cumulative constraint rewards, and taking the average, i.e.,

$$R_q\left(\mathbf{O}, \boldsymbol{\theta}^{(k+1)}\right) \approx \frac{1}{\mathcal{T}} \sum_{\tau=1}^{\mathcal{T}} \sum_{t=0}^{T_\tau} \gamma^t \mathbb{1}\left[h_q\left(\mathbf{X}_o^{(t),\tau}, \mathbf{U}_o^{(t),\tau}\right) \leq 0\right] \tag{25}$$

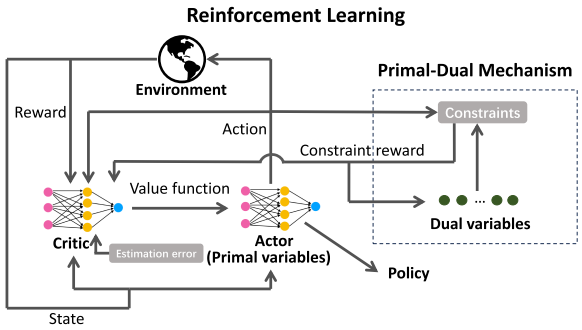


FIGURE 2. Framework of primal-dual reinforcement learning.

where T_τ is the operation time of the τth trajectory. The dual update is model-free because (24) and (25) require only the constraint values $\{h_q(\mathbf{X}_o^{(t)}, \mathbf{U}_o^{(t)})\}_t$ instead of the constraint function models. Algorithm 1 summarizes the approach and Fig. 2 demonstrates the methodology framework.

C. CONSTRAINT GUARANTEES

While the cumulative constraint rewards in the alternative problem (19) is a relaxation of the per-step hard constraints in the original problem (15), we provide constraint guarantees for the alternative problem (19) in the sequel. Specifically, the expectation of the indicator function in (17) is equivalent to the probability of satisfying the constraint, i.e.,

$$\mathbb{E}\left[\mathbb{1}\left[h_q\left(\mathbf{X}_o^{(t)}, \mathbf{U}_o^{(t)}\right) \leq 0\right]\right] = \mathbb{P}\left[h_q\left(\mathbf{X}_o^{(t)}, \mathbf{U}_o^{(t)}\right) \leq 0\right] \quad (26)$$

where $\mathbb{P}[\cdot]$ is the probability measure. This allows to rewrite the cumulative constraint reward as

$$R_q(\mathbf{O}, \boldsymbol{\theta}) = \sum_{t=0}^{\infty} \gamma^t \mathbb{P}\left[h_q\left(\mathbf{X}_o^{(t)}, \mathbf{U}_o^{(t)}\right) \leq 0\right] \geq \mathcal{C}_q \quad (27)$$

for $q = 1, \ldots, Q$ [cf. (18)], which is the discounted sum of the constraint satisfaction probability over all steps t.

However, (27) are not sufficient conditions to satisfy the constraints at each step t, as required by problem (15). We then establish the relation between (27) and the per-step constraints in problem (15). Before proceeding, we define the $(1-\delta)$-constrained solution as follows.

Definition 2 ($(1-\delta)$-constrained solution): A solution of problem (19) is $(1-\delta)$-constrained w.r.t. the constraints $\{h_q\}_{q=1}^Q$ if for each step $t \geq 0$, it holds that

$$\mathbb{P}\left[\cap_{0\leq\tau\leq t}\left\{h_q\left(\mathbf{X}_o^{(\tau)}, \mathbf{U}_o^{(\tau)}\right) \leq 0\right\}\right] \geq 1-\delta \text{ for } q=1, \ldots, Q. \quad (28)$$

The $(1-\delta)$-constrained solution satisfies the constraints at each step t with a probability $1-\delta \in (0, 1]$, which is a feasible solution of problem (15) if $\delta = 0$. With these preliminaries, we analyze the constraint guarantees of the alternative problem (19) in the following theorem.

Theorem 5: Consider problem (19) with the constraint constant taking the form of $\mathcal{C}_q = (1-\delta+\epsilon)/(1-\gamma)$ for $q = 1, \ldots, Q$ [cf. (18)], where δ and ϵ are constraint parameters with $0 \leq \epsilon \leq \delta$. For any δ, there exists ϵ such that the solution of problem (19) is $(1-\delta)$-constrained.

Proof: See Appendix G. ■

Theorem 5 states that the feasible solution of the alternative problem (19) is a $(1-\delta)$-constrained solution of problem (15). The result provides the per-step guarantees for the environment constraints in probability, which demonstrates the applicability of the alternative problem (19). In essence, we have not lost the constraint guarantees by relaxing the hard constraints to the cumulative constraint rewards.

D. INFORMATION PROCESSING ARCHITECTURE

The proposed approach is a general framework that covers various environment optimization scenarios and different information processing architectures can be integrated to solve different variants. We illustrate this fact by analyzing two representative scenarios: (i) offline optimization in discrete environments and (ii) online optimization in continuous environments, for which we use convolutional neural networks (CNNs) and graph neural networks (GNNs), respectively.

CNNs for offline discrete settings: In this setting, we first optimize the obstacles' positions and then navigate the agents. Since computations are performed offline, we collect the states of all obstacles $\mathbf{X}_o$ and agents $\mathbf{X}_a$ apriori, which allows for centralized information processing solutions (e.g., CNNs). CNNs leverage convolutional filters to extract features from image signals and have found wide applications in computer vision [40], [41], [42]. In the discrete environment, the system states can be represented by matrices and the latter are equivalent to image signals. This motivates to consider CNNs for policy parameterization.

GNNs for online continuous settings: In this setting, obstacle positions change while agents move, i.e., the obstacles take actions instantaneously. In large-scale systems with real-time communication and computation constraints, obstacles may not be able to collect the states of all other obstacles/agents and centralized solutions may not be applicable. This requires a decentralized architecture that can be implemented with local neighborhood information. GNNs are inherently decentralizable and are, thus, a suitable candidate.

GNNs are layered architectures that leverage a message passing mechanism to extract features from graph signals [43], [44], [45]. At each layer ℓ, let $\mathbf{X}_{\ell-1}$ be the input signal. The output signal is generated with the message aggregation function $\mathcal{F}_{\ell m}$ and the feature update function $\mathcal{F}_{\ell u}$ as

$$[\mathbf{X}_\ell]_i = \mathcal{F}_{\ell u}\left([\mathbf{X}_{\ell-1}]_i, \sum_{j\in\mathcal{N}_i} \mathcal{F}_{\ell m}\left([\mathbf{X}_{\ell-1}]_i, [\mathbf{X}_{\ell-1}]_j, [\mathbf{E}]_{ij}\right)\right)$$

where $[\mathbf{X}_\ell]_i$ is the signal value at node i, $\mathcal{N}_i$ are the neighboring nodes within the communication radius, $\mathbf{E}$ is the adjacency matrix, and $\mathcal{F}_{\ell m}$, $\mathcal{F}_{\ell u}$ have learnable parameters $\boldsymbol{\theta}_{\ell m}$, $\boldsymbol{\theta}_{\ell u}$. The output signal is computed with local neighborhood information only, and each node can make decisions based on its own

output value; hence, allowing for decentralized implementation [46], [47], [48], [49].

VI. EXPERIMENTS

We evaluate the proposed approach in this section. First, we consider unprioritized environment optimization without constraints [Problem 1]. Then, we consider prioritized environment optimization with constraints [Problem 2]. For both cases, we consider two navigation scenarios, one in which we perform offline optimization with discrete obstacle motion, and another in which we consider online optimization with continuous obstacle motion. The obstacles have rectangular bodies and the agents have circular bodies. The given agent trajectory planner π_a is the Reciprocal Velocity Obstacles (RVO) method [3], which can be applied in continuous space and allows for an efficient computation with decentralized implementation.[3] Two metrics are used: Success weighted by Path Length (SPL) [50] and the percentage of the maximal speed (PCTSpeed). The former is a stringent measure combining the success rate and the path length, which has been widely used as a primary quantifier in comparing the navigation performance [50], [51], [52]. The latter is the ratio of the average speed to the maximal one, which provides complementary information regarding the moving speed along trajectories. Moreover, SPL and PCTSpeed normalize metric values to [0, 1] with a higher value representing a better performance, which provides a unified exposition for performance metrics. Results are averaged over 20 trials with random initial configurations.

A. UNPRIORITIZED ENVIRONMENT OPTIMIZATION

We start with unprioritized environment optimization.

Offline discrete setting: We consider a grid environment of size 8 m $\times$ 8 m with 10 obstacles and 4 agents, which are distributed randomly without overlap. The maximal agent / obstacle velocity is 0.05 m per time step Δt and the maximal time step is 500.

Setup: The environment is modeled as an occupancy grid map. An agent's location is modeled by a one-hot in a matrix of the same dimension. At each step, the policy considers one of the obstacles and moves it to one of the adjacent grid cells. This repeats for m obstacles, referred to as a round, and an episode ends if the maximal round has been reached.

Training: The objective is to make agents reach destinations quickly while avoiding collision. The team reward is the sum of the PCTSpeed and the ratio of the shortest distance to the traveled distance, while the local reward is the collision penalty of individual obstacle [cf. (12)]. We parameterize the policy with a CNN of 4 layers, each containing 25 features with kernel size 2 $\times$ 2, and conduct training with PPO [53].

Baseline: Since exhaustive search methods are intractable for our problem, we develop a strong heuristic method to act as a baseline: At each step, one of the obstacles computes the shortest paths of all agents, checks whether it blocks any of these paths, and moves away randomly if so. This repeats for m obstacles, referred to as a round, and the method ends if the maximal round is reached.

Performance: We train our model on 10 obstacles and test on 10 to 18 obstacles, which varies obstacle density from 10% to 30%. Fig. 3(a) and (b) show the results. The proposed approach consistently outperforms baselines with the highest SPL/PCTSpeed and the lowest variance. The heuristic method takes the second place and the original scenario (without any environment modification) performs worst. As we generalize to higher obstacle densities, all methods degrade as expected. However, our approach maintains a satisfactory performance due to the CNN's cabability for generalization. Fig. 3(c) and (d) show an example of how the proposed approach circumvents the dead-locks by optimizing the obstacle layout. Moreover, it improves the path efficiency such that all agents find collision-free trajectories close to their shortest paths.

Online continuous setting: We proceed to a continuous environment. The agents are initialized randomly in an arbitrarily defined starting region and towards destinations in an arbitrarily defined goal region.

Setup: The agents and obstacles are modeled by positions $\{\mathbf{p}_{a,i}\}_{i=1}^{n}$, $\{\mathbf{p}_{o,j}\}_{j=1}^{m}$ and velocities $\{\mathbf{v}_{a,i}\}_{i=1}^{n}$, $\{\mathbf{v}_{o,j}\}_{j=1}^{m}$. At each step, each obstacle has a local policy that generates the desired velocity with neighborhood information and we integrate an acceleration-constrained mechanism for position changes. An episode ends if all agents reach destinations or the episode times out. The maximal acceleration is 0.05 m/Δt^2, the communication radius is 2 m and the episode time is 500 time steps.

Training: The team reward in (12) guides the agents to their destinations as quickly as possible and is defined as

$$r_{a,i}^{(t)} = \left(\frac{\mathbf{p}_i^{(t)} - \mathbf{d}_i}{\|\mathbf{p}_i^{(t)} - \mathbf{d}_i\|_2} \cdot \frac{\mathbf{v}_i^{(t)}}{\|\mathbf{v}_i^{(t)}\|_2} \right) \|\mathbf{v}_i^{(t)}\|_2 \tag{29}$$

at time step t, which rewards fast movements towards the destination and penalizes moving away from it. The local reward is the collision penalty. We parameterize the policy with a single-layer GNN. The message aggregation function and feature update function are multi-layer perceptrons (MLPs), and we train the model using PPO.

Performance: The results are shown in Fig. 4(a) and (b). The proposed approach exhibits the best performance for both metrics and maintains a good performance for large scenarios. We attribute the latter to the fact that GNNs exploit topological information for feature extraction and are scalable to larger graphs. The heuristic method performs worse but comparably for small number of obstacles, while degrading with the increasing of obstacles. It is note-worthy that the heuristic method is centralized because it requires computing shortest paths of all agents, and hence is not applicable for online optimization and considered here as a benchmark value

[3]The proposed environment optimization approach can be applied in the same manner, irrespective of the chosen agent navigation algorithm.

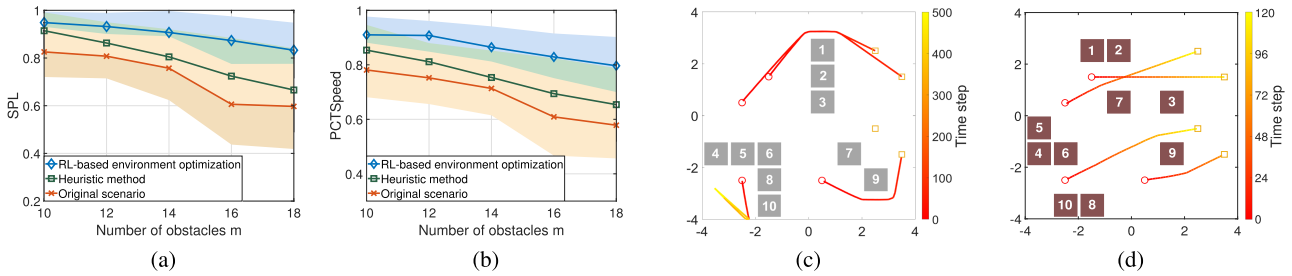


FIGURE 3. (a) And (b) performance of offline environment optimization compared to the baselines. Results are averaged over 20 trials and the shaded area shows the std. dev. The RL system is trained on 10 obstacles and tested on 10 to 18 obstacles. (a) SPL (1 is best). (b) PCTSpeed. (c) And (d) example of offline environment optimization. Red circles are initial positions, yellow squares are destinations, and obstacles are numbered for exposition. Color lines from red to yellow are agent trajectories and the color bar represents the time scale. Obstacle layout and agent trajectories (c) *before* environment optimization and (d) *after* environment optimization.

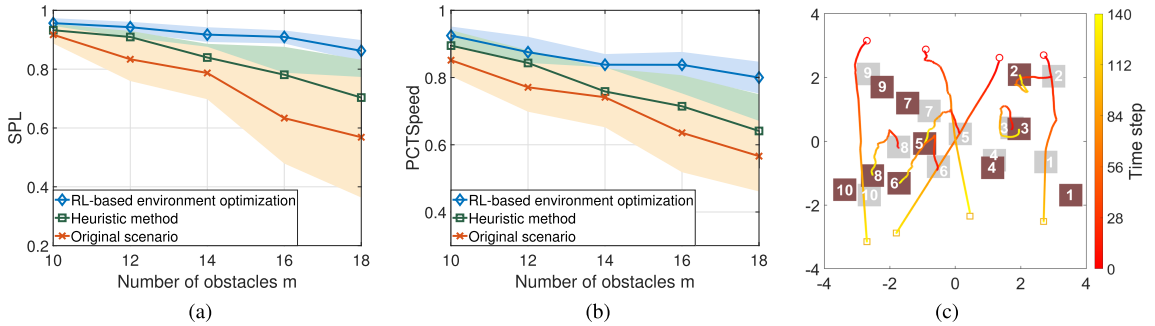


FIGURE 4. (a) And (b) performance of online environment optimization compared to the baselines. Results are averaged over 20 trails and the shaded area shows the std. dev. The RL system is trained on 10 obstacles and tested on 10 to 18 obstacles. (a) SPL (1 is best). (b) PCTSpeed. (c) Example of online environment optimization. Red circles are initial positions and yellow squares are destinations. Grey and brown rectangles are the obstacles before and after environment optimization, and are numbered for exposition. Red-to-yellow lines are trajectories of agents and 5 example obstacles, and the color bar represents the time scale, showing that no agent-agent nor agent-obstacle collision occurs.

only for reference. Fig. 4(c) shows the moving trajectories of agents and example obstacles. We see that obstacles make way for agents to facilitate navigation s.t. agents find trajectories close to their shortest paths.

B. PRIORITIZED ENVIRONMENT OPTIMIZATION

We proceed to prioritized environment optimization with real-world constraints, where the agent priorities are set as $\boldsymbol{\rho} = [2, 1, 0.5, 0.1]^\top$. The environment settings and implementation details are the same as the unprioritized environment optimization in Section VI-A.

Offline discrete setting: We consider the constraint as the maximal round that can be performed by the policy π_o, i.e., the maximal number of grids each obstacle can move, which restricts the capacity of offline environment optimization. This constraint can be satisfied by setting the maximal length of an episode in reinforcement learning and thus, the primal-dual mechanism is not needed in this setting. We consider the maximal round as 8 and measure the performance with the PCTSpeed and the distance ratio, where the latter is the ratio of the shortest distance to the agent's traveled distance.

Performance: We evaluate the proposed approach on 14 obstacles and the results are shown in Fig. 5(a). We see that agent A_1 with the highest priority exhibits the best performance with the highest PCTSpeed and distance ratio. The performance degrades as the agent priority decreases (from A_1 to A_4), which corresponds to theoretical findings in Theorem 3. Fig. 5(c) and (d) display an example of how offline prioritized environment optimization optimizes the obstacle layout. It guarantees the success of all navigation tasks and improves agents' path efficiencies based on their priorities, i.e., it emphasizes the higher-priority agents (A_1 and A_2) over the lower-priority agents (A_3 and A_4).

Partial completeness: We test the trained model on 24 obstacles, where the obstacle region is dense and the obstacle-free area is limited. The success rates of all agents are zero without environment optimization. Fig. 5(b) shows the results with offline prioritized environment optimization. Agent A_1 with the highest priority achieves the best success rate and the success rate decreases with the agent priority, which corroborates the partial completeness in Corollary 1. We note that the success rate of A_1 is not one (100%) because (i) we train our model on 14 obstacles but test it on 24 obstacles and (ii) the proposed approach obtains a local not global solution, leading to inevitable performance degradation compared with theoretical analysis.

Online continuous setting: We consider the constraint as the total deviation distance of the obstacles away from

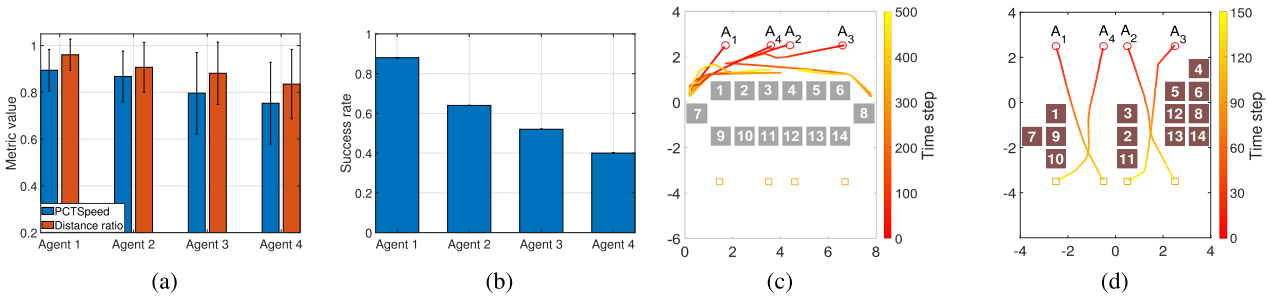


FIGURE 5. (a) And (b) performance of offline prioritized environment optimization with agent priorities $\rho_1 \geq \rho_2 \geq \rho_3 \geq \rho_4$. (a) PCTSpeeds and distance ratios of different agents on 14 obstacles (1 is best). (b) Success rates of different agents on 24 obstacles. (c) And (d) example of offline prioritized environment optimization. Red circles are initial positions, yellow squares are destinations, and obstacles are numbered for exposition. Color lines from red to yellow are agent trajectories and the color bar represents the time scale. Agent priorities $\{\rho_i\}_{i=1}^4$ reduce from A_1 to A_4, i.e., $\rho_1 \geq \cdots \geq \rho_4$. (c) Navigation failures in original environment. (d) Completeness with offline prioritized environment optimization. PCTSpeeds of agents are 0.98, 0.94, 0.88, 0.86 and distance ratios are 0.99, 0.98, 0.95, 0.93, i.e., agent performance increases with the priority.

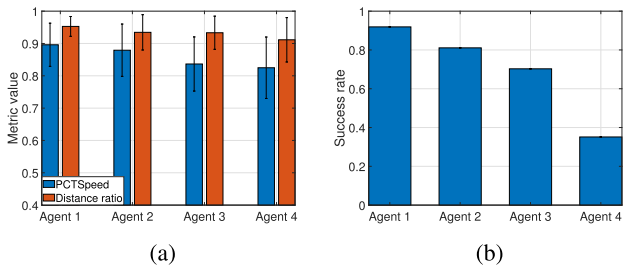


FIGURE 6. Performance of online prioritized environment optimization with agent priorities $\rho_1 \geq \rho_2 \geq \rho_3 \geq \rho_4$. (a) PCTSpeeds and distance ratios of different agents on 14 obstacles (1 is best). (b) Success rates of different agents within 150 time steps.

their initial positions, i.e., $\sum_{j=1}^{m} \|\mathbf{o}_j^{(t)} - \mathbf{o}_j^{(0)}\|_2 \leq C_d$ for all t, where $\mathbf{o}_j^{(t)}$ is the position of obstacle j at time step t and $\mathbf{o}_j^{(0)}$ is the initial position. It corresponds to practical scenarios where the obstacles are not allowed moving too far from their original positions.

Performance: We set $C_d = 10$ and show the results in Fig. 6(a). Agent A_1 with the highest priority performs best, and the performance degrades from A_1 to A_4 with the decreasing of agent priority, corroborating our analysis in Theorem 4. Fig. 8(b) shows the constraint value as a function of time steps. The total deviation distance of the obstacles is smaller than the deviation bound throughout the navigation procedure, which validates the proposed primal-dual method. Fig. 7(b) shows the moving trajectories of agents and example obstacles in online prioritized environment optimization. The obstacles change positions to improve agents' performance and the higher-priority agents (e.g., A_1) are emphasized over the lower-priority agents (e.g., A_4). For example, given the deviation distance constraint, the obstacles create an (almost) shortest path for A_1 but not for A_4.

Partial completeness: We corroborate the partial completeness in Corollary 2 by requiring that all agents arrive at destinations within $T_{\max} = 150$ time steps. Fig. 6(b) shows that the success rate decreases from higher-priority agents to lower-priority ones. The success rate of A_1 is not one (100%) because (i) the obtained solution is local not global and (ii) we impose the constraint of deviation distance on the environment optimization.

Constraint bound: We test different constraint bounds $C_d = 14, 10$ and 4, the decreasing of which corresponds to the increasing of environment restrictions. Fig. 8(b) shows that all variants satisfy the constraints, and Fig. 7 displays three examples of agent and obstacle trajectories. We see that all variants guarantee the success of navigation tasks. The variant with the largest bound performs best with agent trajectories close to the shortest paths, where agent priorities do not play an important role given sufficient resources. The variant with the lowest bound suffers from performance degradation because of the strongest constraint. It puts more emphasis on the higher-priority agents (A_1 and A_2) while overlooking the lower-priority agents (A_3 and A_4), which corroborates theoretical analysis in Section IV.

Velocity constraint: We show that the proposed approach can handle various constraints. Here, we test a constraint on the total speed of the obstacles, i.e., $\sum_{j=1}^{m} \|\mathbf{v}_j^{(t)}\|_2 \leq C_v$ for all t. We set $C_v = 3.5$ and Fig. 8(c) shows that the speed constraint is satisfied throughout the navigation procedure. The obstacle speed first increases to make way for the agents and then slows down to satisfy the constraint. Fig. 8(a) compares the performance of environment optimization with different constraints to the unconstrained counterpart. The constrained variants achieve comparable performance (slightly worse), but saves the traveled distance and the velocity energy of the obstacles.

VII. CONCLUSION

We proposed novel problems of unprioritized and prioritized environment optimization for multi-agent navigation, each of which contains offline and online variants. By conducting the completeness analysis, we provided conditions under which all navigation tasks are guaranteed success and identified the role played by agent priorities in environment optimization. We imposed constraints on the environment optimization corresponding to real-world restrictions, and

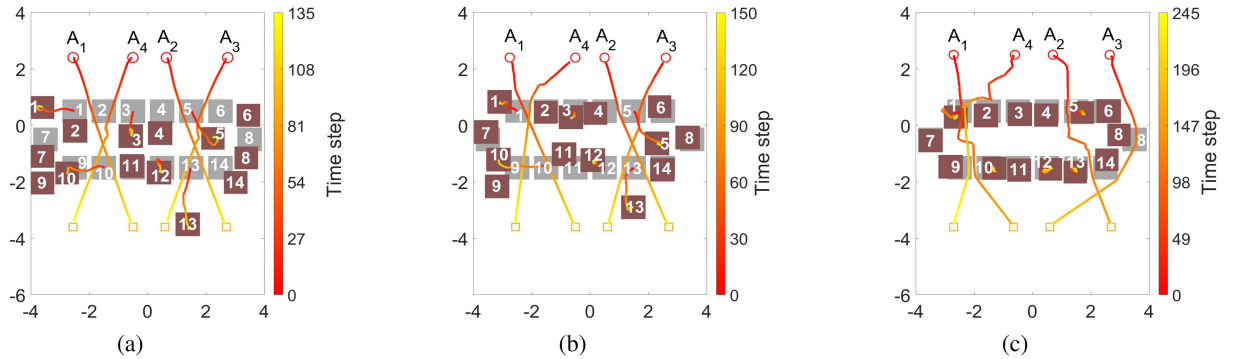


FIGURE 7. **Examples of online prioritized environment optimization. Red circles are initial positions and yellow squares are destinations. Grey and brown rectangles are obstacles before and after environment optimization, and are numbered for exposition. Color lines from red to yellow are agent trajectories and the color bar represents the time scale. Agent priorities $\{\rho_i\}_{i=1}^4$ reduce from A_1 to A_4, i.e., $\rho_1 \geq \cdots \geq \rho_4$. (a) Completeness with deviation distance constraint of $C_d = 14$. PCTSpeeds of agents are 0.98, 0.97, 0.94, 0.91 and distance ratios are 0.99, 0.99, 0.98, 0.97. (b) Completeness with deviation distance constraint of $C_d = 10$. PCTSpeeds of agents are 0.97, 0.95, 0.88, 0.84 and distance ratios are 0.98, 0.98, 0.96, 0.94. (c) Completeness with deviation distance constraint of $C_d = 4$. PCTSpeeds of agents are 0.86, 0.82, 0.76, 0.63 and distance ratios are 0.93, 0.92, 0.85, 0.82. Agent performance increases with the priority.**

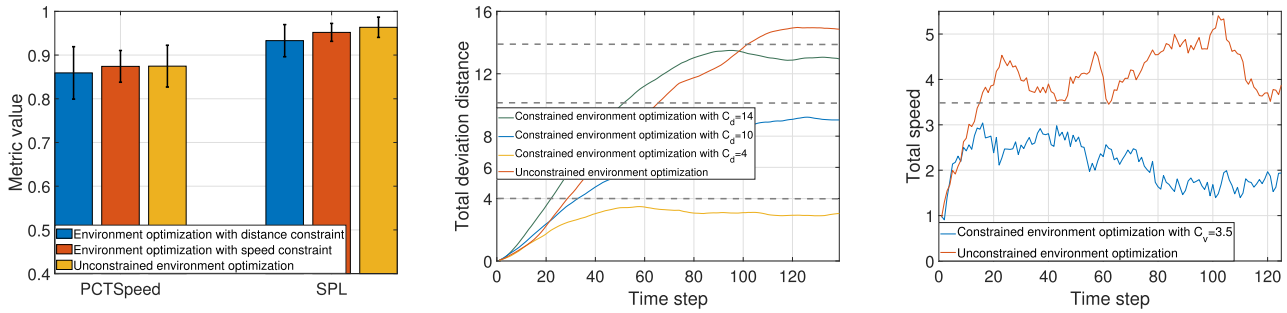


FIGURE 8. **(a) Performance of online prioritized environment optimization with different constraints. (b) Deviation distance constraint of obstacles throughout the navigation procedure. (c) Speed constraint of obstacles throughout the navigation procedure.**

formulated the latter as a constrained stochastic optimization problem. We leveraged model-free reinforcement learning together with a primal-dual mechanism to solve the problem. The former overcomes the challenge of explicitly modeling the relation between agents, environment and navigation performance, while the latter handles the constraints. By integrating different information processing architectures (e.g., CNNs and GNNs) for policy parameterization, the proposed approach can adapt to different implementation requirements. Numerical results corroborate theoretical findings, and show adaptability to various objectives and constraints.

Future work will consider the following aspects. First, we can consider agents and obstacles as a heterogeneous system and re-formulate the proposed problem from this perspective. The latter allows us to develop methods that explore the role of heterogeneity in environment optimization. Second, we only consider 2-dimensional environments. Future works will focus on higher-dimensional environments, e.g., multi-drone navigation in 3-dimensional environments or manipulator motion planning in high-dimensional configuration spaces. Third, we plan to evaluate our method in real-world experiments.

APPENDIX A
PROOF OF THEOREM 1

We prove the theorem as follows. First, we optimize Δ such that the environment is "well-formed", i.e., any initial position in $\mathcal{S}$ and destination in $\mathcal{D}$ can be connected by a collision-free path. Then, we show the optimized environment guarantees the success of all navigation tasks.

Obstacle region optimization: We first optimize Δ based on $\mathcal{S}$ and $\mathcal{D}$ to make the environment "well-formed". To do so, we handle $\mathcal{S}$, $\mathcal{D}$ and the other space $\mathcal{E} \setminus (\mathcal{S} \bigcup \mathcal{D})$ separately.

i) From Assumption 1, the initial positions $\{\mathbf{s}_i\}_{i=1}^n$ in $\mathcal{S}$ are distributed such that $d(\mathbf{s}_i, \mathbf{s}_j) \geq 2\hat{r}$ and $d(\mathbf{s}_i, \partial\mathcal{S}) \geq 2\hat{r}$. Thus, for any $\mathbf{s}_i$, there exists a boundary point $\partial\mathbf{s}_i \in \partial\mathcal{S}$ and a path $\mathbf{p}_{\mathbf{s}_i}^{\partial\mathbf{s}}$ connecting $\mathbf{s}_i$ and $\partial\mathbf{s}_i$ that is collision-free with respect to the other initial positions.

ii) Similar result applies to the destinations $\{\mathbf{d}_i\}_{i=1}^n$ in $\mathcal{D}$, i.e., for any $\mathbf{d}_i$, there exists a boundary point $\partial\mathbf{d}_i \in \partial\mathcal{D}$ and a path $\mathbf{p}_{\partial\mathbf{d}_i}^{\mathbf{d}_i}$ connecting $\partial\mathbf{d}_i$ and $\mathbf{d}_i$ that is collision-free with respect to the other destinations.

iii) Consider $\partial\mathbf{s}_i$ and $\partial\mathbf{d}_i$ for agent A_i. The shortest path $\mathbf{p}_{\partial\mathbf{s}_i}^{\partial\mathbf{d}_i}$ that connects them is the straight path, the area of

which is bounded as

$$\left|\mathbf{p}_{\partial\mathbf{s}_i}^{\partial\mathbf{d}_i}\right| \leq 2d_{\max}\hat{r} \tag{30}$$

because $d_{\max}$ is the maximal distance between $\mathcal{S}$ and $\mathcal{D}$. From $|\mathcal{E}\setminus(\Delta\cup\mathcal{S}\cup\mathcal{D})| \geq 2nd_{\max}\hat{r}$ in (4), the area of the obstacle-free space in $\mathcal{E}\setminus(\mathcal{S}\bigcup\mathcal{D})$ is larger than $2nd_{\max}\hat{r}$. Thus, we can always optimize Δ to Δ^* such that the path $\mathbf{p}_{\partial\mathbf{s}_i}^{\partial\mathbf{d}_i}$ is obstacle-free. If $\mathbf{p}_{\partial\mathbf{s}_i}^{\partial\mathbf{d}_i}$ dose not overlap with $\mathcal{S}$ and $\mathcal{D}$, we can connect $\partial\mathbf{s}_i$ and $\partial\mathbf{d}_i$ with $\mathbf{p}_{\partial\mathbf{s}_i}^{\partial\mathbf{d}_i}$ directly. If $\mathbf{p}_{\partial\mathbf{s}_i}^{\partial\mathbf{d}_i}$ passes through $\mathcal{S}$ for K times, let $\mathbf{s}_{i,e}^{(k)}$ and $\mathbf{s}_{i,l}^{(k)}$ be the entering and leaving positions of $\mathbf{p}_{\partial\mathbf{s}_i}^{\partial\mathbf{d}_i}$ on $\mathcal{S}$ at kth pass for $k=1,\ldots,K$ with $\mathbf{s}_{i,l}^{(0)}=\partial\mathbf{s}_i$ the initial leaving position. First, we can connect $\mathbf{s}_{i,l}^{(k-1)}$ and $\mathbf{s}_{i,e}^{(k)}$ by $\mathbf{p}_{\partial\mathbf{s}_i}^{\partial\mathbf{d}_i}$ because $\mathbf{p}_{\partial\mathbf{s}_i}^{\partial\mathbf{d}_i}$ is obstacle-free. Then, there exists a collision-free path $\mathbf{p}_i^{(k)}$ inside $\mathcal{S}$ that connects $\mathbf{s}_{i,e}^{(k)}$ and $\mathbf{s}_{i,l}^{(k)}$ as described in (i). Same result applies to that $\mathbf{p}_{\partial\mathbf{s}_i}^{\partial\mathbf{d}_i}$ passes through $\mathcal{D}$. Therefore, we can connect $\partial\mathbf{s}_i$ and $\partial\mathbf{d}_i$ with $\mathbf{p}_{\partial\mathbf{s}_i}^{\partial\mathbf{d}_i}$ and $\{\mathbf{p}_i^{(k)}\}_{k=1}^K$.

By concatenating $\mathbf{p}_{\mathbf{s}_i}^{\partial\mathbf{s}_i}$, $\mathbf{p}_{\partial\mathbf{s}_i}^{\partial\mathbf{d}_i}$, $\{\mathbf{p}_i^{(k)}\}_{k=1}^K$ and $\mathbf{p}_{\partial\mathbf{d}_i}^{\mathbf{d}_i}$, we can establish the path $\mathbf{p}_{\mathbf{s}_i}^{\mathbf{d}_i}$ connecting $\mathbf{s}_i$ to $\mathbf{d}_i$ that is collision-free w.r.t. the other initial positions, destinations and the optimized obstacle region Δ^* for $i=1,\ldots,n$, i.e., the optimized environment is "well-formed".

Completeness: From Assumption 2 and the fact that the optimized environment is "well-formed", Theorem 4 in [27] shows that all navigation tasks will be carried out successfully without collision. Therefore, there exists an offline environment optimization scheme that guarantees the success of all navigation tasks completing the proof.

APPENDIX B
PROOF OF THEOREM 2

We prove the theorem as follows. First, we separate the navigation procedure into H time slices. Then, we optimize the obstacle region based on the agent positions at each time slice and show the completeness of individual time-sliced multi-agent navigation. Lastly, we show the completeness of the entire multi-agent navigation by concatenating individual time slices and complete the proof by limiting the number of time slices to the infinity, i.e., $H\to\infty$.

Navigation procedure separation: Let T be the maximal operation time of trajectories $\{\mathbf{p}_i\}_{i=1}^n$ and $\{[(h-1)T/H, hT/H]\}_{h=1}^H$ the separated time slices. This yields intermediate positions $\{\mathbf{p}_i(hT/H)\}_{h=0}^H$ with $\mathbf{p}_i(0)=\mathbf{s}_i$ and $\mathbf{p}_i(T)=\mathbf{d}_i$ for $i=1,\ldots,n$. We can re-formulate the navigation task into H sub-navigation tasks, where the hth sub-navigation task of agent A_i is from $\mathbf{p}_i((h-1)T/H)$ to $\mathbf{p}_i(hT/H)$ and the operation time of the sub-navigation task is $\delta t=T/H$. At each time slice, we first change the obstacle region based on the corresponding sub-navigation task and then navigate the agents until the next time slice.

Obstacle region optimization: We consider each sub-navigation task separately and start from the 1st one. Assume the obstacle region Δ satisfies

$$|\mathcal{E}\setminus(\Delta\cup\mathcal{S}\cup\mathcal{D})| > 2n\hat{r}\|\hat{\mathbf{v}}\|_2\delta t. \tag{31}$$

For the 1st sub-navigation task, the starting region is $\mathcal{S}^{(1)}=\bigcup_{i=1,\ldots,n}B(\mathbf{p}_i(0),r_i)=\mathcal{S}$ and the destination region is $\mathcal{D}^{(1)}=\bigcup_{i=1,\ldots,n}B(\mathbf{p}_i(T/H),r_i)$. We optimize Δ based on $\mathcal{S}^{(1)}$, $\mathcal{D}^{(1)}$ and show the completeness of the 1st sub-navigation task, which consists of two steps. First, we change Δ to $\widetilde{\Delta}$ such that $|\Delta|=|\widetilde{\Delta}|$ and $\widetilde{\Delta}\subset\mathcal{E}\setminus(\mathcal{S}^{(1)}\cup\mathcal{D}^{(1)})$. This can be completed as follows. From the condition $\Delta\subset\mathcal{E}\setminus(\mathcal{S}\cup\mathcal{D})$ and $\mathcal{S}^{(1)}=\mathcal{S}$, there is no overlap between Δ and $\mathcal{S}^{(1)}$. For any overlap region $\Delta\bigcap\mathcal{D}^{(1)}$, we can change it to the obstacle-free region in $\mathcal{D}$ because $\Delta\subset\mathcal{E}\setminus(\mathcal{S}\cup\mathcal{D})$ and $|\mathcal{D}^{(1)}|=|\mathcal{D}|$, and keep the other region in Δ unchanged. The resulting $\widetilde{\Delta}$ satisfies $|\widetilde{\Delta}|=|\Delta|$ and $\widetilde{\Delta}\subset\mathcal{E}\setminus(\mathcal{S}^{(1)}\cup\mathcal{D}^{(1)})$. The changed area from Δ to $\widetilde{\Delta}$ is bounded by $|\mathcal{D}^{(1)}|\leq n\pi\hat{r}^2$. Second, we change $\widetilde{\Delta}$ to $\Delta^{(1)}$ such that the environment is "well-formed" w.r.t. the 1st sub-navigation task. The initial position $\mathbf{p}_i(0)$ and the destination $\mathbf{p}_i(H/T)$ can be connected by a path $\mathbf{p}_i^{(1)}$ that follows the trajectory $\mathbf{p}_i$. Since $\|\hat{\mathbf{v}}\|_2$ is the maximal speed and δt is the operation time, the area of $\mathbf{p}_i^{(1)}$ is bounded by $2\hat{r}\|\hat{\mathbf{v}}\|_2\delta t$. Since this holds for all $i=1,\ldots,n$, we have $\sum_{i=1}^n|\mathbf{p}_i^{(1)}|\leq 2n\hat{r}\|\hat{\mathbf{v}}\|_2\delta t$. From (31), $|\mathcal{S}^{(1)}|=|\mathcal{S}|$, $|\mathcal{D}^{(1)}|=|\mathcal{D}|$ and $|\widetilde{\Delta}|=|\Delta|$, we have

$$|\mathcal{E}\setminus\left(\widetilde{\Delta}\cup\mathcal{S}^{(1)}\cup\mathcal{D}^{(1)}\right)| > 2n\hat{r}\|\hat{\mathbf{v}}\|_2\delta t. \tag{32}$$

This implies that the obstacle-free area in $\mathcal{E}$ is larger than the area of n paths $\{\mathbf{p}_i^{(1)}\}_{i=1}^n$. Following the proof of Theorem 1, we can optimize $\widetilde{\Delta}$ to $\Delta^{(1)}$ to guarantee the success of the 1st sub-navigation task. The changed area from $\widetilde{\Delta}$ to $\Delta^{(1)}$ is bounded by $2n\hat{r}\|\hat{\mathbf{v}}\|_2\delta t-n\pi\hat{r}^2$ because the initial positions $\{\mathbf{p}_i(0)\}_{i=1}^n$ and destinations $\{\mathbf{p}_i(T/H)\}_{i=1}^n$ in $\{\mathbf{p}_i^{(1)}\}_{i=1}^n$ are collision-free from the first step, which dose not require any further change of the obstacle region. The total changed area from Δ to $\Delta^{(1)}$ can be bounded as

$$\frac{\left|\left(\Delta\bigcup\Delta^{(1)}\right)\setminus\left(\Delta\bigcap\Delta^{(1)}\right)\right|}{2} \leq 2n\hat{r}\|\hat{\mathbf{v}}\|_2\delta t. \tag{33}$$

From (32), $|\Delta^{(1)}|=|\widetilde{\Delta}|$ and $\Delta^{(1)}\subset\mathcal{E}\setminus(\mathcal{S}^{(1)}\cup\mathcal{D}^{(1)})$, the optimized $\Delta^{(1)}$ satisfies $|\mathcal{E}\setminus(\Delta^{(1)}\cup\mathcal{S}^{(1)}\cup\mathcal{D}^{(1)})|\geq 2n\hat{r}\|\hat{\mathbf{v}}\|_2\delta t$, which recovers the assumption in (31). Therefore, we can repeat the above process and guarantee the success of H sub-navigation tasks. The entire navigation task is guaranteed success by concatenating these sub-tasks.

Completeness: When $H\to\infty$, we have $\delta t\to 0$. Since the environment optimization time is same as the agent operation time at each sub-navigation task, the obstacle region and the agents can be considered taking actions simultaneously when $\delta t\to 0$. The initial environment condition in (31) becomes

$$\lim_{\delta t\to 0}|\mathcal{E}\setminus(\Delta\cup\mathcal{S}\cup\mathcal{D})| > 2n\hat{r}\|\hat{\mathbf{v}}\|_2\delta t\to 0. \tag{34}$$

which is satisfied from the condition $\mathcal{W} \setminus (\Delta \cup \mathcal{S} \cup \mathcal{D}) \neq \emptyset$. The changed area of the obstacle region in (33) becomes

$$\lim_{\delta t \to 0} \frac{\left| \left(\Delta^{(h)} \bigcup \Delta^{(h+1)}\right) \setminus \left(\Delta^{(h)} \bigcap \Delta^{(h+1)}\right) \right|}{2\delta t} \leq 2n\hat{r}\|\hat{\mathbf{v}}\|_2. \quad (35)$$

That is, if the capacity of the online environment optimization is stronger than $2n\hat{r}\|\hat{\mathbf{v}}\|_2$, i.e., $\dot{\Delta} \geq 2n\hat{r}\|\hat{\mathbf{v}}\|_2$, the navigation task can be carried out successfully without collision. Therefore, there exists an online environment optimization scheme that guarantees the success of all navigation tasks.

APPENDIX C
PROOF OF THEOREM 3

We start by considering agent A_1 with the highest priority. From the condition (6), we have

$$|\mathcal{E} \setminus (\Delta \cup \mathcal{S} \cup \mathcal{D})| \geq 2d_{A_1}\hat{r}. \quad (36)$$

Following the proof of Theorem 1, we can optimize the obstacle region Δ to $\Delta^{(1)}$ such that for the initial position $\mathbf{s}_1$ and destination $\mathbf{d}_1$ of agent A_1, there exists a collision-free path $\mathbf{p}_1$ that connects $\mathbf{s}_1$ and $\mathbf{d}_1$. The area of $\mathbf{p}_1$ outside $\mathcal{S}$ and $\mathcal{D}$ is bounded as

$$|\mathbf{p}_1| \leq 2d_{A_1}\hat{r}, \quad (37)$$

i.e., the traveled distance of agent A_1 is bounded by d_{A_1}.

We then consider the second agent A_2. From Assumption 1, the initial positions $\{\mathbf{s}_i\}_{i=1}^n$ in $\mathcal{S}$ are distributed such that $d(\mathbf{s}_i, \mathbf{s}_j) \geq 2\hat{r}$, $d(\mathbf{s}_i, \partial\mathcal{S}) \geq 2\hat{r}$ and $\mathbf{s}_1, \mathbf{s}_2$ are distributed in the starting components $\mathcal{S}_{A_1}, \mathcal{S}_{A_2}$. If $\mathcal{S}_{A_1} = \mathcal{S}_{A_2}$, there exists a path $\mathbf{p}_{\mathbf{s}_2}^{\mathbf{s}_1}$ in $\mathcal{S}_{A_1}$ without changing the obstacle region. If $\mathcal{S}_{A_1} \neq \mathcal{S}_{A_2}$, there exist boundary points $\partial\mathbf{s}_1 \in \partial\mathcal{S}_{A_1}$, $\partial\mathbf{s}_2 \in \partial\mathcal{S}_{A_2}$ and paths $\mathbf{p}_{\mathbf{s}_1}^{\partial\mathbf{s}_1}$, $\mathbf{p}_{\mathbf{s}_2}^{\partial\mathbf{s}_2}$ that connect $\mathbf{s}_1$, $\partial\mathbf{s}_1$ and $\mathbf{s}_2$, $\partial\mathbf{s}_2$, respectively, and are collision-free with respect to the other initial positions. For the boundary points $\partial\mathbf{s}_2$ and $\partial\mathbf{s}_1$, the shortest path $\mathbf{p}_{\partial\mathbf{s}_2}^{\partial\mathbf{s}_1}$ that connects them is the straight path, the area of which is bounded as

$$\left|\mathbf{p}_{\partial\mathbf{s}_2}^{\partial\mathbf{s}_1}\right| \leq 2d_{\max}(\mathcal{S}_{A_2}, \mathcal{S}_{A_1})\hat{r} \quad (38)$$

where $d_{\max}(\mathcal{S}_{A_2}, \mathcal{S}_{A_1})$ is the distance between $\mathcal{S}_{A_1}$, $\mathcal{S}_{A_2}$ and $d_{A_2,\mathcal{S}} = d_{\max}(\mathcal{S}_{A_2}, \mathcal{S}_{A_1})$ by definition. Similar result applies to the destinations $\mathbf{d}_1$, $\mathbf{d}_2$ in $\mathcal{D}_{A_1}$, $\mathcal{D}_{A_2}$. That is, if $\mathcal{D}_{A_1} = \mathcal{D}_{A_2}$, there exists a path $\mathbf{p}_{\mathbf{d}_1}^{\mathbf{d}_2}$ in $\mathcal{D}_{A_1}$ without changing the obstacle region. If $\mathcal{D}_{A_1} \neq \mathcal{D}_{A_2}$, there exist boundary points $\partial\mathbf{d}_1 \in \partial\mathcal{D}_{A_1}$, $\partial\mathbf{d}_2 \in \partial\mathcal{D}_{A_2}$ and paths $\mathbf{p}_{\partial\mathbf{d}_1}^{\mathbf{d}_1}$, $\mathbf{p}_{\partial\mathbf{d}_2}^{\mathbf{d}_2}$ that connect $\partial\mathbf{d}_1$, $\mathbf{d}_1$ and $\partial\mathbf{d}_2$, $\mathbf{d}_2$, respectively, and are collision-free with respect to the other destinations. The area of the shortest path $\mathbf{p}_{\partial\mathbf{d}_1}^{\partial\mathbf{d}_2}$ that connects the boundary points $\partial\mathbf{d}_1$ and $\partial\mathbf{d}_2$ is bounded as

$$\left|\mathbf{p}_{\partial\mathbf{d}_1}^{\partial\mathbf{d}_2}\right| \leq 2d_{\max}(\mathcal{D}_{A_2}, \mathcal{D}_{A_1})\hat{r} \quad (39)$$

where $d_{\max}(\mathcal{D}_{A_2}, \mathcal{D}_{A_1})$ is the distance between $\mathcal{D}_{A_1}$, $\mathcal{D}_{A_2}$ and $d_{A_2,\mathcal{D}} = d_{\max}(\mathcal{D}_{A_2}, \mathcal{D}_{A_1})$ by definition. From the condition (6), we have

$$|\mathcal{E} \setminus (\Delta \cup \mathcal{S} \cup \mathcal{D})| \geq 2d_{A_1}\hat{r} + 2\left(d_{A_2,\mathcal{S}} + d_{A_2,\mathcal{D}}\right)\hat{r}. \quad (40)$$

Thus, we can optimize $\Delta^{(1)}$ to $\Delta^{(2)}$ such that the paths $\mathbf{p}_{\partial\mathbf{s}_2}^{\partial\mathbf{s}_1}$ and $\mathbf{p}_{\partial\mathbf{d}_1}^{\partial\mathbf{d}_2}$ are obstacle-free. Following the proof of Theorem 1, we can establish the paths $\mathbf{p}_{\mathbf{s}_2}^{\mathbf{s}_1}$ and $\mathbf{p}_{\mathbf{d}_1}^{\mathbf{d}_2}$ that connect $\mathbf{s}_2$, $\mathbf{s}_1$ and $\mathbf{d}_1$, $\mathbf{d}_2$, respectively, and are collision-free w.r.t. the other initial positions, destinations and the optimized obstacle region $\Delta^{(2)}$. By concatenating $\mathbf{p}_{\mathbf{s}_2}^{\mathbf{s}_1}$, $\mathbf{p}_1$ and $\mathbf{p}_{\mathbf{d}_1}^{\mathbf{d}_2}$, we can establish the collision-free path $\mathbf{p}_2$ for agent A_2. The area of $\mathbf{p}_2$ outside $\mathcal{S}$ and $\mathcal{D}$ is bounded as

$$|\mathbf{p}_2| \leq 2d_{A_1}\hat{r} + 2\left(d_{A_2,\mathcal{S}} + d_{A_2,\mathcal{D}}\right)\hat{r}, \quad (41)$$

i.e., the traveled distance of agent A_2 is bounded by $d_{A_1} + d_{A_2,\mathcal{S}} + d_{A_2,\mathcal{D}}$.

Lastly, we follow the above procedure to optimize the obstacle region $\Delta^{(2)}$ to $\Delta^{(n)}$ for agents $A_3, \ldots, A_n$, recursively, and establish the paths $\{\mathbf{p}_i\}_{i=1}^n$ that are collision-free w.r.t. the other initial positions, destinations and the optimized obstacle region. The area of $\mathbf{p}_i$ outside $\mathcal{S}$ and $\mathcal{D}$ is bounded as[4]

$$|\mathbf{p}_i| \leq 2d_{A_1}\hat{r} + \sum_{j=2}^{i} 2\left(d_{A_j,\mathcal{S}} + d_{A_j,\mathcal{D}}\right)\hat{r}, \quad (42)$$

i.e., the traveled distance of agent A_i is bounded by $d_{A_1} + \sum_{j=2}^{i}(d_{A_j,\mathcal{S}} + d_{A_j,\mathcal{D}})$ for $i = 1, \ldots, n$. This shows the optimized environment is "well-formed" for all agents $\{A_i\}_{i=1}^n$ and completes the proof by using Assumption 2 and Theorem 4 in [27], i.e., the navigation tasks of all agents can be carried out successfully without collision and the traveled distance of agent A_i is bounded by $d_{A_1} + \sum_{j=2}^{i}(d_{A_j,\mathcal{S}} + d_{A_j,\mathcal{D}})$ for $i = 1, \ldots, n$.

APPENDIX D
PROOF OF COROLLARY 1

Consider a sub-system of b agents $\{A_i\}_{i=1}^b$ with highest priorities. Following the proof of Theorem 3, we can optimize the obstacle region Δ to Δ^* such that for the initial position $\mathbf{s}_i$ and destination $\mathbf{d}_i$ of agent A_i, there exists a collision-free path $\mathbf{p}_i$ that connects $\mathbf{s}_i$ and $\mathbf{d}_i$ for $i = 1, \ldots, b$. Therefore, Δ^* is "well-formed" w.r.t. the considered sub-system and the navigation tasks of b agents $\{A_i\}_{i=1}^b$ can be carried out successfully without collision.

For the rest of the agents A_j for $j = b+1, \ldots, n$, assume that the initial position $\mathbf{s}_j$ of agent A_j is within the same starting component as $\mathbf{s}_i$ of agent A_i with $i \in \{1, \ldots, b\}$, i.e., $\mathcal{S}_{A_j} = \mathcal{S}_{A_i}$. Denote by $\partial\mathbf{s}_i$ the boundary point where the path $\mathbf{p}_i$ intersects with the boundary of the starting component $\mathcal{S}_{A_i}$. Since the initial positions $\{\mathbf{s}_i\}_{i=1}^n$ in $\mathcal{S}$ are distributed in a way such that $d(\mathbf{s}_{i_1}, \mathbf{s}_{i_2}) \geq 2\hat{r}$ and $d(\mathbf{s}_i, \partial\mathcal{S}) \geq 2\hat{r}$ from Assumption 1 and $\mathbf{s}_j$ is in the same starting component as $\mathbf{s}_i$, there exists a path $\mathbf{p}_{\mathbf{s}_j}^{\partial\mathbf{s}_i}$ in $\mathcal{S}_{A_i}$ that connects $\mathbf{s}_j$, $\partial\mathbf{s}_i$ and is collision-free w.r.t. the other initial positions. Similar result applies to the destinations $\mathbf{d}_j$ and $\mathbf{d}_i$. That is, if $\mathbf{d}_j$ is within the same destination component as $\mathbf{d}_i$, i.e., $\mathcal{D}_{A_j} = \mathcal{D}_{A_i}$,

[4] Without loss of generality, we assume $\sum_a^b = 0$ if $b < a$.

there exists a path $\mathbf{p}_{\mathbf{d}_j}^{\partial\mathbf{d}_i}$ in $\mathcal{D}_{A_i}$ that connects $\mathbf{d}_j$, $\partial\mathbf{d}_i$ and is collision-free w.r.t. the other destinations. Since the boundary points $\partial\mathbf{s}_i$ and $\partial\mathbf{d}_i$ can be connected by the collision-free path $\mathbf{p}_i$, we can establish the collision-free path $\mathbf{p}_j$ that connects $\mathbf{s}_j$ and $\mathbf{d}_j$ by concatenating $\mathbf{p}_{\mathbf{s}_j}^{\partial\mathbf{s}_i}$, $\mathbf{p}_i$ and $\mathbf{p}_{\mathbf{d}_j}^{\partial\mathbf{d}_i}$. Therefore, the optimized obstacle region Δ^* is "well-formed" w.r.t. agent A_j as well, the navigation task of which can be carried out successfully without collision. The same result holds for all agents $A_j \in \{A_{b+1}, \ldots, A_n\}$ satisfying the above conditions, completing the proof.

APPENDIX E
PROOF OF THEOREM 4

We prove the theorem following Theorem 2. First, we reformulate the navigation task as H sub-navigation tasks for each agent. Then, we optimize the obstacle region to guarantee the completeness of sub-navigation tasks successively. Lastly, we show the completeness of the entire navigation by concatenating these sub-navigation tasks and complete the proof by limiting $H \to \infty$.

Let T be the maximal operation time required by trajectories $\{\mathbf{p}_i(t)\}_{i=1}^n$ with velocities $\{\mathbf{v}_i(t)\}_{i=1}^n$ and $\{\mathbf{p}_i(hT/H)\}_{h=0}^H$ be the intermediate positions with $\mathbf{p}_i(0) = \mathbf{s}_i$ and $\mathbf{p}_i(T) = \mathbf{d}_i$ for $i = 1, \ldots, n$. The goal of the hth sub-navigation task for agent A_i is from $\mathbf{p}_i((h-1)T/H)$ to $\mathbf{p}_i(hT/H)$ and the operation time required by each sub-navigation task is $\delta t = T/H$. In this context, we separate the procedure of online environment optimization as a number of time slices with duration $2\delta t$. At each time slice, we change the obstacle region for sub-navigation tasks with duration δt and navigate the agents with duration δt in an alternative manner. From the condition (9), the area of the obstacle region that can be changed at each time slice is

$$2b\hat{r}\|\hat{\mathbf{v}}\|_2\delta t \le \frac{\left|\left(\Delta \bigcup \widetilde{\Delta}\right)\setminus\left(\Delta \bigcap \widetilde{\Delta}\right)\right|}{2} < 2(b+1)\hat{r}\|\hat{\mathbf{v}}\|_2\delta t \tag{43}$$

where Δ is the original obstacle region and $\widetilde{\Delta}$ is the changed obstacle region. From (43) and the proof of Theorem 2, at each time slice, we can change the obstacle region to make the environment "well-formed" w.r.t. the sub-navigation tasks of only b agents instead of all n agents, i.e., it only guarantees the success of b sub-navigation tasks. For the agents whose sub-navigation tasks are not selected for environment optimization, we keep them static until the next time slice when their sub-navigation tasks are selected. By tuning the selections of b sub-navigation tasks across time slices, we prove there exists a selection scheme such that sub-navigation tasks of all agents can be carried out successfully without collision and the navigation time of the agents can be bounded inverse proportionally to their priorities.

Specifically, let $T_1 = H_1(2\delta t)$ with $H_1 \ge H$ be the maximal navigation time required by agent A_1 with the highest priority ρ_1, i.e., agent A_1 requires completing its H sub-navigation tasks within H_1 time slices, and $T_i = H_i(2\delta t) = \lceil H_1\rho_1/\rho_i\rceil(2\delta t)$ be the maximal navigation time required by agent A_i with the priority ρ_i for $i = 2, \ldots, n$. For agent A_1, there exists a scheme that selects the sub-navigation task of A_1 for environment optimization H times within H_1 time slices because $H_1 \ge H$ and thus, its H sub-navigation tasks can be carried out successfully without collision. By concatenating these sub-tasks, the navigation task of A_1 can be carried out successfully without collision within the required time.

For agents A_1 and A_2, if it holds that

$$\min\{1, b\}(H_2 - H_1) + \min\{2, b\}H_1 \ge 2H \tag{44}$$

there exists a scheme that selects the sub-navigation tasks of A_1 and A_2 for environment optimization H times during H_2 time slices, respectively. The minimal operations $\min\{1, b\}$ and $\min\{2, b\}$ in (44) represent the facts that single agent can be selected at most once at each time slice, and each time slice can select at most b agents. Thus, the navigation tasks of A_1 and A_2 can be carried out successfully without collision within the required time. Analogously for agents $\{A_1, \ldots, A_i\}$, if it holds that

$$\sum_{j=1}^{i} \min\{j, b\}\left(H_{i+1-j} - H_{i-j}\right) \ge iH \tag{45}$$

with $H_0 = 0$ by default, there exists a scheme that selects the sub-navigation tasks of $\{A_1, \ldots, A_i\}$ for environment optimization H times during H_i time slices, and their navigation tasks can be carried out successfully without collision within the required time. Therefore, we conclude that if

$$\sum_{j=1}^{n} \min\{j, b\}\left(H_{n+1-j} - H_{n-j}\right) \ge nH \tag{46}$$

there exists a scheme that selects the sub-navigation tasks of all n agents for environment optimization H times during H_n time slices, and all navigation tasks can be carried out successfully without collision within the required time. Since $H_i = \lceil H_1\rho_1/\rho_2\rceil$, i.e., H_i can be represented by H_1 and the associated priorities for $i = 1, \ldots, n$, we can rewrite (46) as

$$\sum_{j=1}^{n} \min\{j,b\}\left(\left\lceil \frac{H_1\rho_1}{\rho_{n+1-j}}\right\rceil - \left\lceil \frac{H_1\rho_1}{\rho_{n-j}}\right\rceil\right) \ge nH. \tag{47}$$

Since $\{\rho_i\}_{i=1}^n$, H are given and the left-hand side of (47) increases with H_1, there exists a large enough H_1 such that (47) holds. Therefore, the navigation tasks of all agents can be carried out successfully and the navigation time of agent A_i is bounded by $T_i = H_i(2\delta t) = \lceil H_1\rho_1/\rho_i\rceil(2\delta t)$.

When H is sufficiently large, i.e., δt is sufficiently small, the agents and the obstacles can be considered moving simultaneously. Since $H_1 \ge H$ is sufficiently large as well, we have $\lceil H_1\rho_1/\rho_i\rceil(2\delta t) \approx 2H_1\rho_1\delta t/\rho_i$. With this observation and the conclusion obtained from (47), we complete the proof, i.e., the navigation tasks of all agents can be carried out successfully without collision and the navigation time of agent A_i is bounded by $T_i = C_T/\rho_i$ for $i = 1, \ldots, n$ where $C_T = 2H_1\rho_1\delta t$ is the time constant.

APPENDIX F
PROOF OF COROLLARY 2

From the proof of Theorem 4, for any integer $1 \leq \eta \leq n$, if it holds that

$$\sum_{j=1}^{\eta} \min\{j, b\} \left(\left\lceil \frac{H_1 \rho_1}{\rho_{\eta+1-j}} \right\rceil - \left\lceil \frac{H_1 \rho_1}{\rho_{\eta-j}} \right\rceil \right) \geq \eta H \tag{48}$$

there exists a scheme with a sufficiently large H_1 that selects the sub-navigation tasks of η agents for environment optimization H times during $H_\eta = \lceil H_1 \rho_1 / \rho_{\eta+1-j} \rceil$ time slices and the navigation tasks of agents $\{A_i\}_{i=1}^{\eta}$ can be carried out successfully without collision within time $\{T_i\}_{i=1}^{\eta}$, respectively. Since the agents are required to reach their destinations within time $T_{\max}$, i.e., the maximal navigation time is $T_{\max}$, the maximal (allowed) number of time slices is $H_{\max} = \lfloor T_{\max} / 2\delta t \rfloor$. This is equivalent to requiring

$$H_\eta = \left\lceil \frac{H_1 \rho_1}{\rho_\eta} \right\rceil \leq \left\lfloor \frac{T_{\max}}{2\delta t} \right\rfloor \tag{49}$$

because H_η is the maximal number of time slices required by the first η agents. Thus, we have

$$H_1 \leq \frac{\left\lfloor \frac{T_{\max}}{2\delta t} \right\rfloor \rho_\eta}{\rho_1}. \tag{50}$$

Since the left-hand side of (48) increases with H_1, substituting (50) into (48) yields

$$\sum_{j=1}^{\eta} \min\{j, b\} \left(\left\lceil \frac{H_1 \rho_1}{\rho_{\eta+1-j}} \right\rceil - \left\lceil \frac{H_1 \rho_1}{\rho_{\eta-j}} \right\rceil \right) \leq \sum_{j=1}^{\eta} \min\{j, b\} \left(\left\lceil \frac{\left\lfloor \frac{T_{\max}}{2\delta t} \right\rfloor \rho_\eta}{\rho_{\eta+1-j}} \right\rceil - \left\lceil \frac{\left\lfloor \frac{T_{\max}}{2\delta t} \right\rfloor \rho_\eta}{\rho_{\eta-j}} \right\rceil \right). \tag{51}$$

By substituting (51) into (48), we have

$$\sum_{j=1}^{\eta} \min\{j, b\} \left(\left\lceil \frac{\left\lfloor \frac{T_{\max}}{2\delta t} \right\rfloor \rho_\eta}{\rho_{\eta+1-j}} \right\rceil - \left\lceil \frac{\left\lfloor \frac{T_{\max}}{2\delta t} \right\rfloor \rho_\eta}{\rho_{\eta-j}} \right\rceil \right) \geq \eta H. \tag{52}$$

By using the fact $T = H\delta t$ where T is the maximal operation time of trajectories $\{\mathbf{p}_i(t)\}_{i=1}^{n}$, we get

$$\sum_{j=1}^{\eta} \min\{j, b\} \left(\left\lceil \frac{\left\lfloor \frac{HT_{\max}}{2T} \right\rfloor \rho_\eta}{\rho_{\eta+1-j}} \right\rceil - \left\lceil \frac{\left\lfloor \frac{HT_{\max}}{2T} \right\rfloor \rho_\eta}{\rho_{\eta-j}} \right\rceil \right) \geq \eta H. \tag{53}$$

When H is sufficiently large, we have $\lceil \lfloor HT_{\max}/(2T) \rfloor \rho_\eta / \rho_{\eta+1-j} \rceil \approx HT_{\max}\rho_\eta/(2T\rho_{\eta+1-j})$ and $\lceil \lfloor HT_{\max}/(2T) \rfloor \rho_\eta / \rho_{\eta-j} \rceil \approx HT_{\max}\rho_\eta/(2T\rho_{\eta-j})$, and (53) is equivalent as

$$\sum_{j=1}^{\eta} \min\{j, b\} \left(\frac{T_{\max}\rho_\eta}{2T\rho_{\eta+1-j}} - \frac{T_{\max}\rho_\eta}{2T\rho_{\eta-j}} \right) \geq \eta. \tag{54}$$

By setting n_p as the maximal η that satisfies (54) and following the proof of Theorem 4, the navigation tasks of $\{A_i\}_{i=1}^{n_p}$ can be carried out successfully without collision within the required time $T_{\max}$, completing the proof.

APPENDIX G
PROOF OF THEOREM 5

For a feasible solution of problem (19) with the constraint constant $\mathcal{C}_q = (1 - \delta + \epsilon)/(1 - \gamma)$ [cf. (18)], it satisfies that

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{1}\left[h_q\left(\mathbf{X}_o^{(t)}, \mathbf{U}_o^{(t)}\right) \leq 0\right]\right] = \sum_{t=0}^{\infty} \gamma^t \mathbb{P}\left[h_q\left(\mathbf{X}_o^{(t)}, \mathbf{U}_o^{(t)}\right) \leq 0\right] \geq \frac{1 - \delta + \epsilon}{1 - \gamma} = \frac{1}{1 - \gamma} - \frac{\delta - \epsilon}{1 - \gamma}, \text{ for } q = 1, \ldots, Q \tag{55}$$

where the linearity of the expectation and (26) are used. For the term $\sum_{t=0}^{\infty} \gamma^t \mathbb{P}[h_q(\mathbf{X}_o^{(t)}, \mathbf{U}_o^{(t)}) \leq 0]$, we have

$$\sum_{t=0}^{\infty} \gamma^t \mathbb{P}\left[h_q\left(\mathbf{X}_o^{(t)}, \mathbf{U}_o^{(t)}\right) \leq 0\right] = \sum_{t=0}^{\infty} \gamma^t - \sum_{t=0}^{\infty} \gamma^t \mathbb{P}\left[h_q\left(\mathbf{X}_o^{(t)}, \mathbf{U}_o^{(t)}\right) > 0\right] = \frac{1}{1 - \gamma} - \sum_{t=0}^{\infty} \gamma^t \mathbb{P}\left[h_q\left(\mathbf{X}_o^{(t)}, \mathbf{U}_o^{(t)}\right) > 0\right]. \tag{56}$$

By comparing (55) and (56), we get

$$\sum_{t=0}^{\infty} \gamma^t \mathbb{P}\left[h_q\left(\mathbf{X}_o^{(t)}, \mathbf{U}_o^{(t)}\right) > 0\right] \leq \frac{\delta - \epsilon}{1 - \gamma}. \tag{57}$$

For the maximal time horizon T, we have

$$\sum_{t=0}^{T} \gamma^t \mathbb{P}\left[h_q\left(\mathbf{X}_o^{(t)}, \mathbf{U}_o^{(t)}\right) > 0\right] \leq \sum_{t=0}^{\infty} \gamma^t \mathbb{P}\left[h_q\left(\mathbf{X}_o^{(t)}, \mathbf{U}_o^{(t)}\right) > 0\right] \leq \frac{\delta - \epsilon}{1 - \gamma} \tag{58}$$

because each term in the summation is non-negative. Then, by setting $\epsilon = \delta(1 - \gamma^T(1 - \gamma)) < \delta$ and substituting the latter into (58), we get $\sum_{t=0}^{T} \gamma^t \mathbb{P}[h_q(\mathbf{X}_o^{(t)}, \mathbf{U}_o^{(t)}) > 0] \leq \gamma^T \delta$. Since $\gamma^T \leq \gamma^t$ for all $t \leq T$, we have $\sum_{t=0}^{T} \gamma^T \mathbb{P}[h_q(\mathbf{X}_o^{(t)}, \mathbf{U}_o^{(t)}) > 0] \leq \gamma^T \delta$ and thus

$$\sum_{t=0}^{T} \mathbb{P}\left[h_q\left(\mathbf{X}_o^{(t)}, \mathbf{U}_o^{(t)}\right) > 0\right] \leq \delta. \tag{59}$$

By leveraging the Boole-Frechet-Bonferroni inequality, we complete the proof that for each step $0 \leq t \leq T$, it holds that

$$\mathbb{P}\left[\cap_{0 \leq \tau \leq t} \left\{h_q\left(\mathbf{X}_o^{(\tau)}, \mathbf{U}_o^{(\tau)}\right) \leq 0\right\}\right] \geq 1 - \delta. \tag{60}$$

REFERENCES

[1] Z. Gao and A. Prorok, "Environment optimization for multi-agent navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 3440–3446.

[2] D. Silver, "Cooperative pathfinding," in *Proc. AAAI Conf. Artif. Intell. Interactive Digit. Entertainment*, 2005, pp. 117–122.

[3] J. v. d. Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2008, pp. 1928–1935.

[4] J. v. d. Berg and M. H. Overmars, "Prioritized motion planning for multiple robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2005, pp. 430–435.

[5] V. R. Desaraju and J. P. How, "Decentralized path planning for multi-agent teams with complex constraints," *Auton. Robots*, vol. 32, no. 4, pp. 385–403, 2012.

[6] T. S. Standley and R. Korf, "Complete algorithms for cooperative pathfinding problems," in *Proc. Int. Joint Conf. Artif. Intell.*, 2011, pp. 668–673.

[7] Z. Gao, G. Yang, and A. Prorok, "Learning environment-aware control barrier functions for safe and feasible multi-robot navigation," 2023, *arXiv:2303.04313*.

[8] N. Mani, V. Garousi, and B. H. Far, "Search-based testing of multi-agent manufacturing systems for deadlocks based on models," *Int. J. Artif. Intell. Tools*, vol. 19, no. 04, pp. 417–437, 2010.

[9] A. Ruderman et al., "Uncovering surprising behaviors in reinforcement learning via worst-case analysis," in *Proc. Int. Conf. Learn. Representations Workshop*, 2019, pp. 1–10.

[10] J. Boudet et al., "From collections of independent, mindless robots to flexible, mobile, and directional superstructures," *Sci. Robot.*, vol. 6, no. 56, 2021, Art. no. eabd0272.

[11] M. Čáp, P. Novák, A. Kleiner, and M. Selecký, "Prioritized planning algorithms for trajectory coordination of multiple mobile robots," *IEEE Trans. Automat. Sci. Eng.*, vol. 12, no. 3, pp. 835–849, Jul. 2015.

[12] Q. Wang, R. McIntosh, and M. Brain, "A new-generation automated warehousing capability," *Int. J. Comput. Integr. Manuf.*, vol. 23, no. 6, pp. 565–573, 2010.

[13] H. Bier, "Robotic building (s)," *Next Gener. Building*, vol. 1, no. 1, pp. 1–10, 2014.

[14] L. Custodio and R. Machado, "Flexible automated warehouse: A literature review and an innovative framework," *Int. J. Adv. Manuf. Technol.*, vol. 106, no. 1, pp. 533–558, 2020.

[15] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI Mag.*, vol. 29, no. 1, pp. 9–19, 2008.

[16] S. Karma et al., "Use of unmanned vehicles in search and rescue operations in forest fires: Advantages and limitations observed in a field trial," *Int. J. Disaster Risk Reduction*, vol. 13, pp. 307–312, 2015.

[17] Z. Drezner and G. O. Wesolowsky, "Selecting an optimum configuration of one-way and two-way routes," *Transp. Sci.*, vol. 31, no. 4, pp. 386–394, 1997.

[18] D. Johnson and J. Wiles, "Computer games with intelligence," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2001, pp. 1355–1358.

[19] T. Tanaka and H. Sandberg, "SDP-based joint sensor and controller design for information-regularized optimal LQG control," in *Proc. IEEE Conf. Decis. Control*, 2015, pp. 4486–4491.

[20] S. Tatikonda and S. Mitter, "Control under communication constraints," *IEEE Trans. Autom. Control*, vol. 49, no. 7, pp. 1056–1068, Jul. 2004.

[21] V. Tzoumas, L. Carlone, G. J. Pappas, and A. Jadbabaie, "Sensing-constrained LQG control," in *Proc. IEEE Amer. Control Conf.*, 2018, pp. 197–202.

[22] H. Lipson and J. B. Pollack, "Automatic design and manufacture of robotic lifeforms," *Nature*, vol. 406, no. 6799, pp. 974–978, 2000.

[23] G. S. Hornby, H. Lipson, and J. B. Pollack, "Generative representations for the automated design of modular physical robots," *IEEE Trans. Robot. Automat.*, vol. 19, no. 4, pp. 703–719, Aug. 2003.

[24] N. Cheney, J. Bongard, V. SunSpiral, and H. Lipson, "Scalable co-optimization of morphology and control in embodied machines," *J. Roy. Soc. Interface*, vol. 15, no. 143, 2018, Art. no. 20170937.

[25] M. Bennewitz, W. Burgard, and S. Thrun, "Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots," *Robot. Auton. Syst.*, vol. 41, no. 2/3, pp. 89–99, 2002.

[26] M. Jager and B. Nebel, "Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2001, pp. 1213–1219.

[27] M. Čáp, J. Vokřínek, and A. Kleiner, "Complete decentralized method for on-line multi-robot trajectory planning in well-formed infrastructures," in *Proc. Int. Conf. Automated Plan. Scheduling*, 2015, pp. 324–332.

[28] W. Wu, S. Bhattacharya, and A. Prorok, "Multi-robot path deconfliction through prioritization by path prospects," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 9809–9815.

[29] I. Gur, N. Jaques, K. Malta, M. Tiwari, H. Lee, and A. Faust, "Adversarial environment generation for learning to navigate the web," 2021, *arXiv:2103.01991*.

[30] H. Zhang, Y. Chen, and D. C. Parkes, "A general approach to environment design with one agent," in *Proc. Int. Joint Conf. Artif. Intell.*, 2009, pp. 2002–2008.

[31] S. Keren, A. Gal, and E. Karpas, "Goal recognition design for non-optimal agents," in *Proc. AAAI Conf. Artif. Intell.*, 2015, pp. 3298–3304.

[32] A. Kulkarni, S. Sreedharan, S. Keren, T. Chakraborti, D. E. Smith, and S. Kambhampati, "Designing environments conducive to interpretable robot behavior," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 10982–10989.

[33] K. K. Hauser, "Minimum constraint displacement motion planning," *Robot.: Sci. Syst.*, vol. 6, pp. 1–8, 2013.

[34] K. K. Hauser, "The minimum constraint removal problem with three robotics applications," *Int. J. Robot. Res.*, vol. 33, no. 1, pp. 5–17, 2014.

[35] M. Bellusci, N. Basilico, and F. Amigoni, "Multi-agent path finding in configurable environments," in *Proc. Int. Conf. Auton. Agents MultiAgent Syst.*, 2020, pp. 159–167.

[36] M. Liu, H. Ma, J. Li, and S. Koenig, "Task and path planning for multi-agent pickup and delivery," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, 2019, pp. 1152–1160.

[37] T. Yamauchi, Y. Miyashita, and T. Sugawara, "Path and action planning in non-uniform environments for multi-agent pickup and delivery tasks," in *Proc. Eur. Conf. Multi-Agent Syst.*, 2021, pp. 37–54.

[38] T. Yamauchi, Y. Miyashita, and T. Sugawara, "Standby-based deadlock avoidance method for multi-agent pickup and delivery tasks," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, 2022, pp. 1427–1435.

[39] B. Li and H. Ma, "Double-deck multi-agent pickup and delivery: Multi-robot rearrangement in large-scale warehouses," *IEEE Robot. Automat. Lett.*, vol. 8, no. 6, pp. 3701–3708, Jun. 2023.

[40] M. Browne and S. S. Ghidary, "Convolutional neural networks for image processing: An application in robot vision," in *Proc. Australas. Joint Conf. Artif. Intell.*, 2003, pp. 641–652.

[41] S. Kumra and C. Kanan, "Robotic grasp detection using deep convolutional neural networks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 769–776.

[42] J. Gu et al., "Recent advances in convolutional neural networks," *Pattern Recognit.*, vol. 77, pp. 354–377, 2018.

[43] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.

[44] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–12.

[45] Z. Gao, E. Isufi, and A. Ribeiro, "Stochastic graph neural networks," *IEEE Trans. Signal Process.*, vol. 69, pp. 4428–4443, 2021.

[46] E. Tolstaya, F. Gama, J. Paulos, G. Pappas, V. Kumar, and A. Ribeiro, "Learning decentralized controllers for robot swarms with graph neural networks," in *Proc. Conf. Robot Learn.*, 2020, pp. 671–682.

[47] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph neural networks for decentralized multi-robot path planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 11785–11792.

[48] Z. Gao, F. Gama, and A. Ribeiro, "Wide and deep graph neural network with distributed online learning," *IEEE Trans. Signal Process.*, vol. 70, pp. 3862–3877, 2022.

[49] Z. Gao, Y. Shao, D. Gunduz, and A. Prorok, "Decentralized channel management in WLANs with graph neural networks," 2022, *arXiv:2210.16949*.

[50] P. Anderson et al., "On evaluation of embodied navigation agents," 2018, *arXiv:1807.06757*.

[51] T. Gervet, S. Chintala, D. Batra, J. Malik, and D. S. Chaplot, "Navigating to objects in the real world," *Sci. Robot.*, vol. 8, no. 79, 2023, Art. no. eadf6991.

[52] X. Wang et al., "Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 6629–6638.

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