2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

# Environment Optimization for Multi-Agent Navigation

Zhan Gao and Amanda Prorok

***Abstract*— Traditional approaches to the design of multi-agent navigation algorithms consider the environment as a fixed constraint, despite the obvious influence of spatial constraints on agents' performance. Yet hand-designing improved environment layouts and structures is inefficient and potentially expensive. The goal of this paper is to consider the environment as a decision variable in a system-level optimization problem, where both agent performance and environment cost can be accounted for. We begin by proposing a novel environment optimization problem. We show, through formal proofs, under which conditions the environment can change while guaranteeing completeness (i.e., all agents reach their navigation goals). Our solution leverages a model-free reinforcement learning approach. In order to accommodate a broad range of implementation scenarios, we include both online and offline optimization, and both discrete and continuous environment representations. Numerical results corroborate our theoretical findings and validate our approach.**

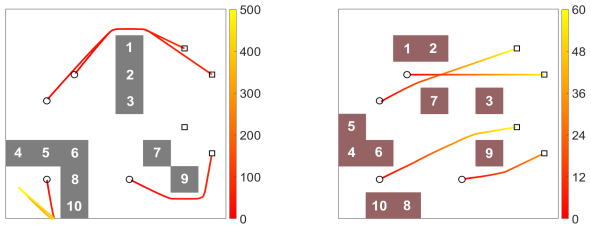
## I. INTRODUCTION

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 [1]–[5]. 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, environmental constraints may result in dead-locks, live-locks and prioritization conflicts, even for state-of-the-art algorithms [6]. 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 [7]. To deal with such bottlenecks, spatial structures (e.g., intersections, roundabouts) and markings (e.g., lanes) are developed to facilitate path de-confliction [8] 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 and live-locks through hand-designed environment templates, such hand-designing process is inefficient [9].

Reconfigurable, automated environments are emerging as a new trend [10]–[12]. In tandem with that enabling technology, the goal of this paper is to consider the environment as a *variable* in pursuit of the agents' incentives. We propose the problem of systematically *optimizing an environment to improve the performance of a given multi-agent system*, an example of which is given in Fig. 1. More in detail, our contributions are as follows:

Department of Computer Science and Technology, University of Cambridge zg292@cam.ac.uk, asp45@cam.ac.uk. This work was supported by European Research Council (ERC) Project 949940 (gAIa).




(a) (b)

Fig. 1. Example of offline environment optimization. Circles are the initial positions, squares are the destinations and the obstacles are numbered for exposition. The color lines from red to yellow are agent trajectories and the color bar represents the time scale. Obstacle layout and agent trajectories (a) *before* environment optimization and (b) *after* environment optimization.

(i) We first define the problem of environment optimization. We then develop two solution variants, i.e., the offline environment optimization and the online environment optimization.

(ii) We analyze the completeness of multi-agent navigation with environment optimization, and identify the conditions under which all agents are guaranteed to reach their navigation goals.

(iii) We develop a reinforcement learning-based method to solve the environment optimization problem, and integrate two variant information processing architectures (i.e., CNNs, GNNs) as a function of the problem setting. The proposed method is able to generalize beyond training instances, adapts to various optimization objectives, and allows for decentralized implementation.

**Related work.** The *role of the environment* on motion planning has been explored by a handful of works [13]–[17]. The works in [13], [14] emphasize the existence of congestion and deadlocks in undesirable environments and develop trajectory planning methods to escape from potential deadlocks. The authors in [15] define the concept of "well-formed" environments, in which the navigation tasks of all agents can be carried out successfully without collisions nor deadlocks. In [16], 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 [17] 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.

The problem of environment optimization is reminiscent of robot co-design [18]–[20], 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 [21]–[23]. A more closely related idea is exploited in [7], wherein the environment is adversarially optimized to fail navigation tasks of state-of-the-art multi-agent systems. It conducts 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.

## II. PROBLEM DEFINITION

Let $\mathcal{E}$ be a 2-D environment described by a starting region $\mathcal{S}$, a destination region $\mathcal{D}$ and an obstacle region $\Delta$ without overlap, i.e., $\mathcal{S}\bigcap\mathcal{D} = \mathcal{S}\bigcap\Delta = \mathcal{D}\bigcap\Delta = \emptyset$. Consider a multi-agent system with a set of agents $\mathcal{A} = \{A_i\}_{i=1}^n$ distributed in $\mathcal{E}$. The agent bodies are contained within circles of radius $\{r_i\}_{i=1}^n$. Agents are initialized at positions $\mathbf{S} = [\mathbf{s}_1, \dots, \mathbf{s}_n]$ in $\mathcal{S}$ and employ a given trajectory planner $\pi_a$ to move towards destinations $\mathbf{D} = [\mathbf{d}_1, \dots, \mathbf{d}_n]$ in $\mathcal{D}$.

Existing literature focuses on developing novel trajectory planners to improve navigation performance. However, the latter depends not only on the implemented planner but also on the surrounding environment $\mathcal{E}$. A "well-formed" environment with an appropriate obstacle region yields good performance for a simple planner, while a "poorly-formed" environment may result in a poor performance even for an advanced planner. This insight implies an important role played by the environment in multi-agent navigation, and motivates the problem of environment optimization.

**Problem 1 (Environment Optimization):** *Given an environment with an initial obstacle region and a multi-agent system with $n$ agents that are required to reach $n$ destinations (in a labeled navigation problem), find a policy that optimizes the obstacle region to improve agents' path efficiency while guaranteeing that they reach their destinations.*

In Section III, we analyze the completeness of multi-agent navigation with environment optimization. In Section IV, we formulate Problem 1 mathematically and leverage reinforcement learning for solutions. Lastly, we present numerical simulations to evaluate the approach in Section IV.

## III. COMPLETENESS ANALYSIS

The multi-agent system may fail to find collision-free (and deadlock-free) trajectories in an environment with an unsatisfactory obstacle region. Environment optimization overcomes this issue by optimizing the layout and guaranteeing navigation success under a mild condition. We propose two variants: the *offline* and the *online* environment optimization.

### A. Offline Environment Optimization

Offline environment optimization optimizes the obstacle region $\Delta$ based on the starting region $\mathcal{S}$, the destination region $\mathcal{D}$ and completes the optimization before the agents start to move. The optimized environment remains fixed during agent movement. We introduce some necessary notation for theoretical analysis. Let $\partial\mathcal{S}$, $\partial\mathcal{D}$, $\partial\Delta$ be the boundaries of the regions $\mathcal{S}$, $\mathcal{D}$, $\Delta$ and $B(\mathbf{s}, r)$ be a closed disk centered at position $\mathbf{s}$ with a radius $r$. We can represent each agent $A_i$ as the disk $B(\mathbf{s}_i, r_i)$ where $\mathbf{s}_i$ is the central position of the agent. 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 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 $\mathbf{z}_\mathcal{S} \in \partial\mathcal{S}$. Similar definitions apply for $\partial\mathcal{D}$ and $\partial\Delta$.

Our analysis assumes the following setup. 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 $i, j = 1, \dots, n$ with $\hat{r} = \max_{i=1,\dots,n} r_i$ the maximal agent radius. The obstacle region $\Delta$ can be controlled / changed **but its area $|\Delta|$ is constant**, i.e., $|\Delta| = |\widetilde{\Delta}|$ where $\widetilde{\Delta}$ is the changed obstacle region. That is, the obstacle region (or any part of it) cannot be removed from the environment. Each agent employs a given trajectory planner $\pi_a$ and any trajectory generated for an agent will be followed precisely. Denote by $\mathbf{p}(t) : [0, \infty) \to \mathbb{R}^2$ 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. $\Delta$ if $d\big(\mathbf{p}_i(t), \partial\Delta\big) \geq r_i$ for all $t \geq 0$. The trajectories $\mathbf{p}_i$, $\mathbf{p}_j$ of two agents $A_i$, $A_j$ are collision-free if $\|\mathbf{p}_i(t) - \mathbf{p}_j(t)\|_2 \geq r_i + r_j$ for all $t \geq 0$. With these preliminaries, we show the completeness in the following.

**Theorem 1:** *Consider the multi-agent system with $n$ agents in the environment $\mathcal{E}$ with starting, destination and obstacle regions $\mathcal{S}$, $\mathcal{D}$ and $\Delta$. 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*

$$|\mathcal{E} \setminus (\Delta \cup \mathcal{S} \cup \mathcal{D})| \geq 2nd_{\max}\hat{r} \tag{1}$$

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

*Proof:* We prove the theorem as follows. First, we optimize $\Delta$ 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 $\Delta$ based on $\mathcal{S}$, $\mathcal{D}$ to make the environment "well-formed". To do so, we handle $\mathcal{S}$, $\mathcal{D}$ and the remaining space $\mathcal{E}\setminus(\mathcal{S}\bigcup\mathcal{D})$ separately.

*(i)* 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 w.r.t. the other initial positions. A 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 w.r.t. the other destinations.

*(ii)* 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 $|\mathbf{p}_{\partial\mathbf{s}_i}^{\partial\mathbf{d}_i}| \leq 2d_{\max}\hat{r}$ 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 (1), 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 $\Delta$ to $\Delta^*$ 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 $\mathcal{S}$ $K$ times, e.g., for $K$ distinct paths, 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 $k$th pass for $k = 1, ..., K$ 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). The same result applies so 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 a 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 $\Delta^*$ for $i = 1, ..., n$, i.e., the optimized environment is "well-formed".

**Completeness.** With the fact that the optimized environment is "well-formed", by Theorem 4 in [15], we complete the proof that all navigation tasks can be carried out successfully under both centralized as well as decentralized (e.g., priority-based) planners. ■

Theorem 1 states offline environment optimization guarantees the success of all navigation tasks under a mild condition. It does not require any initial "well-formed" environment but only a small obstacle-free area [cf. (1)], which is commonly satisfied in real-world scenarios. The offline environment optimization depends on the given starting region $\mathcal{S}$ and destination region $\mathcal{D}$, and completes optimizing the obstacle region *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*

We propose an online environment optimization variant to overcome the above-mentioned issues by changing the obstacle region during agent movement. Different from its offline counterpart, it is interleaved with the deployment online, i.e., it changes the obstacle region based on instantaneous system states, and stays active until the end of the navigation. Specifically, define the starting region as the union of initial positions $\mathcal{S} = \bigcup_{i=1,...,n} B(\mathbf{s}_i, r_i)$ and the destination region as that of destinations $\mathcal{D} = \bigcup_{i=1,...,n} B(\mathbf{d}_i, r_i)$ such that $\mathcal{S} \bigcap \mathcal{D} = \emptyset$. The decentralized trajectory planner $\pi_a$ is given (i.e., each agent plans locally according to $\pi_a$). For scenarios with an empty obstacle region $\Delta = \emptyset$, we assume the environment $\mathcal{E}$ is "well-formed" and all navigation tasks are carried out successfully without collision. We denote these trajectories by $\{\mathbf{p}_i\}_{i=1}^n$. This is a reasonable assumption because an environment without obstacles is the best scenario for multi-agent navigation. For scenarios with obstacles, i.e., $\Delta \neq \emptyset$, these navigation tasks may fail and the online environment optimization handles the latter by changing $\Delta$ during the navigation procedure. Since $\Delta$ 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. The following theorem formally shows the completeness.

**Theorem 2:** *Consider the multi-agent system with $n$ agents $\{A_i\}_{i=1}^n$. For the environment $\mathcal{E}$ without obstacles, i.e., $\Delta = \emptyset$, let $\{\mathbf{p}_i(t)\}_{i=1}^n$ be $n$ collision-free trajectories of $\{A_i\}_{i=1}^n$ generated by the given trajectory planner $\pi_a$ and $\{\mathbf{v}_i(t)\}_{i=1}^n$ be the respective velocities along these trajectories. For the environment $\mathcal{E}$ with an 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*

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

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

*Proof:* We prove the theorem as follows. First, we slice the navigation procedure into $H$ time frames. Then, we optimize the obstacle region based on the agent positions in each frame, and show the completeness of individual frames. Lastly, we show the completeness of the entire multi-agent navigation solution by concatenating individual frames, and complete the proof by considering the whole process when the number of frames tends to infinity, i.e., $H \to \infty$.

**Time slicing.** 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 time frames. 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 $i = 1, ..., n$. We can re-formulate the navigation task into $H$ sub-navigation tasks, where the $h$th sub-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 $\delta t = T/H$. In each frame, we first change the obstacle region based on the corresponding sub-navigation task and then navigate the agents until the next frame.

**Obstacle region optimization.** We consider each sub-navigation task separately and start from the first one. Assume the obstacle region $\Delta$ satisfies

$$|\mathcal{E} \setminus (\Delta \cup \mathcal{S} \cup \mathcal{D})| > 2n\hat{r}\|\hat{\mathbf{v}}\|_2 \delta t. \tag{3}$$

For the 1st sub-navigation task, the starting region is $\mathcal{S}^{(1)} = \bigcup_{i=1,...,n} B(\mathbf{p}_i(0), r_i) = \mathcal{S}$ and the destination region is $\mathcal{D}^{(1)} = \bigcup_{i=1,...,n} B(\mathbf{p}_i(T/H), r_i)$. We optimize $\Delta$ 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 $\Delta$ 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 $\Delta$ 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 $\Delta$ 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 $\Delta$ 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 $\delta t$ 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 $i = 1, ..., n$, we have $\sum_{i=1}^n |\mathbf{p}_i^{(1)}| \leq 2n\hat{r}\|\hat{\mathbf{v}}\|_2\delta t$. From (3), $|\mathcal{S}^{(1)}| = |\mathcal{S}|$, $|\mathcal{D}^{(1)}| = |\mathcal{D}|$ and $|\widetilde{\Delta}| = |\Delta|$, we have

$$|\mathcal{E} \setminus (\widetilde{\Delta} \cup \mathcal{S}^{(1)} \cup \mathcal{D}^{(1)})| > 2n\hat{r}\|\hat{\mathbf{v}}\|_2\delta t. \quad (4)$$

This implies the obstacle-free area in $\mathcal{E}$ is larger than the area of $n$ paths $\{\mathbf{p}_{i\sim}^{(1)}\}_{i=1}^n$. 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 $\Delta$ to $\Delta^{(1)}$ can be bounded as

$$\frac{\left|(\Delta \bigcup \Delta^{(1)}) \setminus (\Delta \bigcap \Delta^{(1)})\right|}{2} \leq 2n\hat{r}\|\hat{\mathbf{v}}\|_2\delta t. \quad (5)$$

From (4), $|\Delta^{(1)}| = |\widetilde{\Delta}|$ 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 (3). Therefore, we can repeat the above process iteratively 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 (3) becomes

$$\lim_{\delta t \to 0} |\mathcal{E} \setminus (\Delta \cup \mathcal{S} \cup \mathcal{D})| > 2n\hat{r}\|\hat{\mathbf{v}}\|_2\delta t \to 0. \quad (6)$$

which is satisfied from the condition $\mathcal{E} \setminus (\Delta \cup \mathcal{S} \cup \mathcal{D}) \neq \emptyset$. The changed area of the obstacle region in (5) becomes

$$\lim_{\delta t \to 0} \frac{\left|(\Delta^{(h)} \bigcup \Delta^{(h+1)}) \setminus (\Delta^{(h)} \bigcap \Delta^{(h+1)})\right|}{2\delta t} \leq 2n\hat{r}\|\hat{\mathbf{v}}\|_2. \quad (7)$$

That is, if the capacity of the online environment optimization is higher 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, which completes the proof. ■

Theorem 2 states online environment optimization guarantees the success of all navigation tasks as well as its offline counterpart. The result is established under a mild condition on the changing rate of the obstacle region [cf. (2)] rather than the initial obstacle-free area [cf. (1)]. This stems from the fact that online environment optimization changes the obstacle region *while the agents are moving*, which improves navigation performance only if timely actions can be taken based on instantaneous system states. The completeness analysis in Theorems 1-2 provide theoretical guarantees, demonstrating the applicability of the proposed idea.[1]

## IV. METHODOLOGY

In this section, we formulate Problem 1 mathematically and solve the latter by leveraging reinforcement learning. Specifically, consider the obstacle region $\Delta$ comprising $m$ obstacles $\mathcal{O} = \{O_1, ..., O_m\}$, which can be of any shape and are located at positions $\mathbf{O} = [\mathbf{o}_1, ..., \mathbf{o}_m]$. These obstacles can change positions to improve system performance (i.e., efficiency of agent navigation and cost of obstacle motion). Denote by $\mathbf{X}_o = [\mathbf{x}_{o1}, ..., \mathbf{x}_{om}]$ the obstacle states, $\mathbf{U}_o = [\mathbf{u}_{o1}, ..., \mathbf{u}_{om}]$ the obstacle actions, $\mathbf{X}_a = [\mathbf{x}_{a1}, ..., \mathbf{x}_{an}]$ the agent states and $\mathbf{U}_a = [\mathbf{u}_{a1}, ..., \mathbf{u}_{an}]$ the agent actions. Let $\pi_o(\mathbf{U}_o|\mathbf{X}_o, \mathbf{X}_a)$ be a policy that controls the obstacles, which is a distribution of $\mathbf{U}_o$ conditioned on $\mathbf{X}_o$, $\mathbf{X}_a$. The objective function $f(\mathbf{S}, \mathbf{D}, \pi_a|\mathbf{O}, \pi_o)$ measures the multi-agent navigation performance, while the cost function $\{g_i(\mathbf{O}, \pi_o|\mathbf{S}, \mathbf{D}, \pi_a)\}_{i=1}^m$ indicates environment restrictions on the obstacles. The goal is to find a policy $\pi_o$ that maximizes the system performance $f(\mathbf{S}, \mathbf{D}, \pi_a|\mathbf{O}, \pi_o)$ regularized by the obstacle costs $\{g_i(\mathbf{O}, \pi_o|\mathbf{S}, \mathbf{D}, \pi_a)\}_{i=1}^m$. With the obstacle action space $\mathcal{U}_o$, we formulate Problem 1 as

$$\begin{aligned} \underset{\pi_o}{\text{argmax}} \quad & f(\mathbf{S}, \mathbf{D}, \pi_a|\mathbf{O}, \pi_o) - \sum_{i=1}^m \beta_i g_i(\mathbf{O}, \pi_o|\mathbf{S}, \mathbf{D}, \pi_a) \\ \text{s.t.} \quad & \pi_o(\mathbf{U}_o|\mathbf{X}_o, \mathbf{X}_a) \in \mathcal{U}_o \end{aligned} \quad (8)$$

Solving (8) is difficult and there are mainly three challenges:

(i) The objective function $f(\mathbf{S}, \mathbf{D}, \pi_a|\mathbf{O}, \pi_o)$ and the cost functions $\{g_i(\mathbf{O}, \pi_o|\mathbf{S}, \mathbf{D}, \pi_a)\}_{i=1}^m$ may be complex, inaccurate or unknown, precluding the application of conventional model-based methods and leading to poor performance of heuristic methods.

(ii) 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.

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

Due to the aforementioned challenges, we develop a model-free policy gradient-based reinforcement learning (RL) method to solve this problem.

### A. Reinforcement Learning

We re-formulate problem (8) within an RL setting, and begin by defining a Markov decision process. At each time $t$, the obstacles $\mathcal{O}$ and 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 policy $\pi_o$ and trajectory planner $\pi_a$. 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 of the states conditioned on the previous states and actions. Let $r_{ai}(\mathbf{X}_o^{(t)}, \mathbf{X}_a^{(t)})$ be the reward function of agent $A_i$, which represents instantaneous navigation performance at time $t$. The reward function of obstacle $O_j$ comprises two components: (i) the multi-agent system reward and (ii) the obstacle individual reward, i.e.,

$$r_{oj} = \frac{1}{n}\sum_{i=1}^n r_{ai} + \beta_j r_{j,\text{local}}, \text{ for all } j = 1, ..., m \quad (9)$$

where $r_{ai}$, $r_{j,\text{local}}$ are concise notations of $r_{ai}(\mathbf{X}_o, \mathbf{X}_a)$, $r_{j,\text{local}}(\mathbf{X}_o, \mathbf{X}_a)$ and $\beta_j$ is the regularization parameter. The first term is the team reward that represents the multi-agent system performance and is shared across all obstacles. The second term is the individual reward that corresponds to environment restrictions on obstacle $O_j$ and may be different among obstacles, such as limitations of moving velocity and distance. This reward definition is novel that differs from common RL scenarios. In particular, the obstacle reward is a combination of a global reward with a local reward. The

[1]The theoretical analysis in Section III is general, i.e., the obstacle region can be of any shape and move continuously.

former is the main goal of all obstacles, while the latter is the imposed penalty of a single obstacle. The regularization parameters $\{\beta_j\}_{j=1}^m \in [0,1]^m$ weigh the environment restriction on the navigation performance. The total reward of the obstacles is defined as $r_o = \sum_{j=1}^m r_{oj}$. Let $\gamma \in [0,1]$ be the discount factor accounting for the future rewards and the expected discounted reward can be represented as

$$R(\mathbf{O},\mathbf{S},\mathbf{D}|\pi_o,\pi_a) = \mathbb{E}\Big[\sum_{t=0}^{\infty}\gamma^t r_o^{(t)}\Big] \tag{10}$$
$$= \mathbb{E}\Big[\sum_{t=0}^{\infty}\gamma^t \sum_{j=1}^{m}\sum_{i=1}^{n}\frac{r_{ai}^{(t)}}{n}\Big] + \sum_{j=1}^{m}\beta_j \mathbb{E}\Big[\sum_{t=0}^{\infty}\gamma^t r_{j,\text{local}}^{(t)}\Big]$$

where $\mathbf{O}$, $\mathbf{S}$ and $\mathbf{D}$ are initial positions and destinations that determine initial states $\mathbf{X}_o^{(0)}$ and $\mathbf{X}_a^{(0)}$, and the expectation $\mathbb{E}[\cdot]$ is w.r.t. the action policy and state transition probability. By comparing (10) with (8), we see equivalent representations of the objective and cost functions in the RL domain. The policy $\pi_o$ is modeled by an information processing architecture $\mathbf{\Phi}(\mathbf{X}_o, \mathbf{X}_a, \boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$, which are updated through gradient ascent using policy gradients [24]. The policy distribution is chosen w.r.t. the action space $\mathcal{U}_o$.

### B. Information Processing Architecture

The above framework covers a variety of 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.

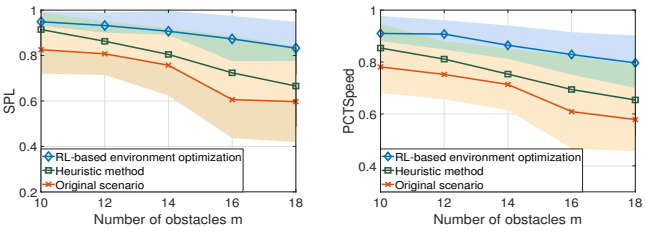
**CNNs for offline discrete settings.** In this setting, we first optimize the obstacles' positions and then navigate the agents. Since computations are performed offline, we collect the states of all obstacles $\mathbf{X}_o$ and agents $\mathbf{X}_a$ apriori, which allows for centralized information processing solutions (e.g., CNNs). CNNs leverage convolutional filters to extract features from image signals and have found wide applications in computer vision [25]–[27]. 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.** For online optimization in continuous environments, 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 [28]–[30]. At each layer $\ell$, 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

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

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 [31]–[34].




(a) (b)

Fig. 2. 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.

## V. EXPERIMENTS

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 $\pi_a$ is Reciprocal Velocity Obstacles (RVO) [2], which is a decentralized method that can be executed with access to local information only.[2] Two metrics are used: Success weighted by Path Length (SPL) [35] and the percentage of the maximal speed (PCTSpeed). The former is a stringent measure combining the success rate and the path length, while the latter is the ratio of the average speed to the maximal one. All results are averaged over 20 trials with random initial configurations.

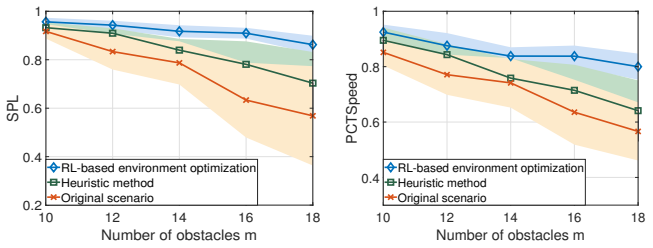
### A. Offline Optimization with Discrete Environment

We consider a grid environment of size $8 \times 8$ with 10 obstacles and 4 agents, which are distributed randomly without overlap. The maximal agent / obstacle velocity is 0.1 per time step 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.

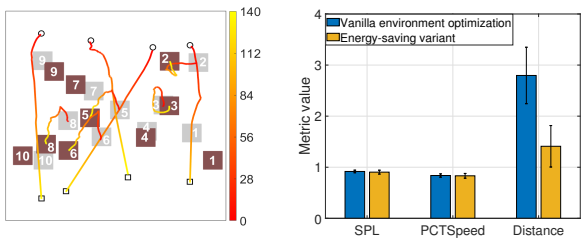
**Training.** The objective is to make agents reach destinations quickly while avoiding collision. The team reward is the sum of the PCTSpeed and the ratio of the shortest distance to the traveled distance, while the local reward is the collision penalty of individual obstacle [cf. (9)]. We parameterize the

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




(a) (b)

Fig. 3. 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.




(a) (b)

Fig. 4. (a) Example of online environment optimization. Circles are the initial positions and squares are the destinations. The grey and brown rectangles are the obstacles before and after environment optimization, and are numbered for exposition. The 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 collisions occur. (b) Performance of the vanilla environment optimization and the energy-saving variant. The results are averaged over 20 trials and the error bar shows the std. dev.

policy with a CNN of 4 layers, each containing 25 features with kernel size $2 \times 2$, and conduct training with PPO [36].

**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. 2 shows 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. 1 shows 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.

### *B. Online Optimization with Continuous Environment*

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 0.1, the communication radius is 2 and the episode time is 500.

**Training.** The team reward in (9) guides the agents to their destinations as quickly as possible and is defined as

$$r_{a,i}^{(t)} = \left( \frac{\mathbf{p}_i^{(t)} - \mathbf{d}_i}{\|\mathbf{p}_i^{(t)} - \mathbf{d}_i\|_2} \cdot \frac{\mathbf{v}_i^{(t)}}{\|\mathbf{v}_i^{(t)}\|_2} \right) \|\mathbf{v}_i^{(t)}\|_2 \qquad (11)$$

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. 3. 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 only for reference. Fig. 4a shows the moving trajectories of the agents and example obstacles. We see that the obstacles make way for the agents to facilitate navigation such that the agents find trajectories close to their shortest paths.

**Optimization criteria.** We show that the objective can capture various criteria. Here, we test whether our approach can learn to also reduce the traveled distance of obstacles. The team reward remains the same, while the local reward becomes the sum of the collision *and speed penalties*. Fig. 4b shows the results with 14 obstacles averaged over 20 trials. This energy-saving variant achieves a comparable performance (slightly worse) across SPL and PCTSpeed, but saves half of the traveled distance, indicating an effective trade-off between performance and energy expenditure.

## VI. CONCLUSION

In this paper, we formulated the problem of environment optimization for multi-agent navigation and developed the offline and online solutions. We conducted the completeness analysis for both variants and provided conditions under which all navigation tasks are guaranteed success. We developed a reinforcement learning-based method to solve the problem in a model-free manner, and integrated two different information processing architectures (i.e., CNNs and GNNs) for policy parameterization w.r.t. specific implementation requirements. The developed method is able to generalize to unseen test instances, captures multiple optimization criteria, and allows for a decentralized implementation. Numerical results show superior performance corroborating theoretical findings, i.e., that environment optimization improves the navigation efficiency by optimizing over obstacle regions.

## REFERENCES


[1] D. Silver, "Cooperative pathfinding," in *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2005, pp. 17–122.

[2] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *IEEE International Conference on Robotics and Automation*, 2008, pp. 1928–1935.

[3] J. P. Van Den Berg and M. H. Overmars, "Prioritized motion planning for multiple robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 430–435.

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

[5] T. S. Standley and R. Korf, "Complete algorithms for cooperative pathfinding problems," in *International Joint Conference on Artificial Intelligence*, 2011, pp. 1–6.

[6] N. Mani, V. Garousi, and B. H. Far, "Search-based testing of multi-agent manufacturing systems for deadlocks based on models," *International Journal on Artificial Intelligence Tools*, vol. 19, no. 04, pp. 417–437, 2010.

[7] A. Ruderman, R. Everett, B. Sikder, H. Soyer, J. Uesato, A. Kumar, C. Beattie, and P. Kohli, "Uncovering Surprising Behaviors in Reinforcement Learning via Worst-case Analysis," *SafeML ICLR 2019 Workshop*, Sep. 2018. [Online]. Available: https://openreview.net/forum?id=SkgZNnR5tX

[8] J. Boudet, J. Lintuvuori, C. Lacouture, T. Barois, A. Deblais, K. Xie, S. Cassagnere, B. Tregon, D. Brückner, J. Baret *et al.*, "From collections of independent, mindless robots to flexible, mobile, and directional superstructures," *Science Robotics*, vol. 6, no. 56, p. eabd0272, 2021.

[9] M. Čáp, P. Novák, A. Kleiner, and M. Selecký, "Prioritized Planning Algorithms for Trajectory Coordination of Multiple Mobile Robots," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 835–849, July 2015.

[10] Q. Wang, R. McIntosh, and M. Brain, "A new-generation automated warehousing capability," *International Journal of Computer Integrated Manufacturing*, vol. 23, no. 6, pp. 565–573, 2010.

[11] H. Bier, "Robotic building (s)," *Next Generation Building*, vol. 1, no. 1, 2014.

[12] L. Custodio and R. Machado, "Flexible automated warehouse: a literature review and an innovative framework," *International Journal of Advanced Manufacturing Technology*, vol. 106, no. 1, pp. 533–558, 2020.

[13] M. Bennewitz, W. Burgard, and S. Thrun, "Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots," *Robotics and Autonomous Systems*, vol. 41, no. 2-3, pp. 89–99, 2002.

[14] M. Jager and B. Nebel, "Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001, pp. 1213–1219.

[15] M. Čáp, J. Vokřínek, and A. Kleiner, "Complete decentralized method for on-line multi-robot trajectory planning in well-formed infrastructures," in *International Conference on Automated Planning and Scheduling*, 2015, pp. 324–332.

[16] W. Wu, S. Bhattacharya, and A. Prorok, "Multi-Robot Path Deconfliction through Prioritization by Path Prospects," *arXiv:1908.02361 [cs]*, Aug. 2019, arXiv: 1908.02361. [Online]. Available: http://arxiv.org/abs/1908.02361

[17] I. Gur, N. Jaques, K. Malta, M. Tiwari, H. Lee, and A. Faust, "Adversarial environment generation for learning to navigate the web," *arXiv:2103.01991 [cs]*, March 2021. [Online]. Available: http://arxiv.org/abs/2103.01991

[18] T. Tanaka and H. Sandberg, "Sdp-based joint sensor and controller design for information-regularized optimal lqg control," in *IEEE Conference on Decision and Control*, 2015, pp. 4486–4491.

[19] S. Tatikonda and S. Mitter, "Control under communication constraints," *IEEE Transactions on Automatic Control*, vol. 49, no. 7, pp. 1056–1068, 2004.

[20] V. Tzoumas, L. Carlone, G. J. Pappas, and A. Jadbabaie, "Sensing-constrained lqg control," in *IEEE American Control Conference*, 2018, pp. 197–202.

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

[22] G. S. Hornby, H. Lipson, and J. B. Pollack, "Generative representations for the automated design of modular physical robots," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 4, pp. 703–719, 2003.

[23] N. Cheney, J. Bongard, V. SunSpiral, and H. Lipson, "Scalable co-optimization of morphology and control in embodied machines," *Journal of The Royal Society Interface*, vol. 15, no. 143, p. 20170937, 2018.

[24] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in Neural Information Processing Systems*, vol. 12, 1999.

[25] M. Browne and S. S. Ghidary, "Convolutional neural networks for image processing: an application in robot vision," in *Australasian Joint Conference on Artificial Intelligence*, 2003, pp. 641–652.

[26] S. Kumra and C. Kanan, "Robotic grasp detection using deep convolutional neural networks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 769–776.

[27] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2018.

[28] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.

[29] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[30] Z. Gao, E. Isufi, and A. Ribeiro, "Stochastic graph neural networks," *IEEE Transactions on Signal Processing*, vol. 69, pp. 4428–4443, 2021.

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

[32] R. Kortvelesy and A. Prorok, "Modgnn: Expert policy approximation in multi-agent systems with a modular graph neural network architecture," in *IEEE International Conference on Robotics and Automation*, 2021, pp. 9161–9167.

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

[34] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph neural networks for decentralized multi-robot path planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 11 785–11 792.

[35] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva *et al.*, "On evaluation of embodied navigation agents," *arXiv:1807.06757 [cs]*, June 2018. [Online]. Available: http://arxiv.org/abs/1807.06757

[36] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv:1707.06347 [cs]*, Aug. 2017. [Online]. Available: http://arxiv.org/abs/1707.06347