# MASAI: Multi-agent Summative Assessment Improvement for Unsupervised Environment Design

**Anonymous Authors**[1]

## Abstract

Reinforcement Learning agents require a distribution of environments for their policy to be trained on. The method or process of defining these environments directly impacts robustness and generalization of the learned agent policies. In single agent reinforcement learning, this problem is often solved by domain randomization, or randomizing the environment and tasks within the scope of the desired operating domain of the agent. The challenge here is to generate both structured and solvable environments that guide the agent's learning process. Most recently, works have sought to produce the environments under the Unsupervised Environment Design (UED) formulation. However, these methods lead to a proliferation of adversarial agents to train one agent for a single agent problem in a discretized task domain. In this work, we aim to automatically generate environments that are solvable and challenging for the continuous multi-agent setting. We base our solution on the Teacher-Student relationship with parameter sharing *Students* where we reimagine the *Teacher* as an environment generator for UED. Our approach uses one environment generator agent (*Teacher*) for any number of learning agents (*Students*). We qualitatively and quantitatively demonstrate that, in terms of multi-agent ($\geq$ 8 agents) navigation and steering, *Students* trained by our approach outperform agents using heuristic search, as well as agents trained by domain randomization. Our code is available at `LinkWithheld`.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

## 1. Introduction

Multi-agent navigation is a well-established but difficult problem. Recent approaches had adapted Reinforcement Learning (RL) to the Multi-agent domain, or Multi-agent Reinforcement Learning (MARL). However, successful navigation learning requires a high degree of generalization to potentially very different environments, which requires designing distribution of tasks and environments that can be used to train and evaluate policies. Nevertheless, manually designing such an effective and appropriate distribution of environments is time-consuming and challenging, as the real-world scenarios are usually complicated and enumerated all of the representative and edge scenes are impractical. We, therefore, aim to automate this process, under the *Unsupervised Environment Design* (UED) (Dennis et al., 2021) framework.

UED is the problem of taking the *underspecified environment* and a policy, and producing a useful distribution of fully specified environments in which that policy can be trained, where *underspecified environment* means the environment that has *free parameters* (*e.g.*, in navigation, the places for the obstacles, agents, and their goals) which control its feature and behavior (Dennis et al., 2021). After training a policy in the distribution of environments generated by UED, an updated policy is obtained and then use UED to generate more diverse environments such that the updated policy can continue to be trained. Hence, the UED formulation can be used to produce capable policies in increasingly complex and difficult environments based on the abilities of the current policy. (Dennis et al., 2021)

In this work, we proposal a novel semi-adversarial training algorithm, MASAI, which aims to solve the UED problem and improve generalization for MARL problems in novel continuous environments. Our algorithm is based on the Teacher-Student relationship with a novel reward signal for UED, and has a simple structure: only two policies 1) the *Student* (*i.e.*, the policy to solve an environment), and 2) the *Teacher* (*i.e.*, the policy to generate free parameters). The motivation for our algorithm is based on the intuitive Teacher-Student relationship. How do Teachers measure whether Students master a concept or not? In general, Teachers prepare two questions of similar difficulty for Students,

one as an example and the other as an exercise. The Environment Generator (*i.e.*, *Teacher*) in every iteration, will produce two sets of environments, one for LEARNING and one for REVIEW based on the *Student*'s performance in the last iteration. The *Student* will collect experience in the LEARNING environment and update its parameters. After the parameters are updated, the *Student* will attempt to solve the REVIEW environments, without updating its parameters. Further, we argue in RL problems, there are usually two classes of rewards: *Task Completion* Reward and *Agent* Reward. *Task Completion* Reward contains the rewards when the agent completes certain tasks (*e.g.*, in navigation, positive rewards for the agent arrives at its goal), and *Agent* Reward includes the cumulative rewards start from initial states until the terminal states (*e.g.*, in navigation, penalties for collisions and positive rewards for moving correctly and reaching intermediate goals). The *Teacher* uses the difference between the reward of the *Student* on REVIEW and LEARNING environments as its reward signal, to update its parameters, and it aims to maximize the *Task Completion* Rewards while minimize the *Agent* Rewards. MASAI requires less parameters than the current state-of-the-art by removing the need for an additional *antagonist* (Dennis et al., 2021).

We qualitatively demonstrate that the *Student* agents trained by MASAI learn and exhibit complex and highly valuable emergent behaviours, and the learned policies also generalize well in unseen difficult steering and navigation scenarios in contrast to the agents trained by classic Domain Randomization (DR) methods. Further, we quantitatively show that under the same conditions, the *Student* agents trained by MASAI have a lower number of collisions and complete goals faster than the agents trained by DR in the domain of multi-agent navigation and steering.

## 2. Unsupervised Environment Design

We briefly summarize the recent formalization of the UED problem in this section (Dennis et al., 2021). The formal definition of UED is the problem of using an underspecified environment to produce a distribution over fully specified environments, which offers the continued or curriculum learning of a particular task-solving policy. We now detail the meaning and relevant distinction of the *fully specified* versus *underspecfied* environments.

Fully specified environments are modeled with a Partially Observable Markov Decision Process (POMDP), represented by a tuple $\langle A, O, S, \mathcal{T}, \mathcal{I}, \mathcal{R}, \gamma \rangle$ where $A$ is a collection of actions, $O$ is a collection of observations, $S$ is a set of states, $\mathcal{T}$ is the transition function of $S \times A \rightarrow \Delta(S)$, $\mathcal{I}$ is the observation function of $S \rightarrow O$, $\mathcal{R}$ is the reward function of $S \rightarrow \mathbb{R}$, and $\gamma$ is the discount factor. This contrasts the underspecified environment which can be modeled as an Underspecified Partially Observable Markov Decision Process (UPOMDP) represented by a tuple $\langle A, O, \Theta, S^{\mathcal{M}}, \mathcal{T}^{\mathcal{M}}, \mathcal{I}^{\mathcal{M}}, \mathcal{R}^{\mathcal{M}}, \gamma \rangle$. The only difference here between a POMDP and a UPOMDP is that a UPOMDP contains a set $\Theta$ representing the free parameters of the environments, and it is these free parameters that can be selected at each time step and integrated into the transition function $\mathcal{T}^{\mathcal{M}}$ as $S \times A \times \Theta \rightarrow \Delta(S)$. The solution to UED proposed in (Dennis et al., 2021) then is to have an environment policy $\Lambda : \Pi \rightarrow \Delta(\Theta^T)$ where $\Pi$ is the set of possible policies and $\Theta^T$ is the set of possible trajectory of environment parameters. In this formulation, UED aims to generate a distribution of environments that caters to the continued learning of a particular agent policy. To solve this, the authors proposed PAIRED (Dennis et al., 2021) to approximate the environment policy. PAIRED solves the UED problem by introducing three agents the *antagonist*, *protagonist*, and *environment-generating adversary*. The antagonist is allied with the environment-generating adversary to guide the adversary from constructing unsolvable environments. The goal of the environment adversary is to build environments in which the antagonist achieves high reward and the protagonist receives a low reward. The protagonist is the agent to be trained for solving the environments.

## 3. Multi-agent Summative Assessment Improvement for Unsupervised Environment Design

We propose a novel semi-adversarial training algorithm, MASAI, which is designed to solve the UED problem and improve generalization, inspired by the intuitive Teacher-Student relationship. How do Teachers measure whether Students master a concept or not? Generally, the *Teacher* prepares two questions of similar difficulty for the Student, and takes one of them as an example to teach (*i.e., Student* will hopefully learn the concept from this example), and then use the other as a quiz to test Student's understanding. MASAI formulates this relationship for multi-agent training. MASAI is summarized in Algorithm 1 and its semantics is shown in Figure 1.

There are two types of trainable agent policies in MASAI, *Teacher* agent (*i.e.,* Unsupervised Environment Generator) and *Student* agent, where the policy is denoted by the symbol $\pi$. *Teacher* agent aims to produce solvable and valuable free environment parameters $\vec{\theta}$, and the *Student* agent aims to learn specific tasks, for instance, navigation and steering. We denote the space of all environment parameters as $\Theta$, and we assume for a *Teacher* agent policy the produced free environment parameters $\vec{\theta}$ follows the distribution of $\Theta_T$. Moreover, we argue that in many RL formulations, it is possible to create two classes of rewards for the agent; the *Task Completion* reward and the *Agent* reward. Task Completion

**Algorithm 1** MASAI

Denote $|S|$ as the number of *Student* agent shares the same *Student* policy $\pi_S$.
Denote $\mathcal{R}(\tau)$ as an operator that sums undiscounted return of $|S|$ *Student* agents in a trajectory $\tau$.
Randomly initialize the *Student* $\pi_S$ and the *Teacher* $\pi_T$.
**repeat**
  Use $\pi_T$ to generate LEARNING environment parameters: $\theta_L \sim \Theta_T$
  Use $\pi_T$ to generate REVIEW environment parameters: $\theta_R \sim \Theta_T$
  Collect the LEARNING trajectory $\tau_L$ using $\pi_S$, and compute Agent Reward $\mathcal{R}_{Agent}(\tau_L)$ and Task Completion Reward $\mathcal{R}_{Task}(\tau_L)$.
  Train policy $\pi_S$ with RL update and discounted return from trajectory $\tau_L$.
  Collect the REVIEW trajectory $\tau_R$ using $\pi_S$, and compute Agent Reward $\mathcal{R}_{Agent}(\tau_R)$ and Task Completion Reward $\mathcal{R}_{Task}(\tau_R)$.
  Compute $\text{GAIN}_{Agent} = \mathcal{R}_{Agent}(\tau_R) - \mathcal{R}_{Agent}(\tau_L)$.
  Compute $\text{GAIN}_{Task} = \mathcal{R}_{Task}(\tau_R) - \mathcal{R}_{Task}(\tau_L)$.
  Train policy $\pi_T$ with RL update and minimize $\text{GAIN}_{Agent}$ while maximize $\text{GAIN}_{Task}$.
**until** convergence

reward is an instant reward, to signal the agent whether it completes the desirable tasks or not. Agent reward contains the cumulative rewards from the initial states until the terminal states, to guide the agent away from low value behaviours (*e.g.,* in the multi-agent navigation problem, collisions with other agents and static/dynamic obstacles need to avoided), and encourage the agent toward the preferable actions and emergent behaviour policies (*e.g.,* in the multi-agent navigation problem, reaching the intermediate waypoints toward the final target should be incentivized, laminar flows and vortices should emerge in conflicting flows).

To train the *Teacher* agent and the *Student* agents, we first let the *Teacher* agent construct $n$ LEARNING environments and $n$ REVIEW environments. Based on the assumption that the produced free environment parameters $\vec{\theta}_T$ follows the distribution of $\Theta_T$, the LEARNING and REVIEW environments share a similar difficulty, and if $n$ is large enough, the difficulty of the constructed environments would be equivalent. Denote $\mathcal{R}(\tau)$ as an operator that sums undiscounted return of all *Student* agents in a trajectory $\tau$. The *Student* agents will interact with the LEARNING environments, collect the experience and trajectories $\tau_L$, and receive the Task Completion Rewards $\mathcal{R}_{Task}(\tau_L)$ and Agent Rewards $\mathcal{R}_{Agent}(\tau_L)$. The weights of the *Student* agent are then updated with the collected trajectories by, for instance, Proximal Policy Optimization (Schulman et al., 2017) or Twin Delayed Deep Deterministic policy gradient algorithm (Fujimoto et al., 2018). With the updated weights, the *Student* agent will apply the learned knowledge from LEARNING environments to the REVIEW environments, and also obtain the Task Completion Rewards $\mathcal{R}_{Task}(\tau_R)$ and the Agent Rewards $\mathcal{R}_{Agent}(\tau_R)$.

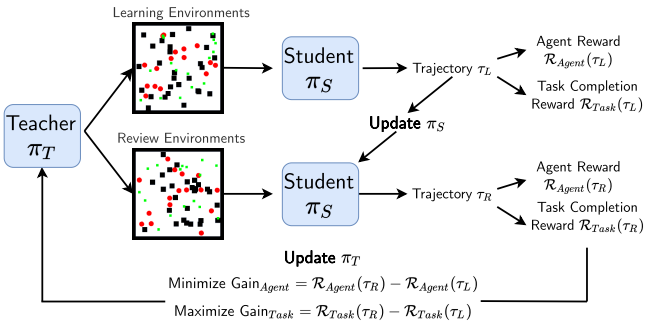


*Figure 1.* Semantics of the MASAI algorithm. The *Teacher* agent first designs the LEARNING and REVIEW environments, and the *Student* agent is updated on the LEARNING environment and applies the learned policy to the REVIEW environment. The *Teacher* agent optimizes its policy with the novel reward signal GAIN. See Algorithm 1 and Section 3 for more details.

Our main contribution is the MASAI methodology to train the *Teacher* agent. We introduce the following novel reward signals, the difference between the Task Completion Rewards and the Agent Rewards of the *Student* agents obtained from the REVIEW and LEARNING environment:

$$\begin{aligned} \text{GAIN}_{Agent} &= \mathcal{R}_{Agent}(\tau_R) - \mathcal{R}_{Agent}(\tau_L) \\ \text{GAIN}_{Task} &= \mathcal{R}_{Task}(\tau_R) - \mathcal{R}_{Task}(\tau_L) \\ \text{GAIN} &= \text{GAIN}_{Task} - \text{GAIN}_{Agent} \end{aligned} \tag{1}$$

The *Student* agent always tries to maximize its Task Completion Rewards and Agent rewards, as defined in the RL formulation. To generate solvable environments, the *Teacher* agent also needs to maximize the $\text{GAIN}_{Task}$, as it needs to produce environments that can teach the *Student* agent to complete more tasks. However, if the *Teacher* agent where to simply maximize the $\text{GAIN}_{Task}$ this would lead to generating trivial environments for the *Student* agent. Thus, we argue that it is essential for the *Teacher* agent to, in addition, *minimize* $\text{GAIN}_{Task}$. In other words, as long as the *Teacher* agent is capable of generating solvable environments (by maximize $\text{GAIN}_{Task}$), it should "set up" the *Student* agent in the produced environments, to construct more edge cases and hard scenarios at the local and global level, to motivate the *Student* agents to deviate from its current parameter distribution to further explore better weight structure, ideally leading to a generalization of skills. The sequence of free parameters $\vec{\theta}$ in each step the *Teacher* agent produced form the training trajectories, with the addition of $\text{GAIN}_{Task}$ (*i.e., maximize*) and $-\text{GAIN}_{Agent}$ (*i.e., minimize*) as its reward

signal. To complete the approach, we train the *Student* agents and *Teacher* agent alternatively until convergence or after a certain number of iterations.

## 4. Experimental Setup

The proposed MASAI is a general algorithm to improve the generalization of any policy. We test MASAI on the task of navigation and steering due to its popularity in the MARL application and its sensitivity to unseen environments.

### 4.1. Teacher

In this section, we outline the details of the *Teacher* agent, including the architecture and state observations in terms of navigation and steering. In our experiments, the *Teacher* agent works in a $16m \times 16m$ environment, with a fixed number of agents (16 agents, coloured in red), their goals (16 coloured in green), and static obstacles (28 coloured in black). In total, there are 60 objects in an environment. See Figure 2 for example generated environments.

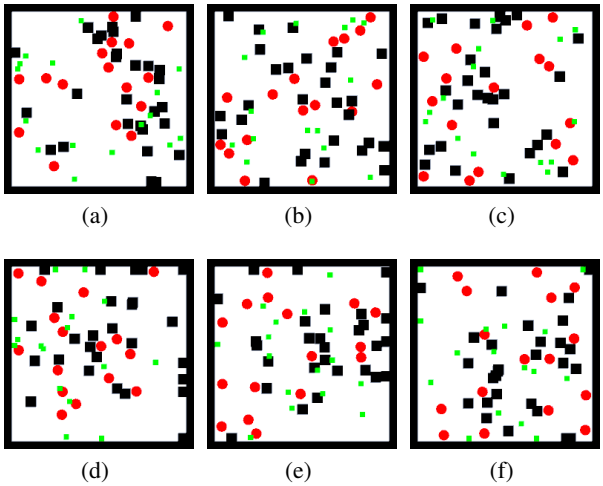


*Figure 2.* Sample environments constructed by the Domain Randomization (top row) and MASAI (bottom row) approaches. The locations to place objects are uniformly randomly selected, without taking the policy of the agent into consideration.

The observation of the *Teacher* agent in each frame are an $84 \times 84 \times 3$ image which captures the whole environment (*e.g.,* Figure 2a), and a scalar $k$ that enumerates the class of the object it should place. Let $k = 1$ for static obstacle, $k = 2$ for goal, and $k = 3$ for agent. Figure 3 illustrates the observations and actions of the *Teacher* agent. In each time step, the *Teacher* agent takes a visual observation of the environment, and a scalar observation to indicate the class of objects it should place next. The action of the the *Teacher* agent is a two-dimensional coordinate within the environment. The *Teacher* agent at Time Step 0 takes an empty environment and a scalar $k = 3$ to place the first agent. At Time Step 1, the *Teacher* agent takes the visual observation of the environment (currently only one agent) and a scalar $k = 2$ to determine the place of the goal for this agent. This process continues after a fixed number of agents and their goals. Finally, the *Teacher* agent starts to place static obstacles by observing the current environment, with scalar $k = 1$ as its input (see Time Step 33). We note the first visual observation is always the same at the beginning of the environment construction. Thus, the policy gradient method is employed to sample actions from the policy.

#### 4.1.1. ARCHITECTURE

The *Teacher* agent network architecture consists of a single convolutional layer that connects to an LSTM and then to two fully connected layers which connected to the policy outputs. The additional scalar observation $k$ is connected directly to the LSTM layer. A second network with identical architecture is used to estimate the value function (Dennis et al., 2021). The convolutional layer contains 128 filters with $3 \times 3$ kernel, and an LSTM size of 256 cells, two fully connected layers of size 64 each, and a fully connected layer of size 9 to process the object class and connects to the LSTM. The policy output is a 2-dimensional location to place the next object in the current environment.

### 4.2. Student

In this section, we describe the *Student* agent for learning local steering and collision avoidance for navigating multi-agent environments, or a synthetic crowds agents. The observation of the *Student* agent in each frame is a 24 dimension vector, consisting of depth detection, relative velocity to the maximum allowed speed, and relative position to the next intermediate waypoint. Moreover, we stack the past 4 frames (temporally widen the observation), which results in a 96-dimensional observation vector for each agent as its policy network input. The policy output is a 2-dimensional virtual force to be applied to the *Student* agent rigidbody. The *Student* agent updates at 12.5Hz. The reward function for the *Student* agent is taken from (Haworth et al., 2020b), which is defined as a combination of distance-based rewards. Let $\Delta = ||\texttt{pos}(\text{agent}_i) - g_i^H||$, where $\texttt{pos}$ computes the current location of agent $i$ and $g_i^H$ denotes the location of the goal of agent $i$:

$$r_{goal} = \begin{cases} 20 & \text{if } \Delta < 1 \\ \exp\left(-\Delta^2\right) & \text{otherwise.} \end{cases} \tag{2}$$

The instantaneous reward signal places a large reward value when reaching goals as soon as possible, while the continuous component guides learning policies toward its goals. A second penalty term $r_{collision}$ checks agent-agent or agent-

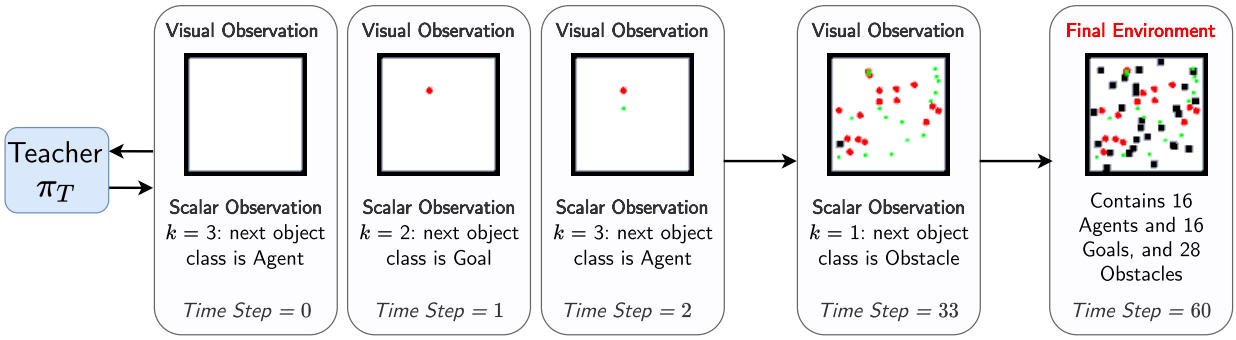


*Figure 3.* Visualization of the input observations and output actions for the *Teacher* agent. The *Teacher* agent takes the current view of the environment and a conditional variable $k$ that indicates the object to place for the next step. In this experiment, we set $k = 1$ for Obstacle, $k = 2$ for Goal and $k = 3$ for Agent. Note the first visual observation will be the same so policy gradient is used to sample actions from the policy. The action of the *Teacher* agent is a two-dimensional coordinate in the environment. See Section 3 for more details.

obstacle collisions.

$$r_{collision} = \begin{cases} -1 & \text{if collision occurs} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

#### 4.2.1. ARCHITECTURE

The *Student* agent network architecture consists of a LSTM and then to two fully connected layers which connect to the policy output. A second network with identical architecture is used to estimate the value function (Dennis et al., 2021). The LSTM is of size 256 cells, and two fully connected layers of 64 which connect to either the policy outputs or the value estimate.

### 4.3. Training

The experiments were conducted on a server with 48 Intel Xeon E5-2650 CPUs. To improve reproducibility, we control the random seed for all training and experiments. For instance, the initialization of neural networks in each experiment will be the same. The inputs and rewards were normalized for both the *Student* agents and the *Teacher* agent, and both were trained with Proximal Policy Optimization (PPO) (Schulman et al., 2017) algorithm with a discount factor of 0.99, a learning rate of 0.0001 with Adam optimizer (Kingma & Ba, 2017), and an 8 parallel training environments to collect a batch of episodes, which was used to complete on training update In other words, *Teacher* agent creates 8 LEARNING and REVIEW environments each iteration. All *Student* agents shared the same policy network, *i.e.*, parameter sharing for multi-agent reinforcement learning. We empirically set the maximum allowed steps of the *Student* agents during training to 250.

## 5. Experiments

We qualitatively demonstrate that the *Student* agent trained by MASAI generalizes better in unseen environments than by DR and heuristic search, which are supported by the quantitative evaluation. We use the DR technique as our primary baseline representing how environments have been generated in past solutions to the multi-agent navigation problem. In the navigation and steering task, DR is employed by uniformly and randomly choosing a location to place objects. In the following experiments, the *only* difference is the training algorithm, *i.e.,* DR versus MASAI.

### 5.1. Qualitative Evaluation

Sample environments constructed by DR and using MASAI are shown in Figure 2. In a small environment, such as $16m \times 16m$, there is an infinite number of configurations, but the computational resource is limited. It is crucial to reduce the free parameter space and produce environments that can instruct the *Student* agent to learn (not simply memorize) quickly. MASAI solves this issue precisely. From Figure 2, we can observe that the DR places objects that are spread around across the environment, due to the uniform distribution sample space. Moreover, there are cases (Figure 2a 2c) the goals are overlapped with the static obstacles and some agents are stuck between static obstacles (Figure 2b), which result in unsolvable cases. While MASAI tends to cluster the objects and put goals near the static obstacles, and create challenging scenarios, as shown in Figure 2. For instance, in Figure 2d and Figure 2e, the central agents are surround by static obstacles, forming challenging training scenario *caves*. In Figure 2f, MASAI creates nearly a wall-like configuration in the left to separate agents and their goals.

To evaluate the learned *Student* agent policy, we create several common navigation and steering scenarios. At the smaller scale, these include *Egress*, *Simple wall*, *Crossing with obstacles*, *Crossing groups*, *Surprise*, and *Rooms egress*. At the larger scale, we use *Bi-directional hallway* and *Diametric goals* scenarios which contain forty-nine dy-

namic agents, as shown in Figure 5. In all evaluations, we use the weights of the *Student* agent saved at the highest average return (see Figure 7) in both DR and MASAI to solve the evaluation scenarios.

Results for the six smaller scale common steering and navigation scenarios are shown in Figure 4. In these scenarios we show that MASAI guides the learning of high value policies for emergent behaviours. In particular, *Student* agent learns to side step and wait for other *Student* agents to reduce collisions. This is a complicated behaviour that extends over time that even industry standard steering simulators struggle with or do not produce. However, DR policies never discover this policy. We also show that polices learned from DR can lead to equilibrium cases where the net force on competing agents reaches equilibrium and the agents never complete the scenario (see *Surprise* and *Crossing with Obstacles* in Figure 4). Policies learned from MASAI avoid this problem through side-stepping and stop-and-wait behaviours. These represent real courtesies people give to each other in such low density scenarios.

From Figure 5 and Figure 6, *Student* agent policy and DR agent policy are able to solve the environments. From the simulation, we notice that the *Student* agent trained by MASAI demonstrates important emergent steering skills. In Figure 5, initially, the lower group and upper group are aiming to avoid collision by steering far right and far left. Moreover, both MASAI *Student* agent and the DR policies cluster in the *Diametric Goals* scenario and created a large number of collisions, but the *Student* agent trained by MASAI reduces the amount of time of clustering much faster than the agent trained by DR through an emergent vortex behaviour. These are unseen environments in the training environment distribution, showing MASAI trains more generalized agents.

### 5.2. Quantitative Evaluation

To quantitatively estimate the performance of the trained *Student* agent policies, we compute the number of collisions and the elapsed time of agents to reach its goal. We also compare the results for agents follows the path planned by Heuristic Search (*i.e.,* A* path planning) due to its wide usage in synthetic crowds applications.

Based on Table 1 and Table 2, it is clear the *Student* agents trained by MASAI has a superior generalization performance than the agent trained by DR in various metrics regarding the quality of steering in a multi-agent navigation setting. We note the mean of time used to reach goals by the agents that are driven by heuristic search is the lowest is due to the fact the agents will move straight up to their goals, without any consideration of steering. This will result in the fastest navigation and the least pleasant steering.

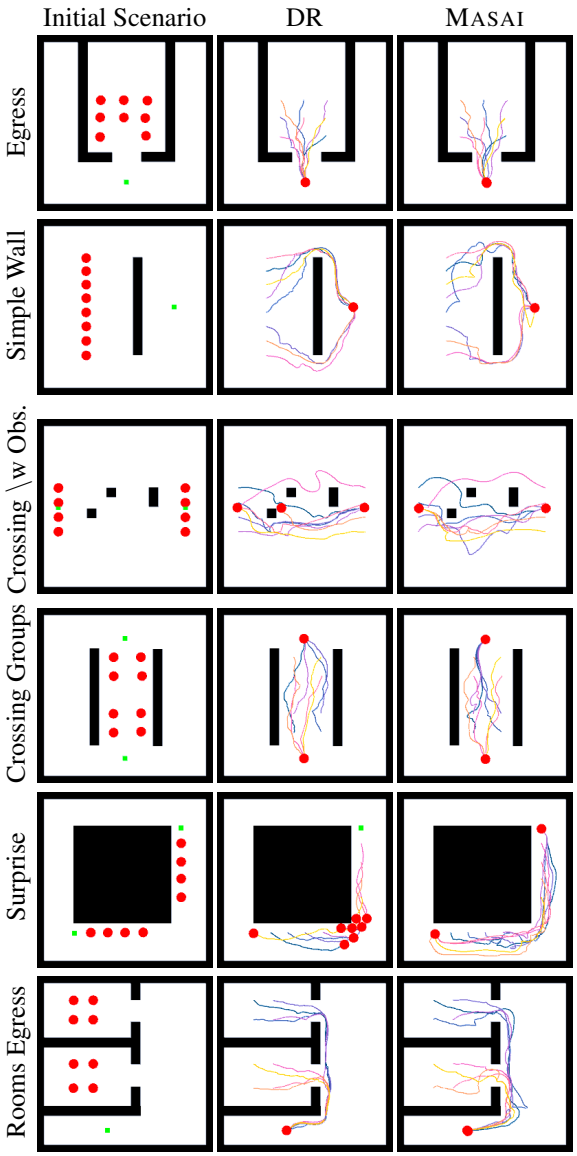


*Figure 4.* Six common navigation and steering evaluation scenarios. Agents trained by Domain Randomization takes 11.92 seconds to complete *Egress*, 14.16 seconds to complete *Simple* Wall, 40 seconds and *not* complete *Crossing with Obstacles*, 13.04 seconds to complete *Crossing Groups*, and 40 seconds and *not* complete *Surprise*, and 24.72 seconds to complete *Room Egress*. *Student* agent trained by MASAI takes 7.52 seconds to complete *Egress*, 15.44 seconds to complete *Simple Wall*, 11.20 seconds to complete *Crossing with Obstacles*, 9.44 seconds to complete *Crossing Groups*, and 23.36 seconds to complete *Surprise*, and 17.68 seconds to complete *Room Egress*.

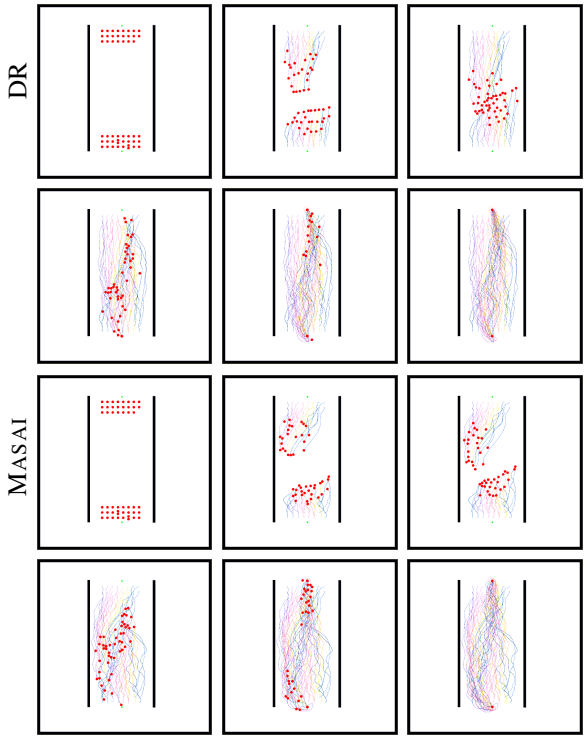


*Figure 5.* Bi-directional hallway with forty-nine agents. Agents trained by DR takes 35.36 seconds to complete whereas *Student* agent trained by MASAI takes 28.96 seconds.

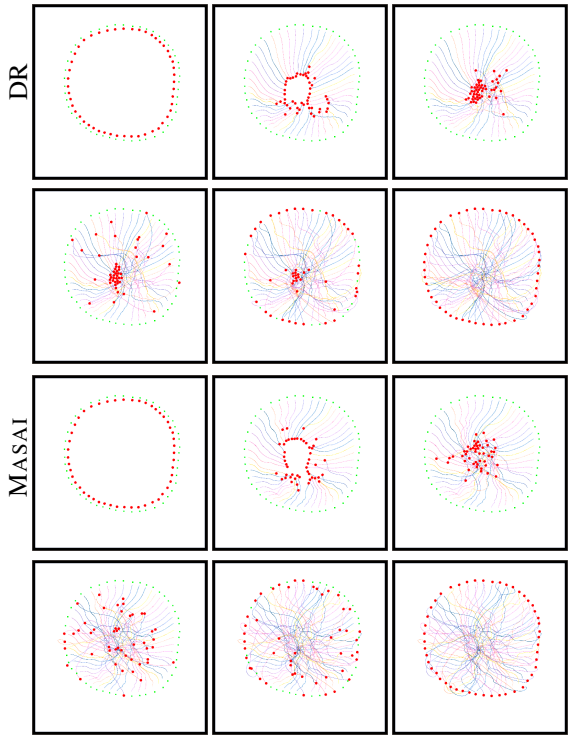


*Figure 6.* Diametric goals with forty-nine agents. Agents trained by DR takes 47.28 seconds to complete whereas *Student* agent trained by MASAI takes 31.68 seconds.

*Table 1.* Quantitative evaluation in the six common navigation scenarios (see Figure 4) of the *Student* agents trained by MASAI, the agents trained by Domain Randomization (DR), and the performance of the agent following the Heuristic Search (HS) plan is provided. Unit of time is second.

| METRICS | HS | DR | MASAI |
|---|---|---|---|
| AGENTS COMPLETED | 48 | 40 | 48 |
| TOTAL COLLISIONS | 325 | 232 | **151** |
| MEAN COLLISIONS | 6.9148 | 5.8 | **3.1458** |
| STD COLLISIONS | 4.9156 | 5.1613 | **2.2172** |
| MEAN TIME TO GOAL | **7.92** | 43.64 | 39.34 |
| STD TIME TO GOAL | **6.93** | 25.07 | 21.54 |

*Table 2.* Quantitative evaluation in the two forty-nine agents evaluation scenarios (see Figure 5) of the *Student* agents trained by MASAI, the agents trained by Domain Randomization (DR), and the performance of the agent following the Heuristic Search (HS) plan is provided. Unit of time is second.

| METRICS | HS | DR | MASAI |
|---|---|---|---|
| AGENTS COMPLETED | 98 | 98 | 98 |
| TOTAL COLLISIONS | 1294 | 1024 | **970** |
| MEAN COLLISIONS | 13.2041 | 10.449 | **9.8980** |
| STD COLLISIONS | 5.4510 | 5.6715 | **4.5546** |
| MEAN TIME TO GOAL | **7.80** | 27.79 | 21.83 |
| STD TIME TO GOAL | **0.93** | 7.76 | 3.27 |

Figure 7 illustrates the average undiscounted return curve during training of the *Student* agents trained by MASAI is higher than and trained by DR, which aligns with the quantitative metrics in Table 1 and qualitative evaluation.

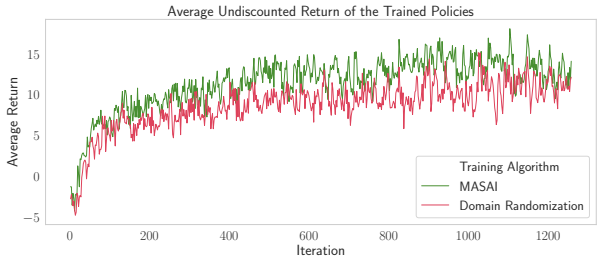


*Figure 7.* Average undiscounted return curves during training of the *Student* agents trained by MASAI and agents by Domain Randomization. We note that the average return curve is not an ideal quantitative evaluation metric as the environments during training are different.

## 6. Related Work

Prior works on automating the process of generating environments include DR and Minimax Adversary. DR aims to generate fully specified environments, without taking the current learning agent policy into consideration (Jakobi,

1997; Savage, 1951; Graves et al., 2017). These environment parameters are randomly sampled for each episode, and then a policy is trained over the generated environment. Nevertheless, this does not guarantee policy performance on a carefully designed configuration of free parameters. Minimax adversary training aims to produce environments by minimizing the reward of the current learning agent policy, *i.e.,* minimax training (Pinto et al., 2016; Morimoto & Doya, 2005; Shen et al., 2019). Although these approaches have demonstrated the capability of producing compelling environments, they can fail to generate truly valuable environments. Uniformly random environments will usually fail to generate challenging and sometimes solvable environments, while minimax adversary is incentivized to create environments that are impossible to solve. In many applications, both of these algorithms fail to produce valuable and solvable structures. The recently introduced PAIRED algorithm is a middle ground of environment generators which maximize *regret*, producing difficult but solvable tasks (Dennis et al., 2021). We discuss PAIRED in detail in Section 2.

Curriculum Learning is a well-studied and widely applied solution to difficult training problems in RL (Bengio et al., 2009; Matiisen et al., 2017; Leibo et al., 2019; Graves et al., 2017; Florensa et al., 2017). In basic curriculum learning, the learning agents is exposed to increasingly difficult tasks to learn or environments to train in. However, it is increasingly difficult to decide what is 'difficult' as well as when, where, and how much a particular element of a curriculum should be used during training. Teacher-Student Curriculum Learning (TSCL) is a framework designed for automatic curriculum learning, where the *Student* tries to learn a complex task and the *Teacher* automatically chooses sub-tasks from a given set for the *Student* to train on (Matiisen et al., 2017). The *Teacher* algorithms rely on the intuition that the *Student* should practice more of those tasks on which it makes the fastest progress. Nevertheless, the *Teacher* in the proposed algorithm is not parameterized or trainable, it maintains an array of probability for each sub-task, *i.e.*, only the *Student* is trainable. The *Teacher* proposes tasks through an epsilon-greedy policy on that probability array. Another Teacher-Student approach for predicting hospital inpatient admission location separates the Teacher and Student into two neural networks where the Teacher learns to select items from a dataset for the Student to learn on (El-Bouri et al., 2020).

Our work proposes to address the shortcomings of prior methods when applied to a more difficult problem domain. We apply a novel Teacher-Student approach top the UED problem for continuous multi-agent reinforcement learning. MASAI fully parameterizes both the *Teacher* agent and *Student* agent using deep neural networks, where the proposed tasks for *Student* agents are produced directly without any prior selection algorithms. In this work, we propose incentivize the *Teacher* agent, (*i.e.,* the Unsupervised Environment Generator), to produce free parameters that, when a policy is trained on it, can observe novel reward signals, GAIN, which guides the *Teacher* agent to construct environments that are not only as difficult as possible but also catered for automatically improving the sophistication of the policy of the *Student* agents. To test our approach, we look to the local steering and collisions avoidance in multi-agent navigation literature for a difficult problem domain. Navigation and steering is a burgeoning application of MARL, especially in terms of crowd simulation (Haworth et al., 2020a; Berseth et al., 2020; Lee et al., 2018; Lowe et al., 2017; Lan et al., 2020; Sun et al., 2019; Bisagno et al., 2019). The problem is continuous, multi-agent, and often semi-chaotic. Additionally, earlier work in comparative crowds analysis proposed a carefully crafted rule-based algorithm to generate challenging environments to evaluate steering algorithms (Kapadia et al., 2011). However, the space of rule based procedural content (environment) generation is beyond the scope of this paper. The proposed MASAI approach is not limited to navigation and steering and ideally can be applied to many other difficult continuous MARL problems.

## 7. Discussion

In this work, we proposed an algorithm for solving the unsupervised environment design problem in the continuous multi-agent reinforcement learning domain. Our method estimates the environment generator policy by carefully formulating a training procedure informed by the Teacher-Student approach and guided by a novel reward signal designed to produce compelling, challenging, and solvable environments in an automatic curriculum for training agents in difficult problem domains. Our method produces such environments and that the resultant training regime leads to highly valuable policies. We showed that the approach solves the UED problem for training multi-agent navigation and the the trained agents learn valuable emergent polices which generalize in entirely unseen environments.

Limitations of the method are mainly related to the underlying methodology for training in the MARL domain. We utilize parameter sharing which has high utility and mitigates issues like non-stationarity. However, there exists many approaches to multi-agent training which may or may not be adapted into this framework. Additionally, it remains to be seen how successful such a method would be in a real world setting with highly uncertain or noisy inputs.

For future work, we plan to propose extensions to the PAIRED algorithm for the multi-agent continuous domain. Moreover, we also want to test MASAI on other tasks, such as computer vision and natural language processing.

## References


Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 41–48, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553380. URL https://doi.org/10.1145/1553374.1553380.

Berseth, G., Haworth, B., Moon, S., Kapadia, M., and Faloutsos, P. Multi-agent hierarchical reinforcement learning for humanoid navigation, 2020. URL https://openreview.net/forum?id=B1ldb6NKDr.

Bisagno, N., Garau, N., Montagner, A., and Conci, N. Virtual crowds: An lstm-based framework for crowd simulation. In *International Conference on Image Analysis and Processing*, pp. 117–127. Springer, 2019.

Dennis, M., Jaques, N., Vinitsky, E., Bayen, A., Russell, S., Critch, A., and Levine, S. Emergent complexity and zero-shot transfer via unsupervised environment design, 2021.

El-Bouri, R., Eyre, D., Watkinson, P., Zhu, T., and Clifton, D. Student-teacher curriculum learning via reinforcement learning: predicting hospital inpatient admission location. In *International Conference on Machine Learning*, pp. 2848–2857. PMLR, 2020.

Florensa, C., Held, D., Wulfmeier, M., Zhang, M., and Abbeel, P. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pp. 482–495. PMLR, 2017.

Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. *CoRR*, abs/1802.09477, 2018. URL http://arxiv.org/abs/1802.09477.

Graves, A., Bellemare, M. G., Menick, J., Munos, R., and Kavukcuoglu, K. Automated curriculum learning for neural networks, 2017.

Haworth, B., Berseth, G., Moon, S., Faloutsos, P., and Kapadia, M. Deep integration of physical humanoid control and crowd navigation. In *Motion, Interaction and Games*, MIG '20, New York, NY, USA, 2020a. Association for Computing Machinery. ISBN 9781450381710. doi: 10.1145/3424636.3426894. URL https://doi.org/10.1145/3424636.3426894.

Haworth, B., Berseth, G., Moon, S., Faloutsos, P., and Kapadia, M. Deep integration of physical humanoid control and crowd navigation. In *Motion, Interaction and Games*, MIG '20, New York, NY, USA, 2020b. Association for Computing Machinery. ISBN 9781450381710. doi: 10.1145/3424636.3426894. URL https://doi.org/10.1145/3424636.3426894.

Jakobi, N. Evolutionary robotics and the radical envelope-of-noise hypothesis. *Adaptive Behavior*, 6(2):325–368, 1997. doi: 10.1177/105971239700600205. URL https://doi.org/10.1177/105971239700600205.

Kapadia, M., Wang, M., Singh, S., Reinman, G., and Faloutsos, P. Scenario space: Characterizing coverage, quality, and failure of steering algorithms. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '11, pp. 53–62, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450309233. doi: 10.1145/2019406.2019414. URL https://doi.org/10.1145/2019406.2019414.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2017.

Lan, X., Liu, Y., and Zhao, Z. Cooperative control for swarming systems based on reinforcement learning in unknown dynamic environment. *Neurocomputing*, 410: 410–418, 2020.

Lee, J., Won, J., and Lee, J. Crowd simulation by deep reinforcement learning. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games*, MIG '18, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450360159. doi: 10.1145/3274247.3274510. URL https://doi.org/10.1145/3274247.3274510.

Leibo, J. Z., Hughes, E., Lanctot, M., and Graepel, T. Autocurricula and the emergence of innovation from social interaction: A manifesto for multi-agent intelligence research, 2019.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6382–6393, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Matiisen, T., Oliver, A., Cohen, T., and Schulman, J. Teacher-student curriculum learning, 2017.

Morimoto, J. and Doya, K. Robust reinforcement learning. *Neural computation*, 17(2):335–359, 2005.

Pinto, L., Davidson, J., and Gupta, A. Supervision via competition: Robot adversaries for learning tasks, 2016.

Savage, L. J. The theory of statistical decision. *Journal of the American Statistical Association*, 46(253):55–67, 1951. doi: 10.1080/01621459.1951.10500768. URL

https://www.tandfonline.com/doi/abs/10.1080/01621459.1951.10500768.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms, 2017.

Shen, Z., He, Z., and Xue, X. Meal: Multi-model ensemble via adversarial learning, 2019.

Sun, L., Zhai, J., and Qin, W. Crowd navigation in an unknown and dynamic environment based on deep reinforcement learning. *IEEE Access*, 7:109544–109554, 2019.