Situated Learners in a Sequential Decision-Making Setting

Timur Kasimov*
Computer Science (Grinnell College)
Grinnell, USA
email: kasimovt@grinnell.edu

Shinon Takei*
Computer Science (Grinnell College)
Grinnell, USA
email: takeishi@grinnell.edu

Hanfeng Lu
Computer Science (Grinnell College)
Grinnell, USA
email: hlu002327@gmail.com

Mingi Lee
Computer Science (Grinnell College)
Grinnell, USA
email: leemingi@grinnell.edu

Fernanda Eliott
Computer Science (Grinnell College)
Grinnell, USA
email: eliottfe@grinnell.edu

***Abstract*—Tricky coordination challenges can emerge from combining distributed settings with independent learners (since these learners have only access to limited information). Still, treating agents as independent learners can help mitigate the problem of observing multiple agents' joint actions. Here, we detail and examine the Discrete Smart Surface benchmark with two goals: 1. Follow a call to the Multi-Agent Systems community to consider the environment as an important entity at the application level, and 2. Show/discuss our experimental results for combining independent learners with the Discrete Smart Surface benchmark under various dimensions and agent-weighting systems. Investigating challenges in Multi-Agent Systems can be particularly insightful for applications that rely on multiple decision-makers; thus, we thoroughly reflect on our experimental results and exemplify challenges, such as action shadowing.**

Keywords-action shadowing; discrete actuator arrays; independent learners; sequential decision making.

I. INTRODUCTION

Given the accelerated advances in computational infrastructure over the past few years, is it still advantageous to study Multi-Agent Systems (MAS) in tabular settings? Or, in other words, what can we gain from investigating tabular worlds? As Gronauer and Diepold [1] overview the landscape of the multiagent deep reinforcement learning literature, they help us answer that question: tabular worlds still offer insights to investigating crucial challenges in MAS – "simple worlds remain a fertile ground for further research, especially for problems like shadowed equilibria, non-stationarity or alter-exploration problems and continue to matter for modern deep learning approaches" [1].

In fact, Matignon, Laurent, and Le Fort-Piat [2] investigate coordination challenges in MAS; more specifically, challenges that independent learners (see Section II) must overcome to learn and accomplish coordination tasks. Inspired by a distributed manipulation and discrete actuator arrays [3] (see Section II), the authors provide the *Discrete Smart Surface* benchmark, which we call by DSSb. The benchmark enlightens the investigation of challenges faced by applications that rely on multiple decision-makers influencing each other's decisions, such as in multi-robot control.

A fixed 2D grid of actuators defines the DSSb, and its successful accomplishment requires the coordination between many situated agents to move an object to a goal location (see our toy example in Figure 1, which is detailed in Section III). With the benchmark, the authors [2] sought to investigate many agents coordinating actions in a Markov Game [4] (they used a 9×30 grid of 270 actuators). A trial starts with an object at the initial location (grid's top-center as a default) and ends when the object reaches the terminal state (grid's bottom-center as a default).

We build on their work [2] to detail the DSSb and experimentally test it under various settings. *But why pick the DSSb instead of another benchmark, such as the popular pursuit domain [5]?* We are particularly interested in the DSSb due to its focus on situated agents: it enables us to run many *situated* agents in a task that requires coordination between them. Moreover, situated agents help to highlight, within the MAS literature, the importance of having the environment as a first-order abstraction [6]. Hence, the DSSb not only helps to investigate challenges in MAS but also illustrates the importance of distinguishing between agent and environment – such importance motivated us to detail the DSSb, which is one of our contributions.

Focusing on challenges in MAS, an interesting question to ask is: in what ways can the DSSb help us visualize and get a better understanding of action shadowing? We sought to answer that question as we contextualize and explain our experimental results. In particular, we investigated questions such as:

- How does the agents' success in coordinating actions change as we increase the grid dimensions?
- How do the action pairings change as the agents learn to coordinate actions?
- How do the agents' weights in the decision process of moving the object impact the task's success? And does an agent's location determine its role (or importance) in a task's success?
- What is the role of the penalty areas, and how does that affect the learning opportunities of an agent situated in a penalty area? Thus, regarding *learning the task*, can we say that agents are "luckier" according to their location

in the grid? Or, in other words, does an agent's location determine its learning process/opportunities?
- What does action shadowing look like in the DSSb?

We investigate these questions in Sections III and IV; and our contributions are to:

1) Detail the DSSb while making a clear distinction between agent and environment – following a call from [6], and in the hope that others will reflect on the importance of such a distinction, especially today, with AI systems merging deeply into our realities and the physical/digital boundaries getting blurrier.
2) Show and discuss our experimental results for combining independent learners with the DSSb under various dimensions and agent-weighting systems.

In future work, we will expand our experiments to cover more agent dimensions and weighting systems; we will analyze and contrast them with other settings, such as the pursuit domain [5], the Michigan Intelligent Coordination Experiment (MICE) [7], multi-turn games [8] and other settings, such as [9] and [10].

This work is organized as follows: we introduce our work in Section I, provide background information and related literature in Section II, detail the functioning of the DSSb in Section III, provide observations to support experimental analysis followed by experimental results in Section IV, and conclude in Section V. Finally, the questions provided in the Introduction (Section I) are addressed in Sections III - IV.

II. Background and Related Work

In this section, we provide background information as we visit related literature and contextualize the DSSb, which addresses situated agents, independent learners, identical payoffs, and Markov games. We start by describing the work [3] that inspired the design of DSSb [2].

Distributed Manipulation and Discrete Actuator Arrays. Luntz and Messner [3] build upon their previous model [11] to create a distributed manipulation system that coordinates to induce motion and manipulate larger objects. The authors explore distributed control after considering it would be impractical for a single centralized controller to control thousands of cells. Their system uses actuator arrays of many small stationary elements called cells. If one wished to visualize the system in the context of a macro-scale actuator, one could do it through a fixed 2D grid of motorized wheels. For intuitive purposes, here is a loose example to mentally picture the system: imagine a 2D grid of motorized wheels moving checked bags at an airport. An object lies on supporting cells, and as the object moves, its supporting cells change. Cells that support the object provide a traction force, and their combined action, through coordination, determines the object's motion (*both* translation and rotation). The modeling considers a) the interaction between actuators and the object, b) the object's weight distribution among the support, and c) the system's discrete nature. In the system, each cell communicates with its neighboring cells, and each cell is equipped with binary sensors to detect the presence of an object [3].

Situated Agents. A situated agent has spatial coordinates and interacts with other agents in a hosting environment [12]. In response to a bottleneck, [12] propose an approach to the modeling and simulation of large-scale situated MAS. The observed bottleneck sits within parallel/distributed simulation of situated agents such that the environment represents a substantial shared variable that requires cautious treatment, and regular agent access to environment information can quickly become a bottleneck that diminishes system performance and scalability.

MAS and Environments. Weyns, Vizzari and Holvoet [6] identify issues among the MAS community in defining environments, e.g., "environment" may refer to a) the logical entity in which agents and resources are embedded, or b) the software infrastructure on which it is executed, or even c) the running underlying hardware infrastructure. They illustrate those issues through a situated setting (like the DSSb, in which agents have an explicit position in the environment), as they propose a three-layer model to help distinguish between the environment and the infrastructure on which the MAS is deployed. According to the authors, although there are aspects conceptually apart from the agents themselves and thus should not be assigned or hosted inside agents, the MAS community often considers the environment as infrastructure instead of an important entity at the application level.

Weyns and colleagues [13] aim to push the community to make the environment's logical functionalities explicit, or in other words, to treat concerns of environments as first-class entities. While describing the MAS application layer, the authors identify two classes of concerns: the ones related to the structure of the environment and the ones related to the environment's activity. "Agents and objects of a MAS share a common environment. The agents as well as the objects are dynamically interrelated to each other. It is the role of the environment to define the rules under which these relationships can exist and can evolve. As such the environment acts as a structuring entity for the MAS" [13]. We follow the authors' call as we make a clear distinction between agent and environment in the DSSb, in addition to treating concerns of environments as first-class entities (see Section III).

Agent Coordination. Malone and Crowston [14] provide a definition useful for highlighting aspects of a task that requires coordination: it is "the act of managing interdependencies between activities performed to achieve a goal". Building on that, "agent coordination is the ability to manage the interdependencies of activities between agents while agent cooperation is the process used for an agent to voluntarily enter a relationship with another to achieve a system derived goal" [15].

Tan [16] compares independent learners with cooperative agents (defining *cooperation* in the sense of agents sharing episodes, learned policies, or instantaneous information). Sukhbaatar, Szlam, and Fergus [8] apply fully cooperative tasks to examine CommNet, a neural model that uses continuous communication; the model uses multiple agents that learn to communicate alongside their policy. According to the

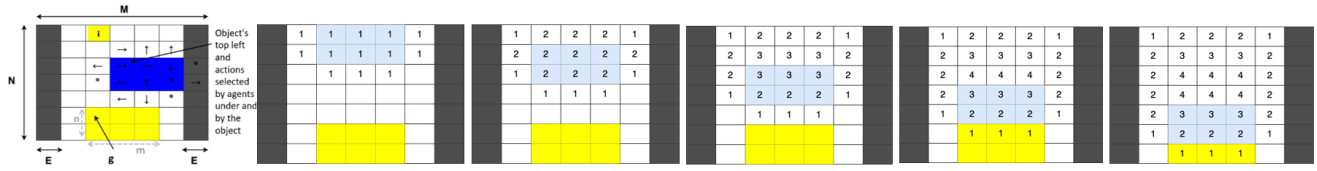

Figure 1. **Left.** Each cell corresponds to a situated agent. Agents task: to move the object from an initial position i into the goal state g (which defines the terminal state). The 3×2 object is color-coded in blue. Arrows inside the grid correspond to actions selected by agents under or by the object (circles for 'stay still'). The leftmost and rightmost columns are penalty areas, which agents also occupy.

authors, their findings show simple but effective strategies for solving the tasks since, in some of the experiments, it is possible to interpret the language used by the agents. Although the DSSb applies non-communicative agents, the work described in [8] is worth mentioning due to one of the tested environments, which is a 14×14 grid. It consists of a 4-way junction: at each time step, a new car enters the grid with a certain probability from each of the four directions; each car occupies a single cell and is randomly assigned to one of three possible routes.

Switching gears to a robot's manipulative repertoire, Stuber and colleagues [17] describe pushing as an essential motion primitive and that, despite a considerable amount of work and models on robot pushing, there would be work to be done on generalizations to novel objects – emphasizing the non-triviality of the problem. For instance, robotic grasping and manipulation would not be trivial even if we considered ideal conditions (described by the authors as structured environments in which an agent has access to a complete model of the environment and ideal sensing abilities). The authors mention the importance of using MAS to move large-scale objects in real-world applications and how control and decision-making are critical issues in those scenarios. For instance, agents must not only coordinate actions, but the point of view changes as a consequence of their actions, bringing yet another challenge: to predict outcomes from pushing given by the action of multiple agents. Finally, Acuña and Schrater [18] provide an interesting study on human decision-making and structure learning in sequential decision tasks.

Coordination and Communication Issues in Robotic Applications. Matignon, Jeanpierre, and Mouaddib [19] study multi-robot exploration under communication breaks constraints. Working under the assumptions of full local observability, limited information sharing between the agents, and breaks in communication, their research addresses global and local coordination of decentralized decision-makers. They develop and apply a method to multi-robot exploration scenarios; according to the authors, experimental results in real-world contexts show that their method is robust to communication breaks and successfully helps to coordinate a team of robots.

Independent Learners and Markov Games. In Robotic tasks that require many agents to coordinate actions to accomplish a task, it is tricky to observe the agents' joint actions [2] (e.g., find and coordinate actions to collect and move an object). To navigate the issue, we can treat the agents as independent (non-communicative) learning agents [20]. In opposition to joint-action learners, independent learners are agents that ignore or cannot observe the actions and reinforcements of other agents in the environment and, therefore, can apply the off-policy temporal difference control algorithm [21] Q-learning [22] in a classic sense and dismiss other agents' existance [20]. However, in the case of distributed settings, the limited information brings about interesting challenges, or "pathologies" [1] for independent learners (which we explore in Section IV).

Boutilier [23] provides an insightful bridge analogy to think of coordination problems when agents have common interests. Consider a team of agents modeled as acting on behalf of a single individual and, therefore, acting to maximize the individual's utility. Suppose a team needs to cross a bridge that can only support one agent at a time; in that case, agents would need to coordinate the ordering of crossing to avoid the bridge collapsing (and destroying the crossing agents along with it). For each agent, the crossing ordering may not be important as long as it crosses it and pursues its goals. Further, Boutilier [23] reflects on scenarios where each agent's abilities are such that it does not really matter what agent pursues which goal (as long as all or most of the goals are pursued). When there is some flexibility in each agent's role, they may end up pursuing the same goal, and lack of coordination can range from delays in accomplishing a task to never really accomplishing it. Such context can be generalized to other team contexts, such as logistics planning [23].

Within robotics applications, Matignon and colleagues [2] are interested in those in which a group of robots can accomplish a task faster than a single robot. The authors, similarly to [23], adopt the terminologies *cooperative robots* and *learning algorithms in fully cooperative MAS* to convey settings in which agents share common interests or the same utility function, *i.e.,* there is a correspondence between each agent's achievement and the group's and therefore, the learning goal is defined as maximizing the common discounted return.

Or, as [23] puts it, fully cooperative MAS in which we assume that it is possible to set a common coordination mechanism and that agents do not have a reason to deliberate strategically. The authors [2] use *Markov Games* instead of *Stochastic Games* to distinguish their settings from *stochastic (non-deterministic) Markov Games*. "Markov games are a superset of Markov decision processes and matrix games, including both multiple agents and multiple states", and all

agents are supposed to observe the complete state s, and the transition and reward functions depend on the joint action of agents [2]. If all agents are fed the same rewards, then the Markov game is called *fully cooperative* by some authors, as "a best-interest action of one agent is also a best-interest action of every agent" [2] – *identical payoff stochastic game* [24] is another possible terminology. We describe the DSSb next.

III. The Discrete Smart Surface Benchmark

In this section, we detail a DSSb environment and the rules that dynamically connect agents and objects. In Figure 1, we illustrate the DSSb: suppose the agents coordinate their actions to move the object to the goal state using the optimum number of steps. Then, **from left to right**, the five images show the object's position step by step within a trial, whereas numbers progressively show how often under/by agents played a role as they moved the object to the goal state. The object reaches the terminal state in the final step, which is not shown here.

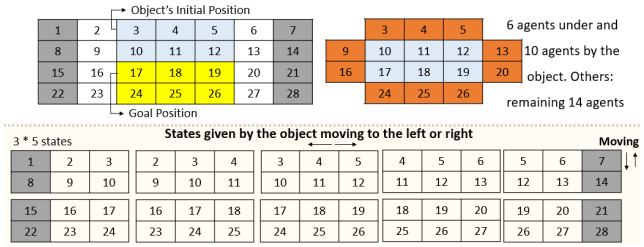

Figure 2. Upper left: a toy example; numbers within cells represent a situated agent. Upper right: a state in which 16 agents' actions play a role in the transition function F; remaining 14 agents are ignored ($w_s = 0$). Lower level: 15 possible states given by the grid and the object's dimensions (intermediary row omitted).

In the DSSb, agents have common interests, and a common coordination mechanism is assumed; the agents' task is to identify a joint sequence of actions that maximize the long-term common discounted return. Let a Discrete Smart Surface $G = \{M, N, E, m, n, S, F, R, A, L\}$, where (see Figure 1 left):

1) M is the width, and N the height of a 2D array of actuators, each driven by an agent. Therefore, there are $M \times N$ actuators – we refer to each actuator as an *agent*.

2) A motion of a 2D object that is placed on the 2D grid's surface is determined by a weighted sum of the agents' actions. The object does not rotate and cannot leave the surface.

3) m is the width, and n is the height of the object ($m < M$ and $n < N$).

4) E is the width of the grid's right and left borders, which are penalty areas. Borders have equal width and height, such as $1 \times N$ each.

5) S is the set of states. Each possible object's position defines a unique state; hence, there are $(M-m+1)(N-n+1)$ states. Initial and terminal states correspond to the object's initial and goal positions (we interchangeably use terminal/goal state).

6) Trials start from placing the object on the initial position i and finish once the object's top left reaches the goal state g.

7) The state transition and reinforcement function depend on the agents' joint actions. F is the transition function that applies a weighted sum of the agents' actions to decide the object's motion/state transition. F categorizes agents into three groups: agents by the object, agents under the object, and remaining agents. Each group has a weight that impacts the transition dynamics: w_b, w_u, w_s, respectively.

8) R is the reinforcement function: agents receive identical payoffs.

9) A is the set of actions available to each agent, and $A = \{\text{left, right, up, down, still}\}$. At each time step, agents select an action from A and simultaneously execute it.

• Finally, L represents the agent structure applied to drive the actuators, such as the agents' implemented reinforcement learning techniques and action-selection mechanism. Although the agents act in an environment that offers affordances [25], [26], agents are distinct from the environment. Thus, within G, we group two classes of concerns: the ones related to the environment's structure and the ones related to its activity.

To conclude, a developer should keep in mind that there are parameters intrinsic to F, R, and L (e.g., weight values, reinforcement values, and learning rates); also, if one wishes to introduce uncertainty to account for actuator errors (e.g., actuators or sensors presenting issues due to unexpected external or internal factors), that can be done by making an agent apply a random action with a small probability [2].

A. Sequential Decision-Making and Trials

In the DSSb, a simulation is a defined number of trials run in sequence, and agents must learn to coordinate actions to move the object to the terminal/goal state. A trial starts with the object at the initial state (even if random) and ends once it reaches the terminal state. A trial lasts for at least the minimum number of steps needed to successfully accomplish the task (object reaching the goal state). Within a trial:

- Agents have access to their own actions and current state (given by the object's top-left position) but *not* to other agents' actions. Agents simultaneously select an action.
- The transition function F uses the agents' actions and corresponding weights w_b, w_u, w_s (which are dynamically matched given an agent's location in relation to the object's position) to set the next state (object's motion).
- The state transition triggers the reinforcement function, which feeds identical payoffs to all agents.
- Agents use their learning mechanisms to learn and select actions in response to the environment.
- A trial ends once the object reaches the goal state (or if the trial meets another pre-defined halt condition corresponding to a task's failure, such as hitting a maximum number of steps).

Does an agent's location interfere with the number of times it will participate in the object's motion? Yes; in fact, given the initial and goal states and weights w_b, w_u, w_s, we can dissect how often an agent's actions are to be considered by the transition function F. Observe Figure 1 left and note

that five steps are needed for the object (3×2 size and color-coded in blue) to reach the goal state. Now, suppose a) an enough number of agents have learned to move the object from the initial state i to the goal state g (both color-coded in yellow), and b) only agents under or by the object matter, in other words, $w_u > 0$, $w_b > 0$, and $w_s = 0$. **The sequential images in Figure 1 show, step by step within a trial, how many times each agent's action plays a role in the state transition (we omit zeros for clarity)**.

Therefore, agents' actions and corresponding locations in the grid are key to contextualizing experimental results for the DSSb. We reflect on that and bridge it to challenges in MAS. Does that mean that an agent's location interferes with its role (or importance) in the task? To answer that question, we built another toy example, shown in Figure 2, upper left.

Consider settings as follows: $M = 7$ and $N = 4$, $m = 3$ and $n = 2$. Therefore, there are 15 possible states, shown in Figure 2, lower part (we omit the third-row states for space reasons). These are the states that follow as the object moves: five columns sideways by three rows in the up/down directions. Within the states, we show the corresponding agents that would be under the object. At each time step, the object's top-left provides s^t, the state s at the time step t. As in the other toy example, we consider $w_b > 0$, $w_u > 0$, and $w_s = 0$. Finally, Figure 2 upper right depicts a state with the following agents under the object: 10-12 and 17-19. Now, see the interplay between state and agent role/importance: in this state, only 16 out of 28 agents play a role (agents by and under the object only). Hence, Figure 2 helps notice that the number of agents by the object varies as the object moves: for this example, 5 agents if at the corners, 7 if at the top or bottom states, 8 sideways, and 10 otherwise – whereas the number of agents under the object remains constant.

IV. Experiments

In this section, we experimentally investigate situated independent learners in a sequential decision-making setting as defined by the DSSb. We test five different grid dimensions: $\{7 \times 7;\ 9 \times 9;\ 11 \times 11;\ 13 \times 13;\ 15 \times 15\}$; therefore, the resulting number of tested agents is, respectively: 49, 81, 121, 169, 225. For clarity, we refer to the "situated independent learners" simply as "agents".

In addition, we ran five weighting settings per grid dimension. To accomplish that, we set different weights for agents by and under the object; the weights are: $\{w_b = 1,\ w_u = 1\}$; and $\{w_b = 1,\ w_u = 5\}$; and $\{w_b = 3,\ w_u = 5\}$; and $\{w_b = 5,\ w_u = 1\}$; and $\{w_b = 5,\ w_u = 3\}$. The higher the weight, the higher the impact of an agent category on deciding the transition dynamics given by F (i.e., on deciding the object's motion/state transition). Note that we kept $w_s = 0$ consistent across all experiments. Other parameters and settings kept consistent across experiments are:

1) A simulation consists of multiple trials run in sequence (simulations are independent of each other).
2) A trial begins with the object at the initial state (grid's top-center) and terminates once the agents manage to move it to the terminal state (grid's bottom-center). Therefore, one can use $N - n$ to obtain the minimum number of steps required to complete a trial.
3) Object size $m = 3$ and $n = 2$.
4) Anytime the object moves to the left or right from the initial state, it enters the penalty area E. To completely avoid a penalty area, the object must remain in the exact middle of the grid throughout an entire trial. If the object remains in the penalty area, it triggers a negative reinforcement at every step it stays there.
5) If multiple actions have the same weighted sum of agents' actions, F chooses randomly among those.
6) As a result of a state transition, all agents receive reinforcement $r = N$ if the object reaches the goal state; a punishment $r = -0.5$ if the object is on a penalty area, and $r = -0.1$ otherwise.
7) Learning phase *vs.* performance phase: we alternate the learning and performance phases such that every *odd* trial is a learning phase whereas every *even* trial is a performance phase. In the latter, there is no learning and agents pick their current best response (randomly between them, if more than one action).
8) Our results depict the performance phase, and they are averaged across 100 simulations for the *p*erformance phase only.
9) We used the Java programming language (Java 17, more specifically), and followed the practices from [27].

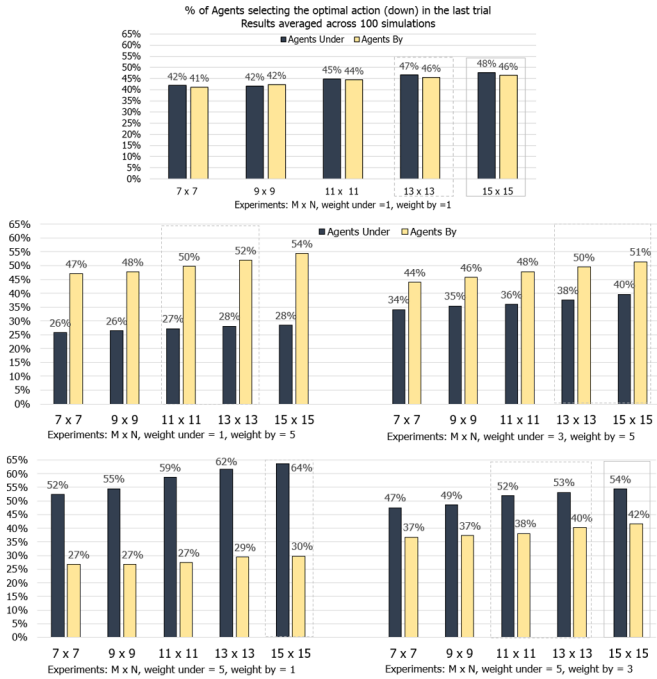

Figure 3. Results for the last trial only: distribution of "down" actions *per* agent category and across experiments. Results averaged across 100 simulations.

Total number of trials. To facilitate the data graphics visualization, we ran a different number of trials per grid dimension (the larger the grid, the higher the number of trials

within a simulation).

Agent Structure. The applied reinforcement learning techniques and related parameters are kept consistent across experiments and are as follows: each agent implements the Q-Learning algorithm [22], [28] with the ϵ-greedy policy with a fixed exploration rate of 10% and a discount rate $\gamma = 0.9$ so that the return objective takes future rewards more strongly [21]. For the learning rate, we use $\alpha = 1/k$, where k is the number of times the state-action pair (s, a) has occurred so far.

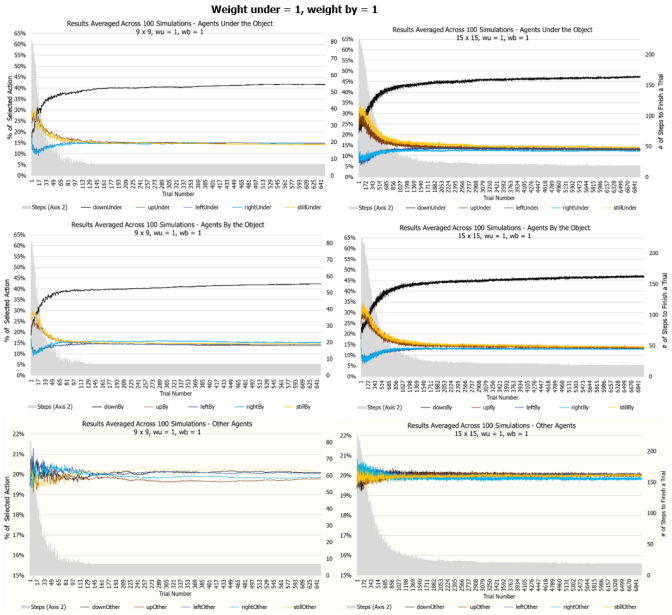

Figure 4. Weights: $w_u = 1$, $w_b = 1$, $w_s = 0$. Over trials, the distribution of actions' percentage *per* agent category, and the number of steps taken to complete a trial. Left: 9×9 grid. Right: 15×15 grid. Results averaged across 100 simulations.

Observations to Support Experimental Analysis:

1) The number of agents under the object is consistent over trials since the object's shape does not change.
2) However, the number of agents by the object varies according to the object's position; see Figure 2. The object's motion implies that the group of agents under and by the object changes over time.
3) Therefore, an agent may participate in the three categories within the same trial: under, by, and remaining. However, once central agents learn to select the down action, agents away from the center of the grid stop participating in F's transition process, as they remain in the "remaining agents" category, and $w_s = 0$.
4) As the agents move the object toward the left or right from its initial center position, it enters the penalty areas. To completely avoid them, the agents must learn to coordinate actions and keep the object in the center columns of the grid throughout an entire trial.
5) A penalty helps agents learn to avoid the left and right borders but also impacts the number of times agents at the borders play a role in the state transition (as agents coordinate actions to avoid those areas).

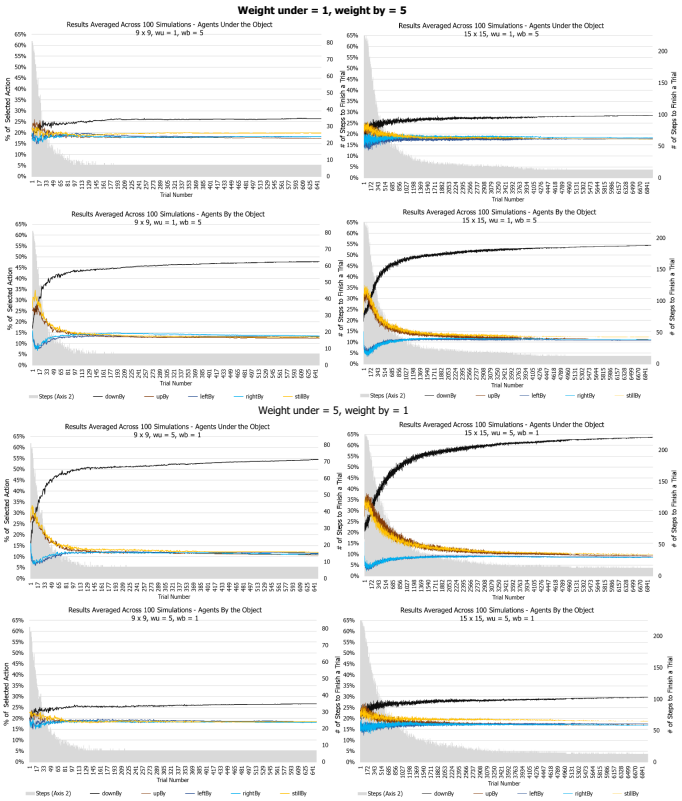

Figure 5. Upper level. Weights: $w_u = 1$, $w_b = 5$. Lower level. Weights: $w_u = 5$, $w_b = 1$. Over trials, the distribution of actions' percentage *per* agent category, and the number of steps taken to complete a trial. Left: 9×9 grid. Right: 15×15 grid. Results averaged across 100 simulations.

A. Results Across Grid Dimensions and Weights

How do different grid dimensions and weights impact agent coordination? Figure 3 shows an overview/comparison across experiments: results depict, within the last recorded trial after convergence, the percentage of agents that selected the *down action*. Down is the optimal action if an enough number of agents continuously select down from the beginning until the end of a trial - which is expected to happen after convergence.

Figure 3 shows one data graphic *per* weight set, in the following order: $\{w_u = 1, w_b = 1\}$; and $\{w_u = 1, w_b = 5\}$; and $\{w_u = 3, w_b = 5\}$; and $\{w_u = 5, w_b = 1\}$; and $\{w_u = 5, w_b = 3\}$. We chose that ordering to facilitate visual comparison across flipped weights. Then, within each data graphic, the results across grid dimensions are split by agent category (if under or by the object).

We added boxes in Figure 3 to highlight experiments in which not all simulations converged to the optimal steps number within a trial (the optimal number is given by $N - n$): a dashed box means that simulations converged to [0, 3) more steps in relation to the optimal, a not-dashed box to [2, 7), and no box to [0, 1). You may ask: "Why are the final number of steps shown as ranges?" Our results are averaged across

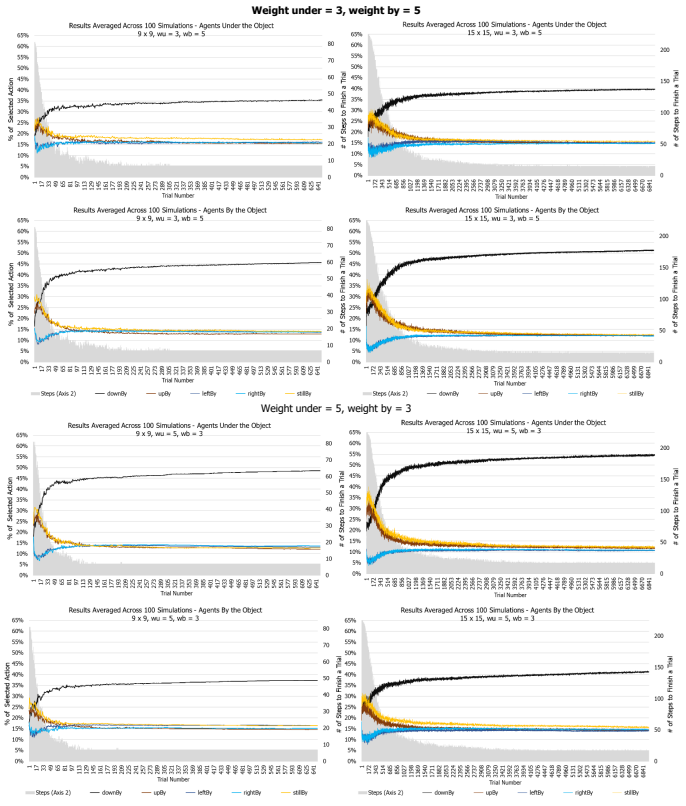

Figure 6. Upper level. Weights: $w_u = 3$, $w_b = 5$. Lower level. Weights: $w_u = 5$, $w_b = 3$. Over trials, the distribution of actions' percentage *per* agent category, and the number of steps taken to complete a trial. Left: 9×9 grid. Right: 15×15 grid. Results averaged across 100 simulations.

simulations and, similar to what the authors [2] observed in their experiments, not all learned policies are stable, and both lack of robustness against exploration and miscoordination penalties interfere with individual policies. As expected, these issues are more visible as we increase the size of the grid; for instance, 15×15, weights $w_u = w_b = 1$, and $w_u = 5$, $w_b = 3$ have the worst results: both converged to [2, 7) more steps than the minimum needed to accomplish the task.

Figure 3's upper level shows that when $w_u = w_b = 1$, agents select the down action with similar percentages. The remaining four graphics show that agent categories with higher weights facilitate their learning to select the optimal action. However, flipping weights lead to a slightly different distribution: observe when agents under the object have higher weight ($w_u = 5$, $w_b = 1$ and $w_u = 5$, $w_b = 3$) and compare it with when they have the lower weight ($w_u = 1$, $w_b = 5$ and $w_u = 3$, $w_b = 5$). If set to higher weights, more agents under the object learn to select the optimal action than would be the case for agents by the object. We suppose that happens due to a combination of the chosen parameters (such as the grid and object's dimensions) and the number of agents under the object (usually smaller than the number of agents by the object, see Figure 2) – therefore, making it easier for the agents under the object to learn to coordinate actions.

B. A Closer Look into the Grids 15×15 *and* 9×9

In this section, we focus on Figures 4, 5, and 6. Here, we investigate our largest grid, 15×15, along with the most successful within the largest ones, 9×9 (by most successful, we mean that the object gets to the goal state within the minimum number of steps for all five sets of examined weights). For easy comparison, we always depict the results side by side: 9×9 at the left and 15×15 at the right. Each figure focuses on a set of weights: Figure 4 shows $w_u = w_b = 1$. Then, Figure 5 shows $w_u = 1, w_b = 5$ at the top, followed by $w_u = 5, w_b = 1$. Finally, Figure 6 shows $w_u = 3, w_b = 5$ at the top, followed by $w_u = 5, w_b = 3$. Here, we examine the action distribution within trials and across simulations to help check if/when the learned policies are close to the best policy. Therefore, Figures 4-6 depict the action distribution (down, up, left, right, still) over trials as well as the total number of steps within trials (we rounded the steps to the nearest whole number to facilitate visualization). Results are split by agent category: depicting first the agents *under the object*, followed by the agents *by the object*.

Given that agents always select an action per step within a trial, percentages are averaged within a trial and across simulations. To facilitate visualization, different grid dimensions have a distinct number of trials within each of the 100 simulations – this is why 15×15 shows a higher number of trials than 9×9 at the x-axis.

In Figure 4 only, we show the results for *the other agents* – we use a different background color to highlight a different scale, which was chosen to facilitate observation. However, we omit *the other agents* from Figures 5-6 since agents who are neither under nor by the object do not influence F's transition process (since $w_s = 0$). Therefore, their action distribution remains within $1\backslash(\#\text{of actions})$, or $1 \setminus 5$, which is 20% *per* action, as Figure 4 shows. Due to the zero weight, actions from that agent category do not matter; consequently, they receive reinforcements in response to the actions of agents either under or by the object.

Overall, our results show that, as agents learn to select the down action, the number of steps used to successfully finish a trial diminishes: if not to the optimal number (given by $N-n$), to a number close to that (see Section IV-A). What is the main difference between Figures 4-6? Figure 4 shows that, in both grid dimensions, results for agents under and by the object are very similar – as one would expect given the identical weights.

Figure 5 shows a two-fold result: when agents by the object have higher weight ($w_u = 1$, $w_b = 5$), their percentages for the down action are higher; also, as one would expect, the opposite happens if we flip the weights ($w_u = 5$, $w_b = 1$). Finally, in Figure 6, although agents with higher weight still show higher percentages for the down action, this setting shows a higher overall selection for that action (under/by agents with more than 35% to that action).

The DSSb is resourceful in investigating challenges faced by a large number of agents to coordinate actions in a Markov game; as the authors [2] point out, the DSSb "brings together

action shadowing induced by penalties, Pareto-selection as there are several possibilities to reach the goal state". In fact, Figures 4-6 provide interesting examples of action shadowing. Still, before getting into action shadowing, let us visit the non-stationarity problem. As mentioned in the background section, independent learners ignore each others' presence and, therefore, can treat other agents as part of the environment. However, from each agent's perspective, the environment no longer appears Markovian and stationary since the "transition probabilities associated with the action of a single agent from one state to another are not stationary and change over time as the action choices of the other agents change. These choices are probably influenced by the past history of play, and so the history of play influences the future transition probabilities when revisiting a state" [2], [29].

Now, suppose agent x selected an optimal action, while most of the other agents selected actions that move the object to a penalty area. Thus, the agent's x action will be shadowed by a transition to a penalty area. Conversely, what happens if agent x selects an action towards a penalty area while most agents select actions away from it? How can agent x learn not to pick poor action choices? In Figures 4-6, note how the actions *l*eft and *r*ight drop initially as agents learn to avoid the penalty areas but then increase a bit. Once an enough number of agents coordinate actions and learn to avoid the penalty areas, it does not matter what agent x selects. There is a point at which just enough agents learn the proper state/action pairs, impacting other agents' learning. To put it in simple terms, it is almost like a portion of agents at each step carries the entire team of agents one step closer to the goal. By looking into Figure 3, one may assume that higher percentages of agents always selecting the optimal action lead to better results overall. However, that is not completely true. Of course, enough agents need to select it so that the object moves to the goal state; however, once that happens, it does not matter how many more agents select that action. For instance, look at Figure 5, 9×9, weights $w_u = 5$, $w_b = 1$; the *d*own action keeps increasing, but trials are such that the optimum number of steps has already been reached. From an agent perspective, it should matter to learn the optimal state/action pairs; however, from a task perspective, successfully accomplishing it is likely to be the overall goal.

V. Conclusion

In this paper, we first detail the Discrete Smart Surface benchmark while making a clear distinction between agent and environment. Then, we address the research questions listed in the Introduction and investigate the benchmark using various grid dimensions and agent weighting values – for example, we show how the distribution of actions per agent category varies over trials as agents learn to coordinate actions. Finally, the situated agents from DSSb offer important lessons for challenges in MAS; for instance, our experimental results contextualize and exemplify action shadowing, which we hope will be insightful for others investigating challenges in MAS. In future work, it would be interesting to study other grid dimensions, object shapes, and sizes, including different weighting systems (to contrast the roles of agents under *versus* by the object). The study would help to investigate questions, such as: if someone wanted to exploit the system and insert a number of agents with a fixed strategy only to deteriorate individual policies, what should be the minimum number of agents, and what grid cells should they occupy so as to maximize their effect?

Acknowledgments

This work would not have been possible without the support from Grinnell College's Harris Faculty Fellowship and the Mentored Advanced Project (MAP) program.

* Authors' equal contributions.

References

[1] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: a survey," *Artificial Intelligence Review*, pp. 1–49, 2022.

[2] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, "Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems," *The Knowledge Engineering Review*, vol. 27, no. 1, pp. 1–31, 2012.

[3] J. E. Luntz, W. Messner, and H. Choset, "Distributed manipulation using discrete actuator arrays," *The International Journal of Robotics Research*, vol. 20, no. 7, pp. 553–583, 2001.

[4] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, ser. ICML'94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, p. 157–163.

[5] M. Benda, V. Jagannathan, and R. Dodhiawala, "On optimal cooperation of knowledge sources-an experimental investigation," *Boeing Advanced Technology Center, Boeing Computing Services, Seattle, Washington, Tech. Rep. BCS-G2010-280*, 1986.

[6] D. Weyns, G. Vizzari, and T. Holvoet, "Environments for situated multi-agent systems: Beyond infrastructure," in *Environments for Multi-Agent Systems II: Second International Workshop, E4MAS 2005, Utrecht, The Netherlands, July 25, 2005, Selected Revised and Invited Papers 2*. Springer, 2006, pp. 1–17.

[7] E. H. Durfee and T. A. Montgomery, "Mice: A flexible testbed for intelligent coordination experiments," in *Proceedings of the 1989 Distributed Artificial Intelligence Workshop*, 1989, pp. 25–40.

[8] S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning multiagent communication with backpropagation," *30th Conference on Neural Information Processing Systems (NIPS)*, vol. 29, pp. 2252–2260, 2016.

[9] M. Hausknecht, P. Mupparaju, S. Subramanian, S. Kalyanakrishnan, and P. Stone, "Half field offense: An environment for multiagent learning and ad hoc teamwork," in *AAMAS Adaptive Learning Agents (ALA) Workshop*, Singapore, May 2016.

[10] M. Kim, J. Oh, Y. Lee, J. Kim, S. Kim, S. Chong, and S.-Y. Yun, "The starcraft multi-agent challenges+: Learning of multi-stage tasks and environmental factors without precise reward functions," *arXiv preprint arXiv:2207.02007*, 2022.

[11] J. E. Luntz, W. Messner, and H. Choset, "Parcel manipulation and dynamics with a distributed actuator array: the virtual vehicle," in *Proceedings of International Conference on Robotics and Automation*, vol. 2, 1997, pp. 1541–1546 vol.2.

[12] F. Cicirelli, A. Giordano, and L. Nigro, "Efficient environment management for distributed simulation of large-scale situated multi-agent systems," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 3, pp. 610–632, 2015.

[13] D. Weyns, H. Van Dyke Parunak, F. Michel, T. Holvoet, and J. Ferber, "Environments for multiagent systems state-of-the-art and research challenges," in *Environments for Multi-Agent Systems: First International Workshop, E4MAS 2004, LNAI 3374*. Springer, 2005, pp. 1–47.

[14] T. W. Malone and K. Crowston, "What is coordination theory and how can it help design cooperative work systems?" in *Proceedings of the 1990 ACM conference on Computer-supported cooperative work*, 1990, pp. 357–370.
[15] A. Consoli, J. Tweedale, and L. Jain, "The link between agent coordination and cooperation," in *Intelligent Information Processing III: IFIP TC12 International Conference on Intelligent Information Processing (IIP 2006), September 20–23, Adelaide, Australia 3*. Springer, 2007, pp. 11–19.
[16] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the tenth international conference on machine learning*, 1993, pp. 330–337.
[17] J. Stüber, C. Zito, and R. Stolkin, "Let's push things forward: A survey on robot pushing," *Frontiers in Robotics and AI*, p. 8, 2020.
[18] D. E. Acuña and P. Schrater, "Structure learning in human sequential decision-making," *PLOS Computational Biology*, vol. 6, no. 12, pp. 1–12, 2010.
[19] L. Matignon, L. Jeanpierre, and A.-I. Mouaddib, "Coordinated multi-robot exploration under communication constraints using decentralized markov decision processes," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 26, no. 1, 2012, pp. 2017–2023.
[20] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," *AAAI/IAAI*, vol. 1998, no. 746-752, p. 2, 1998.
[21] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction (2nd ed.)*. MIT press, 2018 [1998].
[22] C. J. Watkins, "Learning from delayed rewards," 1989.
[23] C. Boutilier, "Sequential optimality and coordination in multiagent systems," in *IJCAI'99: Proceedings of the 16th international joint conference on Artificial intelligence*, vol. 1, 1999, pp. 478–485.
[24] L. Peshkin, K.-E. Kim, N. Meuleau, and L. P. Kaelbling, "Learning to cooperate via policy search," *arXiv preprint cs/0105032*, 2001.
[25] J. J. Gibson, *The senses considered as perceptual systems.* Bloomsbury Academic, 1966.
[26] ——, *The Ecological Approach to Visual Perception: Classic Edition*. Psychology Press, 1979.
[27] C. S. Horstmann, *Core Java: Fundamentals, Volume 1*. Pearson Education, 2022.
[28] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.
[29] M. Bowling and M. Veloso, *An analysis of stochastic game theory for multiagent reinforcement learning*. Tech. rep., Computer Science Department, Carnegie Mellon University, 2000.