Modular Reinforcement Learning For Cooperative Swarms

Erel Shtossel
Bar Ilan University
Ramat Gan, Israel

Gal A. Kaminka
Bar Ilan University
Ramat Gan, Israel

ABSTRACT

A *cooperative robot swarm* is a collective of computationally-limited robots that share a common goal. Each robot can only interact with a small subset of its peers, without knowing how this affects the collective utility. Recent advances in distributed multi-agent reinforcement learning have demonstrated that it is possible for robots to learn how to interact effectively with others, in a manner that is aligned with the common goal, despite each robot learning independently of others. However, this requires each robot to represent a potentially combinatorial number of interaction states, challenging the memory capabilities of the robots. This paper proposes an alternative approach for representing *spatial interaction states* for multi-robot reinforcement learning in swarms. A modular (decomposed) representation is used, where each feature of the state is handled by a separate learning procedure, and the results aggregated. We demonstrate the efficacy of the approach in numerous experiments with simulated robot swarms carrying out *foraging*.

KEYWORDS

Reinforcement Learning, R learning

ACM Reference Format:
Erel Shtossel and Gal A. Kaminka. 2026. Modular Reinforcement Learning For Cooperative Swarms. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 9 pages.

1 INTRODUCTION

A *cooperative robot swarm* is a collective of robots that work towards a common goal, despite each being limited in the range of its perception and the reach of its actions. These limitations allow only *local* interactions for each robot, with a small subset of its peers [2, 3, 5, 7, 13].

Cooperative swarms pose a significant challenge to swarm robot developers. The maximization of the collective utility by the swarm's members, in aggregate, is encumbered by each individual robot's lack of knowledge about its peers or the state of the collective as a whole. Often, this is addressed by carefully designing the individual decision-making procedures, such that local decisions are forced to *align* with the collective goal [35, 39, 43, 51]. Alternatively, multi-agent reinforcement learning (*MARL*) can be used to learn individual action-selection policies [15, 19, 20, 22, 50], in particular when matched with appropriate aligned rewards [6, 16].

A significant challenge to the use of MARL in swarm robots is raised by the limited computational resources of the robots. Limited memory prohibits explicit representation of the large combinatorial number of states (a *state explosion* problem). Limited computation capabilities prohibit the use of deep neural networks as function approximations to generalize and represent states implicitly. Finally, the restrictions on communication range and bandwidth prohibit sharing of information at the swarm level. Thus robots act, for the most part, as *independent learners*, unaware of the decisions of others, nor the individual impact (reward) of these decisions. This is a particularly challenging setting for MARL [19, 28, 30].

While these challenges are known in general [23, 29, 40, 47], the limited computational resources of swarm robots make most solution approaches impractical or even irrelevant. For perspective, research swarm robots are commonly built around microcontrollers running at a few hundred MHz cycles, using total RAM well under 512KB (the capabilities of popular off-the-shelf robots are presented in Table 1).

Robot	Controller	Clock Speed	RAM
e-puck2 [12]	STM32F4	168 MHz	192 KB
Elisa-3 [11]	ATmega2560	8 MHz	8 KB
Pololu3Pi [34]	ATmega328	20 MHz	2 KB
Krembot [32]	Particle Photon (ARM Cortex M3)	120 MHz	128 KB
Arduino Robot [1]	Two ATmega32u4	16 MHz	2.5 KB each
Kilobots [38]	ATmega328	8 MHz	32 KB

Table 1: Specifications of popular off-the-shelf robots.

Multi-robot swarm foraging, a canonical cooperative swarm task, is a good example of how the severe computational limitations of swarm robots pose a challenge to reinforcement learning approaches. In foraging, multiple robots search a shared arena for items that they pick up and bring to their base. The performance of the swarm is measured by the sum total of collected items. Robots have limited sensor and communication ranges, and cannot communicate with all other robots; in some cases, they cannot communicate at all. To navigate, each robot typically represents its spatial interaction state, composed of the range and bearing to all robots and obstacles in its sensed surroundings. As long as only the current state is maintained, this is not a challenge. However, if a naive reinforcement learning mechanism needs to learn how to act in each state, then all possible spatial states must be represented. The number of states grows combinatorially large with the number of entities and the resolution of measurements. A latent representation using deep neural networks is not feasible given the limited computation available (some controllers do not support floating-point operations in hardware).

To address the state explosion problem, this paper presents a modular (decomposed) state representation, where each state feature (e.g., the position of each neighbour) is represented separately and handled by a separate learning process. In this way, rather than a single learner addressing a k-factor state representation ($O(2^k)$ states to be stored), there are k simple learners, requiring storing a

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus.

total of $O(k)$ states. The learners' recommendations are aggregated via a fixed action-fusion mechanism.

We describe the modular representation in detail and evaluate its use in a comprehensive set of experiments with simulated robots in a cooperative foraging task. The experiments show that the modular representation works on par with a full state learning algorithm, despite its modest memory and computational requirements.

2 BACKGROUND

Research on robot swarms focuses on the collective behaviour of decentralized, self-organized systems composed of numerous computationally-limited robots. The research area is very broad, covering many different tasks and approaches. We refer the reader to comprehensive surveys of the area [2, 3, 5, 7, 13].

We focus on **cooperative swarm foraging**, an operation where a group of swarm robots is tasked with continually searching for objects of interest (*items*), and when found, transporting them to collection points (*bases*). This is a canonical swarm task with many studied variants: see [25, 35, 42, 45, 49] for comprehensive surveys.

This paper focuses on using *multi-agent reinforcement learning* (MARL) [4, 9, 15, 19, 20, 22, 48, 50] for swarm foraging. MARL is particularly challenging in swarm settings, because each agent is an *independent learner* [18, 26–28, 30, 31], unable to get feedback on the effects of its individual action on its nearby peers, nor on the swarm as a whole. Nevertheless, a recent line of work has proposed methods for *aligning* the individual rewards received by each robot, such that their maximization by the individual necessarily optimizes the swarm goals. In particular, *difference rewards* [6], which computes the marginal contribution of each robot to the swarm, offers a promising approach to such alignment. By converting robot collision overheads to internal measurement of time, it is possible for swarm robots to approximate the difference reward, in a completely distributed manner, without relying on any communication or public signals [8, 16, 17]. This is the approach we take here.

State Representation. When reinforcement learning is applied to swarm robots, a critical challenge emerges in choosing the state representation to match the limited memory and computation resources available. The problem of *state-explosion*—the combinatorial growth in the number of states as necessary state features are represented—is, of course, well known. It can be tackled in several ways: state *abstraction* [23, 29] or state *decomposition* [29, 40, 47].

State *abstraction* removes state features or replaces them with more general features that summarize important state attributes, while reducing the number of features. Some attempt to learn abstractions [21] or dynamically choose between them [44]. Fully abstracting the state leads to a state representation that includes a single state (sometimes called *stateless*), reducing the stateful reinforcement learning problem to a multiarm bandit. This can be effective [8, 18], but tends to be brittle [8].

Often, domain experts design state-abstractions manually. For instance, and specifically for the use of learning in foraging, Douchan et al. [8] abstracted collision states in foraging by measuring the local density (i.e., the number of neighbours). The loss of detail can be detrimental to the learning task. For instance, Fig. 1 shows two states from the perspective of the marked focal robot. These differ in their *geometrical detail*, but share the same *local density*. The safe actions in the two states are different, but relying on the density information as proposed in [8] would make the two states appear identical.

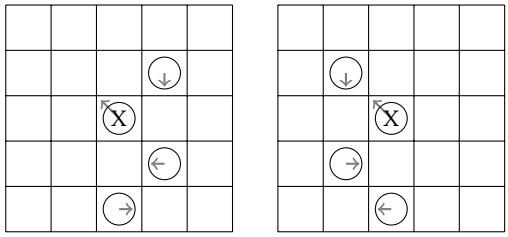

Figure 1: Two states that have the same density, and mirrored geometry. The same action is safe in one, dangerous in the other.

In contrast to abstraction, state *decomposition* breaks the state into smaller sub-states that can be input into the same single RL algorithm to return an action per sub-state [24, 29]. The actions from different sub-states are then aggregated into a single chosen action for the agent. This results in a state representation that grows linearly. At the extreme, a *modular representation*, whereby each different state feature is addressed by separate learning processes, reduces the number of states to be linear in the number of features: for k state features, we utilize k processes, each handling n possible states of the specific feature assigned to it [29].

In this paper, we develop and evaluate a modular state representation for spatial states. Spatial states are very commonly used in robots (sometimes they are known as local obstacle maps, or local occupancy grids), as they form the backbone for navigation in the presence of objects and other robots. A naive (fully factored) state representation of such states tends to explode quickly. For instance, in Fig. 1, the focal robot is positioned in the center of a 5-by-5 discrete sensing grid. Simply marking 0 or 1 to note the presence of a robot or object in the 24 possible locations around it, yields $2^{24} = 16777216$ states. In contrast, a modular representation for the same type would have $2 \cdot 24 = 48$ states in total, divided among 24 learning processes.

3 PRELIMINARIES

We begin by describing the foraging swarm task and draw connections to Markov games so as to pose the individual robot's action-select mechanism as a reinforcement learner, engaging in a highly-distributed multi-agent reinforcement learning process.

Learning Opportunities in Swarm Foraging. From the perspective of a single swarm robot, the foraging task can be divided into several distinct operational states. A straightforward design separates *item-related operations* (searching for items, picking them up, navigating back to base, dropping the items) from *social operations* (monitoring for potential collisions, selecting collision-avoidance and navigation actions, managing communications if available).

There are opportunities for learning—and in particular for reinforcement learning—in many of these operational states. However, we focus on a specific operation: responding to a collision if one is imminent, i.e., *collision-avoidance*.

Collision-avoidance involves significant use of spatial state representation. In non-learning collision-avoidance procedures, the current local state is used as the backbone to any computation of a response, whether myopic or predictive. However, as such procedures do not learn, they only use a single represented state at a time (the current state), and so do not reach the memory limits of simple robots. Such spatial state maps include information about the range and bearing of every robot within the sensing radius; they can sometimes include information about the heading of each such robot, or other information of value. Learning-based methods, however, must store multiple states to respond appropriately to each.

From the point of view of a collision-avoidance learning method, the action-selection point occurs with every collision, where a collision-avoidance action is selected. The reward is received later, as an evaluation of the success of the action in resolving the collision state. In other words, the collision-avoidance learning process discussed in this paper experiences foraging as series of collision-avoidance action-selection opportunities: every impending collision triggers the learner to select an action, and the reward is received upon evaluation—which may be immediate or may be given only on reaching the next collision state (as is commonly modeled in reinforcement learning, the reward $R(s, a, s')$ is given for an action a selected in state s, and reaching the new state s').

Foraging as a Markov Team Game. We adapt the perspective of [16] where the foraging task is modeled as an infinite-horizon, fully-cooperative Markov game (i.e., a Markov Team Game with infinite horizon), where all N members of the swarm interact when colliding. The game is composed of an endless (unknown termination) sequence of *stage* games, where each stage is defined by a collision event as described above. For brevity, we describe below only the necessary formal components needed to define the reinforcement learning problem and methods stemming from this model. We refer the reader to [16] for details.

The collective reward of the swarm is modeled as follows, as the undiscounted *average collective rewards* of a joint policy π:

$$\mathcal{R}(\pi) := \lim_{K\to\infty} \frac{1}{K} \sum_{k=1}^{K} \sum_{s_k \in S} \left[R_k(\pi) \cdot D_k(\pi) \right] \tag{1}$$

Here, K is the number of stage games played (i.e., the horizon tending towards infinity), $R_k(\pi)$ is shorthand notation for $R_k(s_{k-1}, \pi(s_{k-1}), s_k)$, the *joint* reward received for taking an action according to the policy, in stage s_{k-1} resulting in reaching state s_k. Similarly, $D_k(\pi)$ is shorthand for the transition probability function $D_k(s_{k-1}, \pi(s_{k-1}), s_k)$, i.e., the probability of reaching s_k given the application of the policy. Note that all terms here are *joint*: joint states s_{k-1}, s_k (the combination of all agents' states), a joint action a (the combination of all individual actions taken by the agents), and so forth.

Fully-cooperative (team) games have the property that the payoff of any given joint action (the combined individual action of all agents) is *shared* by all, in the sense that all agents get the same reward. In this purely theoretical perspective, if all the agents know $\mathcal{R}$, the collective reward, they can directly measure the effects of the individual actions. Under these settings, it is straightforward to show (i) that their individual self-interested selections will lead to a Nash equilibrium (i.e., the strategy guarantees stability), and (ii) that the Nash equilibrium also maximize $\mathcal{R}$ (i.e., it guarantees collective reward optimality). Allowing the agents to be rational and self-interested in maximizing their rewards will necessarily maximize the collective rewards, as they are one and the same.

Here, the rewards of the collective are said to be *aligned* with those of the individual [6, 46]. In this case, this simply is a result of all agents sharing the rewards, so alignment is trivial. Using reinforcement learning, the agents may evaluate different strategies π by the collective rewards $\mathcal{R}(\pi)$, until converging to the optimal strategy π^*.

Individual, distributed learning. The model (Eq. 1) is a purely theoretical abstraction of actual swarm tasks in general, and of swarm foraging in particular. In reality, no instantaneous reward can be shared; The collective reward $\mathcal{R}$ is unknown to the robots, who only know their own actions, not those of their peers. This is the challenging case of *independent learners* [28]. Thus, instead, the N robots are faced with the collective reward definition based on their individual rewards r^i, where $1 \le i \le N$ denotes an agent.

$$\mathcal{R}(\pi) := \lim_{K\to\infty} \frac{1}{K} \sum_{k=1}^{K} \sum_{i=1}^{N} \sum_{s_k \in S} \left[r_k^i(\pi^i) \cdot d_k^i(\pi^i) \right] \tag{2}$$

Individual rewards. Through the years, different individual rewards have been proposed for the use of the functions $r^i(\cdot)$. A promising approach was introduced in a series of papers [8, 16–18], which seeks to translate the unknowable utility (e.g., number of items collected) into a measure accessible to individual robots, by assuming that the length of time in which robots were active in *item-related tasks* (as discussed above) provide gains for the swarm, while *social tasks*, in particular handling collisions, is detrimental and should be avoided.

For each collision stage k, the robot selects an individual collision-avoidance action a_k. It then measures the duration of the interval until the collision is resolved, and the duration of the subsequent interval in which the robot was free to engage in productive foraging activities (searching for items, collecting them, etc.). The duration of the collision-avoidance interval for agent $i \in N$ is denoted $C_k^i(\pi^i) = C^i(s_{t_{k-1}}, \pi^i(s_{t_{k-1}}), s_{t_k})$, and the duration of the subsequent productive foraging interval is denoted $\mathcal{P}_k^i(\pi^i) = \mathcal{P}^i(s_{t_{k-1}}, \pi^i(s_{t_{k-1}}), s_{t_k})$. Their sum is the total duration of the stage $\left[C_k^i(\pi^i) + \mathcal{P}_k^i(\pi^i) \right]$.

Various reward functions based on this approach have been proposed, including individual self-interest rewards like stage overhead

$$EI(\cdot) := \frac{C_k^i(\pi^i)}{\left[C_k^i(\pi^i) + \mathcal{P}_k^i(\pi^i) \right]},$$

introduced in [18], or its simpler variant, self-interest reward

$$\Omega := \mathcal{P}_k^i(\pi^i) - C_k^i(\pi^i), \tag{3}$$

which we will utilize in some of the experiments. These self-interested rewards are generally not aligned with the swarm collective. Their difference-reward [6] variants do carry such a guarantee: *AEI* is the aligned version of *EI*, presented in [8, 17]. Δ, defined in [16] is what we use in this paper.

$$\Delta_k^i(\pi) := \beta^i \mathcal{P}_k^i(\pi^i) - \alpha^i C_k^i(\pi^i) - (n-1)(\alpha^i + \beta^i)\overline{C_k^i}(\pi^i), \quad (4)$$

where α, β are individual constants defining the rate of productivity loss in collisions, and the rate of productivity gain when foraging, resp. We abuse the notation of N slightly, using it here to denote the set of agents, rather than their number, so $i, j \in N$ denotes agents.

4 MODULAR SPATIAL STATE REPRESENTATION

We propose to decompose the full spatial state of the robot by its features, creating a modular representation whereby each state feature is handled by an independent learning process (described in Section 4.1), and the recommendations of all the learners are aggregated by a fixed procedure (nicknamed, the *council*), which makes a decision on the final action to be taken (described in Section 4.2).

4.1 A Single Learner per Spatial Feature

We divide up the spatial state features by direction. Specifically, for a set of sensors facing a particular direction, we assign a separate learning process. The identification of a particular process with a fixed direction means that the direction feature is implicit in the process and does not need to be explicitly represented.

The various states of each process are composed of the values the sensor(s) can take (jointly). Thus for example a range sensor facing in a particular angle relative to the robot heading (e.g., 45 degrees to the left of the forward direction) is going to take on values related to range: binary values (0/1) can be used to simply indicate the presence of a neighbouring robot within collision radius, while a greater range can indicate other measurements (up to the distance measured).

The learning process must select an action in response to the sensed state. These actions are selected from the action space of *the entire robot*, and are not restricted to actions associated with the direction represented by the learning process. This freedom enables sensors to recommend actions that drive the robot not only in the opposite direction, but also in other approximate forward directions.

We note a subtle but critical caveat with respect to the choice of action-spaces: the actions must be *composable*, as the action for the robot will be decided by a fusion mechanism, fusing together the actions preferred by the modular learning processes, to generate a single action used by the robot. In particular, this presents a departure from the previous work discussed above [8, 16–18, 37], all of which used *collision-avoidance algorithms* as the robot's actions. These serve as macro-actions, but cannot be composed and merged arbitrarily, and perhaps even not at all. In the experiments, we will evaluate the use of algorithmic actions vs composable actions (direction vectors, constant speed).

Given the state-space and action-space as defined above, a reinforcement learning algorithm may now be used to drive the learning process. Its task is to learn a policy mapping the sub-states (values of the associated spatial feature), with recommended actions for the robot. It is important that all the learning processes are triggered together whenever the robot needs to select an action. It can be tempting to think of modular learning as an asynchronous process (no collision in this direction? No problem!), but this is fundamentally incorrect. Breaking down the state into modular features means that the learning occurs in a synchronous manner: all processes are engaged in selecting actions. A corollary of this is that the reward received by selecting the fused action is shared among all the independent learners. There is no attempt at credit assignment.

The implementation used in the experiments serves to ground and clarify the presentation above. The experiments utilize cylindrical robots with 8 range sensors uniformly distributed around their circumference. We therefore have 8 sensing directions, and thus 8 spatial features to model. Each is assigned an identical learning process. We chose to use binary range values, so there are only two states, corresponding to the detected presence of a neighbour within collision distance (*collision*, or lack thereof (*clear*).

The action-space used in the experiments (termed *vectorial*) is made of direction vectors, in the 8 directions used by the sensors. This is for simplicity—there is no immediate connection between the state-space and the action-space, other than the restriction on using composable actions. The robot speed is fixed, and the actions choose direction.

The learning process is triggered if any sensor detects a neighbour within the collision radius. Then *all* modular learning processes are triggered. Every process responds independently as follows: if the state is *clear*, it responds with a fixed action (recommending its own direction). Otherwise, it uses a multi-armed bandit learning algorithm (Continuous UCB-1 [14]) to select one of the eight direction actions. The reward used in most experiments is the difference reward (Δ, Eq. 4), as it is aligned, and has been demonstrated successfully in non-modular learning for foraging.

4.2 The Council

Each of the independent learners selects an action, treated as a preference for the action to be taken by the robot. The aggregation procedure, nicknamed *council*, is used to fuse the preferences to generate compromises.

The learners' preferred vectors disagree (e.g., half the learners prefer one direction, the other half its opposite) or simply offer no clear compromise (e.g., each of the sensors recommends a different direction, so no direction is preferred by more than one learner). This is reminiscent of behavior-fusion control in robotics [36].

We use the following fixed procedure (Alg. 1 below) for generating a selected action that the robot will execute. It accepts the preferred action of each process A_i, and uses it to give a high preference value to the direction indicated. It also gives a (lower) preferred value, based on a Gaussian distribution, to neighbouring directions, to promote satisfying actions [36]. Then the preferences are accumulated, and the accumulated distribution is normalized to generate a discrete probability distribution over the actions. Finally, an action is selected by sampling the probability distribution. The

council process then monitors the execution of the agent, and passes the reward to all independent learners—with no credit assignment.

Input: Direction angles $D \in \{0, 45, 90, 135, 180, 225, 270, 315\}$, standard deviation $\sigma = 2$
Output: Normalized probability vector $P \in \mathbb{R}^8$

define $A \leftarrow \{0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°\}$
define $P \leftarrow \{0, 0, \ldots, 0\}$
foreach $d \in D$ **do**
  for $i \leftarrow 1$ **to** 8 **do**
    diff $\leftarrow |d - A_i|$
    if *diff* > 180 **then**
      diff $\leftarrow 360 -$ diff
    end
    $P[i] \leftarrow P[i] + \exp\left(-\frac{1}{2}\left(\frac{\text{diff}}{\sigma}\right)^2\right)$
  end
end
Normalize P such that $\sum_{i=1}^{8} P[i] = 1$
return P

Algorithm 1: Directional Probability Distribution via Double Gaussian

5 EXPERIMENTS

We evaluate the modular representation in high-fidelity physical simulation, rather than physical robots. This is to allow long training times as needed, and to be able to contrast with algorithms whose computational requirements (run-time and memory) may not be suitable for physical robot resources.

We first describe the experiment settings in Section 5.1. In Section 5.2 we report on the main set of results, contrasting the performance of foraging in policies learned with the modular representation, with algorithms using full state representations, and fixed navigation. We then present results from evaluating the robustness of the performance to reward changes (Section 5.3) and show that the move to the simpler, vectorial action space, necessitated by the modular representation, does not, in general, hurt performance (Section 5.4).

5.1 Experiment setup

We use the ARGoS3 robot swarm simulator [33] to simulate Krembot robots. These are prototypical research-grade swarm robots. They are approximately cylindrical: the diameter is 6.5cm, and the height is 10.6cm. They have 8 visual and range sensors spread uniformly around the robot's circumference, allowing the distinction of color and distance in 8 directions.

We have constructed three simulated environments of 1.5 square meters with a home base of 0.3-meter diameter and randomly scattered 0.1-meter diameter pucks. When a robot picks up a puck, a new puck will spawn around the arena in a uniform distribution.

- **Arena 1**: The home base is in the middle of the arena
- **Arena 2**: The home base is in the corner of the arena
- **Arena 3**: There are two home bases, close to opposite corners.

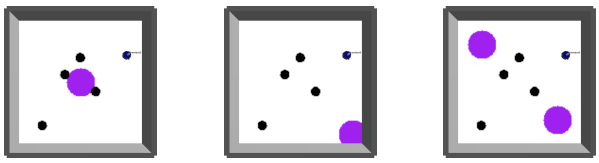

Figure 2: Arenas 1 (left), 2 (middle), and 3 (right). Purple areas show the home base, the black dots are the pucks, and the blue dot is a Krembot for size reference.

The number of robots was varied (4–36 in jumps of 4). Each test was run with twenty different seeds. These seeds determine the random placement of items in the arena and the initial random positions of the robots.

During the foraging, every robot followed the same basic algorithm: When no robots are detected in the *collision radius* of four centimetres around the robot, the robot wanders (random walk) in search of pucks. If it gets to the collision radius from a wall, it will turn towards the arena at a random angle. When a puck is found, the robot is informed where the closest home base is located and its own location, then the robot will calculate the direction to the base and navigate back to it. If at any point a robot is detected in the collision radius, it will stop foraging and call the avoidance method to solve the conflict. Pucks do not fall upon collision, they can only be dropped intentionally, at the base.

5.2 Modular representation vs. full-state RL

We begin by evaluating the performance of the modular representation as described in Section 4. First, we instantiate the modular method for the simulated robots. Each sensor is assigned a separate multi-arm bandit learner: here, a continuous-time UCB1 algorithm [14], with a learning rate of 0.1. It chooses between one of 8 directions; a total of 8 learners make these recommendations, and the council aggregates them. We compare the performance with three alternatives:

- **Random**, which selects actions randomly. This serves as a baseline; there is no representation of the state at all.
- **Dynamic Window** [10], a fixed collision-avoidance procedure. The robot will go to the free space that is closest to its own current direction. This algorithm is *stateful* because it knows where other robots are and what their own direction is.
- **R-Learner**, a myopic version of the algorithm presented in [41] with learning rate and exploration rate of 0.1. This algorithm is *stateful*, and would be too expensive to run on a physical robot. It therefore serves as an upper-bound for how good a full reinforcement learning algorithm may perform. Still, states are represented by assigning a boolean value per sensor, corresponding to whether a neighbour is sensed within the *collision radius*. As there are 8 sensors, the total number of states is 256.
- **Continuous-Time Q-Learning**, an adaptation for Q-Learning to accommodate for differences in the amount of time that

passed at every step. The learning rate for step t α_t is determined by: $\alpha_t = 1 - e^{-\lambda\tau}$, where τ is the duration of the stage.

We chose to use a simple Boolean state representation to compute and compare the Continuous-Time Q-Learning and R-Learning: in each of eight directions where a range sensor is pointing, the value is 0 (false) if no neighbour is sensed, or 1 (true) if a neighbour is sensed. The total number of states in this representation is then $2^8 = 256$ for the full-state learners, and $2 \times 8 = 16$ for the modular representation.

Adding more information can be beneficial to all of the learners—including the modular method—but increases the state space combinatorially (for the full-state learners) or multiplicatively (modular method). For example, the value of each of the eight direction features can take on 0 (no neighbour is sensed), or 1–8 (the heading of the sensed robot, in terms of one of the eight directions), for a total of 9 values. Certainly, knowing whether a sensed robot is heading away from the observer or towards it is of great value. In this case, the state space will expand to $9^8 = 43,046,721$ states. In contrast, the modular method will use $9 \times 8 = 72$ states.

Thus, the use of boolean features is consistent with our goal of evaluating the full-state learners and the modular learner under practical limitations closer to the reality of physical swarm robots. In addition, a greater number of states easily extend the learning period, a common phenomenon in reinforcement learning.

All learning algorithms were allowed to learn for twelve simulated hours. After the learning stage, the learning is stopped, and the evaluation stage of twenty simulated minutes starts. Algorithms without learning are also evaluated for twenty minutes.

Figure 3 shows the results. The Y-axis measures the number of items collected during the evaluation phase of the experiments, while the X-axis shows the number of robots. Each line shows the results for a different algorithm. The points mark mean results over 20 runs (different seeds), while the error bars mark standard errors.

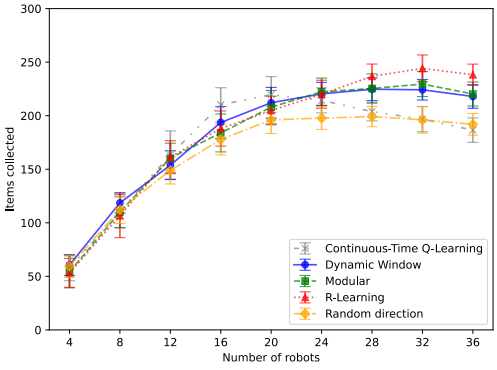

Figure 3: Results in Arena 1

We observe in Figure 3 that up to twelve robots, all algorithms provide essentially indistinguishable results. Dynamic window performs well in lower-density scenarios. At 28 robots the learning algorithms gain an advantage: the stateful R-Learner performed better once the density of the robots is sufficiently high, so that the more informed (and more expensive to store) representation gives an advantage. The modular approach performs stably, on par with dynamic window.

Figure 4 shows the results from Arena 2. Here, the base is close to the corner, resulting in a harder task as more collisions occur when the robots try to return the pucks. All methods are affected by the challenge of this task, performing up to 2.5 worse in comparison to Arena 1 (Fig. 3). Here, too, modular learning closely matches dynamic window, better than R-learning in lower densities, but falling as the density rises. R-learning seems to work only when the density is very high, even losing to the random algorithm in low densities.

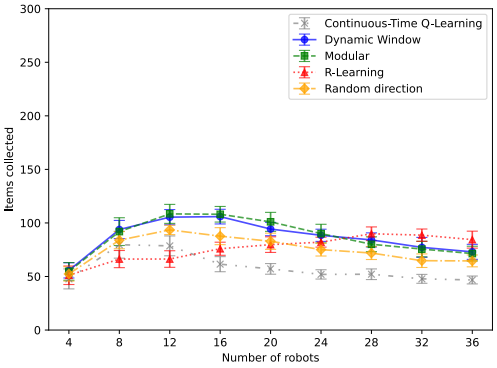

Figure 4: Results in Arena 2

Finally, in Figure 5, multiple bases make the density lower in comparison to a single base, resulting in fewer collisions as the robots can split between the bases. This makes this task easier, as evident by the higher puck collection values in comparison to Arena 1 (Fig. 3). Here, while the modular learning approach is consistently above the random selection baseline, it seems to face difficulties keeping up with dynamic window. We believe that the presence of two bases leads to ambiguous modular states, which the stochastic procedure of the council translates into inconsistent policies.

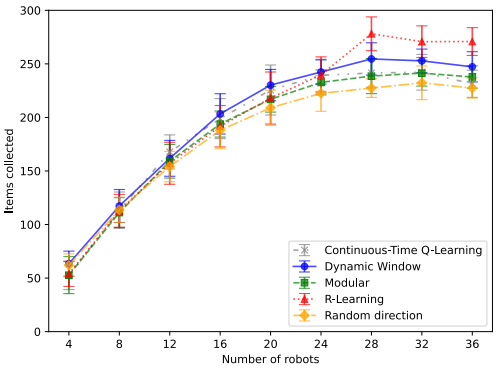

Figure 5: Results in Arena 3

5.3 Modular learning is robust to reward selection

As previously discussed, we chose to use the difference reward (Δ) in all experiments, as it aligns individual and swarm goals. In practice, its use by robots involves a necessary approximation of others' overheads [16]. As it is given that such approximations can be surprisingly brittle, we wanted to test the robustness of the modular representation to changes in the reward value.

We therefore test the modular representation and the R-learning algorithm performance, when the reward was changed from Δ (Eq. 4) to the raw self-interested reward Ω (Eq. 3), treating non-collision intervals as productive, and collision-handling intervals as costly. Figures 6–8 show the results. The modular representation maintains its performance using the changed reward, while the stateful R-learning algorithm shows a dramatic change for the worse. As of now, we do not have an explanation for the robustness of the modular representation, and leave detailed investigation of this to future work.

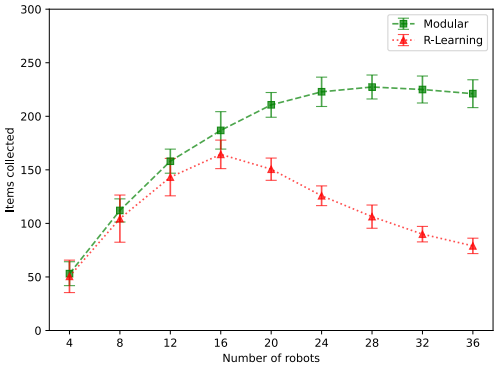

Figure 6: Modular and R-learning algorithms using Δ (Eq. 4) vs Ω (Eq. 3) in Arena 1.

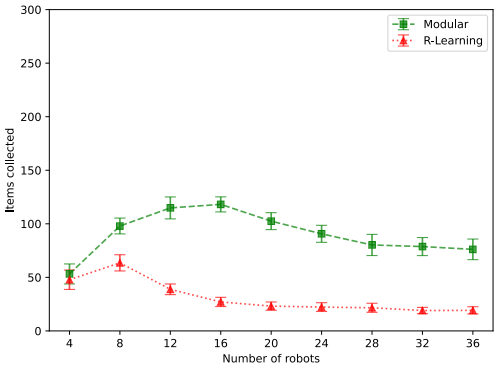

Figure 7: Modular and R-learning algorithms using Δ (Eq. 4) vs Ω (Eq. 3) in Arena 2.

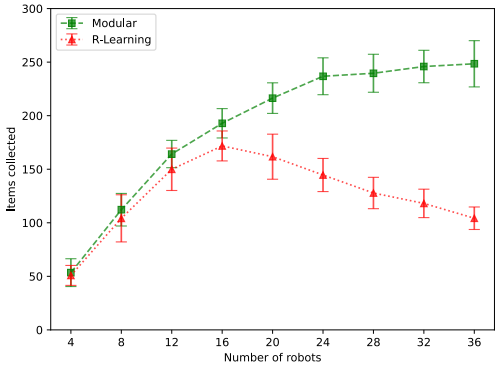

Figure 8: Modular and R-learning algorithms using Δ (Eq. 4) vs Ω (Eq. 3) in Arena 3.

5.4 Vectorial vs Algorithmic Action Spaces

As discussed, the modular method requires a vectorial action space to determine which action to choose, as collision-avoidance algorithms are atomic, so they cannot be merged by the aggregating mechanism of the council. This raises a problem when there is a disagreement between two or more decomposed learning processes. However, given that reliance on such algorithms as the action-space is so prominent in previous work [8, 18, 37, 39, 51], we wanted to evaluate whether the transition to the simpler vectorial action space leads to a loss in overall performance.

Since the modular representation cannot be used with algorithms, we instead used the R-learning algorithm described above, allowing it to select between three algorithms to respond to a collision state: *Repel* [37], *dynamic window* [10] and aggression [51].

The results appear in Figures 9–11. The figure shows that learning to use vectorial actions leads to improved results over learning to use collision-avoidance algorithms to handle collisions.

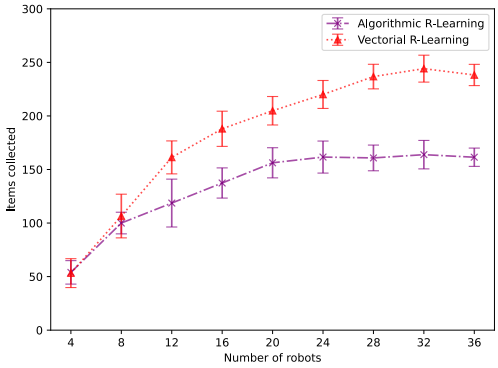

Figure 9: A comparison of learning in vectorial vs. algorithmic action spaces, in Arena 1

6 CONCLUSIONS

Cooperative robot swarms can learn how to interact effectively with others, but to do so, they need to overcome the hurdle of

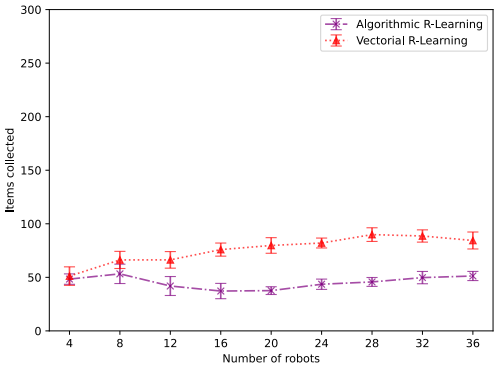

Figure 10: A comparison of learning in vectorial vs. algorithmic action spaces, in Arena 2

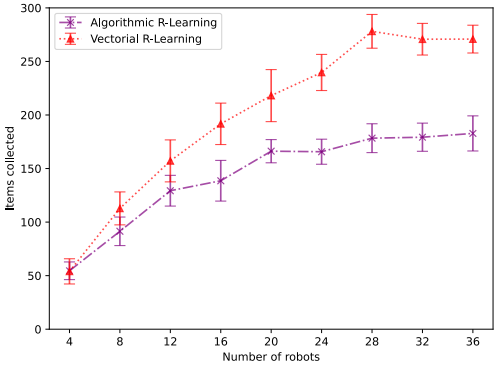

Figure 11: A comparison of learning in vectorial vs. algorithmic action spaces, in Arena 3

state explosion. This is not trivial for physical swarm robots, whose memory is often well below 200KB. We presented a modular state representation approach that takes advantage of the natural decomposition of spatial states and the natural composability of the vectorial action space: a separate learning process is assigned to each range and bearing sensor. This leads to a total number of states that is linear in the number of sensors, rather than exponential. The state-space of each independent learning process is restricted to the readings from the sensor, but its action space is over the entire robot (i.e., all potential headings). The selected actions from all sensors are combined by a mechanism that samples the action preference distribution to choose an action to be executed by the robot.

We have conducted extensive experiments, contrasting the performance of the modular representation with stateful reinforcement learners. The results demonstrate that the modular approach, while not always outperforming others, is robust and generally offers good performance, with dramatic memory savings. In addition, the modular representation can utilize rewards that cause the stateful learners to fail.

We believe modular representations are particularly well suited to spatial states, as they work in similar state spaces as potential fields, and naturally admit a composable action space (vectors of motion). Here, however, they are used as part of a learning process. We intend to explore the use of this modular representation in other swarm robotics tasks as well. In addition, we plan to investigate the use of learning at the aggregation level.

ACKNOWLEDGMENTS

This research was supported in part by ISF Grant # 1373/24. As always, thanks to K. Ushi.

REFERENCES

[1] Arduino 2025. Arduino Robot Technical Specifications. https://docs.arduino.cc/retired/other/arduino-robot/#tech-specs
[2] Levent Bayındır. 2016. A review of swarm robotics tasks. *Neurocomputing* 172 (2016), 292–321.
[3] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. 2013. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence* 7, 1 (2013), 1–41.
[4] Lucian Busoniu, Robert Babuska, and Bart De Schutter. 2008. A Comprehensive Survey of Multiagent Reinforcement Learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38, 2 (2008), 156–172.
[5] N. Correll and A. Martinoli. 2009. Towards Multi-Robot Inspection of Industrial Machinery: From Distributed Coverage Algorithms to Experiments with Miniature Robotic Swarms. *IEEE Robotics and Automation Magazine* 16, 1 (2009), 103–112.
[6] S. Devlin, L. Yliniemi, D. Kudenko, and K. Tumer. 2014. Potential-Based Difference Rewards for Multiagent Reinforcement Learning. In *AAMAS*. Paris, France.
[7] M. Dorigo, G. Theraulaz, and V. Trianni. 2021. Swarm robotics: past, present, and future [point of view]. *Proc. IEEE* 109, 7 (2021), 1152–1165.
[8] Yinon Douchan, Ran Wolf, and Gal A. Kaminka. 2019. Swarms Can be Rational. In *AAMAS*.
[9] Shaheen Fatima, Nicholas R. Jennings, and Michael Wooldridge. 2024. Learning to Resolve Social Dilemmas: A Survey. *JAIR* 79 (2024), 895–969.
[10] D. Fox, W. Burgard, and S. Thrun. 1997. The dynamic window approach to collision avoidance. *IEEE Robotics Automation Magazine* 4, 1 (Mar 1997), 23–33.
[11] GCTronic 2025. Elisa-3 Technical specifications. https://www.gctronic.com/doc/index.php/Elisa-3#Hardware
[12] Generation Robots 2025. e-puck2 technical specifications. https://www.generationrobots.com/en/403090-e-puck2.html
[13] Heiko Hamann. 2018. *Swarm robotics: A formal approach*. Vol. 221. Springer.
[14] Eden R. Hartman. 2022. *Swarming Bandits: A Rational and Practical Model of Swarm Robotic Tasks*. Master's thesis. Bar Ilan University.
[15] Pablo Hernandez-Leal, Michael Kaisers, Tim Baarslag, and Enrique Munoz de Cote. 2019. *A Survey of Learning in Multiagent Environments: Dealing with Non-Stationarity*. Technical Report 1707.09183v2 [cs]. CoRR/arXiv.
[16] Gal A. Kaminka. 2025. Swarms Can be Rational. *Philosophical Transactions of the Royal Society A* 383, 2289 (2025).
[17] Gal A. Kaminka and Yinon Douchan. 2025. Heterogeneous Foraging Swarms Can be Better. *Frontiers in Robotics and AI* 11, 1426282 (2025).
[18] Gal A. Kaminka, Dan Erusalimchik, and Sarit Kraus. 2010. Adaptive Multi-Robot Coordination: A Game-Theoretic Perspective. In *ICRA-10*.
[19] Spiros Kapetanakis and Daniel Kudenko. 2002. Reinforcement learning of coordination in cooperative multi-agent systems. *AAAI/IAAI* 2002 (2002), 326–331.
[20] J. Kober, J. Andrew (Drew) Bagnell, and J. Peters. 2013. Reinforcement Learning in Robotics: A Survey. *IJRR* (July 2013).
[21] Harsha Kokel, Arjun Manoharan, Sriraam Natarajan, Balaraman Ravindran, and Prasad Tadepalli. 2021. RePReL: Integrating Relational Planning and Reinforcement Learning for Effective Abstraction. *Proceedings of the International Conference on Automated Planning and Scheduling* 31, 1 (May 2021), 533–541.
[22] Jonas Kuckling. 2023. Recent Trends in Robot Learning and Evolution for Swarm Robotics. *Frontiers in Robotics and AI* 10 (April 2023).
[23] Timothée Lesort, Natalia Díaz-Rodríguez, Jean-Franois Goudou, and David Filliat. 2018. State representation learning for control: An overview. *Neural Networks* 108 (2018), 379–392.
[24] Marlon Löppenberg, Steve Yuwono, Mochammad Rizky Diprasetya, and Andreas Schwung. 2024. Dynamic robot routing optimization: State–space decomposition for operations research-informed reinforcement learning. *Robotics and Computer-Integrated Manufacturing* 90 (2024), 102812.
[25] Qi Lu, G Matthew Fricke, John C Ericksen, and Melanie E Moses. 2020. Swarm foraging review: Closing the gap between proof and practice. *Current Robotics Reports* (2020), 1–11.
[26] Jason R. Marden and Jeff S. Shamma. 2018. Game-Theoretic Learning in Distributed Control. In *Handbook of Dynamic Game Theory*, Tamer Basar and

Georges Zaccour (Eds.). Springer International Publishing, Cham, 1–36.
[27] Jason R. Marden and Adam Wierman. 2013. Distributed Welfare Games. *Operations Research* 61, 1 (2013), 155–168.
[28] Laetitia Matignon, Guillaume J. Laurent, and Nadine Le Fort-Piat. 2012. Independent Reinforcement Learners in Cooperative Markov Games: A Survey Regarding Coordination Problems. *The Knowledge Engineering Review* 27, 1 (Feb. 2012), 1–31.
[29] Aditya Mohan, Amy Zhang, and Marius Lindauer. 2023. Structure in reinforcement learning: A survey and open problems. *arXiv preprint arXiv:2306.16021* 34 (2023).
[30] Ann Nowé, Peter Vrancx, and Yann-Michaël De Hauwere. 2012. Game theory and multi-agent reinforcement learning. In *Reinforcement Learning*. Springer, 441–470.
[31] L Panait and Sean Luke. 2003. Collaborative multi-agent learning: A survey. *Department of Computer Science, George Mason University, Tech. Rep* (2003).
[32] Particle.io 2025. Photon controller technical specifications. https://docs.particle.io/photon/
[33] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, Mauro Birattari, Luca Maria Gambardella, and Marco Dorigo. 2012. ARGoS: a Modular, Parallel, Multi-Engine Simulator for Multi-Robot Systems. *Swarm Intelligence* 6, 4 (2012), 271–295.
[34] Pololu Robotics and Electronics 2025. 3pi+ robot technical specifications. https://www.pololu.com/product/975
[35] Yinjie Ren, Zhan Xu, Jian Zhao, Jincun Liu, Yang Liu, and Jiahui Cheng. 2023. Collective Foraging Mechanisms and Optimization Algorithms: A Review. In *Chinese Conference on Swarm Intelligence and Cooperative Control*. Springer, 123–135.
[36] J. Kenneth Rosenblatt and David W. Payton. 1989. A fine-grained alternative to the subsumption architecture for mobile robot control. In *International 1989 Joint Conference on Neural Networks*. IEEE, 317–323.
[37] Avi Rosenfeld, Gal A. Kaminka, Sarit Kraus, and Onn Shehory. 2008. A Study of Mechanisms for Improving Robotic Group Performance. *AIJ* 172, 6–7 (2008), 633–655.
[38] Michael Rubenstein, Christian Ahler, and Radhika Nagpal. 2012. Kilobot: A Low Cost Scalable Robot System for Collective Behaviors. In *ICRA*. Computer Society Press of the IEEE, Washington, DC, USA., 3293–3298.
[39] P. Rybski, A. Larson, M. Lindahl, and M. Gini. 1998. Performance evaluation of multiple robots in a search and retrieval task. In *In Proceedings of the Workshop on Artificial Intelligence and Manufacturing*. Albuquerque, NM, 153–160.
[40] Himanshu Sahni, Saurabh Kumar, Farhan Tejani, Yannick Schroecker, and Charles Isbell. 2017. State Space Decomposition and Subgoal Creation for Transfer in Deep Reinforcement Learning. arXiv:1705.08997 [cs.AI]
[41] Anton Schwartz. 1993. A Reinforcement Learning Method for Maximizing Undiscounted Rewards. In *Machine Learning Proceedings 1993*. Elsevier, 298–305.
[42] Dylan A Shell and Maja J Mataric. 2006. On foraging strategies for large-scale multi-robot systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2717–2723.
[43] Z. Song and Richard T. Vaughan. 2013. Sustainable robot foraging: Adaptive fine-grained multi-robot task allocation for maximum sustainable yield of biological resources. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE.
[44] Marco Tamassia, Fabio Zambetta, William L Raffe, Florian'Floyd' Mueller, and Xiaodong Li. 2016. Dynamic choice of state abstraction in q-learning. In *ECAI 2016*. IOS Press, 46–54.
[45] Alan F.T. Winfield. 2009. Foraging Robots. In *Encyclopedia of Complexity and Systems Science*, Robert A. Meyers (Ed.). Springer New York, New York, NY, 3682–3700.
[46] David H. Wolpert and Kagan Tumer. 2002. Collective Intelligence, Data Routing and Braess' Paradox. *Journal of Artificial Intelligence Research* 16 (2002), 359–387.
[47] Esther Wong, Kin Leung, and Tony Field. 2021. State-space decomposition for reinforcement learning. *Dept. Comput., Imperial College London, London, UK, Rep* (2021).
[48] Erfu Yang and Dongbing Gu. 2004. *Multiagent reinforcement learning for multi-robot systems: A survey.* Technical Report. tech. rep.
[49] Ouarda Zedadra, Nicolas Jouandeau, Hamid Seridi, and Giancarlo Fortino. 2017. Multi-Agent Foraging: state-of-the-art and research challenges. *Complex Adaptive Systems Modeling* 5, 1 (2017).
[50] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. 2021. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control* (2021), 321–384.
[51] M. Zuluaga and R. Vaughan. 2005. Reducing spatial interference in robot teams by local-investment aggression. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2798–2805.