Adaptive Team Cooperative Co-Evolution for a Multi-Rover Distribution Problem

Nicolas Fontbonne, Nicolas Maudet, Nicolas Bredeche

Adaptive Team Cooperative Co-Evolution for a Multi-Rover Distribution Problem

Nicolas Fontbonne
Sorbonne Université, CNRS, ISIR, LIP6
F-75005 Paris, France
nicolas.fontbonne@sorbonne-universite.fr

Nicolas Maudet
Sorbonne Université, LIP6
F-75005 Paris, France
nicolas.maudet@lip6.fr

Nicolas Bredeche
Sorbonne Université, CNRS, ISIR
F-75005 Paris, France
nicolas.bredeche@sorbonne-universite.fr

ABSTRACT

This paper deals with policy learning for a team of heterogeneous robotic agents when the whole team shares a single reward. We address the problem of providing an accurate estimation of the contribution of each agent in tasks where coordination between agents requires joint policy updates of two (or more) agents. This is typically the case when two agents must simultaneously modify their behaviors to perform a joint action that leads to a performance gain for the whole team. We propose a cooperative co-evolutionary algorithm extended with a multi-armed bandit algorithm that dynamically adjusts the number of agents that should update their policies simultaneously, aiming both for performance and learning speed. We use a realistic robotic multi-rover task where agents must physically distribute themselves on points of interest of different natures to complete the task. Results show that the algorithm is able to select the best group size for policy updates that reflects the task's coordination requirements. Surprisingly, we also reveal that coupling between agents' actions in a realistic setup can also emerge from interactions at the phenotypical level, hinting at subtle interactions during learning between the control parameter space and the behavioral space.

CCS CONCEPTS

• **Computing methodologies** → **Evolutionary robotics**; • **Theory of computation** → **Multi-agent reinforcement learning**.

KEYWORDS

multi-robots, multi-agent systems, evolutionary robotics, multi-agent reinforcement learning, multi-armed bandits, marginal contribution, fitness evaluation, cooperative coevolution EA

.

1 INTRODUCTION

Multi-robot systems can be used to tackle tasks where many smaller subtasks have to be addressed, sometimes simultaneously. This means that robots often have to coordinate their policies so as to perform complementary actions. This is the case when all robots are part of the same team, where everyone works towards the same goal. However, **the problem of designing efficient individual robot's policies is challenging when the whole team shares a single reward signal**, similar to a football team where all players work together to increase the team's score. In this setup, there is a credit assignment problem as information is not directly accessible to assess a single individual's contribution to the team's performance.

, ,
.

Credit assignments in teams of agents have been explored in both theoretical setups (e.g., N-person coordination games) and experimental setups (e.g., Robocup soccer). Early results in theoretical setups showed that solving an exact formulation is intractable in general as each agent's contribution would require averaging the counterfactual marginal contribution on all possible coalitions [43]. However, many practical implementations have been proposed throughout the years to estimate the contribution of each of the agents through approximation [13, 23, 25] or using various kinds of simplifying assumptions [21, 34, 38, 44, 46, 47, 55]

Learning in Multi-agent systems offers a promising avenue to address coordination problems in multi-robot systems, especially when the state and action spaces are continuous, and only partial observations are available for each robot. In the last few years, multi-robot systems have indeed drawn a lot of attention from both Multi-Agent Deep Reinforcement Learning (MADRL) [24] and Evolutionary Collective Robotics [18, 49]. On the one side, state-of-the-art MARL approaches such as QMIX [40, 41, 45] or MAVEN [31] make use of a combination of individual policy learning agents and shared critics to favour coordination towards better performance.

On the other side, Evolutionary Robotics have pursued different research tracks on whether teams of agents are considered homogeneous (i.e. teams of clones) or heterogeneous. Homogeneous teams are naturally scalable [8, 48] and fast to evaluate [29], and they have been shown to display behavioural specialization by exploiting environmental cues, at least to some extent [12, 20]. When further specialization is required, heterogeneous teams nonetheless allow for the necessary flexibility in terms of role specialization [7, 9, 11, 37, 51].

However, **learning methods for multi-robot systems with a single team-level reward are faced with a dilemma caused by two contradictory issues**:

- the problem of over-generalization [52, 53], which a team faces when the task requires extensive coordination between agents. In this case, joint policy updates may be necessary to discover efficient synergies between agents. This can be solved by updating all policies simultaneously to allow for the discovery of new synergies;
- the problem of exploratory action noise [14, 27, 32], which occurs when an increase in team performance is observed that follows multiple agents updating their policy, without any means to identify who is responsible for such an increase. The actual decrease in the contribution of one particular agent can be shadowed (and sometimes caused) by other suddenly more successful agents. This can be solved by updating only one policy at a time to estimate the benefit (or loss) provided by the agent executing this policy.

Our hypothesis is that the optimal balance between updating a single policy or all policies concurrently is contingent upon the specific problem and may change throughout the learning process. The contribution of the work presented here takes the form of a multiagent evolutionary learning algorithm where the number of agents that undergoes policy updates can be changed to match the problem.

A classical robotic domain task is used for comparison and analysis that involves a team of robots, each following a deterministic learned policy that maps partial observation as continuous states to continuous actions. The

tasks used involve multiple rovers trying to observe a set of points of interest, some of which require close coordination between sub-part of the team. Most importantly, these robotic tasks reveal how deceptive a seemingly simple robotic problem can be: the emergence of coordination may indeed result from joint policy updates *and* subtle behavioural interactions.

We propose combining a Cooperative Coevolution Evolutionary Algorithm with a multi-armed Bandits Algorithm in order to automatically adjust the amplitude of policy updates during the course of learning, to guess which best trade-off should be operated to maximize both the discovery of synergies between agents and the reliability of credit assignment for the agents' contribution to the performance of the team.

In the rest of the paper, previous works are presented that propose solutions for various formulations of the problem to provide an accurate estimation of the contribution of each member of a team, using more or less strong assumptions on the *a priori* knowledge on the task to be done. We then describe two algorithms for the general case and the multi-robot tasks used for evaluation. Finally, results are shown with both algorithms, emphasizing the particularities arising when agents' trajectories are coupled with one another, and the benefits of automatically tuning the number of agents that should simultaneously update their policies.

2 RELATED WORK

Learning complex coordination in a team of agents is a challenging problem, which several methods have tried to address. These approaches can be classified depending on the amount of background knowledge available prior to learning. For example, the D++ algorithm [38] makes strong assumptions about the problem to be solved as requirements for coordination are supposed to be known beforehand (e.g. the number of robots required at a specific location). D++ considers the presence of counterfactual agents at specific times and locations to compute the reward of agents with respect to their possible (rather than actual) contribution to the team's performance. Counterfactual agents are virtual agents used to compute the reward of actual agents *as if* they were not alone. This is particularly relevant for tasks that require coordination among agents who pursue a similar goal (e.g. observing a particular point of interest). However, its requirements for prior knowledge limit its applicability.

Other approaches aim at marginalizing or decomposing the global reward in order to estimate the contribution of each individual to the team performance [47]. The *difference reward* algorithm [3] is a translation in terms of reward shaping [33] of the *Wonderful Life Utility* [54]. The goal is here to extract one agent's marginal contribution by subtracting a counterfactual reward, as if the agent performed a default action, from the gained reward. It can be formalized as:

$$d^{(i)}(\boldsymbol{a}) = r(\boldsymbol{a}) - r(\boldsymbol{a}_{a^{(i)} \leftarrow a^{(i)\prime}}) \tag{1}$$

With $\boldsymbol{a}$ the joint actions of all agents, and $r(\boldsymbol{a})$ is its corresponding global reward. $r(\boldsymbol{a}_{a^{(i)} \leftarrow a^{(i)\prime}})$ is the global reward when agent i perform a default action $a^{(i)\prime}$ instead of the action its policy dictates. $d^{(i)}$ is then, for agent i the advantage of using its policy instead of a default action, freed from the impact of other agents. This method has been successfully applied in several domains including air traffic control [15], rover navigation [5, 28], satellite coordination [1], distributed sensor networks [26, 50], communication system optimization [2], and robot swarm coordination [19].

Multi-Agent Deep Reinforcement Learning combines the decentralized execution of agents' actions and a centralized critic used to approximate the agents' contributions. For example, the QMIX algorithm [39, 42] uses a neural network to map a joint value function to individual value functions. Assuming a counterfactual action is available, the Counterfactual Multi-Agent Policy Gradients algorithm (COMA) [21] estimates the contribution of any single agent by comparing team performance with each agent playing its true action versus the same agent playing the counterfactual action baseline.

Reward shaping can also be used to compute counterfactual rewards for actions that were not taken by agents. This is achieved by using a model of the reward function and the CLEAN shaping structure [27]. One approach is to compare the score of the current action with a new exploratory counterfactual action $a^{(i)\prime}$:

$$c1^{(i)}(\boldsymbol{a}) = r\left(\boldsymbol{a}_{a^{(i)} \leftarrow a^{(i)\prime}}\right) - r(\boldsymbol{a}) \tag{2}$$

This results in the opposite of the difference reward, which was calculated by comparing the reward of the current action to a default counterfactual. Combining these two approaches, a second version of the CLEAN reward can be obtained. This consists of comparing an exploratory action $a^{(i)\prime}$ to a default action $a^{(i)\prime\prime}$:

$$c2^{(i)}(\boldsymbol{a}) = r\left(\boldsymbol{a}_{a^{(i)} \leftarrow a^{(i)\prime}}\right) - r\left(\boldsymbol{a}_{a^{(i)} \leftarrow a^{(i)\prime\prime}}\right) \tag{3}$$

This calculation is used to update the agent's policy. Only the non-exploratory action is visible to other agents, which promotes coordination among them as individual exploration does not impact others.

Additionally, limiting the number of concurrently learning agents across each epoch can be beneficial for reducing agent noise [14]. In this case, agents update their policies depending on a probability that depends on their predicted impact.

In the next Section, we elaborate on works presented here to propose a method that requires neither *a priori* knowledge of the task nor the definition of a default counterfactual action baseline. Similar to the CLEAN reward, we compare different strategies for the team. However, we also enable the modulation of the number of agents simultaneously undergoing a policy update to reduce the exploratory action noise.

3 ALGORITHM

To address the compromise between over-generalization and exploratory action noise, we propose an extension of the Cooperative Co-Evolution Algorithm (CCEA) [30, 35, 36] where the number of agents that are updated at each generation is modulated. We propose two algorithms: (1) the CC-(1+1)-ES$_{k_{\text{fixed}}}$ algorithm updates the policy of a fixed number of agents per learning iteration, and (2) the CC-(1+1)-ES$_{k_{\text{adaptive}}}$ algorithm dynamically adapts the number of agent's policies updated per learning step. The two algorithms are described in Figure 1, and both are extensions of our previous work [22] to the domain of the continuous state, action and parameter spaces for robotics problems.

3.1 CC-(1+1)-ES$_{k_{\text{fixed}}}$ with Fixed Policy Updates

CC-(1+1)-ES$_{k_{\text{fixed}}}$ runs a collection of (1+1)-ES algorithms in parallel. Each (1+1)-ES algorithm i provides the policy parameters $\theta^{(i)}$ for its corresponding agent i. The whole team is evaluated together, and the score is provided for the team. If F is the policy evaluation function, we note $f = F(\theta^{(0)}, \ldots, \theta^{(N-1)})$ the score of a team composed of N agents.

Each (1+1)-ES algorithm maintains a population of two individuals [10], a parent $\theta^{(i)}_{\text{parent}}$ and a child $\theta^{(i)}_{\text{child}}$. Both are candidate policy parameters for agent i. The parent is replaced when the child outperforms it, using the possibly noisy team-level score obtained during evaluation. A new child is created by applying mutation to the new parent. If the child does not outperform its parent, it is replaced by a new child mutated from the current parent.

The mutation operator depends on the problem. In the application we tackle, the mutation operator chooses with probability $p = 0.2$ between a *Gaussian mutation* operator on all parameters of an agent, and with probability $1 - p = 0.8$ a *uniform mutation* operator of a subset of its parameters. The Gaussian mutation operator applies a perturbation centred on the current parameter value, with a variance σ:

$$\theta \leftarrow \mathcal{N}(\theta, \sigma)$$

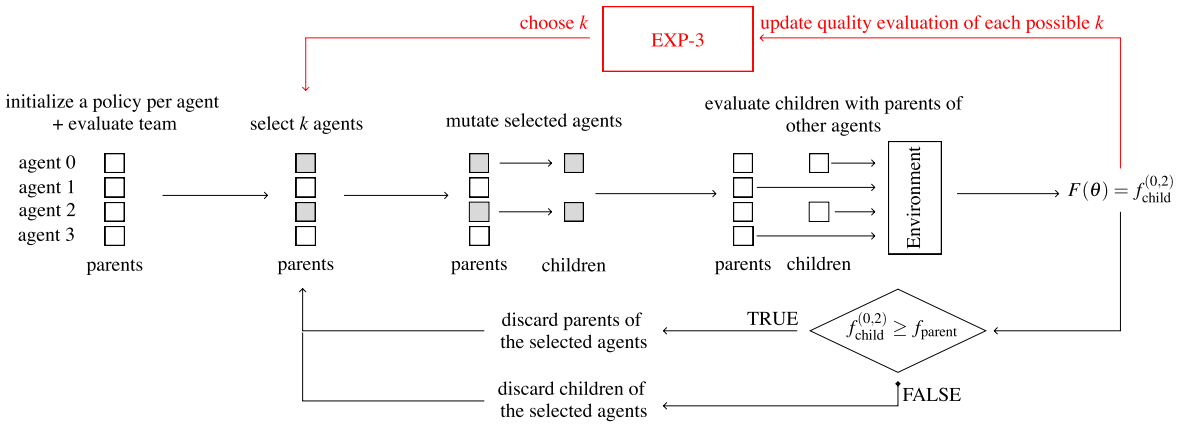

Figure 1: The CC-1+1-ES algorithm. Each square represents some policy parameters $\theta^{(i)}$. In this example, $k = 2$. Agents 0 and 2 are randomly selected and then mutated to produce new children. These children are evaluated alongside the parents of agents 1 and 3 to produce the team score $f^{(0,2)}_{\text{child}}$. If it is greater or equal to the previously evaluated team score (f_{parent}), the children are kept, or else they are discarded. The top red part represents the choice of k in the k_{adaptive} version of the algorithm.

And the uniform mutation operator on range $\mathcal{R}$ of a subset V_m of m randomly selected parameters is defined as:

$$\theta[i] \leftarrow \mathcal{U}(\mathcal{R}) \quad \forall i \in V_m$$

At each new generation, k agents among N are drawn and randomly changed in the team, with $0 < k \leq N$. The k new children are kept only if they increase the team score. Therefore, the expert's challenge is finding the most efficient size k of agents to change in each generation before learning starts.

3.2 CC-(1+1)-ES$_{k_{\text{adaptive}}}$ with Adaptive Updates

So far, k is pre-determined by the user and may benefit from prior knowledge about the task regarding possible necessary coupling between agents' policy updates. However, such prior knowledge may not be available. In particular, a relevant value of k depends on the problem and on the current state of the optimization (e.g., broad initial search steps vs. refined tuning near the optimal solution).

To address this, the CC-(1+1)-ES$_{k_{\text{adaptive}}}$ allows for adapting k during evolution. The value of k is modified at each generation using the multi-armed bandit learning algorithm EXP3 (Exponential-weight algorithm for Exploration and Exploitation [6]). This method allows choosing among a set of possible values for k, according to the performance gains they allowed. It uses an egalitarianism factor γ, which allows modulating the exploration or the exploitation. A γ closer to 0 indicates more exploitation of the best-performing group size k, while a γ closer to 1 will allow more exploration and k will be drawn almost uniformly. The full algorithm is shown in Fig. 1 with the multi-armed bandit for dynamic selection in red. As using $\gamma = 0.1$, 0.5 or 0.9 yielded similar results, we only show results obtained with $\gamma = 0.1$.

4 EXPERIMENTAL SETUP

4.1 Task and Environment

To study the dynamics of these algorithms, we devise a rover exploration task inspired by [4] (and used in further research [16, 17, 56]). It is a good abstraction of a search and retrieve robotic task, where an unknown environment must be explored to retrieve specific objects or resources. The goal here is for a team of rovers to reach some points of interest (POI) that would give the most rewards.

Four rovers start in a small aggregate in the center of an area with some random variation of coordinates and orientation. Each rover senses both its teammates and points of interest (POI) if close enough. The team is expected to coordinate to observe POIs to maximize the score. It depends on the nature of the POIs and the number of rovers that end up observing each POI. Table 1 provides details on the location and orientation of the robots, which are also represented by small coloured squares in Figure 2.

Parameter	Value
Agent 0 x,y,orient.	$x \in [0, 0.5]$, $y \in [0, 0.5]$, $\alpha \in [0, \frac{\pi}{2}]$
Agent 1 x,y,orient.	$x \in [-0.5, 0]$, $y \in [0, 0.5]$, $\alpha \in [\frac{\pi}{2}, \pi]$
Agent 2 x,y,orient.	$x \in [-0.5, 0]$, $y \in [-0.5, 0]$, $\alpha \in [\pi, \frac{3\pi}{4}]$
Agent 3 x,y,orient.	$x \in [0, 0.5]$, $y \in [-0.5, 0]$, $\alpha \in [\frac{3\pi}{4}, 2\pi]$
Number of POIs	10
POI Radius	2 px
POI distance to center	13 px

Table 1: Initial range values for locations of rovers and POI.

Ten POIs placed in a circle around the robots' starting position. The agents have T steps to reach a POI, after which the team reward is computed. Each POI delivers some reward if the n agents that visit it are in their radius of observation = 2. Each POI can be of either of two types:

- solo-POI: $r = 1$ if $n \geq 1$;
- duo-POI: $r = 3$ if $n \geq 2$.

The team thus obtains a global reward R equal to the sum of all rewards delivered by the POI, $R = \sum_{\text{POI}} r$.

We devise two different setups, which are illustrated in Figure 2 and characterized as follows:

- **Setup 1**: 10 solo-POI are placed in a circle around the center of the arena. The maximal score is attained if each agent moves to a unique POI. The number of POIs is larger than the number of agents, and the task requires a low level of coordination between agents.
- **Setup 2**: 8 solo-POI and 2 duo-POI are placed in a circle around the center of the arena. The 2 duo-POI are aligned on the horizontal axis. There are three equilibria in which teams may end up, i.e. situations from which the behavioural change of a single agent will either decrease or leave an unchanged score. The first two equilibria are

suboptimal: (a) agents may spread among solo-POIs, for a score of 4, and (b) agents may split among 2 solo-POIs and 1 duo-POIS for a score of 5. The single optimal equilibrium is reached when agents pair together two by two and spread on the two duo-POIs for a score of 6. The task is assumed to require a high level of coordination between agents.

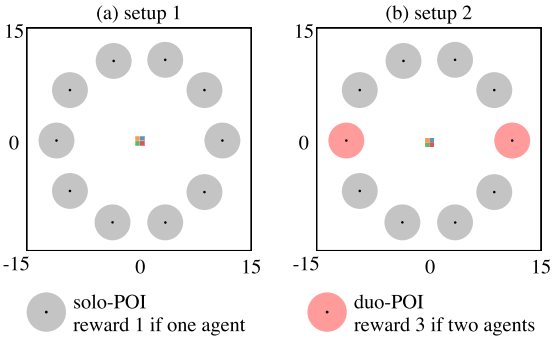

Figure 2: The two setups explored in this study. Setup 1 has 10 solo-POI organized in a circle around the center. We refer to this setup as the 10 solo-POI setup. Setup 2 has the same circular configuration, but the POI on the horizontal axis are duo-POI. We refer to this setup as the 8 solo 2 duo-POI setup. In both figures, the small coloured square in the middle represents the initialization area of the agents described in Table 1. Blue represents the initialization area of agent 1; orange represents agent 2; green represents agent 3; and red represents agent 4.

4.2 Robot model

Agents in the experiment represent robots subject to a kinematic model, with an action space A and observation space O. Concerning the action space, it is a question of controlling a velocity vector. Thus, the agents have a 2-dimensional action space A where the two dimensions represent speed ($a_0 \in [0, 1]$ for [stop, maximum speed]) and angular speed ($a_1 \in [-1, 1]$ for [clockwise, anti-clockwise]).

Their observation space O is composed of :

- 3 sensors per POI that get information about their distance, angle and type;
- 2 sensors per rover that get information about their distance and angle.

Thus, for our setups with 10 POI and 4 rovers, we obtain an observation vector of $\dim(O) = 38$ dimensions.

4.3 Policy

The agent's policy π maps observations $\mathbf{o} \in O$ to actions $\mathbf{a} \in A$. For that purpose, we use a discrete artificial neural network with one hidden layer as the main policy structure. Figure 3 details the full topology between inputs and outputs, and the number of parameters. This architecture imposes a relatively large number of free control parameters θ that need to be optimized. In the present case, this means $dim(\theta) = 658$ parameters.

5 RESULTS

This section describes the results of applying the CC-1+1-ES algorithms to the two setups described earlier. We first evaluate the performance of the CC-(1+1)-ES$_{k_{\text{fixed}}}$ algorithm with different values of k to assert the

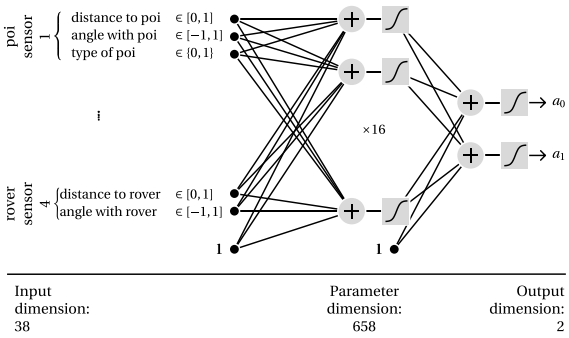

Figure 3: Architecture of the policy function. Each sensor gives information about the distance to the obstacle and the type of POI detected if within range. The policy is a multi-layered neural network, with 38 dimensions as input, a hidden layer of 16 dimensions plus a bias term for each layer. Hyperbolic tangent (tanh) is used as an activation function. The action dimension is 2. The number of parameters is 658.

relevance of tuning the number of joint policy updates depending on the problem at hand. We also provide further experimental studies to uncover the dynamics of coordination between robots. Secondly, we conduct an experimental study of the CC-(1+1)-ES$_{k_{\text{adaptive}}}$ algorithm in order to show that the value of k can indeed be dynamically tuned over time to match the requirement for the problem at hand.

The environment is stochastic, as the same policy parameters can lead to different trajectories and score outcomes due to the random variation in initial conditions. In order to provide a fair estimation of team performance, we compute and use the expected return as a policy evaluation function $F(\boldsymbol{\theta})$ of the team parameters $\boldsymbol{\theta}$.

$$F(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\tau \sim \boldsymbol{\pi}_{\boldsymbol{\theta}}} [R(\tau)] \tag{4}$$

In practice, this means the return (or fitness) of a single trajectory $R(\tau)$ is the sum of the rewards given by the POIs for this particular evaluation as described in Section 4.1. During all the following experiments, we use the term *score* (as it is more general than the term *fitness*) to refer to the experimental approximation of $F(\boldsymbol{\theta})$ by sampling multiple trajectories from different initial conditions (see Section 4.1). More precisely, each *score* represents the average return of the trajectories from 16 different initial conditions that are kept constant from one generation to another. Using as many different repetitions for evaluation ensures that agents learn policies that generalize behaviour regardless of initial conditions. Table 2 lists parameters and values used in all experiments. All code is available on a repository at *removed for review*. Each experiment is performed with 16 independent runs to account for algorithmic stochasticity.

5.1 Setup 1: Coordination via Specialization

In the first setup, each of the 10 points of interest provides a reward of $r = 1$ if at least one agent comes close enough. The maximum team score of 4 is reached if agents spread out on different points of interest. In this setup, agents are not required to perform complex coordination except for avoiding each other to reach separate POI.

With respect to the number of simultaneous policy updates, it is not clear *a priori* which value of k should allow the fastest convergence and best performance. A value of $k = 1$ is less destructive as it ensures that a policy is updated only if it is better. However, using $k > 1$ allows for a more exploratory learning strategy, which may not be a problem as optimal

Parameter	Value
POI parameters	
Number of POI	10
Radius	2
Distance from center	13
Agents parameters	
Number of agents	4
Sensor maximum distance	30 px
Maximum velocity $v_{\max}$	1 px/steps
Maximum angular velocity $\omega_{\max}$	60 degrees/steps
Number of steps per evaluation T	15
Controller	
Initialization range	$[-2, 2]$
Sensory inputs	38
Hidden layer	1
Hidden size	16
Control outputs	2
Total number of parameters	658
Algorithm parameters	
Number of initial conditions per eval.	16
Gaussian mutation probability p	0.2
Gaussian mutation size σ	0.1
Uniform mutation volume V_m	25%
Uniform mutation range $\mathcal{R}$	$[-2, 2]$
Egalitarianism factor γ for EXP3	0.1

Table 2: Experimental parameters

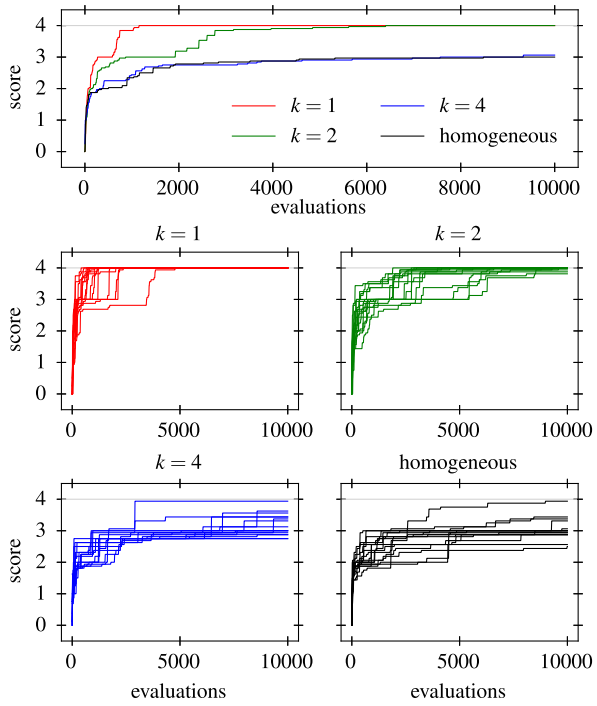

Figure 4: Performance of the algorithm with fixed $k = 1, 2, 4$ and the homogeneous team on Setup 1. Top: each curve represents the median on 16 independent replications. Bottom: The four figures represent the learning curves of the sixteen independent replications of the four cases studied.

policies are largely independent: save for avoiding each other, the task does not require robots to perform complex coordination behaviours.

In order to provide a fair comparison, we use two control experiments built from classic models featured in the literature:

- heterogeneous team with simultaneous policy updates. This corresponds to using $k = 4$, where all policies are updated simultaneously aiming for maximal exploration (see [51] for a comprehensive overview of genetic composition);
- homogeneous team. In this scheme, all individuals are clones, meaning they use the same control parameters. This scheme is particularly well-fitted when the score is assigned to the whole team, as the trick is to consider the team as a single agent, rather than a composite. A successful policy update is then guaranteed to be beneficial for the whole team at once. This approach has been shown by others to provide some level of behavioural specialization [20].

Figure 4 shows the evaluation for group sizes of $k = 1$, $k = 2$, $k = 4$ (control 1) and a *homogeneous* team (control 2). All versions of the algorithm could reach optimal performance in at least one run out of 16, but displayed very different dynamics and overall performance both in speed and quality. $k = 1$ is the fastest to converge, followed by $k = 2$, which is twice slower when the median value is considered. Both control experiments $k = 4$ and *homogeneous* are significantly slower, and their medians never get even close to the maximum score. They struggle in terms of the number of runs that discover the optimal strategy. Indeed, there is a *single* run for each control that reaches the optimal score while using $k = 1$ and $k = 2$ lead to *all* runs converging towards teams with optimal performance.

The first observation from these results is that, as postulated in the introduction of this paper, balancing the number of policy updates is critical to getting the best team performance. Larger values of k experience degradation in performance due to concurrent exploration by multiple agents. This tends to add noise during the learning process since the evaluation variation will depend on the modification of several agents. These are similar to the exploratory action noise highlighted by [14, 27], which is completely avoided for the lower value of $k = 1$, and near-completely with $k = 2$.

The second observation is that the canonical homogeneous team composition approach falls largely behind using $k = 1$ and 2 in the heterogeneous team composition. While the need for coordination between agents might seem limited at first, agents still have to observe different points of interests. With homogeneous team composition, this requires learning a policy that can both target a single point of interest *and* avoid any point of interest already being observed. This can be an issue with agents sharing the same policy as coordination may be more challenging to achieve, a problem that will be explored further in the following Sections.

5.2 Setup 2: Coordination via Limited Synergies

The second setup also features 10 point of interest, with 2 special points of interest (the duo-POIs) that must be observed by 2 agents to provide a reward of $r = 3$ points. As in the previous setup, agents may split among the normal points of interest (the 8 solo-POIs) for a sub-optimal maximal score of 4. In order to obtain the best score, the team must split into two groups, one for each duo-POIs, for a maximum score of 6. The obvious pitfall is that observing solo-POIs is a local optimum from which improvement can be attained only if *two* agents simultaneously change their behaviours to converge towards the same unoccupied duo-POI (see Section 4.1).

Figure 5 shows the score for group sizes of $k = 1$, $k = 2$, $k = 4$ (control 1) and a homogeneous team (control 2). Figure 5-top shows that median values for the four methods converge to a final score of 5, with $k = 4$ ending slightly below this score. The homogeneous team composition is the fastest to converge, followed by $k = 2$ and $k = 1$, which is counter-intuitive as one

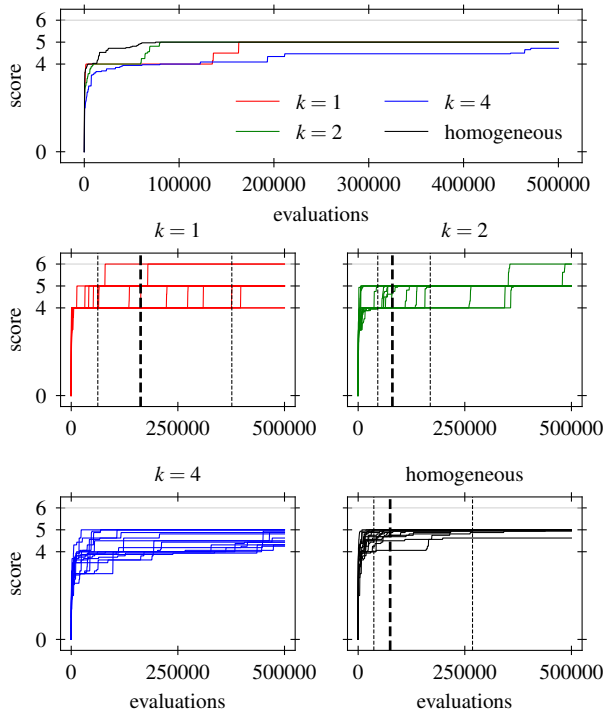

Figure 5: Performance of the algorithm with fixed $k = 1, 2, 4$ and the homogeneous team on Setup 2. Top: each curve represents the median on 16 independent replications. Bottom: The four figures represent the learning curves of the sixteen independent replications of the four cases studied. The vertical dotted lines indicate when a specific number of independent runs were able to reach a score $f \geq 5$: $1/4$ of the runs (thin), $1/2$ (bold) and $3/4$ (thin).

would have expected both the *homogeneous* and $k = 1$ to fail at capturing even one duo-POI. Moreover, all methods converge to suboptimal team performance, with a score of 5 corresponding to the team observing 1 duo-POIs and 2 solo-POIs. $k = 2$ (as well as $k = 4$, at least to some limits) could have been expected to learn successful teams able to capture both duo-POIs by jointly updating policies of more than one agent. This does not seem to be the case at least when median values are considered.

The picture is very different when we look at independent runs for each experiment (the four graphs in Fig 5-bottom). $k = 1$ and $k = 2$ outperform the two control experiments by both succeeding at reaching the optimal score of 6 (2 runs out of 16 for each method). This means that teams can capture both duo-POIs, a feat that is never achieved with either $k = 4$ or the homogeneous team. Moreover, $k = 2$ is faster to converge than $k = 1$ in general (median values and proportion of runs than reach the threshold score of 5).

These results confirm the conclusion obtained with the first setup regarding the benefit of limiting the number of policy updates when confronted with a task that requires agents to coordinate. However, it is surprising that the level of coordination obtained with $k = 1$ is sufficient to reach optimal team performance. By construction, the task was meant to require that *two* agents change their behaviours *simultaneously*, abandoning a suboptimal yet stable equilibrium (which is here illustrated by team scores of either 4 or 5, which are visible in the curves) in order to reach the optimal equilibrium ($score = 6$). The next Section provides an analysis as to why $k = 1$ is able to learn optimal policies even if strong coordination is required to solve the task.

5.3 Coordination in the Phenotypical Space

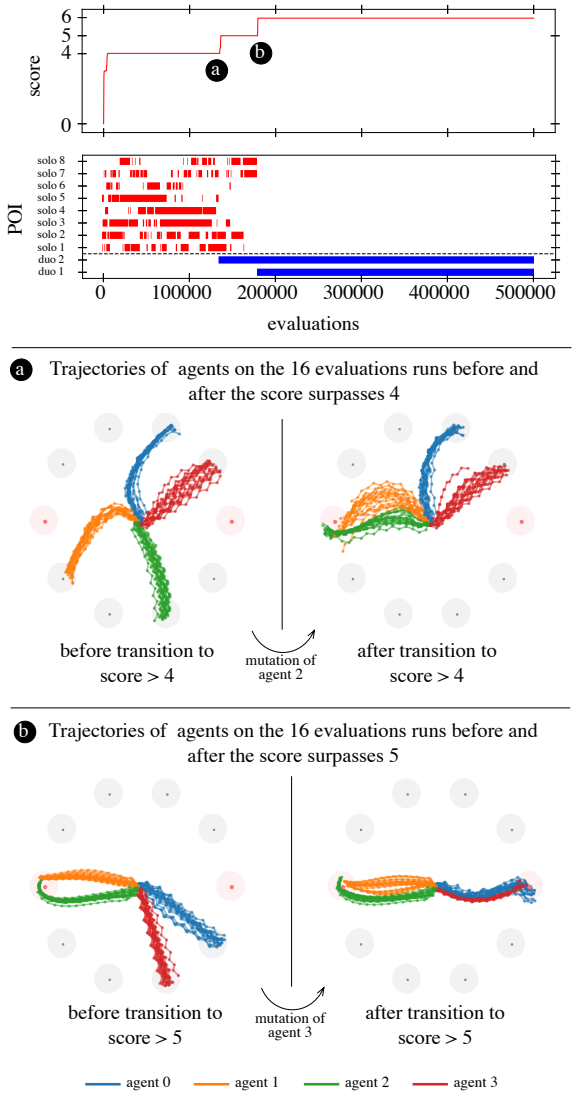

Figure 6: Impact of single mutations on the agents' behaviours in Setup 2 for $k = 1$. Top: results from a single run (run #6) where the score reaches the maximal value of 6, along with the distribution of the POI throughout learning for one of the initial conditions out of 16. Red and blue lines indicate that respectively one and two agents occupy these POI. Bottom: Trajectories of agents from the 16 initial conditions before and after a single agent's policy update yields a significant score increase. The team switches (a) from 4 solo-POIs observed to 2 solo-POIs and 1 duo-POIs observed and (b) from 2 solo-POIs and 1 duo-POIs to 2 duo-POIs.

Each agent perceives the others during the experiment and may either ignore or be influenced by their teammates. Consequently, simultaneous

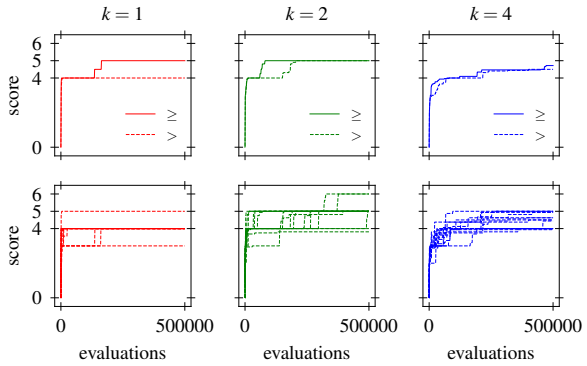

Figure 7: Comparison of the CC-1+1-ES algorithm with fixed $k = 1, 2$ and 4 on Setup 2 with either a soft (>=) or strict (>) selection operator. Top: median values with soft or strict selection. Bottom: scores for each of the 16 independent replications for the strict selection (see Fig. 5 for comparison).

modification of multiple agent trajectories can arise even if a single agent updates its policy. Rather than occurring purely during joint policy updates, coordination may arise between agents solely due to one agent influencing another. This is observed in the second setup when learning with $k = 1$.

Figure 6 shows examples of trajectories for all agents with $k = 1$, extracted from a run where the team reaches the optimal score. The two graphs on Fig.6-top show performance during learning and the distribution of agents among the points of interest starting from one initial condition used during evaluation (taken randomly out of the 16 initial conditions). It confirms the switch from one equilibrium to a better one as two agents jointly stop observing distinct solo-POIs and start observing the same duo-POI.

Figure 6-bottom features trajectories of agents for all 16 initial conditions for this particular run before and after two events: (a) when the team observes its first duo-POI and (b) when the team reaches optimal distribution over the two duo-POIs. Fig. 6-bottom-(a) show how the policy update of a single agent (here, agent no.2) changes the behaviour of other agents whether this change is slight (agents no.0 and no.3 marginally changes trajectories but still observe the same POI) or with significant impact on the score (agent no.1 switches and now observe a duo-POI, jointly with agent no.2). Similarly, Fig.6-bottom-(b) show the same team transitioning to the optimal team coordination behaviour.

It is interesting to note that the right panel of Fig. 6-bottom-(a) and the left panel of Fig. 6-bottom-(b) show different team trajectories while they come from the same run and corresponds to a similar outcome ($score = 5$). This is due to the selection operator used in our evolutionary algorithm, which conserves a policy update as long as it yields a better *or similar* score than previously. This "soft" selection operator enables genetic drift, that is, the possibility to explore different behavioural strategies with a similar score.

Figure 7 shows results obtained using either the default "soft" selection operator (ie. a strategy change is kept if it does not worsen team performance) or a "strict" operator (ie. a strategy change is kept only if it improves team performance). Using a "strict" selection operator, that is, when policy updates are kept *only* if the team score improves, turns out to be detrimental for $k = 1$. It significantly reduces the overall performance for $k = 1$ as the median remains stuck to a score of 4, which corresponds to the worse sub-optimal equilibrium. $k = 2$ moderately suffers from using a strict selection operator, as performance increase is only delayed. Nevertheless, the algorithm using $k = 2$ can still learn optimal team strategies. At the same time, $k = 1$ now fails at completing this task (only one run reaches the sub-optimal score of 5).

Due to the complex and counter-intuitive effect of the genotype-phenotype mapping at work in a robotic task where robots interact, it is no surprise that choosing *a priori* the number of policy updates to perform per learning step is challenging. In the next Section, we present results using an adaptive method for dynamically choosing the k value of our algorithm.

5.4 Adaptive group size

An important lesson from the previous results is that different setups require different values for the number of policies to be updated simultaneously. In the general case, it can be challenging to guess to which value k should be set, and it is natural to aim for an adaptive method that would automatically tune the value of k to match the problem at hand.

In this Section, we evaluate the CC-1+1-ES$_{k_{\text{adaptive}}}$ algorithm presented in Section 3. CC-1+1-ES$_{k_{\text{adaptive}}}$ extends the algorithm used so far by encapsulating the EXP3 Multi-Armed Bandit algorithm in order to dynamically tune the number of policies that should be updated, based on results obtained so far.

Figure 8 shows the median on 16 independent runs of the score for the CC-1+1-ES$_{k_{\text{adaptive}}}$ algorithm, along with results shown previously with the original CC-1+1-ES$_{k\text{fixed}}$ algorithm with $k = 1$, 2 and 4 for reference (see Fig. 4 and 5, here with transparent curves).

Results show that the adaptive method closely matches the best-performing algorithm with fixed k (median values, first row), though it is a bit slower to reach the best value. This is expected as the cost of exploration cannot compete with an algorithm initialized with the best value for k. This is confirmed by looking at the independent runs (second row): in both setups, the adaptive version is shown to be able to reach the optimal team behaviours on par with the non-adaptive method using the optimal value for k, and without the burden to guess this value prior to learning.

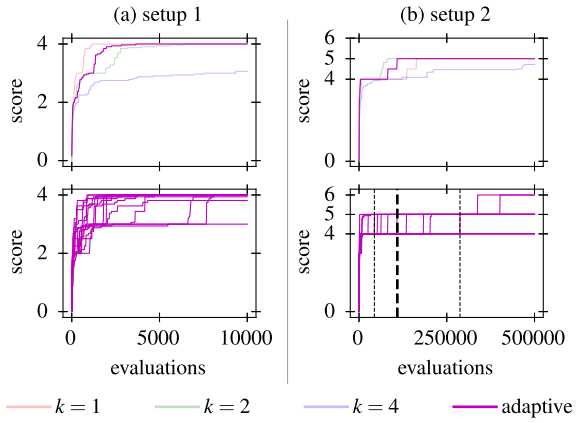

Figure 8: Performance of the CC-1+1-ES$_{k_{\text{adaptive}}}$ algorithm k in the two setups. Top: median scores. Previous results shown with transparent curves. Bottom: scores for each of the 16 replications. The vertical dotted lines indicate when a specific number of independent runs are able to reach a score $f \geq 5$: 1/4 of the runs (thin), 1/2 (bold) and 3/4 (thin).

6 CONCLUSION

In this paper, we addressed the problem of estimating the contribution of robotic agents when performance is measured at the level of the team. When several agents update their policy simultaneously, it is difficult to identify

a single agent's contribution to the team, even though joint updates are necessary to find synergies when a high level of coordination is required. We proposed an algorithm that automatically balances the number of agents' policies that should be updated at each learning step in order to discover the degree of coordination required, without requiring prior assumptions on the task at hand.

ACKNOWLEDGMENTS

This work is funded by ANR grant ANR-18-CE33-0006.

REFERENCES

[1] Adrian K Agogino, Chris HolmesParker, and Kagan Tumer. 2012. Evolving distributed resource sharing for cubesat constellations. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*. 1015–1022.

[2] Adrian K Agogino, Chris HolmesParker, and Kagan Tumer. 2012. Evolving large scale UAV communication system. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*. 1023–1030.

[3] Adrian K Agogino and Kagan Tumer. 2004. Efficient Evaluation Functions for Multi-rover Systems. In *Genetic and Evolutionary Computation – GECCO 2004*, Kalyanmoy Deb (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–11.

[4] Adrian K Agogino and Kagan Tumer. 2004. Efficient evaluation functions for multi-rover systems. In *Genetic and evolutionary computation conference*. Springer, 1–11.

[5] Adrian K Agogino and Kagan Tumer. 2008. Analyzing and visualizing multiagent rewards in dynamic and stochastic domains. *Autonomous Agents and Multi-Agent Systems* 17, 2 (2008), 320–338.

[6] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. 2002. The Nonstochastic Multiarmed Bandit Problem. *SIAM J. Comput.* 32, 1 (2002), 48–77. https://doi.org/10.1137/S0097539701398375 arXiv:https://doi.org/10.1137/S0097539701398375

[7] Gianluca Baldassarre, Stefano Nolfi, and Domenico Parisi. 2003. Evolving mobile robots able to display collective behaviors. *Artificial life* 9, 3 (2003), 255–267.

[8] Cristobal Baray. 1997. Evolving cooperation via communication in homogeneous multi-agent systems. In *Proceedings Intelligent Information Systems. IIS'97*. IEEE, 204–208.

[9] Arthur Bernard, Jean-Baptiste André, and Nicolas Bredeche. 2016. To cooperate or not to cooperate: why behavioural mechanisms matter. *PLoS computational biology* 12, 5 (2016), e1004886.

[10] Hans-Georg Beyer and Hans-Paul Schwefel. 2002. Evolution strategies - A comprehensive introduction. *Natural Computing* 1, 1 (2002), 3–52.

[11] Josh C Bongard and Chandana Paul. 2000. Investigating morphological symmetry and locomotive efficiency using virtual embodied evolution. In *From Animals to Animats: The Sixth International Conference on the Simulation of Adaptive Behaviour*. Citeseer.

[12] Bobby D Bryant and Risto Miikkulainen. 2003. Neuroevolution for adaptive teams. In *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.*, Vol. 3. IEEE, 2194–2201.

[13] Jianbo Chen, Le Song, Martin J Wainwright, and Michael I Jordan. 2018. L-shapley and c-shapley: Efficient model interpretation for structured data. *arXiv preprint arXiv:1808.02610* (2018).

[14] Jen Jen Chung, Scott Chow, and Kagan Tumer. 2018. When less is more: Reducing agent noise with probabilistically learning agents. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. 1900–1902.

[15] Leonardo LBV Cruciol, Antonio C de Arruda Jr, Li Weigang, Leihong Li, and Antonio MF Crespo. 2013. Reward functions for learning to control in air traffic flow management. *Transportation Research Part C: Emerging Technologies* 35 (2013), 141–155.

[16] Gaurav Dixit and Kagan Tumer. 2022. Behavior Exploration and Team Balancing for Heterogeneous Multiagent Coordination. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. 1578–1579.

[17] Gaurav Dixit, Nicholas Zerbel, and Kagan Tumer. 2019. Dirichlet-Multinomial Counterfactual Rewards for Heterogeneous Multiagent Systems. In *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 209–215.

[18] Stephane Doncieux, Nicolas Bredeche, Jean-Baptiste Mouret, and A.E. Eiben. 2015. Evolutionary Robotics: What, Why, and Where to. *Frontiers in Robotics and AI* 2, March (2015), 1–18. https://doi.org/10.3389/frobt.2015.00004

[19] Yinon Douchan, Ran Wolf, and Gal A Kaminka. 2019. Swarms can be rational. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. 149–157.

[20] Eliseo Ferrante, Ali Emre Turgut, Edgar Duéñez-Guzman, Marco Dorigo, and Tom Wenseleers. 2015. Evolution of Self-Organized Task Specialization in Robot Swarms. *PLoS Computational Biology* 11, 8 (2015), e1004273. https://doi.org/10.1371/journal.pcbi.1004273

[21] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2017. Counterfactual Multi-Agent Policy Gradients. arXiv:1705.08926 [cs.AI]

[22] Nicolas Fontbonne, Nicolas Maudet, and Nicolas Bredeche. 2022. Cooperative co-evolution and adaptive team composition for a multi-rover resource allocation problem. In *Genetic Programming: 25th European Conference, EuroGP 2022, Held as Part of EvoStar 2022, Madrid, Spain, April 20–22, 2022, Proceedings*. Springer, 179–193.

[23] Christopher Frye, Colin Rowat, and Ilya Feige. 2020. Asymmetric Shapley values: incorporating causal knowledge into model-agnostic explainability. *Advances in Neural Information Processing Systems* 33 (2020), 1229–1239.

[24] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E Taylor. 2019. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems* 33, 6 (2019), 750–797.

[25] Tom Heskes, Evi Sijben, Ioan Gabriel Bucur, and Tom Claassen. 2020. Causal shapley values: Exploiting causal knowledge to explain individual predictions of complex models. *Advances in neural information processing systems* 33 (2020), 4778–4789.

[26] Chris HolmesParker, Adrian K Agogino, and Kagan Tumer. 2013. Exploiting structure and utilizing agent-centric rewards to promote coordination in large multiagent systems.. In *AAMAS*. 1181–1182.

[27] Chris HolmesParker, Mathew E Taylor, Adrian K Agogino, and Kagan Tumer. 2014. Clean rewards to improve coordination by removing exploratory action noise. In *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, Vol. 3. IEEE, 127–134.

[28] Matt Knudson and Kagan Tumer. 2010. Coevolution of heterogeneous multi-robot teams. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. 127–134.

[29] Sean Luke, Charles Hohn, Jonathan Farris, Gary Jackson, and James Hendler. 1997. Co-evolving soccer softbot team coordination with genetic programming. In *Robot Soccer World Cup*. Springer, 398–411.

[30] Xiaoliang Ma, Xiaodong Li, Qingfu Zhang, Ke Tang, Zhengping Liang, Weixin Xie, and Zexuan Zhu. 2019. A Survey on Cooperative Co-Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation* 23, 3 (2019), 421–441. https://doi.org/10.1109/TEVC.2018.2868770

[31] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. 2019. MAVEN: Multi-Agent Variational Exploration. In *Advances in Neural Information Processing Systems*, Vol. 32. https://proceedings.neurips.cc/paper/2019/file/f816dc0acface7498e10496222e9db10-Paper.pdf

[32] 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 (2012), 1–31.

[33] Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. 1999. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML '99)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 278–287.

[34] Julien Perolat, Bilal Piot, and Olivier Pietquin. 2018. Actor-critic fictitious play in simultaneous move multistage games. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 919–928.

[35] Mitchell A Potter and Kenneth A De Jong. 1994. A cooperative coevolutionary approach to function optimization. In *International Conference on Parallel Problem Solving from Nature*. Springer, 249–257.

[36] Mitchell A Potter and Kenneth A De Jong. 2000. Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents. *Evolutionary Computation* 8 (2000), 1–29.

[37] Matt Quinn, Lincoln Smith, Giles Mayley, and Phil Husbands. 2002. Evolving formation movement for a homogeneous multi-robot system: Teamwork and role-allocation with real robots. *COGNITIVE SCIENCE RESEARCH PAPER-UNIVERSITY OF SUSSEX CSRP* (2002).

[38] Aida Rahmattalabi, Jen Jen Chung, Mitchell Colby, and Kagan Tumer. 2016. D++: Structural credit assignment in tightly coupled multiagent domains. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 4424–4429.

[39] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. 2020. Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. *Advances in neural information processing systems* 33 (2020), 10199–10210.

[40] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. 2020. Weighted QMIX: Expanding monotonic value function factorisation for deep multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*. https://proceedings.neurips.cc/paper/2020/hash/73a427badebe0e32caa2e1fc7530b7f3-Abstract.html

[41] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning*. http://proceedings.mlr.press/v80/

rashid18a.html

[42] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*. PMLR, 4295–4304.

[43] Lloyd S Shapley. 1953. A Value for n-person Games. *Annals of Mathematical Studies* 28 (1953), 307–317. https://doi.org/10.1515/9781400881970-018

[44] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. 2019. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International conference on machine learning*. PMLR, 5887–5896.

[45] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. 2019. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*.

[46] Peter Stone, Gal A. Kaminka, Sarit Kraus, and Jeffrey S. Rosenschein. 2010. Ad Hoc Autonomous Agent Teams: collaboration without pre-coordination. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence* (Atlanta, Georgia) *(AAAI'10)*. AAAI Press, 1504–1509.

[47] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. 2018. Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems* (Stockholm, Sweden) *(AAMAS '18)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2085–2087.

[48] Vito Trianni and Marco Dorigo. 2006. Self-organisation and communication in groups of simulated and physical robots. *Biological cybernetics* 95, 3 (2006), 213–231.

[49] Vito Trianni, Stefano Nolfi, and Marco Dorigo. 2008. Evolution, Self-organization and Swarm Robotics. In *Swarm Intell.* 163–191. https://doi.org/10.1007/978-3-540-74089-6{_}5

[50] Kagan Turner. 2006. Designing agent utilities for coordinated, scalable and robust multi-agent systems. In *Coordination of Large-Scale Multiagent Systems*. Springer, 173–188.

[51] Markus Waibel, Laurent Keller, and Dario Floreano. 2009. Genetic team composition and level of selection in the evolution of cooperation. *IEEE transactions on Evolutionary Computation* 13, 3 (2009), 648–660.

[52] Ermo Wei and Sean Luke. 2016. Lenient learning in independent-learner stochastic cooperative games. *The Journal of Machine Learning Research* (2016). https://www.jmlr.org/papers/v17/15-417.html

[53] Rudolf Paul Wiegand. 2003. *An Analysis of Cooperative Coevolutionary Algorithms*. Ph. D. Dissertation. George Mason University. http://l.academicdirect.org/Horticulture/GAs/Refs/PhD_Wiegand&Jong_2003.pdf

[54] D. Wolpert, Kagan Tumer, and K. Swanson. 2000. Optimal Wonderful Life Utility Functions in Multi-Agent Systems.

[55] David H. Wolpert and Kagan Tumer. 2008. *An introduction to collective intelligence*. Technical Report. NASA. 1–88 pages. arXiv:9908014v1 [cs]

[56] Nick Zerbel and Kagan Tumer. 2020. The Power of Suggestion. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020)*. 1602–1610.