Cooperative Co-Evolution and Adaptive Team Composition for a Multi-Rover Resource Allocation Problem

Nicolas Fontbonne, Nicolas Maudet, Nicolas Bredeche

Cooperative Co-Evolution and Adaptive Team Composition for a Multi-Rover Resource Allocation Problem

Nicolas Fontbonne[1], Nicolas Maudet[2], and Nicolas Bredeche[1]

[1] Sorbonne Université, CNRS, ISIR, F-75005 Paris, France
[2] Sorbonne Université, LIP-6, F-75005 Paris, France

Abstract. In this paper, we are interested in ad hoc autonomous agent team composition using cooperative co-evolutionary algorithms (CCEA). In order to accurately capture the individual contribution of team agents, we propose to limit the number of agents which are updated in-between team evaluations. However, this raises two important problems with respect to (1) the cost of accurately estimating the marginal contribution of agents with respect to the team learning speed and (2) completing tasks where improving team performance requires multiple agents to update their policies in a synchronized manner. We introduce a CCEA algorithm that is capable of learning how to update just the right amount of agents' policies for the task at hand. We use a variation of the El Farol Bar problem, formulated as a multi-robot resource selection problem, to provide an experimental validation of the algorithms proposed.

Keywords: ad hoc autonomous agent teams · multi-agent systems · marginal contribution · team composition · multi-robots · cooperative co-evolutionary evolutionary algorithms (CCEA) · evolutionary computation · evolutionary robotics

1 Introduction

When multiple individuals get together to solve a task, it is sometimes difficult to identify who is actually contributing, and who is not. This is especially problematic when the benefits are equally shared among individuals, including with free-riders who invest a minimal amount of effort. Nature abounds from such examples and various strategies evolved to mitigate the detrimental cost of free-riding, such as partner choice or reputation [11, 22].

The problem of identifying the marginal contribution of individuals has also been studied extensively in cooperative game theory [18]. However, exact methods such as computing the Shapley value [17] require strong assumptions (e.g. ability to replay coalitions) and unrealistic computation time, which have led to a flourishing research into the design of approximate methods [21, 23, 24]. The basic idea of such methods is to identify the individual's contribution by computing the difference between the group performance with and without this

very individual (e.g. by removing it or by replacing it with an individual with a default strategy).

In this research, we are interested in ad hoc autonomous agent teams where agents must act together without pre-coordination [19], which implies that the environment is non-stationary as all agents learn in parallel. This requires using methods that can only alter individuals' strategy, with neither a default strategy being known nor the possibility to remove temporarily one particular individual.

Such problem settings have been explored in evolutionary computation for multiagent systems, and notably with Cooperative Co-Evolutionary Algorithms (CCEA) which were first introduced in [13, 14] and largely explored since then (see [10] for a review of variants and applications). In particular, CCEA have been explored in setups involving multiple robotic agents in tasks such as exploration and foraging [7] and environment monitoring [15, 16, 25].

In this paper, we address the problem of isolating team members so as to identify their contribution within the collective. On the one hand, one could allow only a single agent to learn at a given time, making it possible to measure accurately its contribution as other agents' strategies would remain stationary. On the other hand, several (or all) of the agents' policies could be iterated at the same time, possibly speeding up learning thanks to parallelization.

Balancing between providing accurate estimation of an individual contribution and parallelization of learning actually depends on the context at hand. When rewards are sparse and depend on a single individual's behavioural innovation, it is preferable to bootstrap learning with large-scale exploration. However, whenever team performance increases it is more efficient to turn towards a more conservative search so as to retain previous improvements. Finally, one less obvious situation arises when the synergy between individuals is required to improve performance, for example when two robots are required to open a door, none of which would gain any benefit by acting alone.

The rest of the paper is organized as follows. Section 2 presents the problem of team composition in CCEA. Section 3 presents two CCEA algorithms that enable to tune the number of learning agents within one learning step. The two algorithms differ with respect to how the balance between contribution estimation accuracy and learning parallelization is set: fixed, or self-adaptive. Section 4 presents the problem used for evaluation, which is a modified version of the famous Bar El Farol problem [1] formulated as a multi-robot resource allocation problem where coordination is required (i.e. several resources must be harvested and harvesting is extremely beneficial it the optimal fixed number of robots is met). Results are presented in Section 5 for both the ad hoc version of and the self-adaptive versions of the algorithm.

2 Cooperative Coevolutionary and Team Composition

In its most simple setup, cooperative co-evolutionary algorithms (CCEA) rely on a collection of independent evolutionary algorithms, with each dedicated to optimizing the policy of one particular agent of the team [4,5]. Each independent

algorithm works to improve the performance of one agent's control parameters using an assessment of the team performance.

Each algorithm maintains a population of parameter sets, which define candidate policies for the agent this algorithm is in charge of. At each generation, performance assessment is computed for various teams. Then, each instance of the CCEA tries to improve its agent's performance by using classic evolutionary operators of selection, mutation and recombination.

The problem faced by CCEA is thus a black-box optimization problem, with the additional twist that evaluation is for the *whole* team, and optimization is performed at the individual level, thus implying a weak link between team evaluation and the actual individual behaviour. Defining $\boldsymbol{\theta}$ as the vector containing the parameters provided by each algorithm of the CCEA, F as the fitness function used to assess team performance and f the fitness value, we have:

$$F : \text{team parameters } \boldsymbol{\theta} \longrightarrow \text{fitness value } f$$

Figure 1 illustrates the learning loop of a simple multi-agent black-box optimization procedure for cooperative co-evolution. A given algorithm in charge of a particular agent i will provide policy parameters θ^i for this agent to be evaluated in a team. These parameters will be evaluated alongside parameters provided by the other agents. The team is then evaluated and a fitness value is returned.

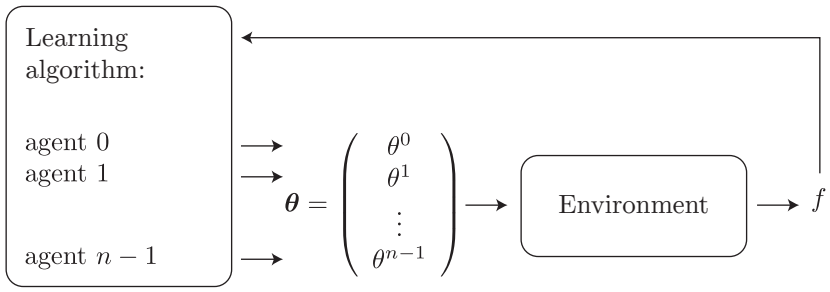


Fig. 1: The learning loop of interaction. All agents submit their own parameters independently for evaluation. They are evaluated at the same time on the environment or task. This return the fitness f of the whole team. Finally this feedback is used by all agent to update their parameters submission for next iteration.

After an evaluation, each agent has to evaluate if the new set of parameters proposed θ^i has contributed to the team fitness in a negative, positive or neutral manner. It is necessary to extract from the fitness $f = F(\boldsymbol{\theta})$, the part which depends only on the parameters of the agent i, $F^i(\theta^i)$.

From an agent's viewpoint, an increase in team performance may be due to others, and may even shadow a decrease in the very contribution the agent performs. In order to mitigate the intrinsically noisy fitness evaluation due to team heterogeneity (team composition changes over time), multiple evaluations of the same set of policy parameters will be performed for a given agent, so that different versions of θ^i can be ranked and further selected to create new candidate policies for the next generation. However, obtaining an exact assessment of one individual's contribution to the team remains elusive unless all other individuals follow static policies. Considering teams formed of n team agents, with each agent's evolutionary algorithm maintaining a population of p candidate policies, complexity would be $\mathcal{O}(p^n)$ at each generation.

In order to provide results in a reasonable time, CCEA generally relies on a partial evaluation of agents' policy contribution, by evaluating a subset of all possible team compositions at each iteration. Though such implementation breaks down complexity, CCEA algorithms have been shown to have a tendency to prematurely converge to stable states because of a deceptive fitness landscape created by the choice of collaborators for evaluation [6,12]. Several methods have been proposed to address these issues, including novelty-based rewards to escape local minima [7] or automatically merging populations when agents' behaviours are similar to address scalability issues [8, 9]. However, the contribution of one specific agent remains approximated rather than precisely measured.

3 Cooperative Co-Evolutionary Algorithms with Limited Team Composition Update

In the simplest case, the global fitness $F(\boldsymbol{\theta})$ is the sum of each agent's individual fitness $F^i(\theta^i)$ for the current team:

$$F(\boldsymbol{\theta}) = \sum_{i \in \text{agents}} F^i(\theta^i)$$

With θ^i the policy parameters for agent i, and $\boldsymbol{\theta} = (\theta^0, \ldots, \theta^i, \ldots, \theta^{N-1})$, i.e. the team policy parameters composed of the policies of N agents.

Whenever a single agent updates its policy parameters, the variation in overall fitness $F(\boldsymbol{\theta}_{t+1}) - F(\boldsymbol{\theta}_t)$ will be equal to the variation in the fitness due to the change in behaviour of the agent concerned δF^i. This can be written as:

$$\begin{aligned} F(\boldsymbol{\theta}_{t+1}) - F(\boldsymbol{\theta}_t) &= F^{-i}(\theta^{-i}_{t+1}) + F^i(\theta^i_{t+1}) - F^{-i}(\theta^{-i}_t) - F^i(\theta^i_t) \\ &= F^i(\theta^i_{t+1}) - F^i(\theta^i_t) \\ &= \delta F^i \end{aligned} \tag{1}$$

With $F^{-i}(\theta^{-i})$ the performance of all individuals minus the agent i, assuming θ^{-i} is stationary between t and $t+1$. Though an exact value for the contribution of agent i remains unavailable, δF^i gives a proxy which provides sufficient

information to measure both the direction and amplitude of the change due to agent i's new policy.

Assuming agents are independent, the above equation holds true and can be used as long as only one agent's policy is changed at a time. However, this assumption incurs two important limitations:

- The computational cost of iterating over a single agent's strategy at a time is high (see previous Section), and there is a trade-off between the quality of one agent's contribution estimation and the expected gain at the level of the team (e.g. whenever a single robot is needed to significantly improve team performance, trying with all robots is relevant);
- The task may require coupling between the agents' behaviour, and any team fitness improvements may require that *several* agents change their policy parameters simultaneously (e.g. moving a heavy object can only be done with two robots). If not, a CCEA can get stuck on a local minimum if we assume independence between agents and change only one agent at a time.

In order to address these issues and still retain the benefit of accurate estimation of the agents' contribution, we propose a CCEA algorithm where it is possible to modulate the number of agents that are updated in-between team evaluations. We use a collection of (1+1)-GA algorithm where each (1+1)-GA algorithm i provides the policy parameters θ^i for its corresponding agent i, and the whole team is evaluated using all agents with their policy parameters, i.e. $\boldsymbol{\theta} = (\theta^0, \ldots, \theta^{N-1})$.

Algorithm 1: CC-(1+1)-GA$_{k_{\text{fixed}}}$
Introducing k mutants per iteration

$k \leftarrow$ number of team members to be updated;
$N \leftarrow$ total number of agents;
$\boldsymbol{\theta}_{\text{parent}} \leftarrow$ parameters initialisation;
$f_{\text{parent}} \leftarrow F(\boldsymbol{\theta}_{\text{parent}})$;
for *gen* $<$ *nb max generation* **do**
  ID $\leftarrow$ randomly sample k agents;
  $\theta^{ID}_{\text{child}} \leftarrow \text{mutate}(\theta^{ID}_{\text{parent}})$;
  $f_{\text{child}} \leftarrow F(\theta^{ID}_{\text{child}}, \theta^{-ID}_{\text{parent}})$;
  if $f_{child} \geq f_{parent}$ **then**
    $\theta^{ID}_{\text{parent}} \leftarrow \theta^{ID}_{\text{child}}$;
    $f_{\text{parent}} \leftarrow f_{\text{child}}$;
  end
end

Algorithm 1 details the complete CCEA, which runs multiple instance of (1+1)-GA in parallel, which we will refer to as the CC-(1+1)-GA$_{k_{\text{fixed}}}$ algorithm from now on. Each (1+1)-GA algorithm maintains a population of two individuals [3], a parent θ^i_{parent} and a child θ^i_{child}. Both are candidate policy parameters

for agent i. The parent is replaced when the child fares better, and a new child is created by applying mutation on the new parent. Whenever a child fails to outperform its parent, it is replaced by a new child mutated from the current parent. The mutation operator depends on the problem (e.g. Gaussian mutation, bit-flip, uniform draw).

At each new iteration, k agents are drawn and randomly changed in the team, with $0 < k \leq N$. The k parameter is used to tune the amount of renewal k/N for the team composition in-between iterations of the CCEA algorithm. The k new team members are kept only if they provide an increase in the team fitness. Therefore, the challenge is to find the most efficient size k of the number of team agents to be modified at each CCEA steps. So far, k is fixed beforehand by the user, and may benefits from prior knowledge on the task regarding possible required coupling between agents, in terms of number of agents to change simultaneously to reach the global optimum in terms of team efficiency.

However, such prior knowledge may not be available and a relevant value of k not only depends on the problem (e.g. some problems may require coupling between agents, others may not), but also on the current state of the optimization (e.g. broad initial search steps vs. refined tuning near the optimal solution). In order to address this, we propose the CC-(1+1)-GA$_{k_{\text{adaptive}}}$, where the k parameter is learned during the course of evolution (see Algorithm 2). We propose to choose the number of team members to be updated by using the adversarial bandit learning algorithm EXP3 (Exponential-weight algorithm for Exploration and Exploitation [2]) that tracks the success rate of various possible values for k so far, which means both exploiting the current best value and exploring alternate values. The goal of the adaptation mechanism is to converge to the best possible value for k for the context at hand, i.e. the value that leads to the largest increase of fitness, whether through rare but important increases or through small but frequent increases.

As described in Algorithm 2, we define a set of J possible values for k $(k_0, ..., k_{J-1})$, each associated with a weight $W(k_j)$ monitoring the success rate of a particular k_j. Lines 10-13 of the algorithm detail which k_j is selected for a particular iteration. We compute the probability distribution of each k_j which depends on the weight $W(k_j)$ and the parameter γ of the algorithm. $\gamma \to 1$ favours exploration (i.e. the choice of k_j will be almost uniform). On the contrary, $\gamma \to 0$ favours exploitation, taken into account the importance of the weights $W(k_j)$. The fitness gain is normalized between $[0, 1]$ (Line 20) and then used to update the weight $W(k_j)$ (Line 21).

4 The Multi-Rover Resource Selection Problem

We define a problem that is a variation of the well-known El Farol bar problem [1, 24] where each individual must choose a day to go to the bar among M possible choices with the criterion of not being too numerous each days. In our setup, we consider the problem where N independent robots must spread over M resources, and where each resource has an optimal capacity c in terms of number

Algorithm 2: CC-(1+1)-GA$_{k_{\text{adaptive}}}$
Replacing a varying number of team agents per iteration

$K \leftarrow$ table of possible number of team members to update simultaneously ;
$W \leftarrow$ table of weights for each k;
$P \leftarrow$ table of probability for each k;
$k \leftarrow$ number of team members to be updated ;
$N \leftarrow$ total number of agents;
$\boldsymbol{\theta}_{\text{parent}} \leftarrow$ parameters initialization;
$\gamma \leftarrow$ real $\in [0, 1]$, parameter for the EXP3 algorithm ;
$f_{\text{parent}} \leftarrow F(\boldsymbol{\theta}_{\text{parent}})$;
for *gen* $<$ *nb max generation* **do**
  for $j = 1, \ldots, J$ **do**
    $P[j] \leftarrow (1-\gamma)\frac{W[j]}{\sum_{i=1}^{J} W[i]} + \frac{\gamma}{J}$
  end
  k$_j \leftarrow$ random draw in $K[]$ with probabilities $P[]$;
  ID $\leftarrow$ randomly sample k agents;
  $\theta^{ID}_{\text{child}} \leftarrow \text{mutate}(\theta^{ID}_{\text{parent}})$;
  $f_{\text{child}} \leftarrow F(\theta^{ID}_{\text{child}}, \theta^{-ID}_{\text{parent}})$;
  if $f_{child} \geq f_{parent}$ **then**
    $\theta^{ID}_{\text{parent}} \leftarrow \theta^{ID}_{\text{child}}$;
    $f_{\text{parent}} \leftarrow f_{\text{child}}$;
    $R \leftarrow \tanh\left(\frac{f_{\text{child}}}{f_{\text{parent}}}\right)$;
    $W[j] \leftarrow W[j] \exp\left(\frac{\gamma R}{P[j]J}\right)$;
  end
end

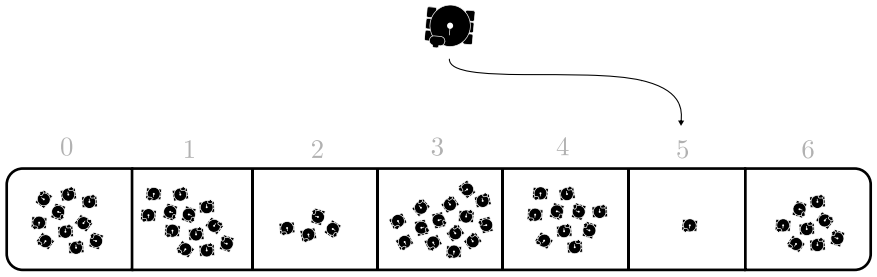
of robots necessary to optimally harvest it. This is illustrated in Figure 2, which provides an example where each robot chooses a resource.

Team performance f is obtained by adding each resource's satisfaction ϕ_c. For each resource, its satisfaction ϕ_c depends on the number of robots r who chose it. This satisfaction is described by the following equation:

$$\phi_c(r) = \begin{cases} Mr\exp(\frac{-r}{c}) & \text{if } r = c \\ r\exp(\frac{-r}{c}) & \text{else.} \end{cases} \tag{2}$$

where r represents the amount of robots on the resource, M the total number of resources, and c controls the optimal number of robots required for the resource.

The satisfaction function diverges from the original formulation of the El Farol Bar problem as the best team performance always implies that the number of robots per resource must be optimal (exactly c robots per resource), even if it implies some resources are left aside or only partially filled. The satisfaction boost for the $r = c$ case ensures that filling a maximum number of resources with the c robots is the optimal strategy. An example of such function with $c = 10$ is plotted on figure 3.



$$f = \phi_c(10) + \phi_c(12) + \phi_c(4) + \phi_c(14) + \phi_c(10) + \phi_c(2) + \phi_c(8)$$
$$= 66.483$$

Fig. 2: The $N = 60$ robots are represented here as little rovers that each must choose between $M = 7$ resources. Here the selected agent chooses resource 5. When all robots have made their choices, the satisfaction for each resources are computed and summed to obtain the fitness f of the team.

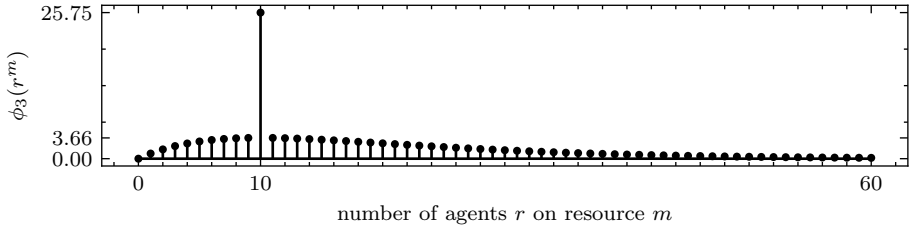


Fig. 3: Satisfaction function with $c = 10$ of a given resource, depending on the number of robots that picked it

The fitness of a team is then the sum of the satisfaction for all resources:

$$f = \sum_{m \in [0, M-1]} \phi_c(r^m)$$

Where r^m is the number of robots at resource m. The robots must coordinate to optimally fill a maximum number of resources.

To increase the value of this function, it is then necessary to move individuals from crowded resources to resources with fewer robots, up to the extent that resources with the exact number of robots are favoured.

The number of robots N, the number of resources M, and the optimal number of robots per resource c can be modified to change the structure of the problem. In the next section, different instances of this problem are used to study various properties of the Algorithms we proposed in the previous Section. In particular, it is possible to set up the problem so that either single or multiple changes in

the team composition may always yield too few or too many permutations in the team distribution over resources for team performance to increase.

5 Results

5.1 Experimental setting

We use the Multi-Rover Resource Selection problem, with different number of resources M, number of agents N, and optimal number of robots per resource c. The three setups used are:

- **Setup 1** with $N = 120$, $M = 300$, and $c = 30$. There are many resources, each requiring a large number of robots. The maximum performance could be reached by a team of exactly $M \times c = 300 \times 30 = 9000$ agents. Given the limited number of agents, they must spread over a few of the resources ($N/c = 120/30 = 4$ resources) so that the team reaches optimal performance;
- **Setup 2** with $N = 900$, $M = 300$, and $c = 3$. The number of robots involved makes it possible to reach the optimal team performance value for this setup ($N = M \times c$) if all agents are uniformly spread over the resources;
- **Setup 3** with $N = 60$, $M = 7$, and $c = 10$. There exists several configurations of pairing agents and resources which are local optima *and* cannot be escaped by updating only one agent in the team. Figure 4 gives an example of a sub-optimal configuration for which using $k = 1$ is detrimental. When the algorithm gets into such a configuration, all possible updates of a unique agent will decrease the team fitness. Escaping such a local optimum requires either exploring new configurations at the cost of a (hopefully temporary) decrease in team performance (see [7] for example using novelty search in CCEA, which is out-of-scope of the current paper) or modifying several agents at the same time (which is possible with $k > 1$).

In the following, we use both the CC-(1+1)-GA$_{k_{\text{fixed}}}$ algorithm with either $k = 1$, 10 or 30, and the CC-(1+1)-GA$_{k_{\text{adaptive}}}$ algorithm (using EXP3) for learning the value of k in $\{1, 10, 30\}$. All experiments are replicated 32 times. Mean and standard deviation for all algorithm variants are traced. Evaluations is used on the x-axis to provide a fair comparison in terms of computational effort.

5.2 Fixed vs. Adaptive Methods for Team Composition Update

Starting with the three variants of the CC-(1+1)-GA$_{k_{\text{fixed}}}$ Algorithm, we can observe different learning dynamics depending both on the value of k and the setup at hand.

In the first setup (Fig. 5(a)), we observe a clear benefit for using $k = 10$ and $k = 30$ during the first iterations. But this initial gain in performance does not allow it to converge faster when using $k = 1$. In particular, a value of $k = 30$ is extremely deleterious for the convergence as it fails to reach the optimum value within the allocated evaluation budget. This tendency is even more visible in the

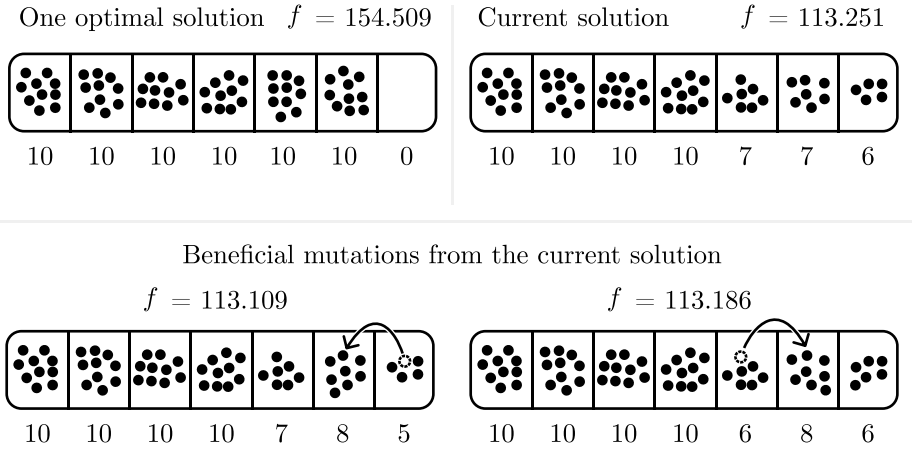


Fig. 4: The resource selection problem has local minimums that can't be escaped by mutating only one agent. In this example, 60 agents must spread on 7 resources by being 10 on 6 of them. In the state where 4 resources are selected by 10 agents, 2 are selected by 7, and 1 by 6, modifying the selection of one agent can only decrease the fitness of the system.

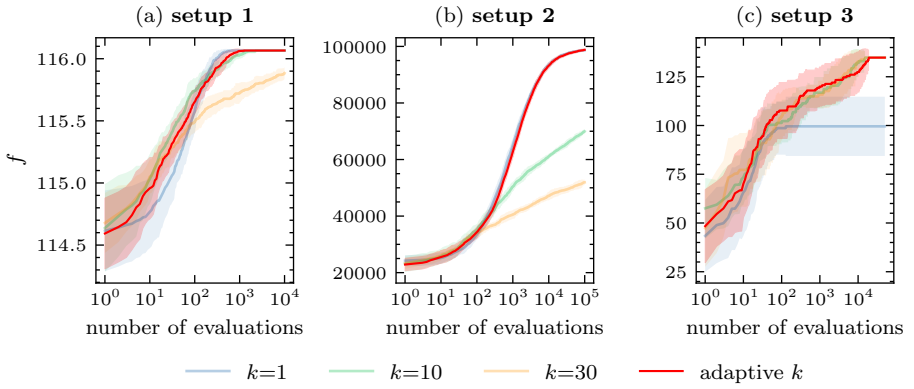


Fig. 5: Performance of the CC-(1+1)-GA$_{k_{\text{fixed}}}$ algorithm with either $k = 1$, $k = 10$ and $k = 30$, as well as CC-(1+1)-GA$_{k_{\text{adaptive}}}$ with $k \in \{1, 10, 30\}$. Performance f is plotted as mean (solid lines) and standard deviation for the three setups considered with respect to the number of fitness evaluations. Curves are plotted on a x-log scale. There are 32 replications for each algorithm and for each experiment.

second setup (Fig. 5(b)). Using larger values for k provides a slight advantage at the beginning but is quickly lost for both $k = 10$ and $k = 30$.

The outcome is rather different in the third setup (Fig. 5(c)) as using $k > 1$ allows to reach better performances and prevent the algorithm to get stuck on

a local optimum. Indeed, the algorithm becomes stalled when using $k = 1$, the structure of the problem making it impossible to improve team performance without considering coupled synergies when updating team members.

Figures 5(a) and (b) show that the CC-(1+1)-GA$_{k_{\text{adaptive}}}$ Algorithm follows the curves of the best performing algorithms using a fixed value of k. Figure 5(c) also shows that the adaptive algorithm is able to adapt to a situation where the CC-(1+1)-GA$_{k_{\text{fixed}}}$ algorithm would fail because of its fixed k value (here, using $k = 1$ withholds convergence to an optimal team composition). Overall, dynamically modulating the number of policies updated in the team composition always results in performance curves closely matching the best out of the algorithmic variants using a fixed value of k. This remains true even when the best variant with a fixed k value is outperformed by another variant with a different value of k, which confirms the relevance of the adaptive algorithm to act as an anytime learning algorithm. In other words, the CC-(1+1)-GA$_{k_{\text{adaptive}}}$ Algorithm presents the best choice when the problem and the evaluation budget are not known.

5.3 Dynamics of Adapting the Number of Team Agents to Update

We analyze how the CC-(1+1)-GA$_{k_{\text{adaptive}}}$ Algorithm is changing the value of k throughout evolution for the three setups at hand. Figure 6 represents the median value of k over the 32 repetitions for each of the three experimental setups. We observe that the algorithm switches between the different values for k, and follows different dynamics depending on the setup.

In the first setup, the method slightly favours $k = 1$ and $k = 10$ after a few iterations of exploration. This bias is consistent with the performances observed for k fixed, where the $k = 30$ version is less efficient (see Fig. 5). In the second setup, the value of k decreases during the learning process to stabilize at $k = 1$, allowing for some fine-tuning of team composition. The third setup displays

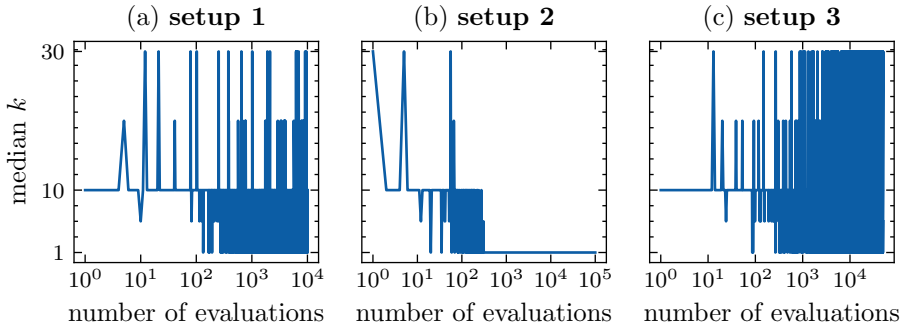


Fig. 6: Median value of k over the 32 repetitions for the first (left), second (middle) and third setups (right). Curves are plotted on a x-log scale.

somewhat different dynamics for the value of k, quickly switching from one value

to the other. The difference in performance between the different group sizes is not large enough to make a radical choice, and the method chooses k uniformly at each step without impacting the performances.

5.4 Sensitivity of meta-parameters

As described in Algorithm 2, CC-(1+1)-GA$_{k_{\text{adaptive}}}$ uses two meta-parameters, which are:

- $K = (k_0, ..., k_{J-1})$, the set of possible values for k;
- the egalitarianism factor γ that determines at each step whether k should be chosen at random (uniform sampling), or selected with respect to the weights of the k values, obtained from the cumulative fitness gain for each particular value of k. The value of γ balances between exploitation and exploration, and the EXP3 algorithm for multi-armed bandit problems has been extensively studied elsewhere [2, 20];

In the previous section, these meta-parameters were fixed as follow: the set of possible k was limited to $\{1, 10, 30\}$, the egalitarianism factor γ was set to 0.1.

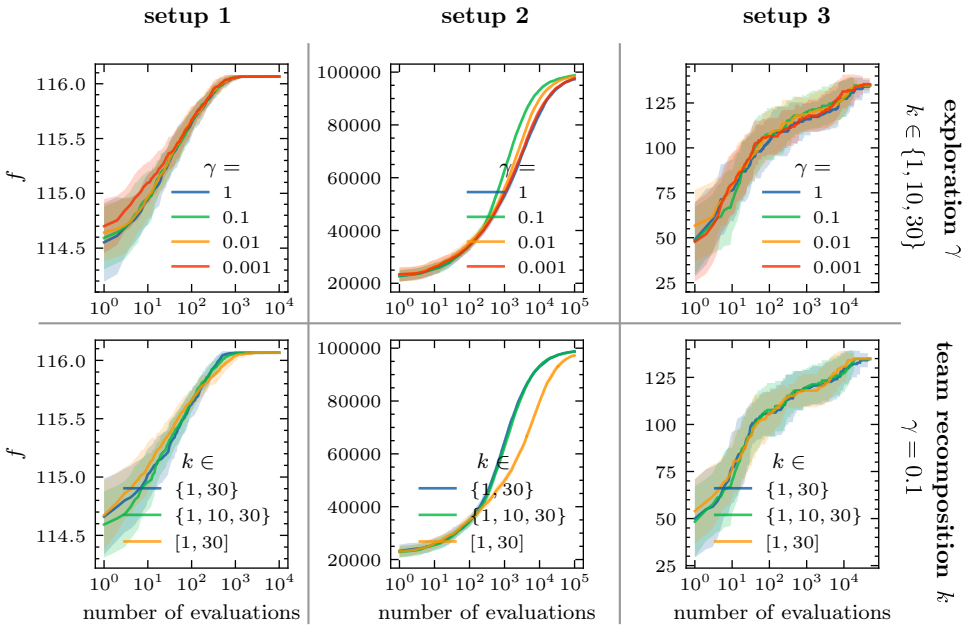


Fig. 7: Sensitivity of the algorithm to meta-parameters. Each column represents one of the different experimental setups. On the rows, one of the meta-parameter is fixed and the other one is varying. On the top, the set of k are fixed but γ varies. At the bottom, γ is fixed but the set of k varies.

Figure 7 shows the sensitivity of the algorithm with respect to the different meta-parameters. From top to bottom, the sensitivity to γ and the set of possible k. The general conclusion from this study is that the algorithm is robust and remains a relevant choice for anytime learning. Learning curves remain close to what has been shown previously, with some exceptions for extreme values. In particular, we can observe that:

- γ does not have a significant impact on the algorithm, provided that it is not too small nor too high to efficiently modulate the exploration and exploitation of k's
- the algorithm is somewhat also sensitive to the cardinal of the set of possible values for k. When there are too many possibilities to explore, the evaluation of each choice takes more time and is, therefore, less accurate if the context changes too fast. The effect of this exploration can especially be observed for the second setup where the algorithm is less accurate when the value for k can be chosen among 30 possible values ($k \in [1, 30]$).

6 Conclusion

In this paper, we present a cooperative co-evolutionary algorithm (CCEA) that implements a collection of (1+1)-GA algorithms, each endowed with the task to optimize the policy parameters of a specific agent while performance is assessed at the level of the team. Our algorithm acts on team composition by continuously updating a limited number of team agents, depending on the task at hand and the level of completion. Therefore, the main contribution of this paper is to describe an algorithm with a self-adapting team composition update mechanism used throughout learning.

We showed that modulating through time the number of new policies added to the current team makes it possible to provide efficient anytime learning, without requiring *a priori* knowledge on the problem to be solved. Moreover, we show that the algorithm can deal with problems where coupling between agents during learning is mandatory to improve team performance.

Experimental validation was conducted using a variant of the El Farol Bar problem, a famous problem in collective decision making, which was modified to capture a multi-agent resource selection problem. Our algorithm is indeed also relevant for multi-robotic setups, which have been recently studied using various CCEA algorithms [7–9, 15, 16, 25], and future works are currently being conducted in this direction.

Acknowledgements

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

References

1. W. Brian Arthur. Inductive reasoning and bounded rationality. *The American Economic Review*, 84(2):406–411, 1994.
2. Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.
3. Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies - a comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.
4. Kenneth A De Jong. Evolutionary computation: a unified approach. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, pages 185–199, 2016.
5. Agoston E Eiben and James E Smith. *Introduction to Evolutionary Computing*, volume 53. Springer, 2003.
6. Pablo Funes and Enrique Pujals. Intransitivity revisited coevolutionary dynamics of numbers games. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, GECCO '05, page 515–521, New York, NY, USA, 2005. Association for Computing Machinery.
7. Jorge Gomes, Pedro Mariano, and Anders Lyhne Christensen. Novelty-driven cooperative coevolution. *Evolutionary computation*, 25(2):275–307, 2017.
8. Jorge Gomes, Pedro Mariano, and Anders Lyhne Christensen. Dynamic team heterogeneity in cooperative coevolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 22(6):934–948, 2018.
9. Jorge Gomes, Pedro Mariano, and Anders Lyhne Christensen. Challenges in cooperative coevolution of physically heterogeneous robot teams. *Natural Computing*, 18(1):29–46, 2019.
10. Xiaoliang Ma, Xiaodong Li, Qingfu Zhang, Ke Tang, Zhengping Liang, Weixin Xie, and Zexuan Zhu. A survey on cooperative co-evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 23(3):421–441, 2019.
11. Ronald Noë and Peter Hammerstein. Biological markets: supply and demand determine the effect of partner choice in cooperation, mutualism and mating. *Behavioral ecology and sociobiology*, 35(1):1–11, 1994.
12. Liviu Panait. Theoretical convergence guarantees for cooperative coevolutionary algorithms. *Evol. Comput.*, 18(4):581–615, Dec 2010.
13. Mitchell A Potter and Kenneth A De Jong. A cooperative coevolutionary approach to function optimization. In *International Conference on Parallel Problem Solving from Nature*, pages 249–257. Springer, 1994.
14. Mitchell A Potter and Kenneth A De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8:1–29, 2000.
15. Aida Rahmattalabi, Jen Jen Chung, Mitchell Colby, and Kagan Tumer. D++: Structural credit assignment in tightly coupled multiagent domains. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4424–4429. IEEE, 2016.
16. Golden Rockefeller, Shauharda Khadka, and Kagan Tumer. Multi-level fitness critics for cooperative coevolution. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020)*, pages 1143–1151, May 9–13, 2020.
17. Lloyd S. Shapley. A value for n-person games. *Contributions to the Theory of Games II (Annals of Mathematics Studies 28)*, pages 307–317, 1953.

18. Yoav Shoham and Kevin Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.
19. Peter Stone, Gal A. Kaminka, Sarit Kraus, and Jeffrey S. Rosenschein. Ad hoc autonomous agent teams: collaboration without pre-coordination. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, page 1504–1509. AAAI Press, 2010.
20. Richard S Sutton and Andrew G Barto. *Reinforcement learning, Second Edition: An introduction*. MIT Press, 2018.
21. Kagan Tumer, Adrian K Agogino, and David H Wolpert. Learning sequences of actions in collectives of autonomous agents. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 378–385, 2002.
22. Stuart A West, Ashleigh S Griffin, and Andy Gardner. Social semantics: altruism, cooperation, mutualism, strong reciprocity and group selection. *Journal of evolutionary biology*, 20(2):415–32, 3 2007.
23. David H. Wolpert and Kagan Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001.
24. David H. Wolpert and Kagan Tumer. An introduction to collective intelligence. Technical report, NASA, 2008.
25. Nick Zerbel and Kagan Tumer. The power of suggestion. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020)*, pages 1602–1610, May 9–13, 2020.