Embodied Evolution versus Cooperative Coevolution in Multi-Robot Optimization: a practical comparison

P. Trueba
Integrated Group for Engineering Research
Universidade da Coruña
Spain
pedro.trueba@udc.es

A. Prieto
Integrated Group for Engineering Research
Universidade da Coruña
Spain
abprieto@udc.es

F. Bellas
Integrated Group for Engineering Research
Universidade da Coruña
Spain
francisco.bellas@udc.es

ABSTRACT

In this paper we show the potentiality of Embodied Evolution in the optimization of general multi-robot systems, as compared to state-of-the-art approaches based on Cooperative Coevolution. The comparison is carried out in a real application problem of coordinating a team of autonomous UAVs for surveillance which, in parallel, must optimize their location accuracy. The results show the advantages of using Embodied Evolution algorithms to notably reduce the number of evolution steps while maintaining the performance level.

CCS CONCEPTS

• **Computing methodologies** → Evolutionary robotics;
• **Computing methodologies** → Cooperation and coordination

KEYWORDS

Embodied Evolution, Cooperative Coevolution, Multi-robot systems

1 INTRODUCTION

We aim to study here the potentiality of Embodied Evolution (EE) [1] to solve general multi-robot problems performing an off-line, on-board evolution with heterogeneous individuals. Two approaches to EE have been considered: a distributed one, where each robot carries only one genotype, and an encapsulated one, where each robot carries a whole population. We have decided to compare them with the most similar approach that has provided successful results in multi-robot optimization, namely, Cooperative Coevolution Evolutionary Algorithms (CCEA) [2]. The CCEA background consists in decomposing the original high-dimensional problem into a set of lower-dimensional subcomponents, which are easier to solve. Typically, each subcomponent is evolved is a separated population. During the process, the only cooperation takes place in fitness evaluation, through an exchange of information between subcomponents. This type of algorithm supports heterogeneous controllers natively (each controller runs in an independent population) although it performs an off-line and off-board evolution.

GECCO '17 Companion, July 15-19, 2017, Berlin, Germany © 2017 Copyright is held by the owner/author(s). ACM ISBN 978-1-4503-4939-0/17/07. http://dx.doi.org/10.1145/3067695.3076083

Algorithm 1 DECC

```
variables ← random() { robots*genomeSize }
bestVariables ← variables
while cycles ≤ maxCycles do
  create groups of variables { random or fixed groups}
  initialize DE populations with groups
  for all population in DE populations do
    while evaluations < maxEvaluations do
      generate new individual in this population
      evaluate team with new individual
      f_new ← fitness {Global or private}
      if f_new > f_ancestor then
        Replace ancestor with new
        if globalFitness > bestGlobalFitness then
          bestGlobalFitness ← globalFitness
          bestParameters ← team
        end if
        variables ← bestParameters
      end if
    end while
  end for
end while
```

2 ALGORITHMS

The three main algorithms that have been used in the comparison are the Differential Evolution Cooperative Coevolution (DECC) [2] with three variations, an encapsulated Embodied Evolution approach with two variations [3], and finally, a distributed Embodied Evolution approach [4]. Their pseudocodes are shown in Algorithm 1, Algorithm 2 and Algorithm 3 correspondingly. For the DECC, the three variations that have been used are the

following: one uses fixed groups and global fitness (DECC-FG), the second uses fixed groups but private fitness (DECC-FG-private), and the last one uses random groups (DECC-RG). For the encapsulated EE algorithm, two variations have been considered, a synchronous one (the evaluations are fixed to a period of time and it runs off-line) and an asynchronous one (the controllers are evaluated until they run out of energy, and it runs on-line).

Algorithm 2 Encapsulated DE

```
for all robot in robot population do
  robot ← DE population
end for
while cycles ≤ maxCycles do
  choose team with tournament {in each robot}
  choose randomly active populations
  generate individuals in active populations
  modify team with generated
  evaluate team
  f_new ← globalFitness
  if f_new > f_ancestor then
    replace ancestor with new
  end if
  update fitness of team members
end while
```

Algorithm 3 Canonical dEE algorithm

```
for all robot in population do
  g⃗ ← generate random genotype {Initialize}
end for
loop
  for all robot in population do
    Fitness assignment {Interaction with the task}
    if random < P_mating then
      candidates ← findCandidatesForReproduction
      s⃗ ← selectCandidates {P_elegibility}
      offspring ← recombination(g⃗, s⃗)
    end if
    if random < P_replacement then
      g⃗ ← offspring
      reset robot state
    end if
  end for
end loop
```

3 EXPERIMENTAL SETUP

To compare the algorithms, we have designed a multi-robot experiment based on a collective surveillance task that must be carried out through a group of UAVs with realistic degradation on the accuracy of the self-location. The experiment details can be seen in [4]. Fig. 1 contains the average exploration level obtained for the six algorithms throughout iterations. Each algorithm was executed 20 times, each of them implying about 4 hours in an i7 4770s processor, and the resulting exploration level was averaged between them. The most significant result that can be extracted from this figure is the time required to reach stable solutions, which is much lower in the on-line algorithms (asynchronous encapsulated and canonical dEE), as expected. They improve their exploration level quickly, and in around 5 million iterations, which constitutes the 12,5% of the total iterations, they reach a stable range that continues until the final iteration. In contrast, the off-line algorithms (encapsulated and DECC variants) require almost half of the total iterations to reach a stable solution. The encapsulated algorithm provides the best results in the off-line approaches, while the DECC-FG obtains the best performance of the DECC variants.

We have noticed during the realization of the tests that there is a clear relation between the performance and the specialization of the population in multi-robot optimization problems. Thus, the on-line algorithms, and specially the dEE, exploit their simplicity in this domain in order to specialize individuals in simple tasks so evolution is much faster than in off-line approaches, where the several evolution processes that are executed in parallel makes the optimization slower and more complex.

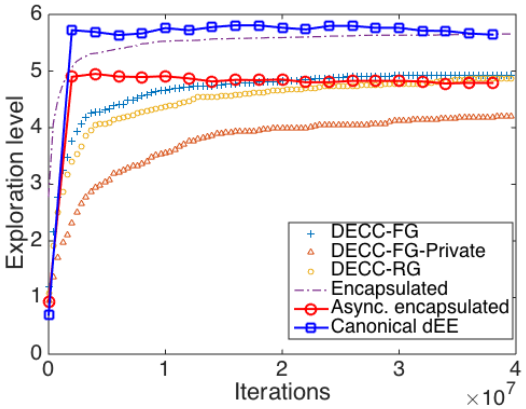

Figure 1: Performance comparison of the algorithms

5 CONCLUSIONS

The main conclusion of this work is that Embodied Evolution is a highly promising evolutionary approach for on-line multi-robot optimization. It is able to obtain specialized individuals that can solve complex tasks without resorting to high time-consuming evaluations.

ACKNOWLEDGMENTS

This work has been partially funded by the EU's H2020 research and innovation programme under grant agreement No 640891 (DREAM project) and by the Xunta de Galicia and European Regional Development Funds under grants GRC 2013-050 and redTEIC network (R2014/037).

REFERENCES

[1] R. Watson, S. Ficici, J. Pollack. 2002. Embodied evolution: Distributing an evolutionary algorithm in a population of robots, *Robotics and Autonomous Systems*, 39(1), Elsevier, 1-18

[2] Z. Yang, K. Tang and Xin Yao. 2008. Multilevel cooperative coevolution for large scale optimization. *IEEE Conf on Evolutionary Computation*, 1663-1670

[3] E. Haasdijk, A.E. Eiben, G. Karafotias. 2010. On-line evolution of robot controllers by an encapsulated evolution strategy. *Proc. IEEE CEC2010*, IEEE Press, 1-7

[4] A. Prieto, F. Bellas, P. Trueba and R.J. Duro. 2016. Real-time optimization of dynamic problems through distributed Embodied Evolution, *Integrated Computer-Aided Engineering*, vol 23, 237 - 253