Improving performance in distributed embodied evolution: Distributed Differential Embodied Evolution

Pedro Trueba[1] and Abraham Prieto[1]

[1]Integrated Group for Engineering Research. Universidad de la Coruna
abraham.prieto@udc.es

Abstract

The field of Embodied Evolution has been strongly developing during the last ten years by more than doubling the yearly number of contributions since 2008 (Bredeche et al., 2018). Many different scenarios and tasks have been addressed and some works have already focus in formalizing and standardizing the paradigm. There hasn't been a lot of effort, however, towards comparing and improving the performance of the algorithms, which is essential to increase the complexity of the experimental setups and therefore the applicability of the technique. This paper extends the work started in (Trueba, 2017) to compare different variations of EE algorithms with the incorporation of a Differential Evolution based distributed EE algorithm.

Introduction

Since its creation in 2002, Embodied Evolution has been applied mainly to evolve (optimize) multi-robot coordination for collectively solving a task.

As an evolutionary approach, EE has potential to provide successful results in complex dynamic scenarios, due to the capability of dealing with high-dimensional non-linear search spaces. However, this potential has still to be exploited to improve its performance and extend its application, which involves comparing and testing EE algorithms to keep improving them.

In this line, a preliminary work (Schut et al., 2009) compared the performance of a traditional implementation of dEE against two variations of a genetic algorithm. Later, (Trueba, 2017) presented a thorough comparison among several EE algorithms and Cooperative Coevolution Evolutionary Algorithms (CCEAs). The comparison run on a complex collective surveillance and location task by a group of UAVs. The experimental tests compared three EE algorithms (one distributed and two encapsulated versions) and three CCEAs variations. As a result of the comparison, the EE algorithm (CDEE) (Prieto et al., 2015) produced the highest performance with the lower number of iterations, significantly outperforming any CCEA implementation.

This papers extends the work of (Trueba, 2017) by presenting a combination of the CDEE structure with some operators used in the differential algorithm (Brest et al., 2007). The new algorithm, *Distributed Differential Embodied Evolution* (DDEE), is also tested on the collective surveillance and location task used on the comparisons in (Trueba, 2017).

Distributed Differential Embodied Evolution

DDEE follows the structure of an EE algorithm and mimics the mating and replacement activation processes of CDEE. The operation of these events is stochastic and task and scenario independent and then they are triggered based on probability distributions rather than on the occurrence of specific events on the scenario. As part of the replacement operator the algorithm runs a genetic recombination which is traditionally a very simple and naive version of crossover and/or mutation that generates a new genotype based on one or two existing ones.

Algorithm 1: Distributed Diff. Embodied Evolution

```
initialize p robots
while evolution is active do
    simulate one step
    map individual robot fitness→f_r
    for robot ← 1 to p do
        if age_r == T_maturity then
            if f_r < f_parent,r then
                active ← parent
            end
        end
        if age_r > T_maturity then
            if rand(0,1) < P_matingandReplacement then
                active ← recombination.DE operator
                f_parent,r ← f_r
                age ← 0
            end
        end
        age_r++
    end
end
```

DDEE aims to improve the genetic recombination process by adapting the more complex genetic recombination implemented in DE to generate a new genotype. As in DE, the new genotype is subject to a evaluation period which takes place

during the maturity time already implemented in CDEE. Algorithms 1 and 2 shows the pseudocode of DDEE including the implementation of the alternative recombination operator.

Algorithm 2: Distributed Diff. Embodied Evolution

```
Operator recombination.DE() is
    pbest ← 0.15;
    create a pool with affine genotypes within the
      population→ P_elig;
    best ← random individual from top pbest from P_elig;
    random1 ← random individual from P_elig;
    random2 ← random individual from P_elig;
    randomGen ← randi[0, genotype.length];
    for i ← 0 to genotype.length do
        if rand(0, 1) < crossRate or i == randomGen
          then
            active.g[i] ← active.g[i] + F*(best.g[i] -
              active.g[i]) + F*(random1.g[i] - random2.g[i]);
        else
            active.g[i] ← current.genotype[i];
        end
        return active
    end
end
```

The concept of *eligible population* is used to promote the creation of species. Only a subset of the population which fulfills an affinity criterion, based on genotypic distance, is selected as candidates. This criterion has been used to define a variation of DDEE which includes this tendency to favor genetically close individuals: DDEE with kin selection.

Results

The experimental setup uses the aforementioned surveillance and location task where each robot (see (Trueba, 2017) for more details) evolves a neural controller which decides what to do and where to go in each time step. The scenario has been tested for three different configurations regarding the degradation rate of the accuracy of the location of the robots. The higher the degradation the more complex the task and the higher the dependence on a good task decomposition. On this setup three algorithms were tested: CDEE, the recently defined DDEE, and its variation including kin selection, DDEE with kin selection. Fig. 1 shows that for low and moderate degradation, DDEE clearly outperforms CDEE and therefore any result obtained in (Trueba, 2017) in terms of final performance. For the high degradation setup, DDEE fails dramatically since it is not capable of exploiting the creation of species adequately but the incorporation of kin selection changes this outcome. CDEE with k.s. appears to be less stable and varies more from run to run but it still outperforms CDEE in low and moderate degradation. In the case of high degradation, although still very oscillating, many runs produce a very efficient behavior which, if the algorithm can be adjusted to eliminate, or reduce, the bad runs we will get a highly improved DDEE. Summarizing, DDEE and DDEE with kin selection have produced the best results obtained so far in this complex task after a exhaustive comparison with several algorithms, and it appears to be much room for improvement.

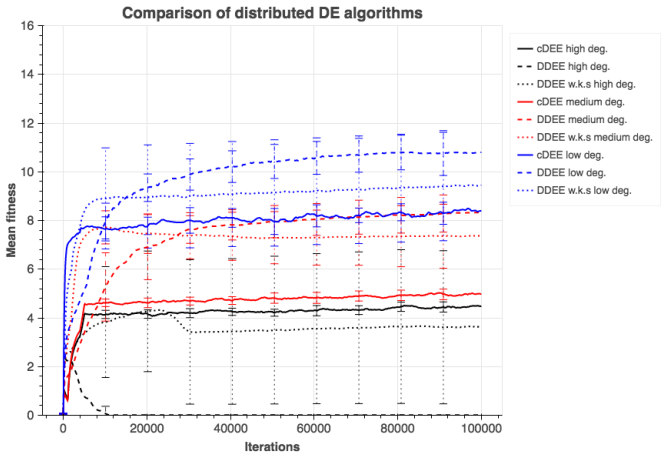

Figure 1: Comparison of distributed DE algorithms for different configurations of the surveillance and location task. Each line represents 25 runs of 10000 time steps

Acknowledgements

This work has been partially funded by the EU's H2020 research programme under grant No 640891 (DREAM) as well as by the Xunta de Galicia and the European Regional Development Funds under grants ED431C 2017/12 and redTEIC network (ED341D R2016/012).

References

Bredeche, N., Haasdijk, E., and Prieto, A. (2018). Embodied evolution in collective robotics: A review. *Frontiers in Robotics and AI*, 5:12.

Brest, J., Greiner, S., Bokovi, B., Mernik, M., and Zumer, V. (2007). Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. 10:646 – 657.

Prieto, A., Bellas, F., Trueba, P., and Duro, R. J. (2015). Towards the standardization of distributed Embodied Evolution. *Information Sciences*, 312:55–77.

Schut, M. C., Haasdijk, E., and Prieto, A. (2009). Is situated evolution an alternative for classical evolution? In *2009 IEEE Congress on Evolutionary Computation, CEC 2009*, pages 2971–2976.

Trueba, P. (2017). Embodied Evolution versus Cooperative Coevolution in Multi- Robot Optimization : a practical comparison. pages 79–80.