HIT-EE: a Novel Embodied Evolutionary Algorithm for Low Cost Swarm Robotics

Nicolas Bredeche
Sorbonne Université, CNRS, Institut des Systèmes Intelligents et de Robotique, ISIR, F-75005 Paris, France
nicolas.bredeche@sorbonne-universite.fr

ABSTRACT

This paper presents a novel distributed on-line evolutionary learning algorithm for swarm robotics that can cope with very limited hardware, as expected from using a swarm of low cost robots. The algorithm is able to deal with hardware constraints over the communication bandwidth by sharing only a limited amount of information, using a recombination operator inspired from bacterial conjugation. Using a classic foraging task, we show that the algorithm converges towards stable and efficient solutions even though, as expected, it converges slower when the bandwidth is limited. However, we also show that the proposed algorithm performs a trade-off between convergence speed and absolute performance that depends on the amount of bandwidth available. The recombination operator yields better performance if communication is limited, as recombination makes the most from the genetic material already present in the population. In other words, quality outweighs convergence speed if the bandwidth is limited.

KEYWORDS

Embodied evolution, low cost robots, swarm robotics

ACM Reference Format:
Nicolas Bredeche. 2019. HIT-EE: a Novel Embodied Evolutionary Algorithm for Low Cost Swarm Robotics. In *Genetic and Evolutionary Computation Conference Companion (GECCO '19 Companion), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3319619.3321928

1 ALGORITHM

The HIT-EE algorithm (short for "Horizontal Information Transfer for Embodied Evolution") differs from other embodied evolution algorithms [1] by introducing two new ideas. Firstly, a *maturation time* is set that allows any new individual to remain protected while its quality with respect to a reward function is estimated. Combined with a sliding window, it allows to maintain an estimation of the average reward of a particular behaviour that can be compared to other.

Secondly, we introduce a recombination operator, termed the *transfer* operator, that can send only a subset of one's genome. Whenever the sender is deemed better than the receiver, the bits of genome sent will overwrite the corresponding bits in the receiver. The amount of information sent will then depends on the available bandwidth. Depending on the $transferRate$ (tsf for short), the receiver's genome may be slightly altered ($tsf = \epsilon$, very low bandwidth), recombined (moderate bandwidth) or completely overwritten ($tsf = 1.0$, large bandwidth).

Our algorithm is loosely inspired from bacterial conjugation, which is a mechanism for horizontal gene transfer used by bacteria. Similar to bacteria where genes are sent from a living donor to a receiver though physical contact, two robots within communication range may send a subset of their control parameters (cf. also [2] for a classic GA implementation of the same idea).

Algorithm 1 The HIT-EE algorithm

```
1:  mutationRate = m // m is a value between 0.0 and 1.0
2:  transferRate = t // t is a value between 0.0 and 1.0
3:  maturationDelay = d // d is an integer value (strictly positive
4:  genome.initialize() // e.g. random values
5:  reward = 0 // similar to "fitness value"
6:  age = 0
7:  newGenome = False
8:  while forever do
9:      move() // execute the agent's controller for one step.
10:     reward = updateReward() // e.g.: use a sliding window of size t
11:     if age > maturationDelay then
12:         age = age + 1
13:         broadcast(genome,transferRate,reward) // (subset of) genome
14:         incomingPackets = listen() // returns received packets since last iteration
15:         for p in incomingPackets do
16:             if p.reward >= reward then
17:                 copy p.genomeBits to genome // i.e. conjugation
18:                 mutate genome using mutationRate
19:                 newGenome = True
20:             end if
21:         end for
22:         if newGenome == True then
23:             reward = 0, age = 0, newGenome = False
24:         end if
25:     end if
26: end while
```

2 EXPERIMENTS

The obvious question is how the HIT-EE algorithm can cope when communication is constrained. We use a foraging task where the reward (i.e. the fitness function) is given by the number of items captured by a robot. Each robot is controlled by a simple Perceptron which weights are evolved. We perform a first set of experiments with two different values for the transfer operator. We experiment with transfer rates where either half ($tsf = 0.5$) or most ($tsf = 0.9$) of the genome (ie. the Perceptron's weights) can be transferred (cf. Table 1 for other settings). Mutation rate is arbitrarily set to zero: while this may be sub-optimal in terms of exploration, this allows a clear understanding of the effect of the transfer operator.

Results for the two different settings are presented in Figures 2 and 3. The $tsf = 0.50$ and $tsf = 0.90$ variants both converge to a

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
GECCO '19 Companion, July 13–17, 2019, Prague, Czech Republic
© 2019 Copyright held by the owner/author(s). Publication rights licensed to the Association for Computing Machinery.
ACM ISBN 978-1-4503-6748-6/19/07…$15.00
https://doi.org/10.1145/3319619.3321928

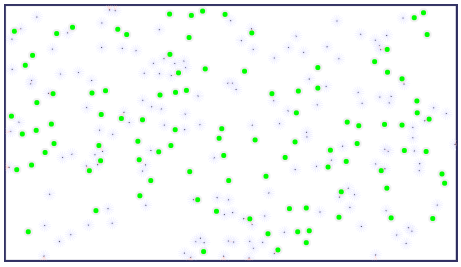

Figure 1: Arena with 150 **robots (small dots) and** 100 **items (big dots). The fitness function for each robot counts the number of items captured (i.e. foraging task).**

Table 1: Control parameters

Parameter	Value	Comments
General parameters		
Population size	150	
Number of items	100	
Arena size	1400x800	
Robot size	5x5	
Sensor&communication range	16	
iterations	800400	*max.2000 gens*
replications	64	
Controller and encoding		
Sensory inputs	163	*16 sensors*
Motor outputs	2	*left and right*
Genome size	326	
HIT-EE parameters		
maturation time	400	
sliding window	400	
transfer	0.5 **or** 0.9	
mutation	0.0	*no mutation here*

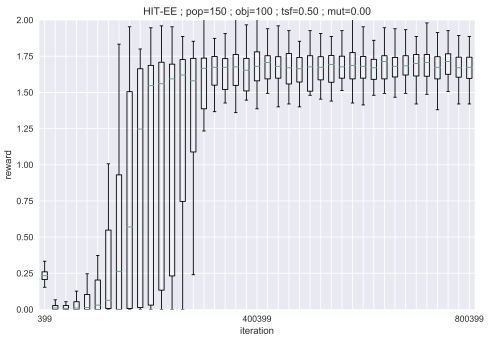

Figure 2: using a 50% genome transfer per encounter. Mean=1.6845614385965**, standard deviation**=0.11280099972487

stable state. The $tsf = 0.90$ variant converges (much) faster, but the $tsf = 0.50$ variant performs better. This is confirmed by performing a two-tailed Mann-Whitney test (*p-value* = .00078).

Changing the transfer rate makes it possible to operate a trade-off between speed of convergence and exploitation of the initial

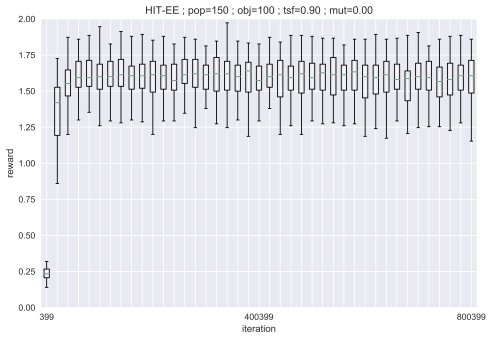

Figure 3: using a 90% genome transfer per encounter. Mean=1.5773333272727**, standard deviation**=0.21354009301578

gene pool. As expected, a lower value (such as $tsf = 0.50$) leads to slower convergence, but it also makes it possible to explore the possible benefits of recombination between control parameters that are originally spread over the population.

This short paper shows that the HIT-EE algorithm can be used when the communication bandwidth is limited, by choosing a suitable amount of information to be transferred. Moreover, it reveals that while hardware limitations may slow down convergence, it can also benefit the quality of learning.

SUPPLEMENTARY INFORMATION

Sources and data for this paper are available[1].

The initial reward value (iteration no.399) does not represent the efficiency of the robots' behaviour at this point but is due to an initial large amount of available items to forage (once an item is harvested, there is a delay of 50 iterations before relocation and reactivation).

Computing means, standard deviations, and p-values is performed using runs that converged to behaviours with non-zero rewards. This is the case for most of the runs presented here, but approx. 7% of the runs actually converged to robots *avoiding* items and maximizing only the spread of their own genome. For further discussion, we refer the reader to previous papers in embodied evolution about the interplay between task-driven and environment-driven selection pressures (see [1] for a review). P-values are computed using the last 1200 iterations of each run.

The relatively large genome size is due to redundant and/or useless sensory information fed to the controller (which is itself a simple Perceptron, with no hidden layer). Such a large input space is useful to avoid the initialisation of a "lucky" candidate at generation 0.

ACKNOWLEDGMENTS

This work was supported by the Agence Nationale pour la Recherche under Grant ANR-18-CE33-0006.

REFERENCES

[1] Nicolas Bredeche, Evert Haasdijk, and Abraham Prieto. 2018. Embodied Evolution in Collective Robotics: A Review. *Frontiers in Robotics and AI* 5 (2018), 12. https://doi.org/10.3389/frobt.2018.00012

[2] Inman Harvey. 2009. The microbial genetic algorithm. In *European Conference on Artificial Life*. Springer, 126–133.

[1] http://pages.isir.upmc.fr/~bredeche/Experiments/GECCO2019_HIT_code.zip