Environment-driven Embodied Evolution in a Population of Autonomous Agents

Nicolas Bredeche, Jean-Marc Montanier

Environment-driven Embodied Evolution in a Population of Autonomous Agents

Nicolas Bredeche and Jean-Marc Montanier

TAO - Univ. Paris-Sud, INRIA, CNRS - F-91405 Orsay, France firstname.name@lri.fr

Abstract. This paper is concerned with a fixed-size population of autonomous agents facing unknown, possibly changing, environments. The motivation is to design an embodied evolutionary algorithm that can cope with the implicit fitness function hidden in the environment so as to provide adaptation in the long run at the level of the population. The proposed algorithm, termed mEDEA, is shown to be both efficient in unknown environment and robust with regards to abrupt, unpredicted, and possibly lethal changes in the environment.

1 Introduction

In this paper, we are interested in a fixed-size population of autonomous physical agents using local communication (e.g. autonomous robots), facing unknown and/or dynamic environments. This class of problems typically arises when the environment remains unknown to the human designer until the population of agents is actually made operational in the real situation [4], or whenever the environment is known to change during operation, without any indication on *when* and *how* these changes will impact survival strategies.

The challenge is to design a distributed online optimization algorithm targeting agent self-adaptation in the long term, that is being able to successfully manage an implicit pressure resulting from environmental particularities *and* algorithmic constraint with regards to the optimization process. Embodied Evolution (EE), as proposed initially in [5], addresses part of this question as it focuses on algorithms for evolutionary optimization of agent behaviors in an on-line, possibly decentralized manner. On the other hand, EE requires an objective function designed from the supervisor, which is unavailable by definition in the problem setting addressed here.

While concepts and methods from EE may be relevant, we can only assume that maximizing the integrity of the agent population as well as maintaining a communication network for exchanging genome are the basic requirements in the present context. To this end, we propose a distributed algorithm for environment-driven self-adaptation based on evolutionary operators that takes into account selection pressure from the environment. The basic assumption behind this algorithm is to consider the strategies as the atomic elements and the population of agents as a distributed resource onto which strategies compete with one another. This approach is better illustrated using the Selfish Gene metaphor [3]: one specific strategy (or set of parameters, or genome) is ”successful” if it manages to spread over the population, which implicitly requires to both minimize risk for its ”vehicles” (ie. the autonomous agents) and maximizing the number of mating opportunities, though the two may be contradictory.

The general motivation behind the work presented here is to study and provide general evolutionary adaptation algorithms that can utlimately be implemented on real robotic hardware. To this end, the main contribution of this paper is to introduce a new and simple distributed evolutionary adaptation algorithm for use in population of autonomous agents. While it is yet to be applied in a real robotic setup, this paper focuses on an indepth experimental analysis of the robustness of the algorithm with regards to unknown, and changing, environments, under realistic constraints (fixed number of agents, limited sensors and actuators, etc.).

2 Environment-driven Distributed Evolutionary Adaptation

As stated in the introduction, our objective is to design a distributed online evolutionary algorithm for a fixed population of autonomous physical agents (e.g. autonomous robots), whenever the human engineer fails to provide a proper description of an objective function. As a consequence, the key issue behind Environment-driven Distributed Evolutionary Adaptation (EDEA) relies in the *implicit* nature of the fitness function. However, this implicit fitness may be seen as the result of two possibly conflicting motivations:

- **extrinsic motivation**: agent must cope with environmental constraints in order to maximize survival, which results solely from the interaction between the agent and the environment around (possibly including other agents as well). ;
- **intrinsic motivation**: set of parameters (ie. ”genomes”) must spread across the population to survive, which is imposed by the algorithmic nature of the evolutionary process. Therefore, genomes are naturally biased towards producing efficient *mating* behaviors as the larger the number of agents met, the greater the opportunity to survive.

The level of correlation between these two motivations does impact problem complexity to a significant amount: high correlation implies that the two motivations may be treated as one while low correlation implies conflicting objectives. An efficient EDEA algorithm should indeed address this trade-off between extrinsic and intrinsic motivations as the ideal optimal genome should reach the point of equilibrium where genome spread is maximum (e.g. looking for mating opportunities) with regards to survival efficiency (e.g. ensuring energetic autonomy).

These assumptions have also been extensively studied in the field of open-ended artificial evolution, with an emphasis on computational model of evolutionary dynamics [1], including a particular focus on the effect of the environment over the evolutionary adaptation process [6]. However, their application within Embodied Evolution is still an open issue as there is a major difference concerning the working hypothesis as EE is concerned with a fixed number of physically grounded agents that are usually ment to target real world environment (e.g. obstacles, energy constraints, etc.).

2.1 MEDEA: a Minimal EDEA algorithm

Based on these considerations, we introduce the MEDEA algorithm (”minimal EDEA”), described in table 1. This algorithm describes how evolution is handled on a

Algorithm 1 The MEDEA algorithm

```
genome.randomInitialize()
while forever do
  if genome.notEmpty() then
    agent.load(genome)
  end if
  for iteration = 0 to lifetime do
    if agent.energy > 0 and genome.notEmpty() then
      agent.move()
      broadcast(genome)
    end if
  end for
  genome.empty()
  if genomeList.size > 0 then
    genome = applyVariation(select_random(genomeList))
  end if
  genomeList.empty()
end while
```

local basis and is copied as is within all agents in the population. This algorithm works along with a communication routine, which purpose is to receive incoming genomes and store these in the Imported Genome List for later use.

At a given moment, a given agent is driven by a control architecture which parameters are extracted from an "active" genome, which remains unchanged for a generation. This genome is continuously broadcasted to all agents within (a limited) communication range. This algorithm actually implements several simple, but crucial, features, that can be interpreted from the viewpoint of a traditional evolutionary algorithm structure:

Selection operator: the selection operator is limited to simple random sampling among the list of imported genomes, ie. no selection pressure *on a local individual basis*. However, cumulated local random selection ultimately favor the most widespread genomes *on a global population basis* as such genomes have greater probability to be randomly picked *on average*. In fact, the larger the population and mating opportunities, the more accurate the selection pressure at the level of the population.

Variation operator: the variation operator is assumed to be rather conservative to ensure a continuity during the course of evolution. Generating altered copies of a genome only make sense if there is some continuity in the genome lineage: if no variation is performed, the algorithm shall simply converge on average towards the best existing genome initially in the population. In the following, we assume a gaussian random mutation operator, inspired from Evolution Strategies [2], which conservative behavior can be easily tuned through a σ parameter.

Replacement operator: lastly, replacement of the current active genome to control a given agent is performed by (1) local deletion of the active genome at the end of one generation and (2) randomly selecting a new active genome among the imported genome list (cf. selection operator). On a population level, this implies that surviving genomes are likely to be correlated with efficient mating strategies as a given genome may only survive through (altered) copies of itself in the long run.

The positive or negative impact of environmental variability on genome performance is smoothed by the very definition of the variation operator as newly created genomes are always more or less closely related to their parent. As a consequence, each genome results from a large number of parallel evaluations, both on the spatial scale as closely related copies sharing the same ancester may evaluated in a population, and on the temporal scale, as one genome is also strongly related to its ancestors. Hence, a single genome may get lucky once in a while, but it's highly unlikely that a "family" of closely related genomes manage to survive in the population if there are more efficient competitors.

3 Experimental Setting

This section provides a description of the experimental setting used hereafter as well as implementation details. The motivation is here to design a setting such that it is possible to address several issues regarding evaluation and validation of the proposed algorithm. In particular, robustness of MEDEA with regards to environmental pressure and to sudden environmental changes shall be studied.

3.1 The problem: surviving in a dynamic unknown environment

Figure 1 shows the environment used for the experiment: a 2D arena with obstacles, possibly containing food items. The figure also illustrates 100 autonomous mobile agents loosely inspired from the ePuck mobile robot specifications. This environment is used to define two different experimental setups, described hereafter:

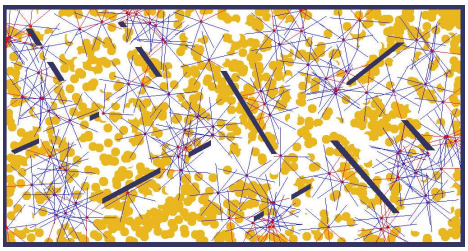

Fig. 1. Snapshot from the simulator with 100 agents. Yellow: food items. Red: agents, modeled after an e-puck robot. Blue: range of proximity sensors (communication range is half this range).

1. **the "free-ride" setup**
   - *Description*: a population of autonomous mobile agents is immersed within an environment with few obstacles. As a consequence, an agent dies only if it was not able to mate with at least one other agent - ie. the current genome is lost for sure as it does not get a chance to survive within any other agents.
   - *Motivation*: this setup makes it possible to evaluate the mechanisms of the MEDEA algorithm as environmental pressure should be limited.

2. **the "energy" setup**
   - *Description*: A set of energy resources ("food items") is spread all over the environment, which can be harvested by the agents. Agents are endowed with an energy level, which depends on harvested food items and power consumption. If the energy level reaches 0, agent dies and genome information is lost. Moreover, harvested food items only "grow" back after a given number of iterations.
   - *Motivation*: In this setup, genomes also compete for agent resources but have to deal with environmental pressure as maximizing mating encounters may not be fully compatible with energy self-sustainability.

The full experimental setup considers starting with the "free-ride" setup, and then suddenly switching to the "energy" setup after a pre-defined fixed number of generations. In the meantime, agents are of course unaware of such a change in the environment and keep on running the same unchanged MEDEA algorithm.

3.2 Representation / Encoding the problem

Specifications for the autonomous agents are inspired from traditional robotic setup, with 8 proximity sensors dispatched all around the agent body and 2 motor outputs (translational and rotational speeds). Moreover, three additional sensory inputs are considered: the angle and direction towards the nearest food item and the current energy level (which is set to a fixed value in the "free-ride" setup). Note that these additional sensor values are useless in the first setup, and may even be considered as distractors. Each agent is controlled by a multiple layer perceptron (MLP) with 5 hidden neurons, which means a total of 72 weights[1].

The variation operator is a gaussian mutation with one σ parameter: a small (resp. large) σ tends to produce similar (resp. different) offsprings. This is indeed a well known scheme from Evolution Strategy where continuous values are solely mutated using a parameterized gaussian mutation, where the σ parameter may be either fixed, updated according to pre-defined heuristics or evolved as part of the genome. In the scope of this work, we rely on self-adaptive mutation, where σ is part of the genome [2] (ie. the full genome contains 73 real values).

The current implementation of the σ update rule is achieved by introducing α, a σ update value, which is used to either decrease ($\sigma_{new} = \sigma_{old} * (1 - \alpha)$) or increase ($\sigma_{new} = \sigma_{old} * (1 + \alpha)$) the value of σ whenever a genome is transmitted. The idea is that whenever an agent broadcast its own genome, probabilities of transmitting an increased or decreased σ values are equivalent. In the following, α is a predefined value set prior to the experiment so that it is possible to switch from the larger σ value to the smaller in a minimum of approx. 20 iterations.

3.3 Experimental settings

The whole experiment lasts for 150 generations, switching from the "free-ride" setup to the "energy" setup at generation 75. During the course of evolution, some agents

[1] 11 input neurons ; 5 hidden neurons ; 2 output neurons ; 1 bias neuron. The bias neuron value is fixed to 1.0 and projects onto all hidden and output neurons.

may come to a halt either because they did not meet any other agents, thus failing to import a new genome for use in the next generation, or because they ran out of energy during the "energy" setup (each agent can store a maximum of 400 energy units and consumes 1 unit/step, one generation lasts 400 steps). In the "free-ride" setup, the agents remain still (or "inactive", ie. without genome), waiting for new genomes imported from "active" agents that eventually come into contact. In the "energy" setup though, agents requires an external "human" intervention for refilling energy. Revived agents remains inactive, but are refilled with enough energy to wait until the end of the current generation, listening for new imported genome that may be used for the next generation. While the reviving procedure makes it possible to avoid progressive extinction in the second setup, extinction is nevertheless possible whenever all agents in the population fail to meet any other agents during one generation, whatever the cause (bad exploratory or harvesting strategies). Also, monitoring the number of active agents in a population provides a reliable indicator of the performance of the algorithm as external intervention may be viewed as one important cost to minimize (e.g. minimizing human intervention in a robotic setup). Detailed parameters used for the experiment presented in the next section are given in figure 2.

Parameter	**Value**
arena width and length	$1024 * 530$ inches
"free-ride" setup duration	75 generations
"energy" setup duration	75 generations
lifetime (ie. generation duration)	400 steps per generation
population size	100 agents
proximity sensor range	64 inches
radio broadcast signal	32 inches
agent rotational velocity	30deg/step
agent translational velocity	2 inches/step
genome length	79 real values (78 MLP weights + σ)
variation operator	gaussian mutation with σ parameter
$\sigma_{minValue}$	0.01
$\sigma_{maxValue}$	0.5
$\sigma_{initialValue}$	0.1
α (ie. σ update parameter)	0.35

"energy" setup only:	
food items	2000
food item diameter	10 inches
food item regrow delay	btw 400 and 4000 steps (see text)
energy per food item	100 energy units
agent energy consumption	1 energy unit per step
agent maximum energy level	400 energy units
agent initial energy level	400 energy units

Fig. 2. Experimental settings (details)

In order to provide a challenging environment, the "energy" setup is designed so that the number of food items in the environment depends on the actual number of active agents. Indeed, a food item grows back whenever harvested, but only after some delay. If the number of active agents is less than half the population size, then $delay_{regrow}$ is set to 400 steps. However, if the number of active agents is between 50 and 100, then the delay linearly increases from 400 steps (fast regrowing) to 4000 steps (slow regrowing, aggressive environment).In the particular setup described here, switching from a possibly efficient population of 100 agents from the "free-ride" setup to the "energy" setup will have a possibly disastrous impact as the number of agents at the beginning of the second setup implies longer regrow delays.

At this point, it is important to note that the motivation behind this experimental setup is both to stress the population for further analysis as well as providing a flexible and challenging experimental settings that could be re-use to evaluate further version and variation over the algorithm presented here. To this end, the source code and pa-

rameters for all experiments presented in the following is available on-line in the Evolutionary Robotics Database[2]. On a practical viewpoint, one experiment takes approx. 15 minutes to be performed using one core of a 2.4GHz Intel Core 2 Duo computer. The home-brew agent simulator is programmed in C++ and features basic robotic-inspired agent dynamics with collision.

4 Results and Analysis

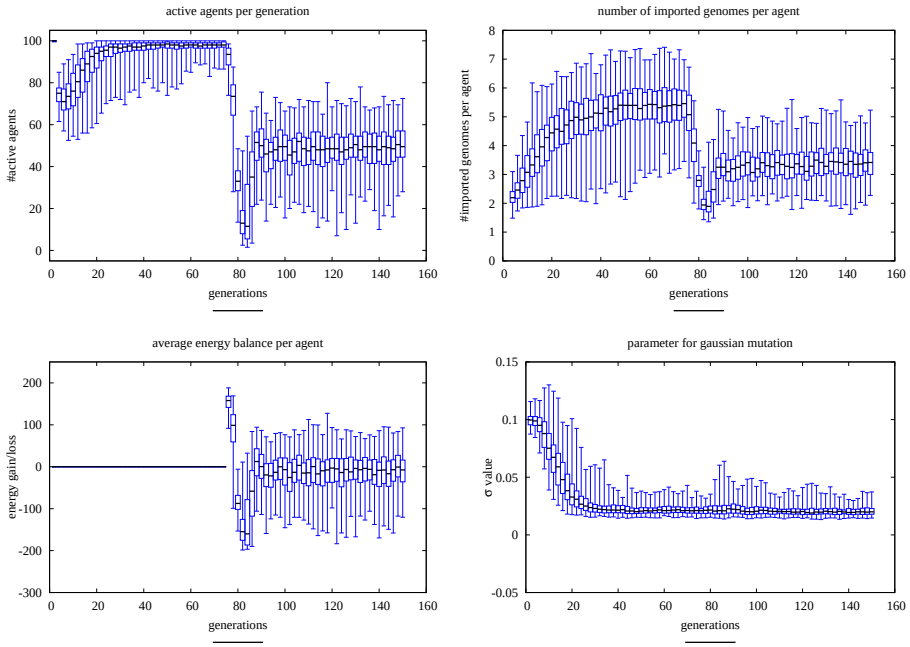

Fig. 3. Experimental results - the experiment starts with the ”free-ride” setup from generation 0 to generation 75, then it switches to the ”energy” setup until generation 150.

The lack of explicit objective function makes it difficult to compare performance during the course of evolution. However, the number of active agents and the average number of imported genomes per generation give a good hint on how the algorithm performs: it can be safely assumed that ”efficient” genomes lead to few deaths and many mating opportunities. Moreover, the number of food items harvested gives some indication in the ”energy” setup. The four graphs in figure 3 give an a synthetic view of the results over 100 independent runs obtained with MEDEA on the experimental scenario described in the previous section. These graphs compile the average values of

[2] Evolutionary Robotics Database: http://www.isir.fr/evorob_db/

selected parameters, or ”indicators”, over generations: number of active agents, average number of imported genomes per agent, average energy balance per agent, and average σ mutation parameter values.

In both setups, all indicators rise to reach stable average values and some conclusions can be drawn: firstly, both setups show an increase in both mating opportunities (number of imported genomes) and survival rate (number of active agents). Secondly, switching setups initially leads to a drop for both indicators, followed by a quick recovery through evolutionary adaptation. This interpretation is reinforced by the increasing value of the energy balance which is a key element for the second setup. A notable remark is that the energy balance stays around zero, which is sufficient to guarantee agent survival. This is not a surprise as over-maximizing harvesting may imply a cost with regards to looking for mating partners. On the other hand, the gaussian mutation parameter is not really influenced by the change of environmental setups (except from a slight increase in maximum values). While the results may vary among runs, with a great difference between minimal and maximal values for each indicator, values between the upper and lower quartiles are remarkably close given the noise inherent to this kind of experiment. Indeed, complete extinctions were even observed after switching to the ”energy” setup in three of the 100 runs (results not shown).

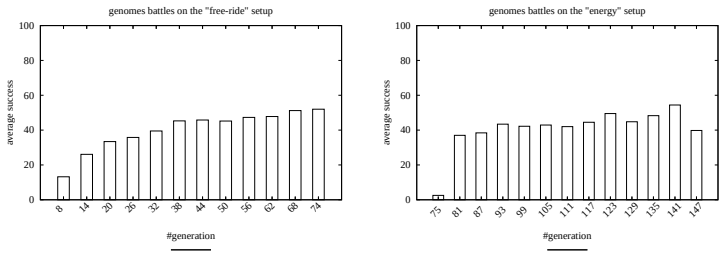

Fig. 4. Genomes battles on both setups (*left*: free-ride setup ; *right*: energy setup). Average success scores for each generation, both histograms are the results of 1000+ battles. See text for details.

However, we must be very cautious with the interpretation of these results. For example, the quality of the equilibrium between maximizing mating opportunity and coping with environmental constraints (ie. avoiding walls, avoiding collisions with other agents, harvesting) is difficult to estimate as such equilibrium may (and appears to) imply sub-optimal values for both related indicators. As a matter of fact, all interpretations provided so far rely on the assumption that values monitored in the experiment are actually correlated with genome survival. In order to support this assumption, a new experimental setup is defined from the results obtained so far: the *post-mortem* battle experiment (or battle experiment, for short). The battle experiment is loosely inspired from competitive coevolution, where each individual competes against a hall-of-fame of the best individuals from every past generations [7], so as to estimate the fitness rank of one individual within all possible (or at least, all available) situations. For the current experiment, one ”battle” is achieved by randomly picking up 10 generations from the

same setup, and extracting one random genome from each of these generations. Then, each genome is copied into ten different agents, resulting in 100 agents that are immersed in the same setup they were evolving in. Variation is turned off, and evolution is re-launched. After 100 generations of random selection and replacement, the number of copies for each genome is accounted for and used to compute a ”survival score”. As an example, one genome gets a maximal score if it succeeds in taking control of all the agents. Average results over 1000 battles are given in figure 4. In both setups, genomes from later generations display better survival capability than early genomes. Moreover, battles on the second setup show a very fast recovery after environmental change, possibly to stable, but limited, strategies as the number of active agents is far from the maximum. Also, these histograms lack the misleading artifacts observed in previous graphs regarding the early generations in both setups: genomes from generation 0 do not benefit from uniform sampling of starting location and genomes from generation 75 do not benefit from high initial energy level.

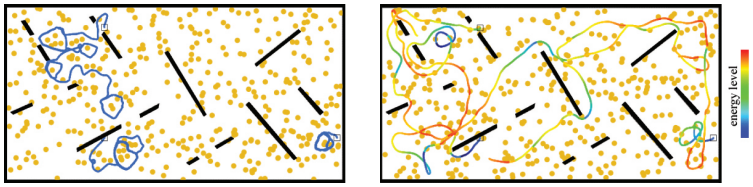

Fig. 5. Typical examples of agent behavioral traces in both setups (*left*: free-ride setup ; *right*: energy setup). In the energy setup, colored traces show the current energy level of the agent (see legend). In both setups, the square symbol shows the agent starting points.

The efficiency of the algorithm is also confirmed by looking at the resulting behavioral strategies. Two examples of behaviors are shown in figure 5, resulting from agent driven by genomes obtained in the late generations of both setups. In the ”free-ride” setup, genomes tend to lead to rather conservative behaviors, with obstacle avoidance, but with limited exploratory behavior. On the other hand, genomes from the later generations of the ”energy” setup show a different behavioral pattern, favoring long distance travel and few circling around, which is an efficient strategy to avoid being stuck in an exhausted area. Moreover, a closer look at trajectories (including, but not limited to what is shown here) show that agents acquired the ability to drive towards a detected food items under certain conditions, such as favoring safe areas with few obstacles whenever energy level is low.

5 Conclusions and Perspectives

This paper provides a proof-of-concept for the viability of environment-driven distributed evolutionary adaptation in a population of autonomous agents. We have presented the mEDEA algorithm, a particular flavor of Embodied Evolution, tailored to address evolutionary adaptation with implicit fitness. The proposed algorithm was evaluated with regards to our initial motivation and proven to be (1) efficient with regards

to providing distributed evolutionary adaptation in unknown environment and (2) robust with regards to unpredicted changes in the environment. Moreover, this algorithm is light-weight and with low complexity, which makes it possible to consider future implementation within hardware/software setups with limited computational capability such as robotic agents.

Many perspectives may be considered from this point, and some are already under investigation. Firstly, the class of problems addressed here is also relevant in the field of embodied Evolutionary Robotics. Secondly, surviving in aggressive environments requires more complex behavioral patterns (e.g. coordination). Sharing similar concerns, previous works in collective intelligence and reinforcement learning have already stressed the issue of the price of anarchy [8], ie. the cost of efficient selfish behavior with regards to population global welfare. Then again, solving this issue remains an open problem, especially if there is no explicit objective function to decompose. Thirdly, the work presented here targets, and is limited to, providing reliable survival strategies. However, our motivational claim is that one should first aim at a reliable, surviving population before even considering to optimize a pre-defined objective function as objective function with extensive goal description often lead to deceiving fitness landscape, and poor results. Of course, this remains to be extensively studied and demonstrated.

Acknowledgements

This work was made possible by the European Union FET Proactive Intiative: Pervasive Adaptation funding the Symbrion project under grant agreement 216342.

References

1. Mark A. Bedau, John S. McCaskill, Norman H. Packard, Steen Rasmussen, Chris Adami, David G. Green, Takashi Ikegami, Kunihiko Kaneko, and Thomas S. Ray. Open problems in artificial life. *Artificial Life*, 6:363–376, 2000.
2. Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies – A comprehensive introduction. *Natural Computing*, 1:3–52, 2002.
3. Richard Dawkins. *The Selfish Gene*. Oxford University Press, 1976.
4. Guy Baele et al. Open-ended on-board evolutionary robotics for robot swarms. In *Proceedings of the IEEE Conference on Evolutionary Computation (CEC 2009)*, 2009.
5. S. Ficici, R. Watson, and J. Pollack. Embodied evolution: A response to challenges in evolutionary robotics. In J. L. Wyatt and J. Demiris, editors, *Proceedings of the Eighth European Workshop on Learning Robots*, pages 14–22, 1999.
6. Jeffrey A. Fletcher, Mark A. Bedau, and Martin Zwick. Effect of environmental structure on evolutionary adaptation. In *Artificial Life VI*, pages 189–198. MIT Press, 1998.
7. Christopher Rosin and Richard Belew. New methods for competitive coevolution. *Evolutionary Computation*, 1996.
8. D.H. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001.