Incremental evolution of animats' behaviors as a multi-objective optimization

Jean-Baptiste Mouret[1] and Stéphane Doncieux[1]

ISIR, Université Pierre et Marie Curie-Paris6, CNRS FRE2507, Paris, F-75005
jean-baptiste.mouret@lip6.fr, stephane.doncieux@upmc.fr

Abstract. Evolutionary algorithms have been successfully used to create controllers for many animats. However, intuitive fitness functions like the survival time of the animat, often do not lead to interesting results because of the bootstrap problem, arguably one of the main challenges in evolutionary robotics: if all the individuals perform equally poorly, the evolutionary process cannot start. To overcome this problem, many authors defined ordered sub-tasks to bootstrap the process, leading to an incremental evolution scheme. Published methods require a deep knowledge of the underlying structure of the analyzed task, which is often not available to the experimenter. In this paper, we propose a new incremental scheme based on multi-objective evolution. This process is able to automatically switch between each sub-task resolution and does not require to order them. The proposed method has been successfully tested on the evolution of a neuro-controller for a complex-light seeking simulated robot, involving 8 sub-tasks.

1 Introduction

One of the main goal of animat research is to design controllers so that the "intelligence" of the animat emerges from its interactions with its environment, avoiding the biases inducted by human analysis and, hopefully, leading to solutions as smart as the ones found by nature [1]. Evolutionary methods have been widely used to that aim (see [2] for an overview), succeeding in creating controllers for complex behaviors like the combination of locomotion, obstacle avoidance and gradient following in an insect-like robot [3].

However, many early researches in evolutionary robotics showed that rewarding the efficiency of the final behavior was not enough to obtain working controllers because of the bootstrap problem: if the objective is so hard that all the individuals in the first generations perform equally poorly, evolution cannot start and no functioning controllers can be found. For instance, it has been found hard to evolve a light-seeking behavior in a complex arena without having before evolved an obstacle-avoidance reflex [4].

In consequence, researchers emphasized the need to help the evolutionary process by adding some kind of reward for intermediate steps. More precisely, we will call an *incremental task* a task such that:

- no fitness gradient exists on most part of the search space, i.e. the fitness of all random individuals for the goal task is minimum;
- the experimenter is able to define some simpler sub-tasks whose completion is useful to the completion of the whole task.

Evolutionary processes specially designed for such tasks will be referred as *incremental evolution*. A lot of experiments have been reported to be more efficiently solved with such approaches than with direct evolution [5, 6, 3, 7].

Despite these successes, published incremental evolution rely on some assumptions that require an accurate knowledge of the problem to solve and can lead the evolutionary algorithm to a local extremum. Most of them, for instance, require to precisely order the different sub-tasks or to determine when to switch from a sub-task to another one. These biases prevent published methods to scale-up well to more complex or more open tasks; noticeably, most of them have been tried with only two or three sub-tasks.

In this article, we propose an original view of incremental evolution based on multi-objective optimization [8], each sub-task defining an objective. Recent multi-objective evolutionary algorithms can then be employed to efficiently solve the bootstrap problem while automatically switching between sub-tasks and not requiring to order the sub-tasks or even to exploit all of them before solving the goal task. This work is a milestone on the way towards the generation of complex behaviors by evolutionary algorithms; nevertheless, to focus ourselves on the evolutionary process, the benchmark task and the genotype have been chosen as simple as possible. To that aim, we evolved feed-forward neuro-controllers for a simulated robot that has to avoid obstacles and turn on a target light, a task that requires to successively switch on at least four different lights in a predefined order. The task has been designed to be extensible and easily modifiable.

This paper is organized in four parts. In the first part, we briefly review the literature concerning incremental evolution. Next, we introduce a way to view incremental evolution as multi-objective optimization. In the third part, we show how multi-objective evolutionary algorithms can be used on incremental tasks. The fourth part describes a simulated robotics experiment using the presented method. A short discussion concludes this paper.

2 Previous work

Papers dealing with incremental evolution can be categorized in four main approaches: staged evolution, environmental complexification, fitness shaping and behavioral decomposition.

In staged evolution, the main task is split into ordered sub-tasks, each with a corresponding fitness function. Individuals are first selected using the first fitness. Once a convergence criterion has been reached for a stage, for instance when a “good enough” fitness is obtained by the best individual or when the best fitness doesn’t change for some generations, the experimenter switches to the next sub-task. This technique has first been successfully employed by Harvey [5] to evolve a vision-based target location task. Three stages were used, from locating a large immobile target to tracking a moving smaller one. Many other examples of staged evolution can be found in the literature [3, 9–11, 7, 12].

Environmental complexification allows to more continuously change the complexity of the task through the tuning of dedicated parameters. Gomez et al. [6] thus worked on a prey-capture task parameterized with the prey speed and the delay before starting the pursuit. Ten ordered sub-tasks were thus defined by specific values of these parameters and their use proved to lead to more efficient solutions.

In behavioral decomposition, the robot controller is divided into sub-controllers, each one evolved separately to solve a sub-task. The sub-controllers are then combined together using a second evolutionary process. This approach led to controllers for a composite behavior where a Lego robot had to push a movable light into a designated goal area [13]. A variation of this concept has been recently used to evolve a position controller for an autonomous helicopter [14].

Fitness shaping is sometimes used to help bootstrapping an evolutionary process, though it is not often seen as an incremental evolution scheme. The fitness is defined as an aggregation (a weighted sum or a product) of different evaluation criteria in order to create a followable fitness gradient. Using this idea, Nolfi et al. [15] successfully evolved a feed-forward neuro-controller for a robot to locate, recognize and grasp a target object. The fitness was increased if the individual was close to the target object, if the target was in front of the robot, if the robot tried to pick-up the object, if the robot had the object in the gripper and if the robot released the object outside the area.

3 Incremental evolution as a multi-objective optimization

While the four previously described approaches allowed the evolution of substantially complex behaviors, they imply an accurate knowledge of the global task and of the defined sub-tasks. Staged evolution methods, for instance, imply a manual switch between the sub-tasks. Such an approach can be represented in the fitness space by a “L” shape (in 2D, figure 1 (a)): the first sub-task T_s is optimized first, while the second T_g is not and, after the switch, the performance on the T_s remains constant – it is at least expected not to decrease either because a good behavior on T_s is required to solve T_g or because T_s controller is frozen – while the performance on T_g starts to climb. This method requires to choose when to trigger the switch: if too early, the first sub-task won’t be completely solved and if too late, generated solutions might be over-specialized and make optimizing T_g even more difficult (figure 1 (b)). More generally, viewing incremental tasks using this figure suggests that a trade-off can exist between T_g and T_s. For instance, a controller could use all the available ressources to obtain an optimal fitness with regard to T_s, leaving nothing left to solve the goal-task.

The order of the sub-tasks is another important bias of current incremental methods. Let us consider the sub-tasks a typical rodent should manage to survive:

– walking;

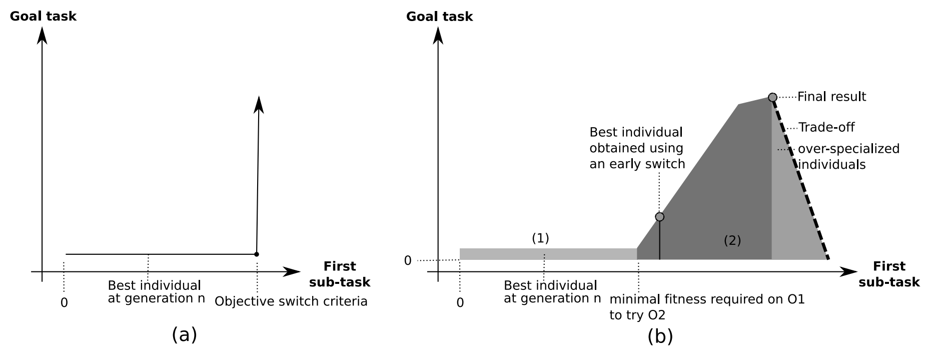

Goal task
First sub-task
0
Best individual at generation n
Objective switch criteria
(a)
Goal task
Final result
Trade-off
over-specialized individuals
Best individual obtained using an early switch
(1)
(2)
0
First sub-task
0
Best individual at generation n
minimal fitness required on O1 to try O2
(b)

Fig. 1. (a) Staged evolution: a goal task depends on the minimal completion of a sub-task. The first sub-task is optimized and, at a given generation, the experimenter changes the fitness to optimize the final task. (b) Typical fitness space for an incremental task with one sub-task. (1) the first sub-task is optimized; (2) the minimal fitness required to try the goal task is reached, therefore it is possible to have a non-minimum fitness for the goal task; a trade-off can exists between the goal-task and the sub-task, leading to over-specialized individuals.

– running;
– avoiding obstacles;
– avoiding predators, implying being able to perceive them and to know an escape strategy;
– eating, implying to be able to explore its environment and locate the food;
– resting, when necessary;
– finding a sexual partner;
– procreating (final goal).

How to successfully learn all these sub-tasks? All of them share complex dependencies but organizing them in a single incremental learning scheme is a challenging problem which could have no solution, even for this simplified rodent.

Consequently, a good incremental algorithm should explore all the sub-tasks and the goal task at the same time, continuing to improve on each tasks at each moment and trying to switch automatically to other tasks when possible. This is exactly what multi-objective evolutionary algorithms do if each task is viewed as an objective.

4 Multi-objective evolutionary algorithms

Recent research in evolutionary computation proposed numerous algorithms to simultaneously optimize several objectives [8]; most of them rely on the concept of domination and generate the so-called Pareto Front (figure 2):

Definition 1. *A solution* $\mathbf{x}^{(1)}$ *is said to dominate another solution* $\mathbf{x}^{(2)}$*, if both conditions 1 and 2 are true:*

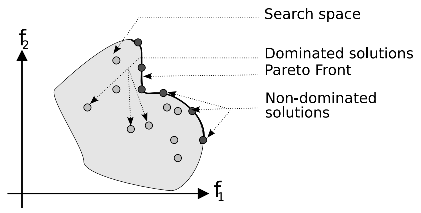


Fig. 2. Dominated and non-dominated solutions for a problem for which the two objectives f1 and f2 must be maximized.

1. *the solution* $\mathbf{x}^{(1)}$ *is not worse than* $\mathbf{x}^{(2)}$ *with respect to all objectives;*
2. *the solution* $\mathbf{x}^{(1)}$ *is strictly better than* $\mathbf{x}^{(2)}$ *with respect to at least one objective.*

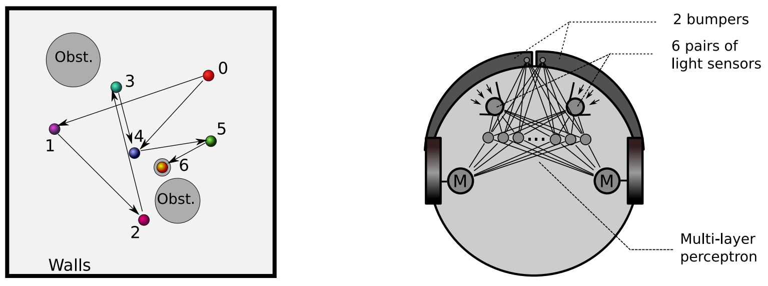
The non-dominated set of the entire feasible search space is the globally Pareto-optimal set (Pareto front).

A typical multi-objective algorithm [8] sorts individuals with respect to domination. Non-dominated individuals may, for instance, be ranked 1, making them the most suitable for reproduction. Individuals which are only dominated by non-dominated ones may be ranked 2, and so on.

Such an algorithm will first try all sub-tasks and the goal-task simultaneously. If the task is incremental, only a small subset of corresponding fitnesses should be different from the minimum. As the fitness on other tasks will be almost equal, the most simple sub-tasks only differentiate individuals and will then be the support of selection. They will be automatically considered as the first sub-tasks to solve. Once individuals are able to test their skills on other sub-tasks or on the goal-task, the algorithm will automatically sort individuals using theses objectives, maintaining the complete set of optimal trade-offs, from the best individuals for each sub-task to the best one for the goal-task.

The aggregation of objectives, as used by Nolfi et al. [15], is a simple way to perform a multi-objective optimization. However, at least three aspects differentiate their approach to the one presented here:

– depending on the shape of the Pareto front, some optimal trade-offs cannot be found using an aggregation approach [8];
– weights have to be chosen using a time-consuming and often not theoretically justified trial-and-error approach;
– best individuals on sub-tasks are not kept, making impossible the simultaneous exploration of different paths potentially leading to more efficient results; for instance, an aggregation approach can be trapped in area (3) of figure 1 or lead to over-specialized individuals;



(a) (b)

Fig. 3. (a) Overview of the benchmark task. Seven colored lights and two obstacles are placed in a arena delimited by four walls. Each light is mounted on a button switch which turns on some other lights, the light circuit (unknown to the algorithm) being represented by arrows. The goal-task is to turn on the sixth light. (b) The simulated robot is equiped with two bumpers and six pairs of binary light sensors, each one sensing a different light. The controller is a multi-layer perceptron with 14 inputs, 14 hidden nodes and 2 outputs (the speed of the motors).

5 Example: a complex light-seeking robot

5.1 Light-seeking robot

To benchmark the proposed approach on a typical incremental task, we designed a variant of the light-seeking task which involves eight sub-tasks with complex dependencies. A simulated wheeled robot is placed in an arena with some obstacles and seven different colored lights (figure 3(a)). Each light is mounted on a button switch which, when pressed, turns on one or more lights in the arena. Once a light is on, it remains in the same state during the whole experiment. The main goal is to turn on a particular light. The connection between lights and switches, a simple explicit dependency between tasks, is to be discovered by the evolutionary algorithm. The only knowledge used to bootstrap the process is that turning on a light should help to turn on the goal-light. As we consider separately each light, turning on a particular light is considered a particular sub-task[1]. To benchmark future incremental algorithms, the task can be easily complexified by adding lights or changing the dependencies between lights.

The underlying structure of this incremental task is depicted on figure 3 (a). The first light turns on two other lights, creating two paths to accomplish the goal-task. A part of the population may choose to learn to turn on lights $\{0, 1, 2, 3, 6\}$ and another one may learn sequence $\{0, 4, 5, 6\}$. This task can therefore be learned in different ways. Moreover, all robots have to avoid obstacles. Despite the simplicity of this problem and of the different sub-tasks, it involves at least five sub-tasks (avoiding obstacles, turn on the good lights of the path having the least number of sub-tasks), making it one of the most complex composite tasks explored with incremental evolution at this time.

[1] This corresponds to seven sub-tasks, the eighth being obstacle avoidance.

The simulated robot (figure 3(b)) is equiped with two bumpers, to avoid obstacles, and six pairs of light-sensors, each one sensing a different light color. Light sensors have a 90 degrees field of view and a binary output (1 if the light is in the field of view, 0 otherwise). Robots are controlled by a simple feed-forward neural network with 14 inputs, 2 outputs and 14 hidden nodes. Neurons use the following activation function, based on $\tanh(x)$, to easily allow the inhibition of a neuron in the case of negative inputs:

$$f(x) = \begin{cases} \tanh(x) \text{ if } x > 0 \\ 0 \text{ otherwise} \end{cases}$$

Cross-over is not used and a Gaussian mutation is applied to weights.

Runs of a direct evolution algorithm designed to minimize the number of time-steps to turn-on the last light, did not find any working controller, all individuals obtaining the minimum fitness. An incremental method seems therefore required to bootstrap the process.

To use the proposed approach, eight objectives are defined. The first one evaluates the obstacle avoidance skills by rewarding how much the robot moves during the whole experiment, thus punishing robots blocked by obstacles[2]:

$$F_0 = \frac{1}{T} \sum_{t=0}^{t=T} \sqrt{\big(x(t) - x(t-1)\big)^2 + \big(y(t) - y(t-1)\big)^2}$$

where T is the length of an experiment and $(x(t), y(t))$ the position of the robot at time-step t. The six remaining objectives are defined as the number of time-steps since the beginning of the experiment before pressing each switch button. To avoid over-learning, each objective is the minimal score for three experiments in which the robot starts from different positions:

$$F_i = \min_{n=1,2,3} \frac{-\varphi(n,i)}{T}$$

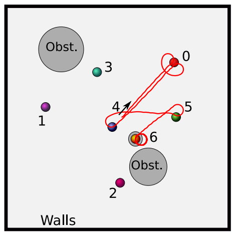
where $i = 1, 2, ..., 7$ is the identifier of the light and $\varphi(n, i)$ denotes the number of time-steps spent before turning on light i, for experiment n.

We launched 10 evolutionary runs, with a population of 150 individuals and the multi-objective evolutionary algorithm ε-MOEA [16]. 60 new individuals were generated at each generation.

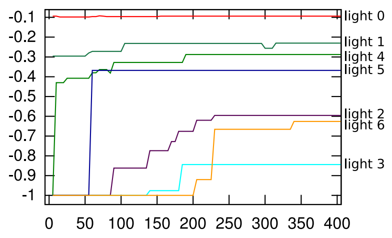
5.2 Results

About 300 generations are needed to obtain efficient neuro-controllers able to reach the sixth light. The best individuals found by evolution are those that used the shortest path, i.e. $\{0, 4, 5, 6\}$. They can be found by looking at the best individuals on the "light 6" criterion. Figure 4(a) depicts a typical trajectory

[2] In the simplified simulation we used, a robot touching an obstacle is systematically stopped until it goes backward.



(a)



(b)

Fig. 4. (a) Typical trajectory of an evolved simulated robot. Lights 0, 4, 5 and 6 are successfully turned on. (b) Best fitness obtained for each fitness with regards to generation. Lights 0, 1 and 4 are first optimized; they are followed by light 5 (gen. 50), 2 (gen. 70), 3 (gen. 130) and 6 (gen 200).

followed by an evolved robot which uses its sensors to avoid obstacles and to locate the successive lights. When the last light has been turned on, it draws an infinite circle around it.

Figure 4(b) shows the best fitness obtained on each sub-task with regards to the generation number. The successive steps followed by the evolutionary process are visible: good controllers are obtained in the first generations for lights 0, 1 and 4; a controller able to reach the fifth light is found around generation 50 and the sixth at generation 230. This switch between sub-tasks was automatic and allowed the process to continue to optimize the first objectives.

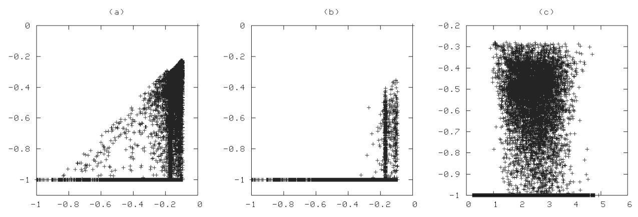


Fig. 5. Typical patterns observed in fitness space. (a) light 4 (y axis) versus light 0 (x-axis); (c) light 5 versus light 0. (c) light 0 versus obstacle avoidance.

Figure 5 shows the explored search space projected on two dimensions. Analyzing these representations may help in finding the dependencies between the criteria, but their interpretation must be careful as they also include historical aspects of the search process[3]. The triangular shape of light 4 versus light 0 criterion (figure 5 (a)) shows the linear dependency of this criterion on light 0 – it is an example of part (2) of figure 1(b). Equivalent figures with light n relative to its predecessor in the path representing the whole task (figure 3(a)) show a

[3] Actually, these sets are only subsets of the whole search space.

similar structure, although it may be shifted or scaled down. The "L" shape of the graph of light 5 relative to light 0 (figure 5 (b)) may be interpreted in different ways. At first glance, it may indicate that a fitness above -0.3 is mandatory before the search can begin on these criteria. But another interpretation may be that the search on these criteria may have started late, at a time when the only remaining individuals were behaving very well on light 0. Anyway, this clearly shows that the search on this criteria has automatically started when some conditions were met and the main point is that we did not tell the algorithm when to start or even what these conditions were. Graph of light criteria relative to obstacle avoidance show a more or less densely populated square (figure 5 (c)) indicating that the two objectives are relatively independent.

6 Discussion and conclusion

In this paper, we proposed an original method to solve the bootstrap problem observed in evolutionary robotics. It removes some important biases present in the previously published methods by requiring less knowledge from the experimenter and less constrains the search. The sub-tasks does not have to be ordered and actually different orders are automatically explored by the evolutionary process. Moreover, the system behaves as an automatic switch between the stages while not stopping to improve already explored sub-tasks.

This method relies on a multi-objective view of incremental evolution which could lead to powerful future algorithms by exploiting the particular shape of the fitness space. One of the difficulties for the proposed method to scale up is the high number of objectives involved. Current multi-objective algorithms are designed to handle a few objectives (typically up to four) and are not as efficient in presence of more objectives. However, the particular shape of the fitness space suggests that the number of dimensions could be reduced during the evolutionary process, for instance using principal component analysis methods. This kind of dimension reduction could allow to neglect the evaluation of sub-tasks during some generations and possibly to reduce the computational cost induced by the simultaneous evaluation of the sub-tasks. Indeed, in the light-seeking task, an agent is observed during a certain amount of time and some fitness functions are derived. Consequently, adding new sub-tasks does not require new evaluations; but some other problems may require sub-tasks to be evaluated in different contexts, thus adding a significant computational cost.

The cross-over operator, not used in this work, could benefit from the proposed method and should be investigated in future work. Since the process maintains specialists in each sub-task, the cross-over operator could allow the evolutionary process to combine sub-parts of each specialist to create more efficient individuals. Such investigations probably requires a modular encoding for neural-networks, as ModNet[17]. Increasing the complexity of the neural-network during evolution also seems to be a key element. All this pleads for considering both selection algorithms and controller encodings as a whole in future work on incremental evolution.

References

1. Meyer, J.A.: From natural to artificial life: Biomimetic mechanisms in animat design. Robotics and Autonomous Systems **22**(3-21) (1997)
2. Meyer, J.A., Husbands, P., Harvey, I.: Evolutionary robotics: a survey of applications and problems. In Husbands, P., Meyer, J.A., eds.: Proceedings of The First European Workshop on Evolutionary Robotics - EvoRobot98, Springer-Verlag (1998)
3. Kodjabachian, J., Meyer, J.A.: Evolution and development of neural networks controlling locomotion, gradient-following, and obstacle-avoidance in artificial insects. IEEE Transactions on Neural Networks **9** (1997) 796–812
4. Urzelai, J., Floreano, D., Dorigo, M., Colombetti, M.: Incremental robot shaping. Connection Science Journal **10**(384) (1998) 341–360
5. Harvey, I., Husbands, P., Cliff, D.: Seeing the light: artificial evolution; real vision. (1994)
6. Gomez, F., Miikkulainen, R.: Incremental evolution of complex general behavior. Adaptive Behavior **5** (1997) 317–342
7. Urzelai, J., Floreano, D.: Incremental Evolution with Minimal Resources. Proceedings of IKW99 (1999)
8. Deb, K.: Multi-objectives optimization using evolutionnary algorithms. Wiley (2001)
9. Winkeler, J., Manjunath, B.: Incremental evolution in genetic programming. Genetic Programming (1998) 403–411
10. Walker, M.: Comparing the performance of incremental evolution to direct evolution. 2nd International Conference on Autonomous Robots and Agents (2004)
11. Parker, G.: The Incremental Evolution of Gaits for Hexapod Robots. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001) (2001) 1114–1121
12. Mouret, J.B., Doncieux, S., Meyer, J.A.: Incremental evolution of target-following neuro-controllers for flapping-wing animats. In: From Animals to Animats: Proceedings of the 9th International Conference on the Simulation of Adaptive Behavior (SAB). (2006) 606–618
13. Larsen, T., Hansen, S.: Evolving composite robot behaviour-a modular architecture. Robot Motion and Control, 2005. RoMoCo'05. Proceedings of the Fifth International Workshop on (2005) 271–276
14. De Nardi, R., Togelius, J., Holland, O., Lucas, S.: Evolution of Neural Networks for Helicopter Control: Why Modularity Matters. Evolutionary Computation, 2006. CEC 2006. IEEE Congress on (2006) 1799–1806
15. Nolfi, S., Paris, D.: Evolving non-Trivial Behaviors on Real Robots: an Autonomous Robot that Picks up Objects. Topics in Artificial Intelligence: 4th Conference of the Italian Association for Artificial Intelligence, AI* IA'95, Florence, Italy, October 11-13, 1995: Proceedings (1995)
16. Deb, K., Mohan, M., Mishra, S.: Evaluating the ε-domination based multi-objective evolutionary algorithm for a quick computation of pareto-optimal solutions. Evolutionary Computatition **13**(4) (2005) 501–525
17. Doncieux, S., Meyer, J.A.: Evolving PID-like neurocontrollers for non-linear control problems. International Journal of Control and Intelligent Systems (IJCIS). Special Issue on nonlinear adaptive PID control **33**(1) (2005) 55–62