Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity

Jean-Baptiste Mouret Stéphane Doncieux

Abstract— **The bootstrap problem is often recognized as one of the main challenges of evolutionary robotics: if all individuals from the first randomly generated population perform equally poorly, the evolutionary process won't generate any interesting solution. To overcome this lack of fitness gradient, we propose to efficiently explore behaviors until the evolutionary process finds an individual with a non-minimal fitness. To that aim, we introduce an original diversity-preservation mechanism, called behavioral diversity, that relies on a distance between behaviors (instead of genotypes or phenotypes) and multi-objective evolutionary optimization.**

This approach has been successfully tested and compared to a recently published incremental evolution method (multi-subgoal evolution) on the evolution of a neuro-controller for a light-seeking mobile robot. Results obtained with these two approaches are qualitatively similar although the introduced one is less directed than multi-subgoal evolution.

I. Introduction

Evolutionary algorithms have been widely used to design controllers, and especially neuro-controllers, for robots (see [1], [2] and [3] for an overview). Compared to other learning methods, one of their main strength is their ability to evolve both the structure and the parameters of complex architectures using a high-level reward function. However, this huge amount of work hides many unsuccessful attempts to evolve complex behaviors by only rewarding the performance of the global behavior. The bootstrap problem is often viewed as the main cause of this difficulty, and consequently as one of the main challenges of evolutionary robotics: if the objective is so hard that all the individuals in the first generation perform equally poorly, evolution cannot start and no functioning controllers will be found. For instance, it has been found hard to evolve a light-seeking behavior for a robot in a complex arena without having first evolved an obstacle-avoidance reflex [4].

In consequence, many researchers added some kind of rewards for intermediate steps, leading to successful *incremental evolution processes* [4]–[9]. Nonetheless, these evolutionary approaches 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 them from scaling up well to more complex or more open tasks; remarkably, most of them have been tried with only two or three sub-tasks.

Jean-Baptiste Mouret and Stéphane Doncieux are with the Université Pierre et Marie Curie (UPMC) - Paris 6, Institut des Systèmes Intelligents et de Robotique (ISIR), CNRS UMR 7222, F-75005, Paris, France; e-mail: {mouret,doncieux}@isir.fr

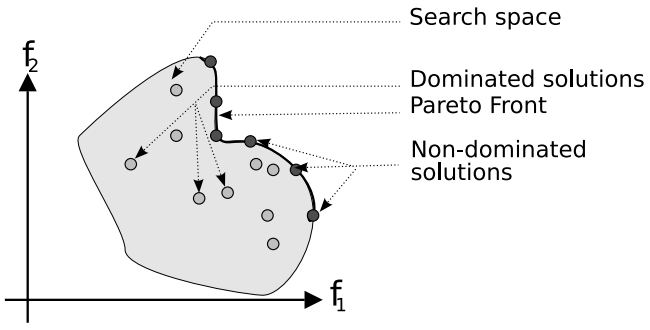

Fig. 1. Dominated and non-dominated solutions for a problem in which the two objectives f_1 and f_2 must be maximized.

Another idea to bootstrap an evolutionary process is to efficiently explore the neighborhood of current candidate solutions to find one with a non-minimal fitness. This concept is close to diversity-preserving mechanisms, widely investigated in evolutionary computation (e.g [10]–[12]), which typically rely on a distance between the genotypes or the phenotypes. However, such distances are conceptually and computationally difficult to employ with complex structures as neural networks.

Drawing inspiration from these diversity-preserving mechanisms and adapting them to evolutionary robotics, we introduce in this paper the *behavioral diversity*, an original method to maintain the diversity in a population that relies on a distance between behaviors and a multiobjective evolutionary algorithm (MOEA, see [13]). We then show how such a diversity preservation mechanism can efficiently overcome the bootstrap problem in an evolutionary robotics experiment in which a simulated robot that has to successively reach some lights in an arena. As a reference point, we compare the results obtained using this new approach with the ones obtained with multi-subgoal evolution [9], a recently published incremental method based on multiobjective evolution.

The remainder of this paper is organized into four parts. In section II, we briefly review the related literature about multi-objective evolutionary algorithms, incremental evolution and diversity preservation. The proposed method is detailed in section III. Section IV describes an evolutionary robotics experiment and reports the results of both methods. A short discussion concludes this paper.

II. Previous Work

A. Multi-objective evolutionary algorithms

Most problems tackled in evolutionary robotics involve trade-offs between numerous objectives, such as energy versus efficiency (see e.g. [14]). The classical approach is to aggregate objectives using a weighted sum or a product to define a composite fitness. Hence, a particular set of weights defines a particular trade-off. Despite its experimental success, this method has several substantial drawbacks. First, it can be theoretically shown that some trade-offs cannot be found (see [13]), thus silently constraining the study to a subset of possible trade-offs. Second, weights have usually to be inferred from a time-consuming trial-and-error approach because no theoretical methods to define them are available. Last, the exploration of each trade-off requires launching a new evolutionary process. These different implications generally limit analysis and discussions of results to one or two particular trade-offs, associated to a particular set of weight parameters that may be difficult to explain.

Recent research in evolutionary computation proposed numerous algorithms to simultaneously optimize several objectives without aggregating them [13]; most of them rely on the concept of domination and generate the so-called Pareto Front (figure 1):

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

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).

Multi-objective evolutionary algorithms are designed to find the Pareto-front, i.e. the set of all the Pareto-optimal trade-offs. Most of them rely on two mechanisms:

- a selection procedure that ensures that non-dominated individuals have a selective advantage;
- a diversity-preservation method designed to spread non-dominated individuals along the whole Pareto front, i.e. that as many different trade-offs as possible are explored.

A typical MOEA 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. The diversity on the Pareto front can be maintained using several ways such as fitness sharing [15] or clustering methods [16].

Despite their considerable success in the optimization field (see [17]), MOEAs deserved little attention in the evolutionary robotic literature (notable exceptions being [14], [18]–[21]). One of the ambitions of this paper is to highlight how they can be useful in evolutionary robotics, in particular to solve the bootstrap problem.

B. Incremental evolution

Four main approaches emerge from the different attempts to tackle the bootstrap problem in evolutionary robotics: staged evolution, environmental complexification, fitness shaping and behavioral decomposition.

Staged evolution is probably the most intuitive – and most used – incremental process. The main task is split into ordered sub-tasks, each with a corresponding fitness function. During each stage, individuals are selected only using the associated fitness. The process is switched from each stage to the next one once a convergence criterion is reached, for instance when a "good enough" fitness is obtained by the best individual or when the best fitness doesn't change for some generations. This technique has first been successfully employed by Harvey et al. [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 (e.g. [7], [8], [14], [22]–[24]).

Following the same idea, environmental complexification aims at more continuously changing the complexity of the task through the tuning of dedicated parameters. Gomez and Miikkulainen [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, each sub-task is solved by a sub-controller evolved using a dedicated fitness, different from the main goal. Another evolutionary process is then used to combine all the sub-controllers together. This approach has been recently used to evolve a position controller for an autonomous helicopter [25].

Fitness shaping is sometimes used to help to bootstrap 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 fitness gradient. For example, Nolfi and Parisi [26] 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.

C. Multi-subgoal evolution

The previously mentioned approaches allowed to generate non trivial behaviors, but they require an important help from the human designer: intermediate steps have to be defined, ordered and their combination also requires the intervention of the human designer, may it be to trigger the switch between these steps or to gather all the elementary solutions in a wider controller. The method introduced in [9], called multi-subgoal evolution, intends to relax some of the constraints of these incremental approaches.

In multi-subgoal optimization, intermediate steps still have to be defined, but once this is done, the evolutionary process will do the rest, without the need to order them or to choose when to switch between the goals. Another interesting aspect with this approach is that all those intermediate steps are optional. We can therefore provide more information than what is actually necessary without endangering the convergence of the evolutionary run. This feature also differentiates it from fitness shaping in which the fitness pushes towards individuals that fulfill *all* the intermediate behaviors even if some of them are not mandatory.

The principle of multi-subgoal optimization consists in defining an objective for each of the intermediate steps and another one for the final goal and then to use multiobjective evolutionary algorithms to perform the search. In such a context, difficult objectives that require to solve simpler tasks won't be explored at all by the algorithm at the beginning of the search process, as all individuals will perform equally poorly, while simpler objectives will draw much of the search effort. As soon as performances will be high enough on simpler tasks so that more complex tasks resolution will be reachable, individuals able to solve them will appear and the algorithm will then automatically start to concentrate on this part of the search space. The other advantage is that this "switch" is not irremediable because the search with respect to the simpler tasks will never stop, thus leaving the possibility to obtain individuals relying on more efficient solutions of the simpler problems. On the other side, the search on the complex task will start as soon as possible, thus avoiding to consider only individuals that may be overspecialized on the simple task which could be difficult to adapt to the complex one, as it can happen with manual switches. Formally, this approach can be considered as a multiobjective formulation of fitness shaping.

More details about multi-subgoal evolution can be found in [9].

D. Diversity

The previously described approach requires to optimize many objectives whereas current multiobjective evolutionary algorithms have poor performance when more than three objectives are used [27], [28]. We consequently explored a different idea relying on the exploration of new behaviors.

The bootstrap problem, and similarly the local minima problem, takes its roots in a lack of a useful selective gradient to select potentially good individuals: all of them share the same poor fitness (sometimes, all of them get a null fitness). In the best case, the evolution is then equivalent to a random search; in the worst case, we observe a premature convergence of the evolutionary process to one, often degenerate, solution. Among genotypes with the same fitness, we can heuristically choose the ones which are the most original. This idea, which can be seen as a push in favor of diversity, has a long history in evolutionary algorithm literature, one of the most popular method being fitness sharing [10]. In this approach, the search landscape is modified by decreasing the fitness of similar individuals, thus encouraging search in unexplored regions. More precisely, if d_{ij} is the distance between the genotype of i and j, the *sharing function* $sh(d_{ij})$ is defined as follows:

$$Sh(d_{ij}) = \begin{cases} 1 - (\frac{d_{ij}}{\sigma_{\text{share}}})^{\alpha}, & \text{if } d \leq \sigma_{\text{share}} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where σ_{share} is a user-defined parameter. The fitness of individual i is then divided by the niche count c_i:

$$F_i = \frac{F(x_i)}{c_i} \text{ where } c_i = \sum_{j=1}^{N} Sh(d_{ij}) \quad (2)$$

Dividing the fitness by the niche count can be seen as an aggregation of two objectives, the fitness and the diversity. Given the recent progress of Pareto-based multiobjective evolutionary algorithms, some researchers suggested to use a multiobjective approach instead of the aggregation. This explicitly models the classic trade-off between exploration and exploitation [29] and let the evolutionary algorithm automatically tune it. Different ways to improve the diversity by adding an objective have been investigated and compared, from the distance to the nearest neighbor to a simple random value [20], [30], [31]. All the available studies conclude that (1) multiobjective methods to preserve the diversity are more efficient than their single-objective counterparts and (2) the mean distance between the individual and the remaining of the population leads to the best results.

III. Behavioral Diversity

Computing a distance between individuals is, however, difficult in evolutionary robotics, especially when evolving the topology of neuro-controllers. The distance between genotypes is often hard to compute because typical distances between two graphs, thereby two neural networks, are NP-complete [32]. The permutation problem [33] prevents the definition of simple metrics on neural-networks with a fixed topology. Although some methods have been published to allow the fitness sharing of neural networks [34], an infinity of very different neural-networks can lead to a similar behavior. This is especially striking with degenerate individuals as those observed when the evolutionary process is unable to start: any neural network not linked to the motors will lead to an immobile robot, thus fitness sharing won't help here. Moreover, fitness sharing has not originally been formulated for multiobjective evolution while the interest of using these methods in evolutionary robotics is growing (see [14], [35], [36], for instance). Hence, maintaining the diversity of genotypes with fitness sharing when evolving neuro-controllers for robots is hard to implement and often inefficient.

The goal of fitness sharing was to differentiate individuals on other aspects than fitness values. Considering the genotype to provide some more information to push towards diversity was natural in classical evolutionary algorithms, but evolutionary robotics experiments offers some other possibilities: the behavior can be used to distinguish individuals. Fitness values are based on an observation of the behavior,

but they are focused on a small set of behavior consequences, not on behaviors on their own. As an alternative approach to fitness sharing, we propose to compute a distance between behaviors instead of genotypes, thus explicitly promoting new behaviors. This distance could be used to modulate the fitness, as in fitness sharing. However, using a multiobjective evolutionary algorithm allows to rely on the good results obtained with multiobjective approaches to preserve the diversity and to explicitly model the exploration/exploitation trade-off.

Let us denote by D_{ij} the distance between the behaviors of individuals i and j. In its simplest form, the behavior sharing function $B(i)$ can be formulated as the mean distance between the behavior of i and the one of each individual of the population. Instead of maximizing the fitness $F(i)$ only, we have to maximize two objectives:

$$\begin{cases} F(i) \\ B(i) = \frac{1}{N}\sum_{j=1}^{N} D_{ij} \end{cases} \quad (3)$$

Designing a distance between behaviors of robots may seem to be a challenging task. However, we have found that in many cases a distance built on basic features tied to robot behavior was sufficient to bootstrap the evolutionary process and avoid many local minima. A simple and efficient approach for mobile robotics is to associate to each individual i a vector $\mathbf{v}^{(i)}$ describing the state of the environment at the end of the experiment. D_{ij} can then be defined as the Euclidean distance between $\mathbf{v}^{(i)}$ and $\mathbf{v}^{(j)}$. For instance, if the robot has to move objects in an arena, a $\mathbf{v}^{(i)}$ may contain the positions of the objects. If the robot has to explore a maze, the final positions can also be used. As a last example, when evolving a n inputs logic function, e.g. XOR, the output for each of the 2^n set of inputs can be stored in a 2^n-dimensional vector.

The behavioral diversity approach shares some similarities with the recently published novelty search [37]. In this paper, Lehman and Stanley proposed to maximize the novelty of behaviors *instead of* fitness to define an open-ended evolutionary process. Using a distance between behaviors $D(i,j)$, similar to the one discussed here, they defined the sparseness ρ at a point x: $\rho(x) = \frac{1}{k}\sum_{i=0}^{k} D(x,\mu_i)$ where μ_i is the i-th nearest neighbor of x with respect to the distance D. The neighbors are computed using the current population and an *archive* of all the previously explored behaviors. Three main points differentiate the two approaches:

- they optimize only novelty whereas we propose to use a multiobjective algorithm that does not ignore the main goal;
- they use an archive of all the previously explored behaviors;
- since the distance is computed using the archive *and* the current population, they mixed a diversity measure with a novelty one.

Despite the substantially high computational cost induced by the growing archive, novelty search could avoid evolutionary cycles that could occur with behavioral diversity. Nonetheless, we will see in the next section that some intuitive behavior distance can not be employed with novelty search while they can be used with behavioral diversity.

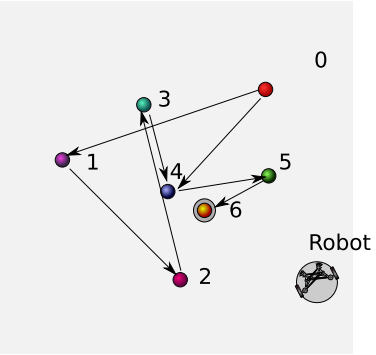

Fig. 2. Overview of the benchmark task. Seven colored lights are placed in an arena, each of them being mounted on a button switch which turns on some other lights; the light circuit (unknown to the algorithm) is represented by arrows. The goal-task is to turn on light 6.

IV. Light-Seeking Robot

A. Main task

To benchmark the proposed approach on a typical incremental task, we designed a variant of the light-seeking task which involves seven sub-tasks with complex dependencies. A similar but simple version of this task has, for example, been studied in [38]. A simulated wheeled robot is placed in an arena with seven different colored lights (figure 2). 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. To benchmark future incremental algorithms, the task can be easily complexified by adding lights or changing the dependencies between lights. This task has been chosen because it does not require nor complex genotypes nor complex fitness functions to be solved: its main challenge is its incremental nature.

The underlying structure of this incremental task is depicted on figure 2. Each lights turn on only if the previous related light is on, following the relations depicted on figure 2. The first light turns on two other lights, creating two paths to solve the goal-task. A part of the population may choose to learn to turn on lights $\{0,1,2,3,4,5,6\}$ and another one may learn sequence $\{0,4,5,6\}$. This task can therefore be learned in different ways. Despite the simplicity of this problem and of the different sub-tasks, it involves at least four sub-tasks (turn on each of the lights belonging to the shortest path to the goal light), making it one of the most complex composite tasks explored with incremental evolution at this time.

The simulated robot (figure 3) is equipped with seven 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).

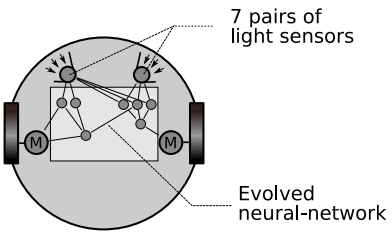

Fig. 3. The simulated robot is equipped with seven pairs of binary light sensors, each one sensing a different light. The controller is an evolved neural-network with 14 inputs and 2 outputs (the speed of the motors).

A robot is controlled by an evolved neural network with 14 inputs and 2 outputs whose topology and synaptic weights are encoded by a simple direct encoding [1]: the network is represented by a weighted directed graph in which neurons and connections can be randomly added or removed. Two kinds of mutations are possible:

- structural mutation: addition/removal of a neuron or a connection;
- parametric mutation: change of a randomly chosen synaptic weight or a neuronal bias; we used here a change in a set of 15 possible values (similar results were obtained using real-valued weights and Gaussian mutation).

The mutation rates of each operator are detailed in appendix.

Cross-over is not used. Neurons use the following activation function, based on $\tanh(x)$, to easily allow the inhibition of a neuron in the case of negative inputs:

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

B. Experimental setup

Behavioral diversity and multi-subgoal evolution have been tested on this problem to compare them.

To use the multi-subgoal approach, one objective for each light ($F_i^{msb}, i \in 0,1,\cdots,6$) was defined. To avoid over-learning and check the robustness of the evolved behaviors, each objective is the minimal score for three experiments in which the robot starts from different positions:

$$F_i^{msb}(x) = \min_{n=1,2,3} \frac{-\varphi(n,i)}{T} \tag{4}$$

where $i = 0,1,...,6$ is the identifier of the light, T is the length (in time-steps) of each experiments and $\varphi(n,i)$ denotes the number of time-steps spent before reaching light i, for experiment n.

The behavioral diversity approach first requires to define a classic fitness F^{bd} corresponding to the goal to be reached. Here we used the number of time-steps to reach the last light:

$$F^{bd}(x) = F_6^{msb}(x) = \min_{n=1,2,3} \frac{-\varphi(n,i)}{T} \tag{5}$$

We then have to define a distance D_{ij} between the behavior of i and j. We chose to describe the behavior of i by a Boolean vector $\mathbf{v}^{(i)}$ such that:

$$\mathbf{v}_k^{(i)} = \begin{cases} 1 & \text{if the } k\text{-th switch has been pressed} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

D_{ij} can be computed using an Euclidean distance between $\mathbf{v}^{(i)}$ and $\mathbf{v}^{(j)}$:

$$D_{ij} = \left|\left|\mathbf{v}^{(i)} - \mathbf{v}^{(j)}\right|\right| \tag{7}$$

This distance lead to surprising results if we use it with novelty search. To distinguish novelty and diversity, we only consider here the archive when computing ρ. Thus, we will not mix the diversity with the novelty. At the end of the random generation, the archive typically contains the following behaviors, due to the incremental structure of the task: 0000000, 10000000, 1000100 and 1100000. We are clearly more interested in 1000100 and 1100000 because the corresponding robot reaches more lights. Since the archive contains only 4 entries, only 4 kinds of behaviors are present in the population. Let's compute their novelty score to know which ones will be selected for the reproduction:

$$\begin{aligned} 0000000: &\quad \rho = 0.957 \\ 1000000: &\quad \rho = 0.75 \\ 1000100: &\quad \rho = 0.957 \\ 1100000: &\quad \rho = 0.957 \end{aligned}$$

The most interesting individuals have the same fitness than individuals with the 0000000 behavior. Since the latter is the least interesting and easiest to obtain, we can predict that the evolutionary process will never start. This prediction has been confirmed by some preliminary experiments.

We used NSGA-2 [39], a state-of-the-art MOEA, to maximize the objectives in the multi-subgoal case and in the behavioral diversity case, with a population of 200 individuals. 30 runs of 1000 generations for each of the proposed methods have been launched.

	Multi-subgoal Evol..	Behavioral div.	T-test
Bootstrap gen.	138(38)	218(75)	$p < 0.001$
Goal objective	$-0.27(0.04)$	$-0.27(0.05)$	$p < 0.853$

TABLE I
MEAN BOOTSTRAP GENERATION AND MEAN OBJECTIVE VALUE AFTER 500 GENERATIONS.

V. RESULTS

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. This contrasts with the results obtained with the two presented methods, summarized in table I. In both cases, all the runs converged to very similar values of the goal fitness. The bootstrap generation (the generation where at least one individual reach a main fitness greater than -0.8) was significantly lower with multi-subgoal evolution than with the behavioral diversity (138 versus 218) but both values

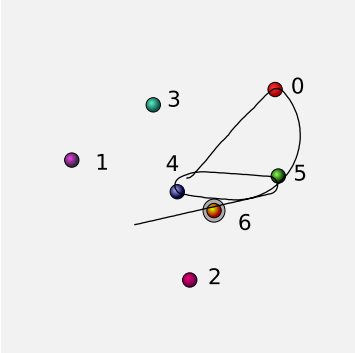

Fig. 4. Trajectory of a typical non-dominated individual after 500 generation. The robot follows the shortest path (0,4,5,6).

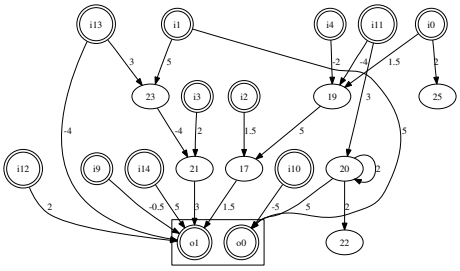

Fig. 5. Neural network of a typical non-dominated individual, after 500 generations. Input and output neurons are double-circled (some inputs are unused); i0 and i1 are connected to the sensors able to see the first light, i2 and i3 the second one, etc.

are of the same order of magnitude: the bootstrap problem was overcome in a few hundred of generations and similar fitness values were obtained at comparable generations.

All observed dominant individuals for the last light objective followed the shortest path (i.e. $0,4,5,6$) whereas a slightly longer path, using more lights, was possible (figure 4). An aggregation of all the subgoals into one single objective would have lead to individuals trying to push all the switch buttons. All the obtained neuro-controllers were robust to the robot starting point: from any reasonable starting position, the robot efficiently performed the good sequence. As a consequence, not overlearned behavior was observed.

Figure 5 displays the neural network of a typical non-dominated individual, after 500 generations. This neural network is surprisingly simple as it uses only 4 hidden neurons and some sensors are not connected to any neuron. This low complexity emphasizes that the investigated problem is mostly a selection pressure problem and not a genotype problem. Moreover, it can be noticed that such simple neural network is unlikely to learn the complex sequence "by heart" and more likely to rely on its sensors.

To analyze the bootstrap process, we plotted the proportion of the population which reaches each light in a typical run of each method (figures 6 and 7).

The diagram corresponding to multi-subgoals evolution shows a clear staircase structure. Almost 50 generations are required to obtain the first individuals to reach the light 0 but after 100 generations, the whole population reaches it. Individuals can then switch the next lights, 1 and 4. These new skills are shared by the whole population in a few generations, allowing some individuals to reach light 6 in less than 150 generations. This incremental structure gives support to our assumptions about the effect of the multiobjective evolutionary pressure: the first attainable objectives are first optimized and the process automatically switches to new ones as soon as possible; the two hypothesis – the two available paths to solve the task – are explored simultaneously; and solved objectives apply almost no pressure once they are solved.

The diagram corresponding to behavioral diversity shows a very similar structure in which gradients show that each subtasks is successively solved. However, the picture is noisier because the diversity approach also maintains individuals that don't solve each task, since at a given generation, they may be different from the remaining of the population. Significantly fewer generations are required to converge with the multi-subgoal method than with the behavioral diversity.

VI. Conclusion and Discussion

We proposed a new method to solve the bootstrap problem observed in evolutionary robotics and compared it with a recently published approach. Compared to previous methods, both approaches don't require to order sub-tasks or to specify when to switch between them since the evolutionary process automatically explores all the available orders. Moreover, the performance on the subgoals can be continuously improved and some may be ignored. We showed that both methods efficiently solved the bootstrap problem on a typical incremental robotics task although the newly introduced method is less directive than the previously published one. Hence, our contribution is twofold: (1) we propose a new way to maintain the diversity of a population, especially adapted to evolutionary robotics and (2) we show that such a diversity preservation mechanism can efficiently overcome the bootstrap problem in an evolutionary robotics experiment.

Although they seem at first glance very different, the two approaches actually share the same basic principle: their common goal is to provide a selective pressure towards exploration and they only differ by the method used to supply it. Both behavioral diversity and multi-subgoal will select individuals which are able to switch on lights that no other can[1]. However, the behavioral diversity ranks individuals according to an averaged Euclidean distance whereas multi-subgoal uses a Pareto sort. Consequently, behavioral diversity will also select individuals that can't switch a single light, as long as they are the only one to do so, whereas an individual that switches any light will always dominate an individual that doesn't switch any light at all in multi-subgoal. This

[1]They would be even more similar with a binary value for the subgoals in the multi-subgoal approach: 0 light not switched on, 1 light switched on during evaluation.

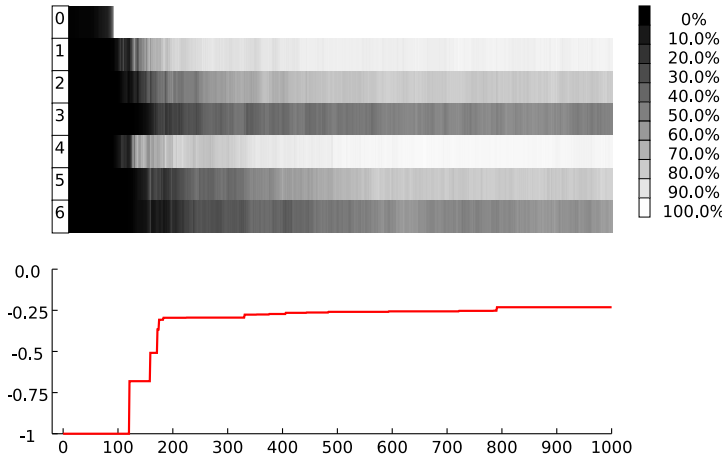

Fig. 6. Multi-subgoal evolution. (top) Proportion of individuals which reach each light as a function the generation, in a typical run. (bottom) Best fitness for the main objective.

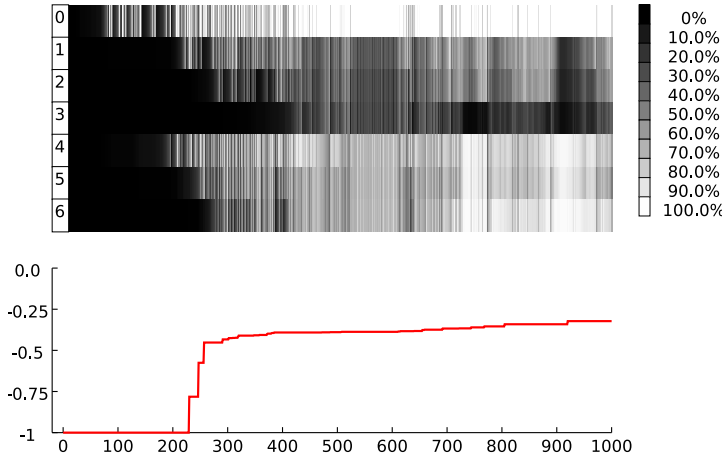

Fig. 7. Behavioral diversity. (top) Proportion of individuals which reach each light as a function of generation, in a typical run. (bottom) Best fitness for the main objective.

doesn't make a huge difference here, as individuals that don't switch any light are easy to find and then won't capture the search effort for long. More formally, the main difference lies in the fact that multi-subgoal is an oriented search. It means that the designer has to know how to sort good and bad behaviors and how to evaluate the robot's performance for each of the subgoals. To use behavioral diversity, only a frame in which behaviors can be described needs to be provided. On the toy problem we considered here, this doesn't make a huge difference, neither on the results, nor on the knowledge we have provided to the algorithm, but it might be more significant in other tasks.

The simplicity of behavioral diversity makes this approach appealing and easy to integrate in an existent framework. However, in particular problems, it may be hard to design a relevant distance between behaviors. A generic way to describe behaviors of mobile robots would be useful to design such distances, hence making easier the application of behavioral diversity to new tasks.

Behavioral diversity makes explicit the exploration and exploitation part of the algorithm through the behavioral diversity objective and the goal fitness. Compared to other learning algorithms like reinforcement learning, there is no need to explicit here the balance between the two objectives, it is autonomously managed by the multiobjective algorithm. The true ability of the algorithm to handle this important aspect of the search should be more carefully studied.

The cross-over operator, not used in this work for the sake of simplicity, could have an effect on the two investigated methods. Since multi-subgoal evolution and behavioral diversity maintain individuals able to solve different subtasks, each part of them could be combined to create more efficient individuals. Using this operator to evolve neurocontrollers would probably require a modular encoding of

neural networks, such as MENNAG [40].

REFERENCES

[1] J.-A. Meyer, P. Husbands, and I. Harvey, "Evolutionary robotics: a survey of applications and problems," in *Proceedings of The First European Workshop on Evolutionary Robotics - EvoRobot98*, 1998.

[2] S. Nolfi and D. Floreano, *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. Bradford Book, 2000.

[3] D. Floreano, P. Dürr, and C. Mattiussi, "Neuroevolution: from architectures to learning," *Evolutionary Intelligence*, vol. 1, no. 1, pp. 47–62, 2008.

[4] J. Urzelai, D. Floreano, M. Dorigo, and M. Colombetti, "Incremental robot shaping," *Connection Science Journal*, vol. 10, no. 384, pp. 341–360, 1998.

[5] I. Harvey, P. Husbands, and D. Cliff, "Seeing the light: artificial evolution; real vision," in *From Animals to Animats 3, Proceedings of the third international conference on Simulation of Adaptive Behavior*, 1994.

[6] F. Gomez and R. Miikkulainen, "Incremental evolution of complex general behavior," *Adaptive Behavior*, vol. 5, pp. 317–342, 1997.

[7] J. Kodjabachian and J.-A. Meyer, "Evolution and development of neural networks controlling locomotion, gradient-following, and obstacle-avoidance in artificial insects," *IEEE Transactions on Neural Networks*, vol. 9, pp. 796–812, 1997.

[8] J. Urzelai and D. Floreano, "Incremental Evolution with Minimal Resources," *Proceedings of IKW99*, 1999.

[9] J.-B. Mouret and S. Doncieux, "Incremental evolution of animats' behaviors as a multi-objective optimization," in *Proceedings of the tenth International Conference on the Simulation of Adaptive Behavior (SAB'08)*, 2008, pp. 210–219.

[10] D. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, 1987, pp. 41–49.

[11] B. Sareni and L. Krahenbuhl, "Fitness sharing and niching methods revisited," *Evolutionary Computation, IEEE Transactions on*, vol. 2, no. 3, pp. 97–106, 1998.

[12] S. Mahfoud, *Evolutionary Computation 2*. Institute of Physics Publishing, 2000, ch. Niching methods, pp. 87–92.

[13] K. Deb, *Multi-objectives optimization using evolutionnary algorithms*. Wiley, 2001.

[14] J.-B. Mouret, S. Doncieux, and J.-A. Meyer, "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, pp. 606–618.

[15] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization : formulation, discussion and generalization," in *Proceedings of the Fourth International Conference on Evolutionary Programming*, 1993, pp. 416–423.

[16] E. Zitzler, M. Laumanns, L. Thiele, *et al.*, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm," *EUROGEN*, pp. 95–100, 2001.

[17] C. A. C. Coello and G. B. Lamont, *Applications Of Multi-Objective Evolutionary Algorithms*. World Scientific, 2004.

[18] J. Shan, C. Junshi, and C. Jiapin, "Design of central pattern generator for humanoid robot walking based on multi-objective ga," in *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2000, pp. 1930–1935.

[19] J. Teo and H. A. Abbass, "Coordination and synchronization of locomotion in a virtual robot," in *Proceedings of the 9th International Conference on Neural Information Processing (ICONIP02)*, vol. 4, 2002, pp. 1931–1935.

[20] L. Bui, H. Abbass, and J. Branke, "Multiobjective optimization for dynamic environments," *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 3, pp. 2349–2356 Vol. 3, Sept. 2005.

[21] A. Moshaiov and A. Ashram-Wittenberg, "Multi-objective Evolution of Robot Neuro-Controllers," in *Evolutionary Computation, 2009. CEC 2009. IEEE Congress on*, 2009.

[22] J. Winkeler and B. Manjunath, "Incremental evolution in genetic programming," *Genetic Programming*, pp. 403–411, 1998.

[23] M. Walker, "Comparing the performance of incremental evolution to direct evolution," *2nd International Conference on Autonomous Robots and Agents*, 2004.

[24] G. Parker, "The Incremental Evolution of Gaits for Hexapod Robots," *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*, pp. 1114–1121, 2001.

[25] R. de Nardi, J. Togelius, O. E. Holland, and S. M. Lucas, "Evolution of Neural Networks for Helicopter Control: Why Modularity Matters," in *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, 2006, pp. 1799–1806.

[26] S. Nolfi and D. Parisi, "Evolving non-Trivial Behaviors on Real Robots: An Autonomous Robot that Picks up Objects," *Lecture notes in computer science*, pp. 243–243, 1995.

[27] R. Purshouse and P. Fleming, "On the Evolutionary Optimization of Many Conflicting Objectives," *Evolutionary Computation, IEEE Transactions on*, vol. 11, no. 6, pp. 770–784, 2007.

[28] O. Teytaud, "On the hardness of offline multi-objective optimization," *Evolutionary Computation*, vol. 15, no. 4, pp. 475–491, 2007.

[29] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[30] H. Abbass and K. Deb, "Searching under multi-evolutionary pressures," *Proceedings of the Fourth Conference on Evolutionary Multi-Criterion Optimization, Spain*, 2003.

[31] E. de Jong, R. Watson, and J. Pollack, "Reducing bloat and promoting diversity using multi-objective methods," *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 11–18, 2001.

[32] H. Bunke and K. Shearer, "A graph distance metric based on the maximal common subgraph," *Pattern Recognition Letters*, vol. 19, no. 3-4, pp. 255–259, 1998.

[33] X. Yao, "Evolving artificial neural networks," in *Proceedings of the IEEE*, vol. 87, no. 9, 1999, pp. 1423–1447.

[34] K. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10(2), no. 99-127, 2002.

[35] J. Teo and H. Abbass, "Automatic Generation of Controllers for Embodied Legged Organisms: A Pareto Evolutionary Multi-Objective Approach," *Evolutionary Computation*, vol. 12, no. 3, pp. 355–394, 2004.

[36] E. de Margerie, J. Mouret, S. Doncieux, and J. Meyer, "Artificial evolution of the morphology and kinematics in a flapping-wing mini-UAV," *Bioinspiration and biomimetics*, vol. 2, no. 4, p. 65, 2007.

[37] J. Lehman and K. Stanley, "Exploiting Open-Endedness to Solve Problems Through the Search for Novelty," in *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, 2008, pp. 329–336.

[38] D. Floreano and J. Urzelai, "Evolution of plastic control networks," *Autonomous Robots*, vol. 11, pp. 311–317, 2001.

[39] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182–197, 2002.

[40] J.-B. Mouret and S. Doncieux, "Mennag: a modular, regular and hierarchical encoding for neural-networks based on attribute grammars," *Evolutionary Intelligence*, vol. 1, no. 3, pp. 187–207, 2008.

APPENDIX

- population size: 200
- number of runs : 30 for each method
- weights:
  $\{-5.0,-4.0,-3.0,-2.0,-1.5,-1.0,-0.5,$
  $0.0,0.5,1.0,1.5,2.0,3.0,4.0,5.0\}$
- weight mutation rate: 0.2
- number of inputs: 15
- number of outputs: 2
- min. neurons (random gen.): 10
- max. neurons (random gen.) : 20
- min. connections (random gen.) : 20
- max. connections (random gen.) : 35
- rate of connection addition: 0.15
- rate of connection removal: 0.25
- rate of connection change: 0.1
- rate of neuron addition: 0.025
- rate of neuron removal: 0.025
- activation function: $y_i = \tanh\left(-b + 5 \cdot \sum_j w_{ij} x_j\right)$