Why and How to Measure Exploration in Behavioral Space

Charles Ollion
ISIR, UPMC
Paris, France
ollion@isir.upmc.fr

Stéphane Doncieux
ISIR, UPMC
Paris, France
doncieux@isir.upmc.fr

ABSTRACT

Exploration and exploitation are two complementary aspects of Evolutionary Algorithms. Exploration, in particular, is promoted by specific diversity keeping mechanisms generally relying on the genotype or on the fitness value. Recent works suggest that, in the case of Evolutionary Robotics or more generally behavioral system evolution, promoting exploration directly in the behavioral space is of critical importance. In this work an exploration indicator is proposed, based on the sparseness of the population in the behavioral space. This exploration measure is used on two challenging neuro-evolution experiments and validated by showing the dependence of the fitness at the end of the run on the exploration measure during the very first generations. Such a prediction ability could be used to design parameter settings algorithms or selection algorithms dedicated to the evolution of behavioral systems. Several other potential uses of this measure are also proposed and discussed.

Categories and Subject Descriptors

I.2.6 [**Computing Methodologies**]: Artificial Intelligence—*Learning*; I.2.9 [**Computing Methodologies**]: Artificial Intelligence—*Robotics*

General Terms

Algorithms

1. INTRODUCTION

The versatility of Evolutionary Algorithms (EA) allows consideration of the evolution of robot parts, whether it be their controllers, morphologies, or both [1]. For such an application of EA, the fitness value results from the observation of the robot in interaction with its environment. The mapping between a robot genotype and its behavior is complex, as some evolved parts may not have any impact on the output, while others may completely change the behavior: in the case of the evolution of neuro-controllers, for instance, changing any part of the neural network not connected to the outputs has no impact on function behavior, while switching the sign of a single connection weight may completely change it.

The consequence of such a complex mapping is that promoting diversity in the genotypic space only is not sufficient. Promoting diversity in the behavioral space is an efficient solution to the problem of premature convergence, a frequent issue in deceptive behavioral systems [14] or in ER in general [13]. This point is so critical that some studies rely on promoting novelty in the behavioral space only. It can lead to a given task resolution more surely than rewarding the efficiency [9, 10].

This suggests that exploration in the behavioral space is of critical importance. Measuring such an exploration may have many applications:

- definition of fitness-independent criterion for run comparisons, *in particular when they do not converge*: it may be a means of studying very challenging problems;
- parameter setting: finding the parameters that maximize the measure of exploration independently from the task to be solved;
- definition of new Evolutionary Algorithms relying on this measure and dedicated to the evolution of behavioral systems in general and ER in particular, for instance with a specific restart mechanism.

In this paper, we propose a simple measure of exploration in the behavioral space. With the hypothesis that current algorithms are not limited by their exploitation capacity, as suggested by previous work on behavioral diversity [13, 14] and novelty search [9, 10], exploration is a critical factor and will then be strongly correlated to the performance. To validate our indicator, we will then measure the correlation between exploration at the beginning of a run and final performance on two problems:

- a deceptive neuro-evolution problem where a neural network must be able to compute a Xor And Xor function;
- a complex ER task where a simulated robot has to collect balls in an arena.

2. RELATED WORK

2.1 Measuring Exploration

The impact of exploration and exploitation on the performance of genetic algorithms has often been discussed [5]. However, only a few articles deal with explicitely measuring exploration. One of the first approaches is the study of the influence of mutation operators on an algorithm's "exploration power" in the genotypic space [2]. Paenke et al. [15] use different approaches to perform a "diversity analysis" in which they measure the genotypic diversity of populations obtained with different parameters.

The problem of measuring exploration is also naturally addressed in diversity maintenance techniques [7]. In [19], the author calculates a diversity score based on the average distance between individuals and a so-called "average" individual. This measure allows one to know in which state—exploration or exploitation—the algorithm is. Recently, Lehman and Stanley proposed the novelty search algorithm [9], which relies on the sparseness of the individual among its generation, in addition to an archive of previous behaviors. Such an approach explicitly rewards exploration, and thus evaluates it.

More recently, Risi et al. tried to characterize the genotypic space of an evolutionary robotics experiment by projecting it into a two dimensional space [16]. That way, they represented graphically the covered genotypic space of the population obtained with different setups, thus performing a qualitative analysis of exploration.

2.2 Behavioral approaches

In the previously mentioned works, the characterization of exploration or diversity maintenance techniques are usually performed in the *genotypic space*. However, in Evolutionary Robotics and behavioral systems the mapping between genotype and behavior is very complex, and comparing genotypes is not enough. Recent methods [9, 4] have shown that using behavioral comparisons is more meaningful for those systems. Another recent study focuses on analysing the different spaces, by measuring neutrality and ruggedness, which can both influence exploration [17].

The diversity [14] or novelty [9] techniques use those distances in a behavioral space to foster exploration. Such behavioral-based approaches require the definition of distances between behaviors. The following distances have been explored: Edit distance between trajectories [18], NCD (Normalized Compression Distance, an approximation of the Kolmogorov complexity) and Entropy-based distance [8], Hamming distance and Fourrier analysis [4], as well as more problem-dependent distances [9, 14]. These distances are a key factor for comparing behaviors, and so for computing a behavioral exploration measure. Despite all these works, there is no common agreement at the time of writing on the best way to compare behaviors.

The archive of the *Novelty search* algorithm [9] can also reveal information about behavioral exploration. The algorithm adds individuals in an archive if their novelty score is higher than an adaptive threshold ρ_{min}. If many (resp. too few) individuals are added during a single generation, ρ_{min} decreases (resp. increases). As a consequence, the analysis of ρ_{min} can give an insight into the exploration fluctuations. This measure, however, is biased by the update mechanism of ρ_{min}, which depends on many parameters.

3. METHODS

3.1 Exploration measure

In order to measure how the evolutionary algorithm explores through the behavioral space, we need to define an exploration indicator.

We propose an indicator based on the sparseness of each individual among the population because intuitively, the more scattered the population is, the higher the exploration capacity of the algorithm should be.

Assuming we have a distance $d_b(x, y)$ between the behaviors of individuals x and y, we can compute the sparseness of x in its generation with the following formula:

$$s_k(x) = \frac{1}{k} \sum_{j=0}^{k} d_b(x, \sigma_j) \qquad (1)$$

where σ_j is the j-th nearest neighbor of x, and k a fixed parameter determined experimentally. The exploration (or sparseness) measure at generation n is defined as the mean sparseness of the population:

$$E^n = \frac{1}{N} \sum_{x \in pop^n} s_k(x) \qquad (2)$$

where N is the size of the population. If $k = N$ the exploration is then:

$$E^n = \frac{1}{N^2} \sum_{x,y} d_b(x, y)$$

Preliminary tests showed better results with $k \sim 15$ for a population size of 200. If E^n is high, then the population is scattered and we will show empirically that the search is more likely to find a solution quickly. However, the distance $d_b(x, y)$ is yet to be defined.

To define the distance between behaviors $d_b(x, y)$, we will consider *behavior descriptors*. The choice of the descriptors obviously affects how the exploration is computed. For instance, in an ER experiment, one can consider to describe the behavior of the robot by tracing its trajectory, or the stream of its sensors, etc.

The next section covers two neuro-evolution experiments which will be used to validate our exploration measure. In those experiments we will show that exploration is a critical factor, therefore it will have a strong impact on the performance of the experiment.

3.2 Xor And Xor

3.2.1 Artificial Neural Network Setup

The goal of this experiment is to evolve an artificial neural network to compute a Xor And Xor Boolean function $\big[(a \oplus b) \wedge (c \oplus d)\big]$, where a, b, c and d are Boolean values and $\oplus$ is the exclusive "or" operator (xor). The neural network outputs one floating point value for each possible value of its four inputs a, b, c and d, and the fitness function is the sum of errors for each possible value of a, b, c and d.

The problem is solved using an artificial network whose structure and parameters are evolved. See figure 1 for an illustration. For simplicity reasons, we employed a typical graph-based direct encoding for neural networks named **DNN** for Direct Neural Network in which two kinds of mutations are possible:

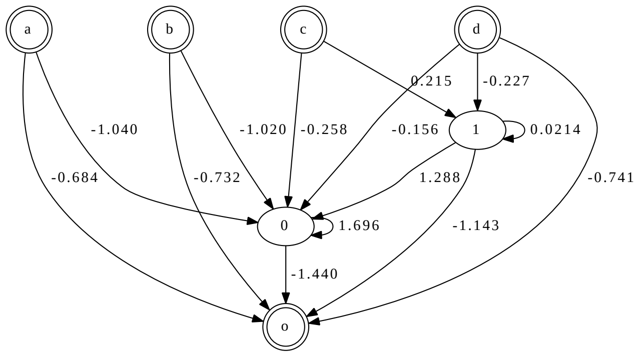

Figure 1: An evolved neural network in the Xor And Xor problem. For each combination of a, b, c and d, the network is simulated during 100 time steps, and if the network is stable, we compare the output o with the expected value of $\left[(a \oplus b) \wedge (c \oplus d)\right]$. If the output is oscillating or unstable, the fitness associated to the network is arbitrarily set to 0. Only connection weights are represented. This network solves the problem with maximum fitness $F_x = 1.0$.

- structural mutation: addition/removal of a random neuron or a random connection;
- parametric mutation: change of a randomly chosen synaptic weight or a neuronal bias;

Cross-over was not used. The parameters of the evolutionary algorithm are described in the appendix.

This problem is very deceptive: by returning 0 for any possible Boolean input the network would be 75% correct (for example if the output neuron o is not connected at all to the input neurons). This way, those degenerate networks are good solutions and are selected by the evolutionary algorithm.

The fitness of an individual x is the sum of errors for each possible set of inputs:

$$F_x = 1 - \frac{1}{16} \sum_{\{a,b,c,d\}=\{0,1\}^4} \left| o_x(a,b,c,d) - \left[(a \oplus b) \wedge (c \oplus d)\right] \right|$$

where $o_x(a,b,c,d)$ is the output of the neural network x for the input set (a,b,c,d). Each neural network is simulated during 100 time-steps. Since we do not constrain the structure of the neural network, nothing prevents it from oscillating (See Figure 1). To avoid such behaviors, we attribute an arbitrary low fitness if the variation of the output is above 0.0001 during the last 10 time-steps.

3.2.2 Behavior description for the exploration measure

To describe behaviors, each individual x is associated with a vector v^x that contains the outputs of the neural network for each of the 16 different input sets:

$$v^x_j = o_x(i_j)$$

where i_j is one of the 16 possible inputs for a, b, c and d. The chosen distance between behaviors is the Euclidian distance:

$$d_b(x,y) = \left|\left|v^y - v^x\right|\right|$$

Table 1: Proposed setups for the Xor And Xor Experiment.

Setup	Goal-oriented objective	Diversity mechanism
1	No	Novelty
2	Yes	Novelty
3	Yes	Diversity
4	Yes	Graph Probing
5	Yes	None

3.2.3 Setups description

Describe here are several setups based on those described in [14]. In this paper, Mouret and Doncieux used a Pareto-based multi-objective optimization in which a second objective is added to explicitly foster diversity. There are many ways to do so, in both genotypic (comparing the *neural networks*) and behavioral spaces (comparing the *behaviors*). The following are the diversity setups we chose:

- genotypic diversity based on graph probing [11]. This method gives a score allowing comparison of the networks based on their topologies.
- behavioral diversity using the previously defined behavior description. The diversity is based on the behavioral distances between itself and individuals in the current population.
- behavioral novelty using the previously defined behavior description. The novelty score is based on the sparseness of the individual among the population and an archive of previously selected behaviors. We will consider this objective alone (as defined in [9]) or together with a goal-oriented one in a multi-objective scheme [14].

The implementation is made using the NSGA-II algorithm [3]. The population size is 200 and the algorithms stops after 1500 generations. The setups are described in Table 1. More information can be found in [14], and the source code is available online at `http://pages.isir.upmc.fr/evorob_db`

3.3 Evolutionary Robotics Experiment

3.3.1 Ball Collector description

In this section we present a challenging simulated Evolutionary Robotics experiment. The task is derived from [4]. It consists of picking up balls in a virtual arena and placing them into a basket (fig. 2). The robot is a two-wheeled robot with the following sensors:

- three wall distance sensors (normalized between 0 and 1);
- two bumpers (1 if it touches a wall, 0 otherwise);
- two ball detection sensors (1 if a ball is in the view field of the sensor, 0 otherwise);
- two switch detectors (1 if the switch is in the view field of the sensor, 0 otherwise);

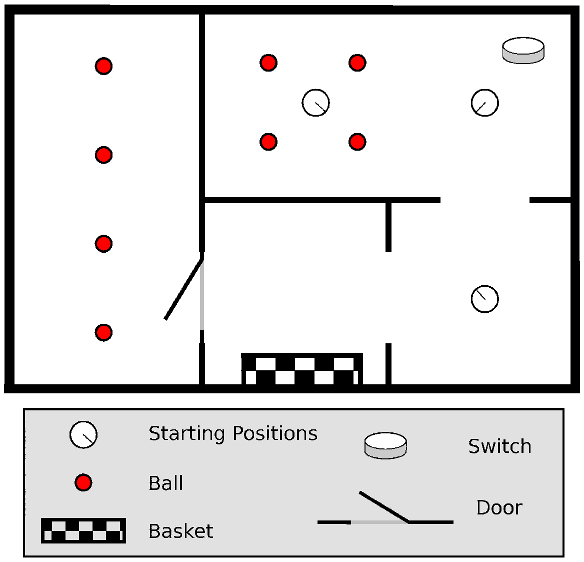

Figure 2: Overview of the arena and of the robot. The goal of the experiment is to place as many balls as possible into the basket. A robot controller is evaluated three times at three different initial positions. To reach the four balls in the left room, the robot has to open the door by first pressing the switch.

- two basket detection sensors (1 if the basket is in the view field of the sensor, 0 otherwise);
- one "ball carrying" sensor (1 if a ball is carried, 0 otherwise).

Two effectors set the left and right wheel motor speeds and the third is an "action" motor: if greater than 0.5, pick up a ball if possible, else throw the carried ball if any; if no ball is carried and greater than 0.5, press the switch if possible). A ball disappears when it is released, and if the robot is close enough to the basket, the ball is collected. The switch has to be pushed only once, and the door remains open for the rest of the run. Each individual is simulated during 4000 time-steps for each of the 3 possible initial positions; the balls and the switch are reinitialized each time.

The main motivation behind the choice of the experiment is that it is very challenging:

- Collecting a ball implies catching a ball, avoiding the walls, reaching the basket, and releasing the ball at the right moment. The individuals are rewarded only when the full sequence is performed.
- After releasing a ball, the robot has to perform a half-turn, and make its way to new balls, which may be hidden behind walls.
- To foster the need of exploration, the robot has to press the switch—which has no direct impact on performance or exploration—to widen its search space, and have more balls within reach.

The goal-oriented objective of a robot is simply the number of balls it collects divided by the maximum number of balls.

Table 2: Proposed setups for the ER experiment.

Setup	Genotype	Goal-oriented objective	Diversity mechanism
1	DNN	No	Novelty
2	DNN	Yes	Novelty
3	DNN	Yes	Diversity
4	DNN	Yes	-
5	ELMAN	No	Novelty
6	ELMAN	Yes	Novelty
7	ELMAN	Yes	Diversity
8	ELMAN	Yes	-

3.3.2 *Encoding*

A robot's genotype is the neural network that controls it. As in [4], we have chosen two different setups for the controller:

- **DNN** - The previously defined encoding with evolving structure.
- **ELMAN** - A network with fixed structure as described in [6]. The number of hidden context units was arbitrarily set to 20.

3.3.3 *Behavior description for the exploration measure*

Partial *behavior descriptors* are used to characterize a robot behavior. The descriptor should carry enough information to differentiate very different behaviors from close ones. Several approaches have been tried, such as a Hamming description, Normalized Compressed Distance, Fourier analysis, etc. [4]. Based on preliminary tests, we selected the following two descriptors:

- **Trajectory Description**: The position of the robot is recorded and discretized every 50 time-steps. The vector of positions represents the behavior. The distance between two behaviors is the Edit distance between vectors, derived from [18]:
  - the cost of an insertion or deletion is 1;
  - the cost of a substitution is
  $$c(a^{(t)}, b^{(t)}) = \left|x_{a^{(t)}} - x_{b^{(t)}}\right| + \left|y_{a^{(t)}} - y_{b^{(t)}}\right|$$
  where $x_{a^{(t)}}$ and $y_{a^{(t)}}$ are the normalized coordinates of the robot a at time-step t.

  The substitution cost can never exceed the cost of a deletion plus an insertion. This distance outperforms a basic Euclidian distance between trajectories because it acts as an alignment distance: the distance between two trajectories that are only shifted in time will be smaller than with the Euclidian distance, which allows a better comparison of trajectories.
- **Ad hoc Description**: This task is centered on what the robot does with the balls. As a consequence, a natural *ad hoc* descriptor is the latest position of the balls, whether they stayed at their initial position or were moved and dropped. The descriptor is then a vector v with all the final positions of the balls, and the

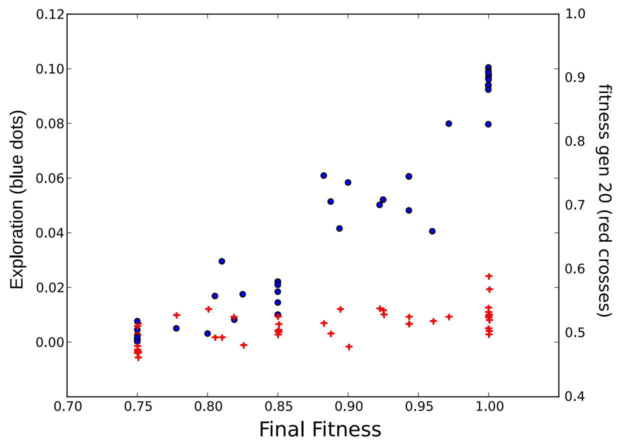

Figure 3: (dots) Scatterplot between fitness after 800 generations and exploration generation 20 for the Xor And Xor Experiment. The runs are picked from each of the previous setups except for the single objective novelty search. There is a positive correlation between the two variables. (crosses) Scatterplot between final fitness and fitness at generation 20. There is almost no correlation.

distance between two vectors v and u is the Euclidian distance between the two vectors $||u - v||$

Being the most generic, the **Trajectory Description** is chosen to compute the exploration indicator. The **Ad hoc Description** will be used in some of the setups described below.

3.3.4 *Description of the Setups*

As in the Xor And Xor problem, we define several setups supposed to have a different exploration capacity. For some of the setups we added a diversity maintenance goal-independant objective.

Different behavior descriptors are used for exploration measure and diversity maintenance to check if promoting behavioral diversity in a behavioral space also enhances exploration in another behavioral space.

Here are the considered diversity mechanisms:

- Behavioral diversity based on the previously defined **Ad hoc Description**.
- Behavioral novelty based on the **Ad hoc Description**.

As in the Xor And Xor experiment, both single-objective and multi-objective approaches are considered. Table 2 summarizes the considered setups.

The eight previous setups were implemented using the $Sferes_{v2}$ framework [12]. The parameters are detailed in the Appendix.

4. RESULTS

4.1 Xor And Xor Experiment

In Figure 4 the leftmost plot shows the fitness of the resulting neural networks for the Xor And Xor problem, between

Table 3: Correlation between fitness at generation 800 and sparseness or fitness at generation g for all the Xor And Xor Experiments.

Generation g	0	20	40	60	80	100
Corr. Final Fit/ Sparseness	0.46	0.71	0.71	0.69	0.70	0.74
Corr. Final Fit/ Fitness at gen	0.05	0.22	0.14	0.22	0.17	0.26

1 and 1500 generations. These results confirm the previous work on this problem as reported in [14]:

- The objective-only search falls into the deceptive trap in 90% of the runs, so its fitness almost never exceeds 0.75.
- The novelty-only approach had much more diverse scores, sometimes over the 0.75 deceptive trap, but could never attain the maximum scores.
- Most of the Novelty and Diversity Multiobjective runs could achieve maximum fitness before the 1500 generations.

The rightmost graph shows two results. If we temporarily put aside the case of novelty-only, we can see that the exploration measure converges to a stable value after 20 generations only and we notice statistical differences between the four setups ($p < 0.0001$ for each four setups compared from top to bottom, Mann-Withney U test). Moreover, there is a strong correlation between the exploration measure and the final fitness of the run: setups with low exploration do not achieve the maximum fitness while those with a high exploration always achieve greater fitness, and faster. This can be seen on the scatterplot (Figure 3) of fitness at generation 800 (the relative performance of the setups is well represented by fitness at generation 800) against the measure of exploration at generation 20. In Table 3 the first line presents the Pearson's correlation values between exploration and final fitness for all runs. The correlations between variable X and Y are computed using the following formula:

$$\text{corr}(X, Y) = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$

where μ_X is the expected value of X and σ_X its standard deviation.

The correlation is high even during the first generations: one can rank the exploration ability of the setups after generation 20, while in the first generations the fitness of the different setups are indistinguishable (at generation 100 for instance, we have $p > 0.1$ for most of the setup comparisons - Novelty only still set aside). This is confirmed by Table 3, which shows that the correlation between the fitness at the first generations and the final fitness is much weaker than the correlation between the exploration measure and the final fitness. Therefore, using only fitness, one must wait for many more generations (often even until the end of the runs) to infer the final fitness of the setup. **This makes the exploration measure a good and fast predictor to the relative global performance of different setups: the more exploration, the better the final performance.**

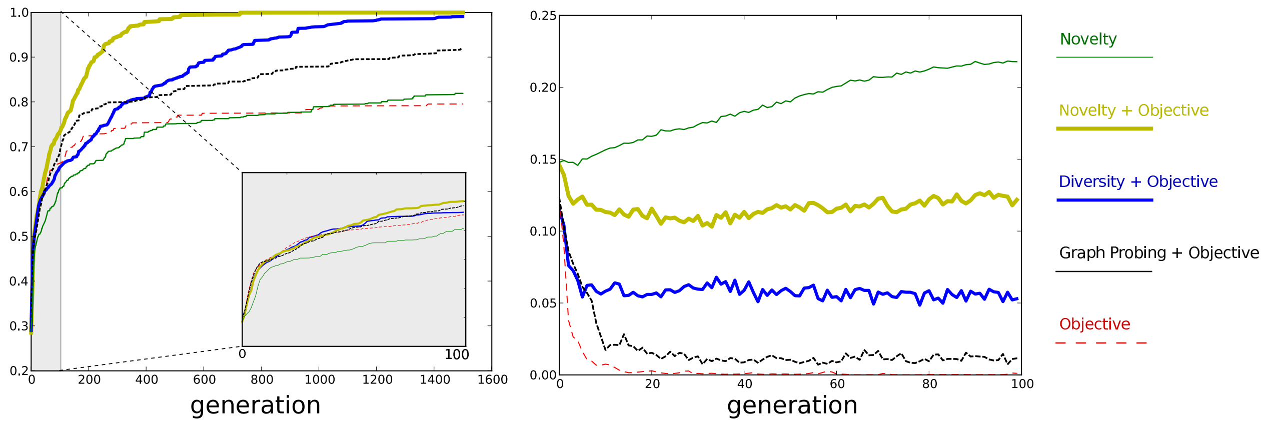

Figure 4: Fitness results for the Xor And Xor Setup. The left plot corresponds to the fitness for 5 different setups (the inset is a zoom on the first 100 generations). The right plot shows the exploration measure based on the behavioral distance, for the 100 first generations. The curves are averaged over 30 runs, for a population of 200 individuals, and the neural network encoding is DNN.

Table 4: Correlation between final fitness and sparseness or fitness at generation g for the Evolutionary Robotics experiment with DNN.

Generation g	0	20	40	60	80	100
Corr. Final Fit/ Sparseness	0.25	0.62	0.70	0.72	0.79	0.76
Corr. Final Fit/ Fitness at gen	0.0	0.38	0.48	0.53	0.62	0.63

Table 5: Correlation between final fitness and sparseness or fitness at generation g for the Evolutionary Robotics experiment with ELMAN.

Generation g	0	20	40	60	80	100
Corr. Final Fit/ Sparseness	0.46	0.54	0.58	0.61	0.64	0.64
Corr. Final Fit/ Fitness at gen	0.0	0.02	0.02	0.25	0.26	0.26

Meanwhile, the novelty only (which rewards only the exploration) has the best exploration measure, but has a poor fitness. It probably results from the fact that there is no pressure towards exploitation. On the other hand, the objective only search, which has a poor exploration capacity, also has poor final fitness. This confirms the importance of a good trade-off between exploration and exploitation in this setup. The use of exploration measure as a relative performance predictor is then only relevant to compare setups that have a similar exploitation capacity.

4.2 Ball Collecting Experiment

The previous results show that the measure of exploration can bring useful information about the performance of the search even within the first generations. This section will now cover the ER setup, which has a much higher complexity.

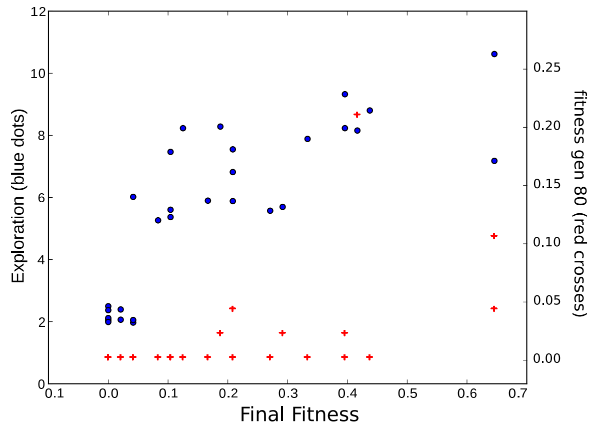

Figure 5: (dots) Scatterplot between final fitness after and exploration at generation 80 for the ER setup. The points correspond to each of the six previous setups. There is a positive correlation between the two variables. (crosses) Scatterplot between final fitness and fitness at generation 80. There is almost no correlation.

In Figure 6, the leftmost plot presents the mean fitness of the resulting robot behavior. As the problem is very complex, the mean fitness can sometimes be very low. However, some of the best individuals could collect up to 21 out of 24 balls. Even if the fitness is low because of the difficulty of the task, the plot exhibits a clear separation between all final fitnesses of the runs (comparison from top to bottom: $p < 0.001$ for each pair of setups). Meanwhile at generation 100, only Novelty+Objective setup can be statistically separated from all others, on the basis of the fitness only.

We can see that without artificially fostering behavioral diversity, both of the neural networks setups have very poor results. The runs using ELMAN also performed poorly compared to those using DNN: This could be a result of the arbitrary chosen parameters of the network topology. The novelty only setup is much less efficient than multi-objective ones, but also outperforms the objective-only setup.

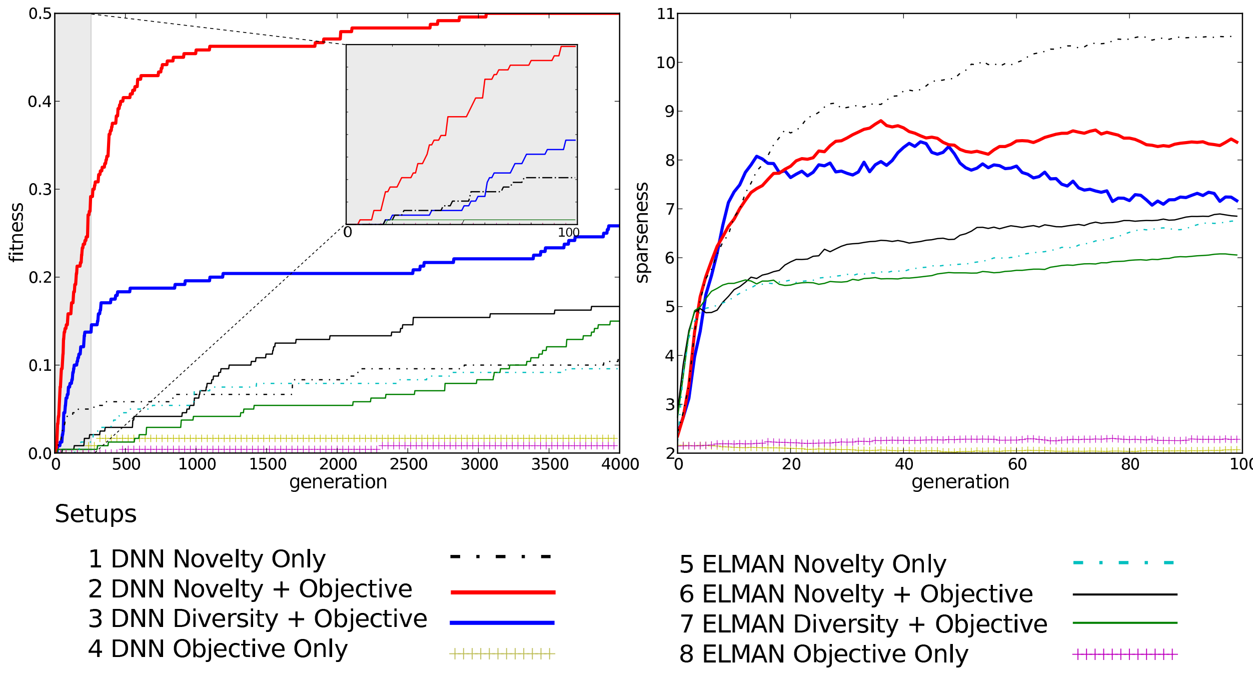

Figure 6: (left) Mean number of collected balls for Elman or DNN encoding, with different behavior diversity preserving setups. The inset shows the fitness for the first 100 generations. (right) Exploration measure based on the trajectory distance for the 100 first generations. As the run are much longer to perform, the curves are averaged over 10 runs only, for a population of 200 individuals.

As in the Xor And Xor case, the novelty-only setup has a high exploration measure compared to other runs, but lacks selective pressure towards exploiting the best individuals; therefore its final fitness is rather low.

Considering the other 6 runs, the right plot of Figure 6 indicates that the final fitness is correlated to the exploration measure, even in this complex setup. This can be interpreted in the following way: this task requires the robot to move around the arena; if the exploration in the trajectory space is high, the robot will more likely solve the task. Figure 5 is a scatterplot of the exploration during the first generations against the final fitness for each run of each setup.

The correlation at different generations is indicated in Table 4 for DNN and Table 5 for ELMAN. In both experiments, the correlation between exploration and final fitness is higher than the correlation between fitness in the first generations and final fitness.

5. DISCUSSION

Exploration is a key factor, and considering the complexity of the mapping between the genotype and the exhibited behavior, characterizing how well a setup explores in the behavioral space is of critical importance. Our straightforward approach can already catch a glimpse of how the exploration is related to performance. In the considered experiments, the *novelty* objective outperforms *diversity* in terms of fitness as well as exploration. The novelty alone, however, has its results damaged by the lack of exploitation. The conflict between exploration and exploitation has been highlighted by our results: the Novelty + Objective traded a part of its exploration capacity for exploitation. Likewise, for all setups using the problem-dependent objective, for which we may then hypothesize that exploitation is at the same level, exploration revealed to be strongly correlated to performance.

When evolving behavioral systems on challenging problems, it is frequent to get only failed runs, *i.e.* runs with a low fitness. In this case, no conclusion can be drawn and the experimenter usually starts to dive in a frustrating trial and error loop. By measuring the exploration, differences between runs can be identified and those that explored better can be considered as more promising.

This measure may also help monitor a run: a premature convergence can be identified by a sudden drop of exploration and likewise, a rapid increase in the exploration may reveal an important discovery. For instance, in the ball collector experiment, there is an increase in the exploration measure when the robot manages to open the door.

When looking at the correlation tables (Tables 3, 4 and 5), correlation between fitness at the first generation and final fitness is surprisingly high. We would expect—or at least hope—that the random initialization of the population has almost no impact on the run. Further analysis shows that the correlation at generation 0 for the ELMAN setup is much higher than for the DNN setup. As the number of parameters in ELMAN are much higher at generation 0 (more than 500 parameters evolved here), while in DNN all starting neural networks have the same topology (no hidden neuron) and have only about 40 parameters (even though this number grows fast when the topology evolves). This search space dimensionality could partly explain why the random initialization has a stronger impact on ELMAN than on DNN. The exploration at generation 0 could prove useful to identify the impact of initialization on the run, an impact that we may want to minimize.

The exploration during the first generations of the algorithm gives good insight to the final fitness. This could lead to several possibilities:

- a Restart Algorithm. Using an on-line measure of the exploration, determine a criterion for generating a new population and restart the run. As the exploration indicator is a relative indicator, it may rely on a previously evaluated threshold or on a sudden drop of exploration;
- as in [19], an on-line exploration measure could determine whether the algorithm should explore or exploit;
- a parameter setting application. Parameter setting requires a lot of evolutionary runs, which usually are, in ER for instance, CPU-intensive. Using the proposed method we could reduce the number of generations drastically and still be able to determine the parameter set quality, thus significantly reducing the computational time.

6. CONCLUSION

We have shown that the behavioral sparseness of the population in difficult neuro-evolution problems is highly related to the exploration of the evolutionary algorithm in the behavioral space. In many problems this exploration rather than the exploitation might be the bottleneck of the search. In those problems, there is a strong correlation between the performance of the algorithm and the exploration, while the fitness is not always a good indicator of how good an algorithm is (because of deceptive problems for instance). We have shown that the behavioral exploration measure is a good alternative: exploration can bring better information on a setup than just fitness comparisons. Future work might use it, for instance, as a tool to compare different setups, to tune algorithms parameters or to monitor runs.

7. REFERENCES

[1] F. Dario and N. Stefano. *Evolutionary Robotics.* MIT Press, 2000.

[2] K. De Jong and W. Spears. A formal analysis of the role of multi-point crossover in genetic algorithms. *Annals of Mathematics and Artificial Intelligence*, 5(1):1–26, Mar. 1992.

[3] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Parallel Problem Solving from Nature PPSN VI*, pages 849–858. Springer, 2000.

[4] S. Doncieux and J.-b. Mouret. Behavioral diversity measures for Evolutionary Robotics. *IEEE Congress on Evolutionary Computation*, pages 1–8, 2010.

[5] A. Eiben and C. Schippers. On evolutionary exploration and exploitation. *Fundamenta Informaticae*, 35(1-4):35–50, 1998.

[6] J. L. Elman. Finding Structure in Time. *Cognitive Science*, 14(2):179–211, Mar. 1990.

[7] T. Friedrich, P. S. Oliveto, D. Sudholt, and C. Witt. Theoretical analysis of diversity mechanisms for global exploration. *Proceedings of GECCO '08*, pages 945–952, 2008.

[8] F. J. Gomez. Sustaining Diversity using Behavioral Information Distance. *Proceedings of GECCO '09*, pages 113–120, 2009.

[9] J. Lehman. Exploiting open-endedness to solve problems through the search for novelty. *Proceedings of Alife XI*, (Alife Xi):71–80, 2008.

[10] J. Lehman and K. O. Stanley. Abandoning Objectives: Evolution through the Search for Novelty Alone. *Evolutionary Computation*, 0(ja):1–34, 2010.

[11] D. Lopresti and G. Wilfong. A fast technique for comparing graph representations with applications to performance evaluation. *International Journal on Document Analysis and Recognition*, 6(4):219–229, Apr. 2003.

[12] J.-b. Mouret. Sferes v2: Evolvin-in the multicore world. *IEEE World Congress on Computational Intelligence, CEC*, pages 4079–4086, 2010.

[13] J.-B. Mouret and S. Doncieux. Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity. *2009 IEEE Congress on Evolutionary Computation*, pages 1161–1168, May 2009.

[14] J.-b. Mouret and S. Doncieux. Using Behavioral Exploration Objectives to Solve Deceptive Problems in Neuro-evolution. *Proceedings of GECCO'09*, pages 627–634, 2009.

[15] I. Paenke, J. Branke, and Y. Jin. On the Influence of Phenotype Plasticity on Genotype Diversity. *2007 IEEE Symposium on Foundations of Computational Intelligence*, (Foci):33–40, Apr. 2007.

[16] S. Risi, C. E. Hughes, and K. O. Stanley. Evolving Plastic Neural Networks with Novelty Search. *Adaptive Behavior*, 18(6):470–491, 2010.

[17] T. Smith, A. Philippides, and P. Husbands. Neutrality and ruggedness in robot landscapes. *Proceedings of CEC*, pages 1348–1353, 2002.

[18] L. Trujillo and G. Olague. Discovering several robot beaviors through speciation. *Applications of Evolutionary Computing*, 4974:164–174, 2008.

[19] R. Ursem. Diversity-guided evolutionary algorithms. *Parallel Problem Solving from Nature PPSN VII*, 2439:462–471, 2002.

APPENDIX

A. PARAMETERS

- MOEA: NSGA-II (pop. size : 200)
- DNN (direct encoding):
  - prob. of changing weight/bias: 0.1
  - prob. of adding/deleting a conn.: 0.15/0.25
  - prob. of changing a conn.: 0.1
  - prob. of adding/deleting a neuron: 0.025/0.025
  - activation function for neurons: $y_i = \varphi\left(\sum_j w_{ij} x_j\right)$ where $\varphi(x) = \frac{1}{1+\exp(b-kx)}$
- ELMAN (fixed structure):
  - prob. of weight/bias change: 0.1
  - prob. of changing a conn.: 0.1
  - number of hidden and context units: 20
- Source code : `http://pages.isir.upmc.fr/evorob_db`