How Do Different Encodings Influence the Performance of the MAP-Elites Algorithm?

Danesh Tarapore
University of York
Department of Electronics
danesh.tarapore@york.ac.uk

Jeff Clune
University of Wyoming
Evolving AI Lab
jeffclune@uwyo.edu

Antoine Cully
Imperial College
Personal Robotics Lab
a.cully@imperial.ac.uk

Jean-Baptiste Mouret
Inria Nancy Grand-Est
CNRS UMR 7503
Université de Lorraine,
jean-baptiste.mouret@inria.fr

ABSTRACT

The recently introduced Intelligent Trial and Error algorithm (IT&E) both improves the ability to automatically generate controllers that transfer to real robots, and enables robots to creatively adapt to damage in less than 2 minutes. A key component of IT&E is a new evolutionary algorithm called MAP-Elites, which creates a behavior-performance map that is provided as a set of "creative" ideas to an online learning algorithm. To date, all experiments with MAP-Elites have been performed with a directly encoded list of parameters: it is therefore unknown how MAP-Elites would behave with more advanced encodings, like HyperNeat and SUPG. In addition, because we ultimately want robots that respond to their environments via sensors, we investigate the ability of MAP-Elites to evolve closed-loop controllers, which are more complicated, but also more powerful. Our results show that the encoding critically impacts the quality of the results of MAP-Elites, and that the differences are likely linked to the *locality* of the encoding (the likelihood of generating a similar behavior after a single mutation). Overall, these results improve our understanding of both the dynamics of the MAP-Elites algorithm and how to best harness MAP-Elites to evolve effective and adaptable robotic controllers.

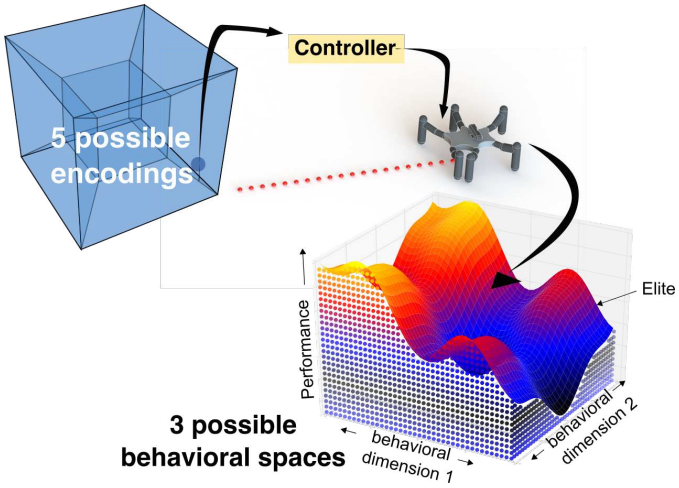

Figure 1: After a user chooses phenotypic (or behavioral) dimensions of interest, MAP-Elites searches for the highest performing solution of each type. We investigate the impact on the performance of MAP-Elites of 5 different robot controller encodings.

Keywords

MAP-Elites; neuroevolution; evolutionary robotics; intelligent trial and error; divergent search

1. INTRODUCTION

Despite recent, impressive advances in robotics [24], robots are still unable to adapt to situations that were not expected by their designers [5]. Instead of relying on diagnosis and planning procedures, which is the traditionnal approach to deal with unexpected situations [27, 2], robots could learn new behaviors by trial-and-error, as animals do. However, trial-and-error learning algorithms face a trade-off [15]: on one hand, they have to explore a search space that is as small as possible to be fast enough to work on robots in real time; on the other hand, they have to explore a search space that is as large as possible, so that they can be creative enough to handle truly unexpected situations.

The recently published "Intelligent Trial & Error" algorithm (IT&E) [9] made a first step to reconcile these two antagonistic objectives. In this algorithm, creativity is achieved thanks to an evolutionary algorithm that explores a large search space with a simulation of the basic, intact robot. Fast on-line adaptation is then achieved with a Bayesian optimization algorithm [4] based on Gaussian processes, which operates on a low-dimensional search space created by the

GECCO '16 July 20-24, 2016, Denver, CO, USA

ACM ISBN 978-1-4503-4206-3/16/07.
DOI: http://dx.doi.org/10.1145/2908812.2908875


evolutionary algorithm. IT&E enabled a 6-legged robot to adapt to 5 different damage conditions in less than two minutes (about 15 trials) [9], and a planar robotic arm to adapt to 14 different damage conditions in less than 1 minute [9].

This paper focuses on the evolutionary algorithm used in the first step of IT&E, called MAP-Elites [19]. Traditional evolutionary algorithms aim to return the single highest-performing solution in a search space; instead, MAP-Elites creates a *map* of high-performing solutions (the elites) at each point in a behavioral space defined by *behavioral dimensions*, which are dimensions of variation of interest to the user (e.g. height, cost, energy efficiency). In summary, MAP-Elites searches for the best behavior for each type of phenotype of behavior and stores them in a map such that different areas of the map contain different types of solutions. These new kind of algorithms are called *illumination algorithms* [19] or *quality diversity algorithms* [23].

In addition to the damage recovery experiments in the IT&E paper [9], MAP-Elites has also illuminated the relationships between connection costs, modularity, and fitness in neural networks [19, 7], the design space for walking "soft robots" encoded with a CPPN [19, 25], and the working space of a physical, soft robotic arm [19]. It has also been used to evolve neural networks that drive a simulated mobile robot trough mazes [23]. In another line of work, Nguyen et al. harnessed MAP-Elites to generate images that "fool" deep neural networks [21] and to produce "Innovation Engines," which attempt to automate creativity by generating a large number of different, functional, and interesting solutions (artistic images, in this case) [22].

Being an illumination algorithm, MAP-Elites has important differences from traditional evolutionary algorithms (EAs), despite having a variation/selection loop. Specifically, (1) It is not an optimization algorithm because it does not aim at finding a single solution, but instead has many niches representing different types of solutions and returns the highest performing solution of each type; (2) It does not have a fixed-size population; (3) It maintains a very large number of niches (10,000+ in the damage recovery experiments in [9]), which are all potential stepping stones to build more complex solutions [19, 22]. These differences create different algorithmic dynamics—for example, increasing evolvability by allowing frequent "goal switching" [22]—and they may force us to revise some of our intuitions about evolutionary algorithms. MAP-Elites can be seen as simplistic model of natural evolution in which new niches can be "discovered" (when the cell is initially empty) or "invaded" by individuals that come from other niches. Because individuals that reach a new niche are kept regardless of their performance, there is an implicit bonus for novelty. Because only the best individual of each niche can have offspring, there is also a performance pressure.

Here, we investigate the influence of the encoding on MAP-Elites. To avoid conclusions specific to a particular behavior space, each experiment is conducted in 3 different behavior spaces. We focus on the evolution of legged locomotion controllers because MAP-Elites has demonstrated its usefulness for this task [9], and because legged locomotion has been widely used as a benchmark for encodings [13, 3, 6, 8, 18, 26]. In addition, real walking robots would benefit from controllers that can adapt gaits online to small variations in terrain, which is why we chose to mainly evolve closed-loop controllers.

We consider five different state-of-the-art encodings for locomotion controllers: (1) for comparison, the original, directly encoded, parameterized open loop controller from [9], (2) a CPPN [25] that specifies the parameters of a network of non-linear oscillators (Central Pattern Generators) [14], (3) a CPPN that specifies the weights of a neural network [6, 8], (4) a CPPN used as a Single Unit Pattern Generator [18] (SUPG), and (5) a frequency constrained SUPG variant, which was required to avoid breaking the physical robot.

It has been previously shown that the selected encodings have very different evolvability signatures [26], which is why we expect the results of MAP-Elites to be different for each of them. For instance, mutating a high-performing individual in SUPG can generate very different, yet still high-performing, individuals [18, 26], while mutations to the direct encoding of periodic signals generates individuals with similar behavior and fitness [9, 26]. Additionally, HyperNEAT can produce very different individuals with a significant drop in fitness [8, 26]. All these encodings are capable of generating high-performing gaits with a traditional evolutionary algorithm (NSGA-II + behavioral diversity) [26]. However, SUPG and HyperNEAT provided more evolvability when the body was changed (e.g. a leg of the simulated robot was removed) during evolution [26].

All comparisons are performed in simulation. Nevertheless, to understand how a different encoding can affect MAP-Elites and, in turn, IT&E for damage recovery, we compare two very different encodings from our experiments on a real, physical robot. As described below, we chose an encoding that performed well in our experiments yet was closed-loop, in an effort to advance IT&E to the type of controller required for real-world robotics challenges.

2. METHODS

2.1 The MAP-Elites algorithm

Evolutionary Algorithms are traditionally viewed as optimization algorithms that can work on large search spaces and do not require a differentiable objective function. Nevertheless, while the ability of natural evolution to optimize lifeforms in many aspects is certainly impressive, its creativity might be even more striking. As Darwin put it: "there is grandeur in this view of life [...] that, from so simple a beginning, endless forms most beautiful and most wonderful have been, and are being, evolved" [12].

The Novelty Search (NS) algorithm [16] made the evolutionary computation community realize that, instead of optimizing an objective, an EA could instead search for individuals that are different from those previously encountered. By doing so, the EA becomes an endless generator of new, potentially interesting things: that is, it becomes a creative algorithm. To compare candidate solutions, NS introduced the concept of the *behavior space*: when evolving dynamical systems, and in particular when evolving robot controllers, many genotypes lead to the same behavior (e.g. not moving at all); as a result, searching for things that behave differently is much more interesting than simply searching for new genotypes. Experimental results suggest that encouraging exploration in the behavior space is always beneficial, whatever the behavior description is [20].

A pure exploration algorithm like NS could, for example, find many ways for a robot to walk. However, we are often

Algorithm 1 A pseudocode description MAP-Elites.

```
procedure MAP-ELITES
    (P ← ∅, X ← ∅)
    for iter = 1 → I do
        if iter < G then
            x' ← random_solution()
        else
            x ← random_selection(X)
            x' ← random_variation(x)
        b' ←behavior_descriptor(x')
        p' ←performance(x')
        if P(b') = ∅ or P(b') < p' then
            P(b') ← p'
            X(b') ← x'
    return feature-performance map (P and X)
```

not only interested in finding these many different types, but also in finding the best (e.g. fastest) of each type. For the walking robot example, knowing that a 6-legged robot can move by using only four legs is interesting, but it is often both more useful and more computationally tractable to find the best way to walk with four legs, along with the best way to walk with 6 legs, etc.

We call such solutions the *elites* of the search space, that is, the best solution of type. More precisely, an elite is the best solution within a subspace of the behavior space. Biologically, an elite corresponds to the most fit species within a specific niche. Because finding all the elites is a promising approach to visualize a high-dimensional search space and to gain insights about it, we call such algorithms *illumination algorithms* [19]. Finding the elites is also a novel way to discover many high-performing solutions at the same time, which is why Pugh *et al.* proposed the name *quality diversity algorithms* [23]. For example, a quality diversity algorithm can allow a walking robot to create a repertoire of controllers that each specialize in helping a robot walk in a different direction [11].

The closest algorithm to MAP-Elites is BR-Evolution [11, 10], which is a derivative of Novelty Search with Local Competition [17] (NSLC). Both BR-Evolution and NSLC are multi-objective EAs that rank individuals according to two objectives: their novelty (average distance to the n closest neighbors), and their local performance (rank in performance vs. the n closest neighbors). However, the result of in NSLC is the current population, whereas the result in BR-Evolution is the novelty archive. Both BR-Evolution and MAP-Elites search for the elites of the search space [23, 19] but MAP-Elites is simpler, both conceptually and to implement, and at least as good at finding elites [23].

Algorithm 1 shows the pseudo-code of MAP-Elites. Given a performance measure performance($\mathbf{x}$), where $\mathbf{x}$ represents a candidate solution, the user chooses N behavioral dimensions that are measured by behavioral descriptors. For instance, the dimensions for evolving robot morphologies could be the height and weight of the robot. Each dimension of variation is then discretized based on user preference or available computational resources, leading to an N-dimensional grid where each cell can be seen as a niche—a different type of solution—that MAP-Elites will try to fill with the best performing solution of that type.

MAP-Elites starts by randomly generating G genomes and determining the performance and phenotypic features of each. Those genomes are placed into the cells to which they belong in the behavior space (if multiple genomes map to the same cell, the highest-performing one per cell is retained). At that point the algorithm is initialized, and the following steps are repeated until a termination criterion is reached. (1) An individual from the current map (an elite) is randomly chosen and its genome produces an offspring via mutation (or possibly crossover, which we did not use for simplicity). (2) The behavior and performance of that offspring are determined, and the offspring is placed in the cell if the cell is empty or if the organism is higher-performing than the current occupant of the cell, in which case that occupant is discarded.

2.2 Direct encoding of periodic signals

Walking controllers evolved with direct encodings are designed to be simple. The actuation along each robot DOF is governed by mathematical functions that are directly parametrized by the genotype. Following [9], in this paper that function is a periodic signal of an open-loop oscillator generated by a signal of frequency 1Hz, parametrized by its amplitude, its phase, and its duty cycle (the proportion of each period that the joint angle is positive). There are thus 3 parameters for each DOF, meaning 6 parameters per leg (each leg has three actuated servos, but the angular position of one of them is not controlled). Therefore each hexapod locomotion controller is fully described by 36 parameters (details in [9]).

2.3 Indirect/generative encodings

Generative encodings create an indirect (e.g. non-linear) mapping between the genotype and the phenotype (e.g. a neural network controller). For this study, Compositional Pattern Producing Networks (CPPNs) [25] generate this indirect mapping. CPPNs abstract the processes of embryonic development by determining the attributes of phenotypic components as a function of their geometric location in the phenotype. Our CPPN genome is a directed graph of nodes connected by weighted links, with the following possible activation functions for each node: *Sine*, *Gaussian*, *Sigmoid*, and *Linear*. The CPPNs themselves are directly encoded and evolved with traditional mutation operators for neural networks that add and remove nodes or change connection weights.

Closed-loop CPGs with CPPNs (CPG). Our closed-loop, generatively encoded CPG is composed of 12 coupled amplitude-controlled phase oscillators [14], governing the actuation of the 12 servos (two per leg). We designed the oscillators to exhibit a limit cycle behavior, producing a stable periodic output. All 12 CPG oscillators have the same frequency (1 Hz), and inter-oscillator couplings are bilaterally symmetrical. Each oscillator is modeled by the following set of ordinary differential equations:

$$\dot{\theta}_i = 2\pi + \sum_j \alpha_j w_{i,j} \sin\left(\theta_j - \theta_i - \phi_{i,j}\right) \quad (1)$$

$$\ddot{\alpha}_i = \beta\left(\frac{\beta}{4}\left(\mathrm{A}_i - \alpha_i\right) - \dot{\alpha}_i\right), \gamma_i = \alpha_i \cos\left(\theta_i\right) \quad (2)$$

where α_i and θ_i, respectively, denote the amplitude and phase of the ith oscillator, and A_i represents its intrinsic amplitude (in rad); α_i converges to A_i at a rate determined by

positive constant β (set to 10 rad/s for rapid convergence). Inter-oscillator couplings are defined by weights $w_{i,j}$, and phase biases $\phi_{i,j}$. The coupling weights influence the time to synchronize between oscillators (set to 20 to enable rapid inter-oscillator synchronization). The phase bias $\phi_{i,j}$ determines the phase difference between oscillators i and j. The output of the oscillator γ_i governs servo actuation.

A CPPN encodes the intrinsic amplitudes A_i and the inter-oscillator phase biases $\phi_{i,j}$ of all 12 oscillators. The oscillators are placed in a 2-D Cartesian grid called the *substrate* [25] in a way that reflects the hexapod robot morphology while providing each oscillator a distinct (x, y) coordinate (exact locations are listed in [26]). The CPPN is then queried with these geometric locations and outputs the intrinsic amplitudes and phase biases for each servo. The closed-loop aspect of the system is achieved by phase-resetting mechanism [1], which is triggered by the touch sensors attached to each of the six legs of the hexapod. The total number of CPG parameters generatively encoded by the CPPN is 23 (12 intrinsic amplitudes and 11 phase biases, for details see [26])

ANNs with CPPNs (HyperNN). The second generative encoding scheme is a simplified version of the HyperNEAT indirect encoding, in which NEAT is replaced by a basic direct encoding for neural networks. The CPPNs encode the weights of input-hidden and hidden-output neuron connections of a fixed topology, single-layer feedforward ANN. ANN neurons are positioned in the substrate to reflect the hexapod robot morphology (details in [26]). For each connection, these locations are input to the CPPN and it specifies the weight of that connection (details in [26]).

The ANN inputs are the previously requested angles for each of the 12 pivot joints of the 6-legged robot. Additionally, to facilitate periodic oscillations, sine and cosine waves of frequency 1 Hz are input to the ANN. The ANN outputs at each time-step are 12 numbers, scaled to the allowable angular range, and indicate the next position of each of the motors (for more details, see [26]).

SUPGs with CPPNs (SUPG). In the third generative encoding scheme, the CPPN encodes the attributes of a SUPG. The SUPG is a macro-neuron that, upon being triggered, produces a single cycle of a CPPN encoded activation pattern. Consequently, the repeated triggering of the SUPG results in temporal oscillations. The CPPN is input the position of the SUPG in the substrate, and the elapsed time (in interval $[0, 1]$) since the SUPG was last triggered. During the period of the SUPG, its internal, individual timer ramps upwards with each simulation time-step, from an initial value of 0 to a maximum value of 1. Consequently, the resultant activation pattern output by the SUPG is a function of both its position in the substrate and the time since the last cycle was initiated. Applying the SUPGs for hexapod locomotion, the substrate comprises 12 SUPGs positioned to reflect the robot morphology. The outputs of the SUPGs at each time-step specify the desired angles for the two servos on each leg of the hexapod (details in [26]).

For each SUPG, the internal timer can be restarted following the occurrence of an external trigger event. Consequently, the period of the SUPG-generated oscillations does not need to be predefined. Rather, the oscillation period can be adjusted depending on the gait and the terrain by restarting the SUPG whenever its associated foot touches the ground. Therefore, the two SUPGs on each leg of our hexapod robot are triggered by that leg's foot touching the ground, producing closed-loop control.

SUPGs with constrained CPPNs (C-SUPG). The SUPG is capable of generating oscillatory signals with frequencies far exceeding 1 Hz (see [26]). Such high frequency gaits are likely to result in overly taxed servos [28]. To constrain the SUPG oscillatory signal frequency to 1 Hz, we constrain the CPPN so that the sum of weights of neural connections between the timer input and phase output neurons is hard-wired to 2π (1 Hz = 2π radians/s). Furthermore, mutations to add and remove neurons and connections are disabled between these two neurons so that evolution cannot produce higher frequency gaits. Thus, behavior-performance maps evolved with C-SUPG may be tested on the physical robot.

2.4 Behavioral descriptors

We analyze the encodings with 3 different behavioral descriptors (denoted $\mathbf{x}$) — duty factor, body orientation, and relative ground reaction force (GRF) — each described by a vector of 6 behavioral descriptor values (described below), meaning a 6-dimensional behavior map for each of the 3 behavioral descriptors. Each dimension is in the range $[0, 1]$, and discretized at these five values: $\{0, 0.25, 0.5, 0.75, 1\}$. The three descriptors were selected for the differences in the maximum number of solutions they allow to be stored in the map: 15625 for the duty factor, 3375 for the body orientation, and 235 for the relative ground reaction force (computed analytically and from experiment data); these differences were consequent to constraints imposed by the dependencies between the 6 dimensions of the descriptors (for instance, an angle cannot be both positive and negative at the same time, or the weight of the robot has to be distributed among each contact point).

Duty factor: the proportion of time that each leg is in contact with the ground.

$$\mathbf{x} = \left[\frac{\sum_t C_1(t)}{\text{numTimesteps}}, \cdots, \frac{\sum_t C_6(t)}{\text{numTimesteps}} \right] \tag{3}$$

where $C_i(t)$ denotes the Boolean value of whether leg i is in contact with the ground at time t (1: contact, 0: no contact), recorded at each time step (every 15 ms) the controller is simulated, averaged over numTimesteps of simulation.

Body orientation: changes in the angular orientation of the hexapod during locomotion, measured as the proportion of 15ms intervals that each of the pitch, roll and yaw angles of the robot frame are positive (three dimensions) and negative (three additional dimensions).

$$\mathbf{x} = \left[\begin{array}{l} \frac{1}{K}\sum_k U(\Theta(k) - 0.005\pi) \\ \frac{1}{K}\sum_k U(-\Theta(k) - 0.005\pi) \\ \frac{1}{K}\sum_k U(\Psi(k) - 0.005\pi) \\ \frac{1}{K}\sum_k U(-\Psi(k) - 0.005\pi) \\ \frac{1}{K}\sum_k U(\Phi(k) - 0.005\pi) \\ \frac{1}{K}\sum_k U(-\Phi(k) - 0.005\pi) \end{array} \right]^T \tag{4}$$

where $\Theta(k)$, $\Psi(k)$ and $\Phi(k)$ denote the pitch, roll and yaw angles, respectively, of the robot torso at the end of 15ms interval k, and K denotes the number of such intervals during the 5 seconds of simulated movement (here, $K = 5s/0.015s \approx 334$). The unit step function $U(\cdot)$ returns 1 if its argument exceeds 0, and returns 0 otherwise. To discount for insignificant motion around 0 rad, orientation

angles are only defined as positive if they exceed 0.5% of π rad. Similarly, orientation angles are only defined as negative if they are less than -0.5% of π rad.

Relative GRF: the amount of GRF each leg applies to the ground during walking, relative to that of all the legs:

$$\mathbf{x} = \left[\frac{F_1}{\sum_{i=1..6} F_i}, \cdots, \frac{F_6}{\sum_{i=1..6} F_i} \right] \quad (5)$$

where F_i denotes the GRF each leg i generates, averaged over 5 seconds of simulated movement.

2.5 Robot and fitness

We simulate the 6-legged robot in the ODE dynamics library, as in Cully et al. [9]. Each leg has three degrees of freedom (thus, 18 total across all 6 legs), but only the first two joints are controlled. The angular position of the last joint is automatically computed such that the last segment of the leg always stays vertical. We thus evolve controllers for 12 servos in total. As in [9], the fitness function is the distance covered in 5 seconds.

2.6 Quality Metrics

We here use the quality metrics introduced by Mouret and Clune [19] for MAP-Elites.

Global fitness. For each run, the single highest-performing solution found by that algorithm anywhere in the search space divided by the highest performance possible in that domain. If it is not known what the maximum theoretical performance is, as is the case for all of our domains, we estimate it by dividing by the highest performance found by any algorithm in any run. This metric is independent of the behavior space.

Coverage. Measures how many cells of the feature space a run of an algorithm is able to fill of the total number that are possible to fill. This metric depends on the behavior space because some combinations of behavioral descriptors might be impossible.

Global reliability. This metric computes the average fitness for all the cells of the map. For each run, the average across all cells of the highest-performing solution the algorithm found for each cell (0 if it did not produce a solution in that cell) divided by the best known performance for that cell as found by any run of any algorithm. Cells for which no solution was found by any run of any algorithm are not included in the calculation (to avoid dividing by zero, and because it may not be possible to fill such cells and algorithms thus should not be penalized for not doing so). This metric depends on the behavior space.

Precision (opt-in reliability). This metric computes the average fitness for the filled cells only. For each run, if (and only if) *this* run creates a solution in a cell, the average across all such cells of the highest performing solution produced for that cell divided by the highest performing solution any algorithm found for that cell. This metric depends on the behavior space.

3. RESULTS

For each experiment, MAP-Elites has a budget of 10 million evaluations for each of the 5 replicates. These experiments required approximately 3 days on 24 cores, for a single replicate. We did not have the computational resources to perform more replicates.

Figure 2 shows typical maps produced by each treatment. Figure 3 aggregates the results for all the metrics and all the combinations of encodings and behavioral spaces.

The results show that the best global fitness values are reached with the direct encoding and the CPPN-encoded CPG (Fig. 3). The SUPG, the C-SUPG, and the HyperNN all led to similar lower fitness values. The best median fitness was reached with the orientation behavioral descriptors (and the CPG controllers), but the best fitness for all the runs was reached with the relative GRF descriptors. Nevertheless, the duty factor behavior space consistently led to lower-performing controller, for all the considered encodings.

While the CPG led to the best fitness, the coverage metric shows that it makes it hard for MAP-Elites to fill the map. In particular, it led to the worse coverage for the relative GRF descriptor and for the Duty Factor. This encoding therefore appears to strongly constrain the behavioral search space: controllers found are good, but it is hard to generate different behaviors. The direct encoding performs the best for all the behavior descriptors, but the difference with SUPG, C-SUPG and HyperNN are small. While the map are less filled with the Duty Factor descriptors than with the two other considered descriptors, it may only result from the fact that the physics simulation makes some behaviors very unlikely or even impossible (e.g. flight-like gaits that involve almost no contact with the ground).

The best reliability (average fitness for all cells) is achieved with the direct encoding, that is, MAP-Elites found the most high-performing gaits with this encoding, whatever the behavior description. This result is easy to spot on Fig. 2, where the cells of the maps generated by the direct encoding appear both well filled and high-performing. A good reliability is also achieved with the CPG for the orientation descriptors, but not for the two other descriptors. The three remaining encodings (SUPG, C-SUPG, and HyperNN) do not perform well on this metric.

Overall, the direct encoding dominates on all the metric except for the global fitness, for which it may be outperformed by the CPG. The CPPN-encoded CPG performs well in term of fitness, but has difficulties to fill the map. The SUPG, C-SUPG, and HyperNN are often similar and it is hard to distinguish them given the low number of replicates.

We hypothesized that the differences between the encodings stem from different approaches to mutation. In particular, previous results have shown that the SUPG encoding generates very different behaviors after a single mutation, whereas directly encoding periodic signals tends to generate similar behaviors in parents and offspring [26]. We therefore plotted the distance in behavior space between each ancestor and its offspring over all the runs of MAP-Elites (Fig. 4, all differences are highly significant because of the high number of samples). For all the behavior spaces, the direct encoding always generates the smallest ancestor-offspring distances. The CPG leads to higher distances but it can often generates neutral or almost neutral mutations. C-SUPG, SUPG and HyperNN all leads to greater ancestor-offspring distances. This analysis of the ancestor-offspring distance mirrors our conclusions about the overall performance of each encoding, which suggests that a low ancestor-offspring is beneficial for MAP-Elites. In preliminary experiments, we tried to change lower the mutation rates to make generative encoding gen-

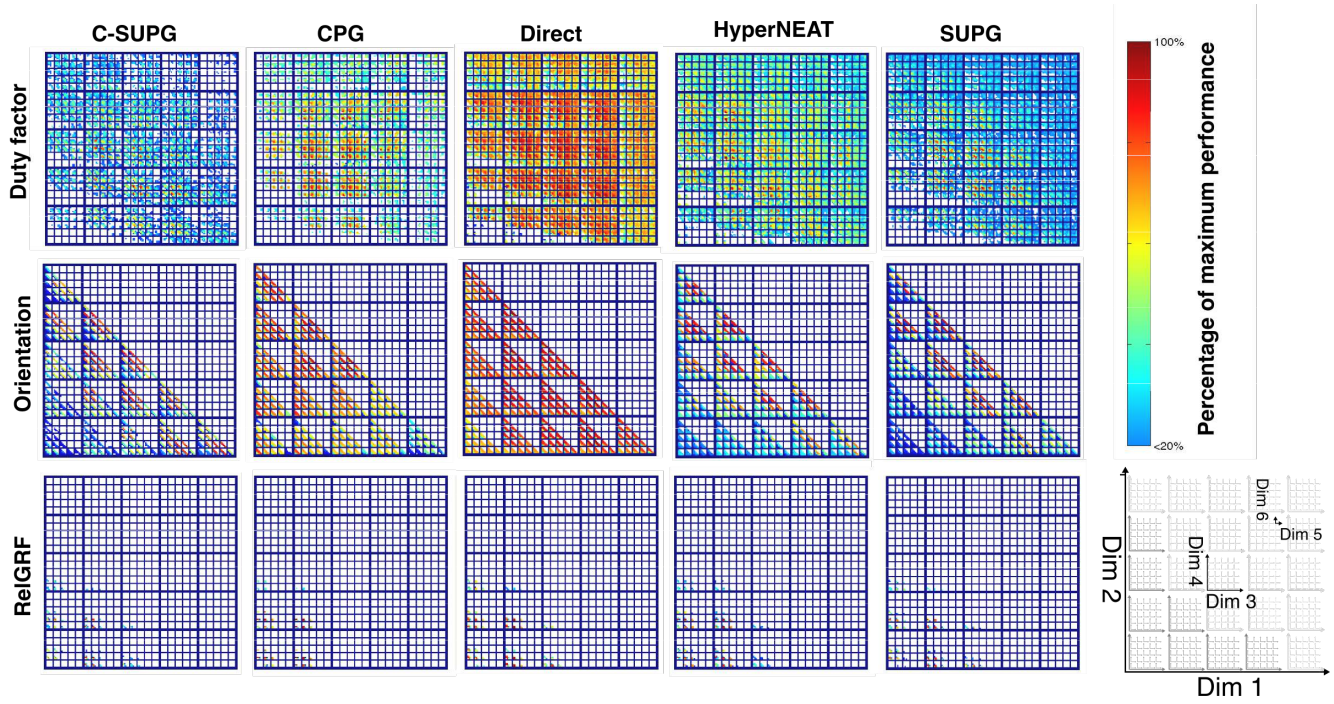

Figure 2: Typical 6-dimensional maps obtained with each of the studied treatments. The matrices visualize the six-dimensional behavioral space in two dimensions according to the legend in the bottom-left. The maximum number of cells that can be filled is 15625 for the duty factor, 3375 for the body orientation, and 235 for the relative ground reaction force.

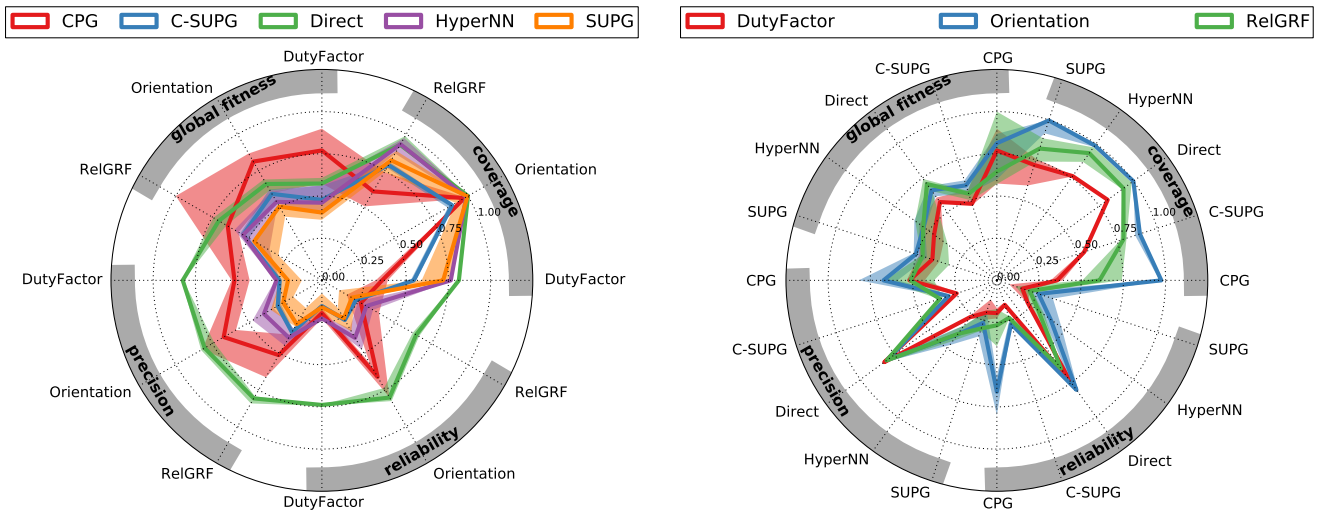

Figure 3: Influence of the encodings and the behavioral descriptors on the global fitness, the precision, the reliability, and the coverage of the maps produced by MAP-Elites. (Left) Comparison of the different encodings. Each colored line corresponds to one encoding. For each quality metric and each behavioral descriptor, the best encoding is the one that with the highest value. (Right) The same data as in the left panel is pictured from the perspective of the behavioral descriptors. Each combination of one encoding and one behavioral descriptor has been replicated 5 times. The solid lines represent the median value over the 5 replications, while the shade areas indicate the minimum and maximum regions over all the replicates.

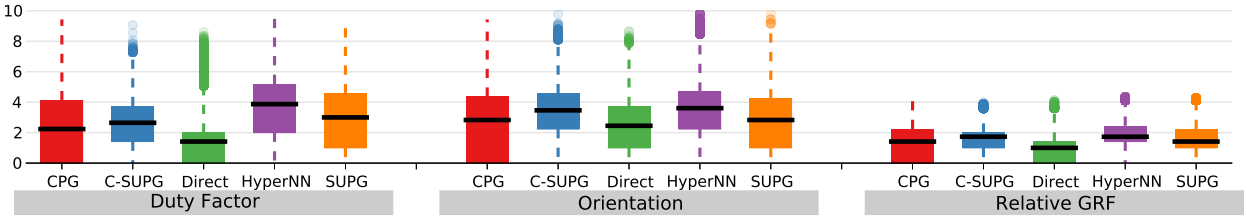

Figure 4: Average distance (in cells) between parent and offspring in the behavioral space (behavioral locality), for each encoding and each behavior space (there are 5 replicates for 25,000 generations; data are logged every 50 generations and correspond to 400 offspring; as a result 1,000,000 mutations are taken into account for each treatment). In spite of the variance, all differences are highly statistically significant ($p < 10^{-100}$, Mann-Whitney-Wilcoxon U-test) because of the high number of samples.

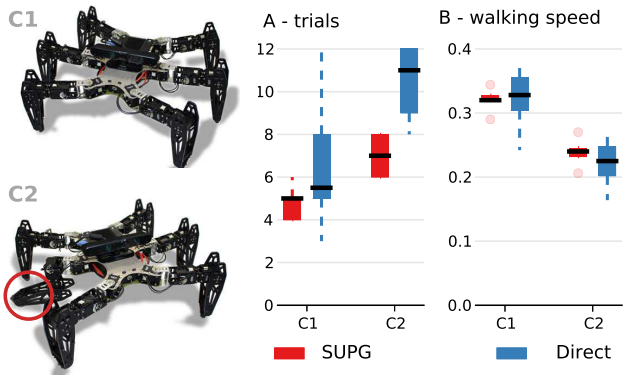

Figure 5: Results of IT&E on the physical robot. (Left) Experiments were performed on the undamaged robot (condition C1), and robot with indicated damaged right-middle leg (condition C2). (Right) Number of trials required by the robot to adapt to conditions C1 and C2, and the walking speed (distance moved in 5s), after adaptation. The only statistically significant difference is the number of trials for C2 ($p = 0.0007$, two-sided Mann-Whitney-Wilcoxon U-test; $p = 0.078$ for C1).

erate more similar behaviors, but without success so far (see [26] for an analysis of the mutation rates for each encoding).

IT&E adaptation on physical robot. While the direct encoding leads to the best performance for MAP-Elites, robots need closed-loop controllers that can take advantage of sensory information. The SUPG or CPG-based controllers are good candidates for this. As a consequence, we needed to check if, in spite of their relatively disappointing scores with MAP-Elites, these closed-loop controllers could still be used for a robot to adapt to damage with the IT&E algorithm.

As experiments with physical robots are expensive and time-consuming, we selected SUPG because it is the most different from the direct encoding, with which all the previous experiments about IT&E that have been published. We chose the C-SUPG variant to avoid damaging the robot. The C-SUPG was triggered with a periodic sawtooth signal of 1 Hz (as the current version of our robot was not equipped with foot contact sensors). A single map of the selected encoding was evolved for 12M evaluations, in which a quasi-stable equilibrium was reached. After 12M evaluations, the MAP-Elites algorithm was able to discover 3112 different gaits, 98.1% of the total 3375 possible behaviors that can be encoded in the body orientation map.

Using the selected map, experiments with the hexapod were replicated five times for each of conditions C1 and C2. On the undamaged hexapod (Figure 5: condition C1), the IT&E was able to discover a working gait for the robot in 5 trials (Median ±IQR: 5 ± 1.25 trials). The robot moved efficiently in the 5 s evaluation time, and achieved a performance of 0.32 ± 0.02 m/s. In order to test the adaptability provided by the encoding, the robot was then damaged by unpowering its right-middle leg (Figure 5: condition C2). Despite the damage to the robot, the algorithm was able to find a working gait in no more than 7 trials (7 ± 2 trials across 5 independent replicates), and the damaged robot was able locomote at 0.24 ± 0.03 m/s (videos of walking gaits available as supplementary material).

These results demonstrate the C-SUPG encoding does not impact negatively the performance of IT&E for damage recovery, in spite of the lower-quality maps. They also suggest that even if the maps generated with C-SUPG are worse than with other encodings, the C-SUPG can still be a good candidate for closed-loop controllers for damage recovery.

4. DISCUSSION AND CONCLUSION

Generative encodings allow evolution to generate controllers with diverse behaviors with a single mutation [8, 26]. This ability is often beneficial for traditional evolutionary algorithms with both static and dynamic fitness functions [26] because it increases exploration, which is often a limiting factor in artificial evolution [20].

The results presented here demonstrate that this ability can be a disadvantage in illumination algorithms: the direct encoding of periodic function, which generate offspring with a behavior close to its ancestors, leads to better results than all the other considered encoding; the more advanced encodings like SUPG, which generates more different behaviors, led to worse performance for all the considered criteria.

Intuitively, generative encodings could have an advantage to quickly fill the behavioral space. On the other hand, they could be unable to generate an offspring with a behavior that is close its ancestor's one, which would make it difficult to fill *all* the cells in a gradual fashion. From the results described in this paper, it seems this second view dominates for MAP-Elites: locality appears more important for MAP-Elites than

the ability to generate highly diverse behaviors, even when there is only a small fitness drop (like for SUPG, see [26]). Because MAP-Elites already maintains a high diversity, it is possible that it does not benefit as much as traditional evolutionary algorithms from the diversity boost provided by generative encodings.

While illumination algorithms are worth investigating on their own, our experiments with the physical robots suggest that the quality of the maps does not impact much the ability of IT&E to help a robot to recover from damage. This surprising result will be investigated in future work as the critical aspect that make the IT&E algorithm perform well are still unclear, but better controllers are needed for real-world applications.

Overall, our results show that illumination algorithms and traditional evolutionary algorithms can have opposite needs, which highlights how these two families are fundamentally different in spite of their common roots.

5. ACKNOWLEDGMENTS

DT is supported by a Marie Curie Intra-European Fellowship (Project: GiFteD-MrS, EC Grant No. 623620), JC by an NSF CAREER award (CAREER: 1453549), and JBM by European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Project: ResiBots, grant agreement No 637972).

6. SUPPLEMENTARY MATERIAL

Code: http://github.com/resibots/tarapore_2016_gecco/
Videos: https://goo.gl/13FJrX

7. REFERENCES

[1] S. Aoi and K. Tsuchiya. Locomotion control of a biped robot using nonlinear oscillators. *Autonomous Robots*, 19(3):219–232, 2005.

[2] J. Bongard, V. Zykov, and H. Lipson. Resilient machines through continuous self-modeling. *Science*, 314(5802):1118–1121, 2006.

[3] J. C. Bongard. Evolutionary Robotics. *Communications of the ACM*, 56(8):74–83, 2013.

[4] E. Brochu, V. M. Cora, and N. De Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[5] J. Carlson and R. Murphy. How UGVs physically fail in the field. *IEEE Transactions on Robotics*, 21(3):423–437, 2005.

[6] J. Clune, B. Beckmann, C. Ofria, and R. Pennock. Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In *Proc. of IEEE CEC*, 2009.

[7] J. Clune, J.-B. Mouret, and H. Lipson. The evolutionary origins of modularity. *Proceedings of the Royal Society B*, 280(1755):20122863, Mar. 2013.

[8] J. Clune, K. Stanley, R. Pennock, and C. Ofria. On the Performance of Indirect Encoding Across the Continuum of Regularity. *IEEE Transactions on Evolutionary Computation*, 15(3):346–367, 2011.

[9] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015.

[10] A. Cully and J.-B. Mouret. Behavioral repertoire learning in robotics. In *Proc. of GECCO*, 2013.

[11] A. Cully and J.-B. Mouret. Evolving a Behavioral Repertoire for a Walking Robot. *Evolutionary Computation*, 2015.

[12] Darwin, Charles. *On the Origin of Species by Means of Natural Selection*. 1859.

[13] S. Doncieux, N. Bredeche, J.-B. Mouret, and A. E. G. Eiben. Evolutionary robotics: what, why, and where to. *Frontiers in Evolutionary Robotics*, 2, 2015.

[14] A. J. Ijspeert, A. Crespi, D. Ryczko, and J.-M. Cabelguen. From swimming to walking with a salamander robot driven by a spinal cord model. *Science*, 315(5817):1416–1420, 2007.

[15] J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[16] J. Lehman and K. O. Stanley. Abandoning Objectives: Evolution Through the Search for Novelty Alone. *Evolutionary Computation*, 19(2):189–223, 2011.

[17] J. Lehman and K. O. Stanley. Evolving a Diversity of Virtual Creatures Through Novelty Search and Local Competition. In *Proc. of GECCO*, 2011.

[18] G. Morse, S. Risi, C. R. Snyder, and K. O. Stanley. Single-unit Pattern Generators for Quadruped Locomotion. In *Proc. of GECCO*, 2013.

[19] J.-B. Mouret and J. Clune. Illuminating search spaces by mapping elites. *arXiv:1504.04909 [cs, q-bio]*, Apr. 2015. arXiv: 1504.04909.

[20] J.-B. Mouret and S. Doncieux. Encouraging Behavioral Diversity in Evolutionary Robotics: An Empirical Study. *Evolutionary Computation*, 20(1):91–133, 2011.

[21] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proc. of CVPR*, 2015.

[22] A. Nguyen, J. Yosinski, and J. Clune. Innovation engines: Automated creativity and improved stochastic optimization via deep learning. In *Proc. of GECCO*, 2015.

[23] J. K. Pugh, L. B. Soros, P. A. Szerlip, and K. O. Stanley. Confronting the Challenge of Quality Diversity. In *Proc. of GECCO*, 2015.

[24] B. Siciliano and O. Khatib. *Springer handbook of robotics*. Springer Science & Business Media, 2008.

[25] K. O. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines*, 8(2):131–162, 2007.

[26] D. Tarapore and J.-B. Mouret. Evolvability signatures of generative encodings: Beyond standard performance benchmarks. *Information Sciences*, 313:43–61, 2015.

[27] V. Verma, G. Gordon, R. Simmons, and S. Thrun. Real-time fault diagnosis. *IEEE Robotics Automation Magazine*, 11(2):56–66, June 2004.

[28] J. Yosinski, J. Clune, D. Hidalgo, S. Nguyen, J. Zagal, and H. Lipson. Evolving robot gaits in hardware: the hyperneat generative encoding vs. parameter optimization. In *Proc. of ECAL*, 2011.