Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers

Sylvain Koos
ISIR, CNRS UMR 7222
Univ. Pierre et Marie Curie
F-75005, Paris, France
koos@isir.upmc.fr

Jean-Baptiste Mouret
ISIR, CNRS UMR 7222
Univ. Pierre et Marie Curie
F-75005, Paris, France
mouret@isir.upmc.fr

Stéphane Doncieux
ISIR, CNRS UMR 7222
Univ. Pierre et Marie Curie
F-75005, Paris, France
doncieux@isir.upmc.fr

ABSTRACT

The reality gap, that often makes controllers evolved in simulation inefficient once transferred onto the real system, remains a critical issue in Evolutionary Robotics (ER); it prevents ER application to real-world problems. We hypothesize that this gap mainly stems from a conflict between the efficiency of the solutions in simulation and their transferability from simulation to reality: best solutions in simulation often rely on bad simulated phenomena (e.g. the most dynamic ones). This hypothesis leads to a multi-objective formulation of ER in which two main objectives are optimized via a Pareto-based Multi-Objective Evolutionary Algorithm: (1) the fitness and (2) the transferability. To evaluate this second objective, a simulation-to-reality disparity value is approximated for each controller. The proposed method is applied to the evolution of walking controllers for a real 8-DOF quadrupedal robot. It successfully finds efficient and well-transferable controllers with only a few experiments in reality.

Categories and Subject Descriptors

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

General Terms

Experimentation

1. INTRODUCTION

In Evolutionary Robotics [18], solutions are commonly evolved in simulation for the purpose of speeding the search. The best controllers found in silico are then transferred onto the real device. However, evolutionary algorithms often exploit simulation's discrepancies in an opportunistic manner to achieve high fitness values with unrealistic behaviors. This problem is called reality gap [18, 12]: if one transfers a controller designed in simulation that relies on badly modeled phenomena, the behavior observed in simulation doesn't match with the one observed in reality. This problem remains a critical issue in Evolutionary Robotics as it often prevents from applying evolutionary results to real robots. More generally, it occurs whenever a controller is designed in simulation before application to a physical target robot. For instance, Palmer et al. reported numerous reality gap problems when working on a 12-DOF bipedal walking robot [19].

A straightforward approach to deal with the reality gap is to rely as little as possible on simulations, for instance by evolving solutions directly on the physical robot [6]. Such a method is clearly not possible with the many robots for which transfer experiments are time consuming or even dangerous for the device: in these common cases, it is necessary to limit the number of transfers by relying on simulations. Other works handle the gap between simulation and reality as environmental variations that can be overcome by robust or adaptive controllers [7]. But, such approaches assume that the solution obtained in simulation behaves similarly in reality, as it can't efficiently re-adapt its behavior from scratch. Simulations can also be improved online by coevolving with robot's controllers: transfer experiments are conducted to generate relevant learning data to evaluate the evolved simulators [2]. However, when the simulator has to model complex phenomena, it can't be easily evolved online.

This paper introduces an alternative method based on an evaluation of the *transferability*. A controller is said transferable if the corresponding trajectories of the robot (expressed in a relevant state space) in simulation and in reality are quantitatively similar. Based on this definition, we propose a new evolutionary approach that aims to:

- find controllers that are both relevant for a given task and transferable from simulation to reality;
- minimize the number of transfers by relying at most on the simulation and only conducting a few experiments on the real robot during the evolutionary run.

Solutions that behave at best in simulation frequently exploit bugs or badly modeled phenomena, making them not transferable. Transferability and efficiency therefore appear to be conflicting objectives. In order to look for relevant trade-off solutions, we propose to optimize solutions with a Pareto-based Multi-Objective Evolutionary Algorithm (MOEA) in which two objectives are defined: a task-dependent fitness only computed in simulation and a transferability objective.

To estimate the transferability of a given controller from simulation to reality, we introduce a simulation-to-reality disparity (STR disparity) measure that evaluates the disparities between the corresponding simulated and real behav-

GECCO'10, July 7–11, 2010, Portland, Oregon, USA.

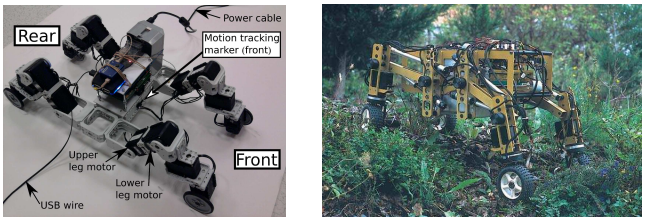

Figure 1: Left, the real robot used for approach's validation, designed as a reduced scale of the robot Hylos, pictured right.

iors of the robot: the higher the STR disparity, the worse the transferability. However, as the number of transfers has to be minimized, the exact STR disparity value for each controller can't be obtained. Consequently, we also introduce an approximation of this STR disparity that depends on the exact STR disparities of already transferred controllers and on behavioral distances [16, 14] to these controllers.

This approach is validated on an 8-DOF wheeled-legged quadrupedal walking robot (Figure 1), as such a device brings into play various gaits more or less dynamic and complex to simulate, then possibly more or less transferable.

2. RELATED WORK

As the reality gap springs from inadequacies between the reality and the simulation, a first attempt to deal with this problem consists in evolving the solutions directly on the real device. Such experiments have been done in [6] with a Khepera mobile robot to find robust controllers that can adapt online to the variations encountered by the robot: environment, battery lifetime, etc. Nevertheless, as experiments are time consuming and dangerous for physical devices, it is often attractive to rely more on simulations and to minimize the number of transfers. As part of the GOLEM project, one of whose goals consists in coevolving morphologies and controllers, Pollack et al. [20] propose to evolve the solutions in simulation and to finish the evolutionary run with some generations on the real robot. First, the robot's morphology and its controller are coevolved within a realistic simulation. Next, an embodied evolution takes place on a population of physical robots with the best morphology to overcome the reality gap. But, such an approach assumes that the solutions' real behavior and simulated behavior are related, i.e. that the gap is sufficiently small.

Another attempt to deal with the reality gap consists in building a minimal simulation [12] by only modeling meaningful parts of the simulation related to the target behavior. The unwanted phenomena are hidden in an *envelope of noise* so that the evolved solutions can't exploit them and have to be robust enough to achieve high fitness values. This approach has been successfully applied to design walking gaits for an octopod robot. Moreover, the more realistic the amount of noise is, the better the transfer should be [15]. The robustness of the behaviors can also be obtained by evaluating the solutions in different simulated environments as in [21]. Nevertheless, it is hard to find the right level of robustness needed to overcome the reality gap for a given task.

Some other works deal with the reality gap as an environment variation to be overcome online. The goal is then to find flexible enough controllers that behave well in simulation, but can adapt online to the gap once transferred onto the real robot. In [10], an anticipation module allows to build a model of the motor consequences in the simulated environment. Then, if some differences are encountered once in reality between this model and the current environment, a correction module performs online adaptation to improve the behavior and overcome the gap. The flexibility can also be intrinsic to the controller structure. In [7], synaptic plasticity of neural network controllers is used to learn several sub-behaviors and also to overcome the gap when a solution is transferred onto the real device by adapting online to the "new" environment. Once again, if the real behavior differs a lot from the simulated one, adaptation is not possible.

In order to evolve solutions while enhancing simulations, some authors resort to coevolution to improve both controllers and simulators at the same time. In [2], Bongard et al. introduce the Exploration-Estimation Algorithm that evolves two populations: simulators and controllers. The simulators have to model the previously observed real data and the controller that discriminates at most between these simulators is transferred onto the real device to generate new meaningful learning data for the modeling part. This process is iterated until a good simulator is found and relevant controllers for a given task can next be built using it. Such a method allows to efficiently explore the solution space with a few transfer experiments. A similar method based on multi-objective evaluation of the solutions is applied to a simulated quadrotor helicopter in [13]. It looks for controllers that maximize simulators' disagreement and achieve a given stabilization task. Such coevolutionary methods rely on the assumption that the simulator can easily be upgraded with only few experiments, that is when modeling simple dynamics or simply adjusting a few parameters.

Also based on coevolution between simulators and controllers, the Back-to-Reality algorithm [22] doesn't resort to a disagreement measure, but try to reduce the fitness variation observed between simulation and reality. Once the controllers have sufficiently converged within the best simulator, they are transferred onto the real robot and the fitness variations of the individuals that behave at best in reality are used to evolve better simulators, and so on. As for the Exploration-Estimation Algorithm, the coevolution process ends when a good simulator and a good controller are found, nevertheless it needs more experiments on the physical device. The approach is applied to a ball-kicking task with a Sony AIBO robot.

In this paper, contrary to coevolutionary approaches, the simulator is designed once and not improved afterwards. Off-the-shelf simulators therefore don't require to be parametrized, making them easy to use. Explicitly idenfiying incorrectly simulated phenomena, for instance to add noise, is also not necessary to employ the method introduced here.

3. APPROACH

Our goal consists in evolving controllers that are both relevant for a given task and well-transferable from simulation to reality. Each of these aspects is evaluated by an objective in a multi-objective way: the task-dependent fitness and the approximated STR disparity objective. We first describe the underlying framework, before detailing the algorithm. An outline of the algorithm is shown on Figure 2.

3.1 Definitions

Behavioral features and distance.

For each controller evaluated in simulation, we assume that its corresponding behavior can be summed up by n values, called behavioral features. Once computed, these features allow to define a *behavioral distance* between individuals. Let $\mathbf{b}^{(1)}$ and $\mathbf{b}^{(2)}$ be the vectors of n behavioral features corresponding to the controllers $c^{(1)}$ and $c^{(2)}$, the behavioral distance *b_dist* between these controllers is:

$$b\_dist(c^{(1)}, c^{(2)}) = ||\mathbf{b}^{(1)} - \mathbf{b}^{(2)}||^2$$

This behavioral distance allows to compare controllers in a simple and fast manner without any dependence on controllers' genotype/phenotype or assumption about it.

Behavioral diversity.

To quantify the diversity of a controller from the already transferred ones, we define a behavioral diversity value as follows[1]. Let C_T be the set of the already transferred controllers and *b_dist* the behavioral distance, the behavioral diversity value $diversity(c)$ for a given controller c is:

$$diversity(c) = min_{c_i \in C_T}\ b\_dist(c,\ c_i)$$

STR disparity.

To estimate controller's transferability, an exact simulation-to-reality disparity value is computed by transferring a controller and comparing the corresponding real and simulated behaviors of the robot. Let us assume that some controllers have already been transferred onto the real robot, the behavioral distance defined above allows to compute an *approximated STR disparity* value for any controller c. Let C_T be the set of the already transferred controllers and $D^*(c_i)$ the exact STR disparity value corresponding to the controller $c_i \in C_T$, the approximated STR disparity $\hat{D}$ of c is:

$$\hat{D}(c) = \frac{1}{N} \sum_{c_i \in C_T} \frac{D^*(c_i)}{b\_dist(c,\ c_i)},$$

$$\text{with } N = \sum_{c_i \in C_T} \frac{1}{b\_dist(c,\ c_i)}$$

Evaluation objectives.

The evaluation process only takes place in simulation. Each controller is evaluated by 2 main objectives in a multi-objective manner:

1. the task-dependent fitness, to find good controllers;
2. the corresponding approximated STR disparity, to find transferable controllers.

Moreover, in order to minimize the number of transfers we want to efficiently explore the controller state space. Exploration can be improved by maintaining behavioral diversity among the population with the help of a third objective [16]:

3. the behavioral diversity value defined above.

[1]Such a diversity is neither genotypic nor phenotypic but behavioral, as it only is derived from the robot's behavior.

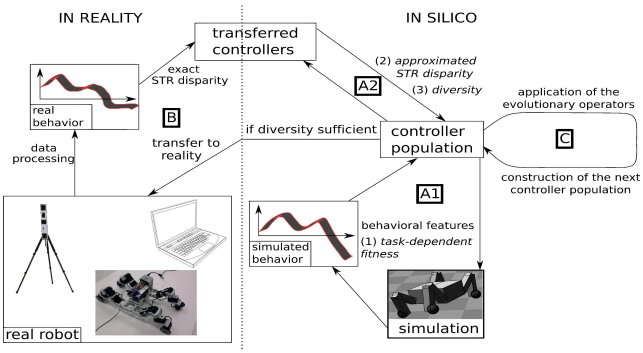

Figure 2: Steps of the proposed algorithm at each generation – A1. The behavioral features and the task-dependent fitness are evaluated for each controller in simulation. A2. The behavioral features of a given controller allow to compute the corresponding approximated STR disparity along with the diversity value based on behavioral distances to the already transferred controllers. B. If this behavioral diversity value is high enough for some controllers, one among them is transferred onto the real system and the corresponding exact STR disparity is computed by comparing the observed simulated and real behaviors of the robot. C. The evolutionary operators are applied to controllers and the selection step builds the next population.

Such a diversity objective looks for solutions that show the more different behaviors from those of the already transferred controllers and ensures that any new experiment is meaningful.

3.2 Algorithm outline

To compute the approximated STR disparity values at the beginning of a run, we assume that a controller c_0 has already been transferred onto the real system. The corresponding exact STR disparity $D^*(c_0)$ and its behavioral features are then available and an approximated STR disparity value can be computed for each controller.

Each generation of the algorithm takes place as follows (Figure 2):

A. evaluation of the controllers:
   - A1. computation of the task-dependent fitness objective and the behavioral features;
   - A2. evaluation of the 2 other objectives in relation to the already transferred controllers (approximated STR disparity and diversity objective);

B. if some controllers have a high enough diversity, one among them is transferred onto the real system;

C. application of evolutionary operators and generation of the next population.

The transfer step B occurs at each generation: once all controllers are evaluated, at most one controller from the population is transferred onto the real system. In order to transfer different enough behaviors from those corresponding to the already transferred controllers and then to limit

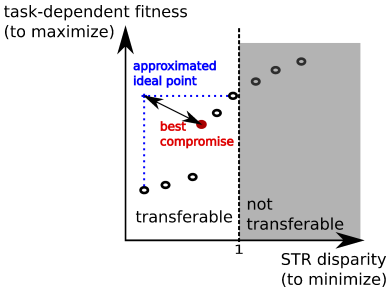

Figure 3: Illustration of the best compromise definition on a non-dominated set.

the number of experiments, we rely on a *diversity threshold* τ_{div}: one controller among those in the current population whose behavioral diversity value is greater than τ_{div} is transferred. As several selection schemes are possible, some algorithm variants have been implemented. These variants will be detailed in the next section.

The diversity threshold is designed by hand to achieve a given number of transfers on average during the whole evolutionary run. When no controller has a sufficiently high diversity value, there is no transfer for this generation. When a controller is transferred, the corresponding exact STR disparity is recorded. It will be used along with its behavioral features in simulation to compute both approximated STR disparity and diversity values for the evolved controllers from the next generation to the end of the run.

3.3 Best solution of a run

At the end of any evolutionary multi-objective run, there is a set of optimal solutions instead of a single one: the non-dominated set. Nevertheless, we want to keep only one "best" solution by run.

As STR disparity values greater than 1 empirically means bad transfers, we call *transferable non-dominated set* the part of the non-dominated set that corresponds to STR disparities lower than 1. There are two possible cases: if the transferable non-dominated set is empty, the best solution of the run is the solution with the lowest STR disparity in the non-dominated set, although it should not transfer well; otherwise, we have to define a best compromise solution on this transferable set.

Let us construct the approximated ideal point whose coordinates are the optimal values for each objective in the transferable non-dominated set. We then select as *best compromise solution* the solution whose distance to the approximated ideal point is minimal. It is illustrated on Figure 3.

4. EXPERIMENTS

4.1 Robot and experimental setups

Locomotion problems have often been addressed in Evolutionary Robotics. In particular, quadrupedal walking offers the advantage of various kinds of gaits: from static and easy to model walks to more dynamic and complex ones. As these gaits don't need the same level of accuracy to be correctly modeled in a simulation, they are expected to achieve different transferability performances on the real device. To exploit such a gait variety, the task we want to achieve here consists in finding transferable walking gaits as fast as possible on a quadrupedal 8-DOF robot (Figure 1). The physical device is made from Bioloid Kit and has been built as a reduced scale model of the wheeled-legged robot Hylos [8] (Figure 1), designed for autonomous planetary/volcanic exploration. Each leg includes 4 Dynamixel AX-12+ Robot Actuators. As we deal with walking gaits, we only control 2 actuators by leg and wheels' positions are fixed. Each leg then includes an upper leg motor and a lower leg motor, all controlled in position. For each actuator, the speed depends on the position error. During the experiments, the robot is supplied with a power cable and controlled with a USB2Dynamixel device connected to a laptop.

We also use a simulator relying on the Bullet Physics Library, an open source physics engine [1]. The following points have been carefully modeled: dimensions of the robot, masses of the different parts, mass asymmetry of the main body, contact areas of the wheels. The simulated robot is made of 14 rigid bodies and 8 hinge constraints to model joints. In order to validate our approach, we rely on two versions of this simulator: an accurate simulator that also models the servos' build-in microcontroller as described in the Dynamixel documentation and a simple simulator that relies on a proportional relation between the speed and the position error.

Two experimental setups are designed with these simulators:

- Exp_1 uses the accurate simulation as "real" system and the simple one as simulated system;
- Exp_2 relies on the accurate simulation and the real robot.

The setup Exp_1 allows to test the algorithm variants in a more systematic way as the whole process in silico is fast and the results obtained with the setup Exp_2 aims at validating the approach. Each algorithm variant has been repeated 10 times in setup Exp_1 and 5 times in setup Exp_2.

Experiments conducted in simulation and in reality follow the same outline. At the beginning of each experiment, all joint angles are set to 0. The 3D trajectory of robot's geometric center is then sampled at 20 Hz for 10 seconds (i.e. 200 data points). New motor positions are sent each 0.1 seconds according to the controller. Once an evaluation is done on the real robot, the initial position is reset to (0, 0, 0) in the dataset by subtracting the initial coordinate values from each data point. It allows not to depend on initial positions when comparing trajectories.

For the experiments in reality, robot's 3D trajectory is recorded with three CODA cx1 scanners (Charnwood Dynamics Ltd, UK). To track geometric center motion, we rely on 2 markers, 1 on each side of the robot (front and rear, see Figure 1). The recording is valid if markers' visibility is greater than 95%, otherwise the experiment has to be done again. The trajectory of the geometric center is then obtained by averaging the 2 markers' positions.

4.2 Controllers, behaviors and STR disparity

To study the reality gap problem in minimal conditions, we rely on one simple sinusoidal controller by motor. All the sinusoidal controllers depend on the same two real parameters $(p_1,\ p_2) \in [0, 1]^2$. The desired angular position α^d of the motor i at time t is obtained by:

$$\alpha^d(i,t) = dir(i) * p_1 - p_2 * sin(\tfrac{2*\pi*t}{20} - \phi(i))$$

$dir(i)$ is 1 for motors of the front-right and rear-left legs, -1 else (see Figure 1 for orientation). The phase angle $\phi(i)$ is 0 for the upper leg motors and $\pi/2$ for the lower leg motors.

Three behavioral features are introduced to sum up the behavior of a given controller in simulation:

- the distance covered during the experiment;
- the mean height of the geometric center of the robot;
- the angular orientation of the robot at the end of its behavior.

These features are normalized by their upper bounds in the considered simulation before any use. At each controller evaluation made within the simulator, these behavioral features are computed. If the controller has also been transferred onto the real robot, the real and simulated distances from the origin $\sqrt{x^2+y^2}$ of the robot's geometric center are computed respectively from the recorded real and simulated trajectories for each sampled data point. These distances are then used to evaluate the corresponding controller's *exact STR disparity*. Let S and R be the distances from the origin respectively obtained in simulation and in reality, let $\overline{S}$ (resp. $\overline{R}$) be the mean of S (resp. R), the exact STR disparity $D^*(c)$ of the controller c is the normalized Mean Square Error (nMSE) between S and R:

$$D^*(c) = \sum_{i=1}^{200} \frac{(S_i - R_i)^2}{\overline{S}\ \overline{R}}$$

Such a STR disparity measure allows to accurately evaluate variations between simulated and real trajectories that correspond to a given controller. It ensures that only transferable controllers will correspond to low STR disparities.

4.3 Algorithms

4.3.1 Proposed Algorithm

The proposed algorithm relies on multi-objective optimization with a Pareto-based ranking scheme. Regarding the implementation[2], we use NSGA-II [5], an efficient state-of-the-art MOEA based on non-dominated sorting and elitist tournament selection. The evolved genotypes are the two parameters of the controllers described in the previous section, $(p_1, p_2) \in [0,1]^2$. Two operators are defined on the genotype: Gaussian point mutation[3] and recombination crossover. The mutation probability is 0.5. For each run, 40 individuals are evolved during 100 generations.

As we look for fast walking gaits, the task-dependent fitness is the covered distance during the experiment. Several variants of the algorithm are implemented (Table 1) depending on: (1) the selection scheme of the candidate controller to transfer at each generation, and (2) the presence/lack of the diversity objective.

The random transfer selection scheme consists in choosing at random the controller to transfer among those whose diversity value is greater than the diversity threshold τ_{div}. The "max. diversity" scheme consists in transferring the controller that maximizes the diversity value if it is greater than τ_{div}.

[2]This work has been implemented within the Sferes$_{v2}$ framework [17]. The source code is available at:
`http://www.isir.fr/evorob_db`

[3]Mutation parameters: 0 mean, 0.2 standard deviation (sd)

Table 1: Algorithm variants.

Variants	Diversity objective	Transfer selection scheme
RandomT & Div	×	random
MaxDivT & Div	×	max. diversity
RandomT & NoDiv		random

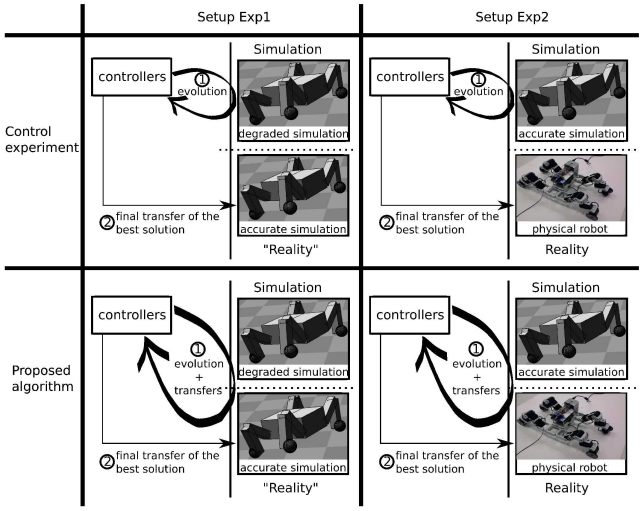

Figure 4: Outline of the 4 scenarios depending on the algorithm and the setup.

For the setup Exp_1, 2 values for diversity threshold τ_{div} are envisaged: 0.05 and 0.025, respectively corresponding to expected numbers of transfers 25 and 45 during a run.

4.3.2 Control Algorithm

For the Control Algorithm, each controller is evaluated by a single objective: the covered distance in simulation. There is no transfer during the run and the best solution of the run is the controller that maximizes the covered distance. Evolutionary operators and parameters are the same as those described for the Proposed Algorithm.

Each of the envisaged algorithms (Control and Proposed) are tested in both setups (Figure 4).

5. RESULTS

5.1 Control algorithm in both setups

For the setup Exp_1, 10 runs of the Control Algorithm have been made. In this setup, relying on both simulations, the best solutions achieve 12712 mm on average (sd = 687 mm) in the simple simulation and 309 mm on average (sd = 200 mm) once transferred in the accurate one. The corresponding exact STR disparities are 55.51 on average with standard deviation 26.41. Such performance loss from accurate to simple simulation highlights a very strong reality gap problem with the setup Exp_1 and prevents to resort to a classic single-objective fitness-based evolution.

The Control Algorithm has been repeated 5 times for the setup Exp_2, relying on the accurate simulation and the real robot. The best solutions achieve 1327 mm on average (sd = 67 mm) in the accurate simulation and 535 mm on average (sd = 442 mm) on the real robot. The corresponding STR disparities are 1.70 on average with standard deviation 1.46.

Table 2: Expected and observed numbers of transfers for each variant and diversity threshold value τ_{div} in both setups.

Setup	τ_{div}	Expected number	Variants	Observed nb. mean	Observed nb. sd
Exp_1	0.05	25	*RandomT & Div*	26	6
			MaxDivT & Div	25	5
			RandomT & NoDiv	28	6
	0.025	45	*RandomT & Div*	44	8
			MaxDivT & Div	46	9
			RandomT & NoDiv	57	6
Exp_2	0.1	10	*RandomT & Div*	11	2

The gap problem is more complex than for setup Exp_1 because among the best solutions found in 5 runs, 2 controllers transfer well on the physical robot and 3 don't. Two conclusions can be drawn from these results:

- the accurate simulation is worthwhile because it allows *sometimes* to find good solutions that transfer well on the real robot;
- there is a reality gap problem anyway with setup Exp_2.

Now that the reality gap problem has been emphasized in both setups, results obtained with our algorithm are shown.

5.2 Proposed algorithm in setup Exp_1

Each algorithm variant has been repeated 10 times, with diversity threshold values τ_{div} 0.05 and 0.025. For the normalization of the features the vector {14.0, 0.76, 3.14} is used. It corresponds to their upper bound values in the simple simulator. The obtained results are shown on the Figure 6. The number of transferred controllers for setup Exp_1 are shown in Table 2. The observed number of transfers are comparable to the target values 25 and 45: it justifies the diversity threshold values τ_{div} chosen by hand.

As the approximation of the STR disparity is modified after each transfer experiment, the algorithm has to deal with a dynamic optimization problem: the optimal solutions change with time. Then, it has to be checked that the found non-dominated solutions regarding the fitness and the approximated STR disparity are also non-dominated regarding the fitness and the exact STR disparity. The Figure 5 shows the non-dominated set along with the last population of a typical run (Proposed Algorithm, variant *RandomT & Div*), expressed respectively in terms of approximated STR disparity on graph A and exact STR disparity on graph B. The non-dominated solutions obtained with the approximated disparity are either non-dominated or near to the non-dominated set regarding the exact one. It tends to justify the way of building the approximated STR disparity.

The *RandomT & Div* and *MaxDivT & Div* variants behave significantly better than the Control Algorithm. Globally, the *RandomT & Div* variant shows best results, as it looks for better trade-off solutions. It notably validates the random selection scheme. One of the best trade-off solutions obtained during *RandomT & Div* variant's runs achieves 1004 mm in the simple simulation and 989 mm once transferred in the accurate simulation with a 0.08 disparity value. On average for this variant with $\tau_{div} = 0.05$, the best solutions achieve 859 mm in the simple simulation and 701 mm in the accurate one with a 0.29 disparity value; on average for $\tau_{div} = 0.025$, we obtain 860 mm in the simple simulation and 921 in the accurate one with a 0.25 disparity value.

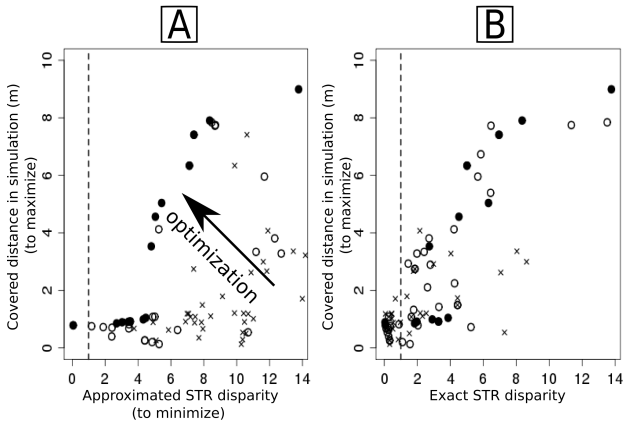

Figure 5: Graph A: non-dominated individuals (circle) along with last population (cross) obtained with a typical run in setup Exp_1 (Proposed Algorithm, variant *RandomT & Div*) in terms of covered distance in the simple simulation and approximated disparity. The black circles are the non-dominated set without taking into account the diversity objective. Graph B: same individuals (setup Exp_1) expressed in terms of covered distance in the simple simulation and exact disparity.

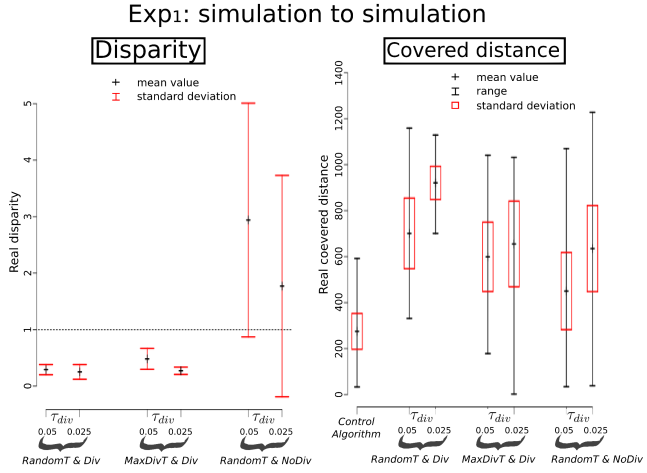

Figure 6: Results for setup Exp_1: exact disparity (left, except for Control Algorithm: $\overline{D^*} = 55.51, \sigma_{D^*} = 26.41$) and covered distance (mm, right) in accurate simulation of the best solutions obtained with each algorithm. The diversity threshold values τ_{div} are written above the variant names. All variants behave better than the Control Algorithm both in terms of disparity (Student T-test p-values $< 10^{-4}$) and distance (p-values $< 10^{-2}$) except for *RandomT & NoDiv* variant (p-value $= 8.01\ 10^{-2}$). Moreover, for *RandomT & Div* and *MaxDivT & Div* variants, the disparities are clearly lower than 1: the found best solutions show good transferability properties.

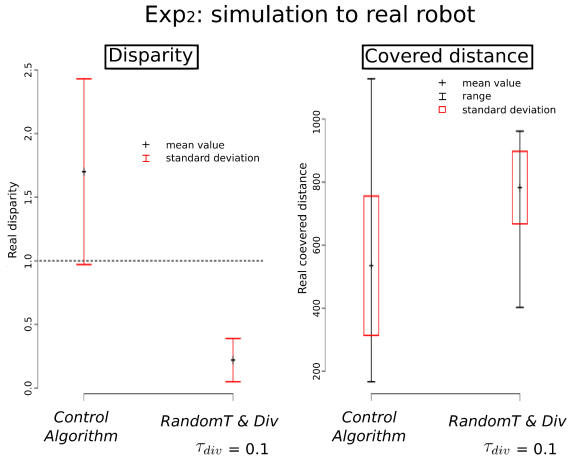

Figure 7: Results for setup Exp_2: exact disparity (left) and covered distance in reality (mm, right) of the best solutions obtained with each algorithm. Due to the few repetitions (5), it is hazardous to compute any relevant statistical significance. However, the *RandomT & Div* variant behaves clearly better than the Control Algorithm with disparities lower than 1 and finds more efficient controllers on average regarding the covered distance objective.

MaxDivT & Div variant finds worse trade-off solutions on average than *RandomT & Div* variant, in terms of disparity with $\tau_{div} = 0.05$ (p-value=0.071) or of distance with $\tau_{div} = 0.025$ (p-value=0.029). It shows that transferring the more different controller from the already transferred ones is not ideal. Compared to a more intermediate controller, its neighborhood indeed contains fewer individuals and the corresponding exact disparity value therefore is less informative. Then, with similar numbers of transfers, the random selection scheme for the transfer step (*RandomT & Div*) variant has to be preferred to *MaxDivT & Div* variant.

The poor results obtained with *RandomT & NoDiv* show that the behavioral diversity objective is necessary, as it drives the search towards individuals different from those already transferred, thus probably increasing the accuracy of the approximated STR disparity.

Surprisingly, the number of transfers has no significant effect on the quality of the best solutions. The proposed algorithm doesn't behave better on average with 45 transfers than with 25. It proves that the algorithm should work with relatively few transfer experiments.

To conclude, the proposed algorithm:

- finds relevant transferable controllers with STR disparity values lesser than 1;
- works at best with a random transfer selection scheme;
- may work with few transfer experiments by run.

5.3 Proposed algorithm in setup Exp_2

As setup Exp_2 is very time consuming compared to setup Exp_1, we only allow 10 transfers on average during each run with a diversity threshold value $\tau_{div} = 0.1$. For the same reason, we didn't study the relevance of the approximated STR disparity compared to the exact one as in setup Exp_1.

The *RandomT & Div* variant has been repeated 5 times. The features are normalized with the values {1.5, 0.2, 3.14}. It corresponds to their upper bound values in the accurate simulator. The obtained results are shown on the Figure 7. The observed numbers of transfers for setup Exp_2 are presented in Table 2 and validate the diversity threshold $\tau_{div} = 0.1$ designed for 10 transfers on average by run.

As for setup Exp_1, the *RandomT & Div* variant behaves better than the Control Algorithm on average despite the very small number of transfer experiments, especially regarding the STR disparity. The best trade-off individual among the best solutions obtained during *RandomT & Div* variant's runs achieves 1031 mm in the accurate simulation and 962 mm on the real robot with a 0.005 STR disparity value. On average for this variant, the best solutions achieve 914 mm in the accurate simulation and 783 mm on the real robot with a 0.22 disparity value. The algorithm then finds good solutions that transfer well on the real robot with only few transfer experiments.

6. DISCUSSION

In our approach, controllers are evaluated in simulation by a task-dependent fitness and a STR disparity value. As we hypothesized that these 2 objectives are conflicting, we proposed to evaluate individuals in a multi-objective way. This antagonism has to be discussed according to the results obtained in both setups.

For the setup Exp_1 (simple simulation / accurate simulation), the shape of the non-dominated set shown on the graph A of Figure 5 is consistent with the hypothesis of conflicting objectives: the higher the fitness in simulation, the worse the approximated STR disparity. Moreover, as it can be seen on the graph B of the same Figure, this antagonism is also observed with the exact STR disparity. For the setup Exp_2 (accurate simulation / real robot) not enough runs have been conducted to conclude, but the few experiments also suggest conflicting objectives.

Despite this antagonism, using a soft constraint based on the STR disparity value could provide an alternative to multi-objective optimization. The most popular method to handle soft constraints in evolutionary computation is probably the penalty method [4]; however such a method is highly dependent on the threshold value used to determine if the constraint is satisfied. Multi-objective optimization is another notable way to deal with soft constraints by treating them as additional objectives in a multi-objective manner [3, 4]. Handling a constraint in such a way would be equivalent to the method introduced in the present paper.

The possible implementation of our approach on different problems has also to be discussed. The approach depends on three main points that have to be defined in relation to the setup: (1) a trajectory to compute the STR disparity measure; (2) a way to record this trajectory on the real robot; (3) relevant behavioral features. For instance, let us assume that we want to apply the approach to Jakobi's experiment described in [11]: controllers for Khepera mobile robots are evolved to turn at the junction of a T-maze depending on the position of a previously encountered light. The fitness is the covered Manhattan distance to the start position, plus a bonus if the robot has turned in the right corridor. Here, the reality gap problem arises from how the junction is modeled in the simulation. In this experiment: (1) the STR disparity could be the nMSE between the simulated and real

Manhattan distances to the start position; (2) the Manhattan distance in reality could be inferred from a trajectory recorded by CODA scanners and a marker fixed at the top of the robot; (3) the behavioral features could be the covered Manhattan distance and the distances on average to the left and right walls.

The reality gap problem also arises for systems like bird-sized unmanned aerial vehicles that correspond to little-known dynamical phenomena. Then, we could use the approach for instance to evolve transferable controllers for the stabilization of a quadrotor helicopter like in [13]. Here, the state of the quadrotor includes the attitude angles along with the angular velocities. The fitness is defined as the Euclidean distance between the final state and the target one. In this setup, (1) the exact STR disparity could be the Euclidean distance between the simulated and real trajectories in the state space; (2) in reality, the quadrotor could be fixed on a Whitman training stand [9] and an inertial measurement unit could be used to record the attitude angles along with the angular velocities; (3) choosing meaningful behavioral features is more difficult for this application. We could envisage to select pitch and roll variations as they are linked to quadrotor's stability, but further investigations have to be done before any choice.

7. CONCLUSION

This paper addressed the reality gap problem in the case of controller transfer, a critical issue in Evolutionary Robotics, which often relies on simulators. An evolutionary algorithm that looks for not only good controllers but also transferable ones has been implemented. Controllers are evaluated by 2 objectives in a multi-objective manner: a task-dependent fitness and a simulation-to-reality disparity that estimates controller's transferability.

The application to an 8-DOF quadrupedal walking robot shows promising results by finding controllers that are relevant regarding a walking task and that transfer well to reality. The proposed algorithm outperforms a more classic evolutionary approach regarding both exact STR disparity and covered distance in reality despite the low number of transfers conducted during the runs.

Based on these successful results, the approach appears to be a simple and relevant method to efficiently cross the reality gap in Evolutionary Robotics.

8. REFERENCES

[1] A. Boeing and T. Bräunl. Evaluation of real-time physics simulation systems. In *Proc. of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, pages 281–288. ACM New York, NY, USA, 2007.

[2] J. Bongard and H. Lipson. Once more unto the breach: Co-evolving a robot and its simulator. In *Proc. of Artificial life IX*, page 57. MIT Press, 2004.

[3] C. Coello. Constraint-handling using an evolutionary multiobjective optimization technique. *Civil Engineering and Environmental Systems*, 17(4):319–346, 2000.

[4] K. Deb. *Multi-objective optimization using evolutionary algorithms*. Wiley, 2001.

[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[6] D. Floreano and F. Mondada. Evolutionary neurocontrollers for autonomous mobile robots. *Neural Networks*, 11(7-8):1461–1478, 1998.

[7] D. Floreano and J. Urzelai. Evolution of plastic control networks. *Autonomous Robots*, 11(3):311–317, 2001.

[8] C. Grand, F. BenAmar, F. Plumet, and P. Bidaud. Decoupled control of posture and trajectory of the hybrid wheel-legged robot Hylos. In *IEEE International Conference on Robotics and Automation*, volume 5, pages 5111–5116, 2004.

[9] S. Hanford, L. Long, and J. Horn. A small semi-autonomous rotary-wing unmanned air vehicle (UAV). In *2003 AIAA Atmospheric Flight Mechanics Conference & Exhibit*, 2003.

[10] C. Hartland and N. Bredèche. Evolutionary Robotics, Anticipation and the Reality Gap. In *Proc. of ROBIO'06*, pages 1640–1645, 2006.

[11] N. Jakobi. *Minimal Simulations for Evolutionary Robotics*. PhD thesis, University of Sussex, 1998.

[12] N. Jakobi. Running across the reality gap: Octopod locomotion evolved in a minimal simulation. *Lecture Notes in Computer Science*, 1468:39–58, 1998.

[13] S. Koos, J. Mouret, and S. Doncieux. Automatic system identification based on coevolution of models and tests. In *Proc. of IEEE CEC*, volume 2009, 2009.

[14] J. Lehman and K. Stanley. Exploiting open-endedness to solve problems through the search for novelty. In *Proc. of ALIFE XI*, volume 54, 2008.

[15] O. Miglino, H. Lund, and S. Nolfi. Evolving Mobile Robots in Simulated and Real Environments. *Artificial Life*, 2(4):417–434, 1995.

[16] J.-B. Mouret and S. Doncieux. Using behavioral exploration objectives to solve deceptive problems in neuro-evolution. In *Proc. of GECCO'09*, pages 627–634. ACM, 2009.

[17] J.-B. Mouret and S. Doncieux. Sferes_v2: Evolvin' in the multicore world. In *Proc. of IEEE CEC'2010*, 2010.

[18] S. Nolfi and D. Floreano. *Evolutionary robotics*. MIT Press, 2000.

[19] M. Palmer, D. Miller, and T. Blackwell. An Evolved Neural Controller for Bipedal Walking: Transitioning from Simulator to Hardware. In *Proc. of IROS 2009 Workshop on Exploring new horizons in Evolutionary Design of Robots*, 2009.

[20] J. Pollack, H. Lipson, S. Ficici, P. Funes, G. Hornby, and R. Watson. Evolutionary techniques in physical robotics. In *Evolvable systems: from biology to hardware: third international conference, ICES 2000, Proceedings*, pages 175–186. Springer Verlag, 2000.

[21] A. Thompson, P. Layzell, and R. Zebulum. Explorations in design space: Unconventional electronics design through artificial evolution. *IEEE Transactions on Evolutionary Computation*, 3(3):167–196, 1999.

[22] J. Zagal and J. Ruiz-del Solar. Combining simulation and reality in evolutionary robotics. *Journal of Intelligent and Robotic Systems*, 50(1):19–39, 2007.