# The Transferability Approach: Crossing the Reality Gap in Evolutionary Robotics

Sylvain Koos, Jean-Baptiste Mouret, Stéphane Doncieux

# The Transferability Approach: Crossing the Reality Gap in Evolutionary Robotics

Sylvain Koos, Jean-Baptiste Mouret and Stéphane Doncieux

***Abstract*—The reality gap, that often makes controllers evolved in simulation inefficient once transferred onto the physical robot, remains a critical issue in Evolutionary Robotics (ER). We hypothesize that this gap highlights a conflict between the efficiency of the solutions in simulation and their transferability from simulation to reality: the most efficient solutions in simulation often exploit badly modeled phenomena to achieve high fitness values with unrealistic behaviors. This hypothesis leads to the Transferability approach, 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, estimated by a simulation-to-reality (STR) disparity measure. To evaluate this second objective, a surrogate model of the exact STR disparity is built during the optimization. This Transferability approach has been compared to two reality-based optimization methods, a noise-based approach inspired from Jakobi's minimal simulation methodology and a local search approach. It has been validated on two robotic applications: 1) a navigation task with an e-puck robot; 2) a walking task with an 8-DOF quadrupedal robot. For both experimental set-ups, our approach successfully finds efficient and well-transferable controllers only with about ten experiments on the physical robot.**

## I. Introduction

EVOLUTIONARY ROBOTICS (ER) [39], [46] deals with the use of Evolutionary Algorithms (EA) in robotics. Such algorithms are indeed attractive black-box optimization methods that put only few constraints on the optimal behavior by relying on a fitness function to compare the potential solutions. This fitness function links each evaluated solution to a value that reflects its efficiency on the task to achieve and, as ER concerns robots, it should theoretically be computed on the studied robot [16]. In practice, each evaluation on a physical device can be very time-consuming. Besides, as the behavior that corresponds to a given solution is not known before its evaluation, harmful behaviors can be transferred onto the robot. Consequently, the few works in which controllers have been directly evolved on the robot often optimized few individuals during few generations, which reduces the efficiency of the evolutionary methods. For instance in [19], controllers for a small helicopter have been evolved with a population of 20 individuals during 30 generations, with few minutes between generations to avoid over-heating, that is only 600 evaluations during the optimization process. In [52], optimization has directly been applied to a prototype ornithopter machine to maximize its lift with 3000 evaluations on the physical system during the optimization, which seems more consistent with evolutionary techniques, but other optimization tasks would require several tens of thousands of evaluations [15].

For these several reasons, simulation models are an appealing way to evaluate the fitness in a fully secure set-up, while significantly speeding up the optimization process [22]. Accurate simulators can be even slower than experiments in reality, which lead to prohibitively long optimization processes. To obtain simulation models with lower computational costs, it is sometimes necessary to neglect some complex physical phenomena, which leads to simpler simulators, of course less accurate, but also faster. The dynamics of the robot can also not be fully known: for instance, bird-size UAVs or small helicopters bring into play little-known dynamics, which leads to approximate simulation models. Consequently, the dynamic model of the robot used to build a simulation model can itself be inaccurate. When the fitness is computed in simulation, the evolutionary process is likely to exploit such inaccuracies between the simulation model and the reality in an opportunistic manner to achieve high fitness values with unrealistic behaviors. In practice, even if many works in ER are successful to build non-trivial and efficient controllers that correspond to original and complex behaviors [51], [55], these attractive results are often locked in the simulated world because of bad transfers from simulation to reality. This transfer problem is called *reality gap* [29] and is arguably the most critical issue that currently prevents the use of ER for practical robotic applications. For instance, numerous reality gap problems have been reported when applying an EA to a 12-DOF bipedal walking robot [48]. It should be noted that the reality gap problem is not specific to ER, as any optimization method based on a simulation model encounters reality gap issues (for instance in [3] when designing control structures for a quadrotor helicopter). A gap can even exist when the controllers are directly evolved on the real system, if the experimental set-up which allows to evaluate the individuals is too different from the real environment of the robot. It has notably been observed in [19] on a small helicopter.

The goal of this work is to introduce the Transferability approach, a general methodology to help crossing the reality gap and to bring Evolutionary Robotics and simulators back together. This approach aims at:

- finding controllers that are both relevant for a given task in simulation and transferable from simulation to reality;
- conducting as few experiments as possible on the physical robot during the optimization process.

Our first insight concerns the simulation models: even if a simulation model is somehow inaccurate, it also contains

Sylvain Koos, Jean-Baptiste Mouret and Stéphane Doncieux are with the ISIR, CNRS UMR 7222, Université Pierre et Marie Curie, F-75005, Paris, France. Contact: koos@isir.upmc.fr

realistic parts as it is designed to accurately mimic some physical phenomena. Efficient behaviors that mainly rely on these realistic parts of the simulation model should transfer pretty well onto the physical device and then achieve good performances in reality.

A controller is said transferable if the corresponding behaviors of the robot observed in simulation and in reality are similar. Our approach takes into account the transfer quality of the evaluated controllers under the form of a transferability measure. For a given controller, this transferability measure compares the corresponding real and simulated behaviors and becomes an objective to optimize during the optimization while looking for efficient controllers. As solutions that behave at best in simulation frequently exploit bugs or badly modeled phenomena, making them not transferable, transferability and efficiency appear to be conflicting objectives. In order to look for relevant trade-off solutions, we then propose to optimize solutions with a Pareto-based Multi-Objective Evolutionary Algorithm (MOEA) in which two objectives are defined: a task-dependent fitness computed in simulation only 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 behaviors 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 cannot be obtained. Consequently, we build during the optimization process a surrogate model that approximates the STR disparity function with function interpolation techniques.

Preliminary results have been obtained with the Transferability approach with an 8-DOF wheeled-legged quadrupedal robot on a walking task [35]. In the current paper, the approach is additionally applied to a navigation task in a T-maze [26] and both applications allow systematic comparisons between the Transferability approach and state-of-the-art methods.

After presenting some previous work on the reality gap problem, we introduce the Transferability approach in a robotic context. The approach is next validated on two robotic applications (cf. above) and compared to two robot-based optimization methods, a noise-based approach inspired from Jakobi's minimal simulation methodology and also a local search approach. Three main aspects are next investigated, notably regarding the quality of the approximated STR disparity, before discussing the underlying hypotheses of the approach.

## II. Previous work on the reality gap problem

Several works deal with the reality gap problem and one can distinguish three main types of approaches: (1) reality-based optimization approaches where optimization takes place, fully or partly, on the physical robot; (2) simulation-based optimization approaches with an entire optimization process in simulation; (3) robot-in-the-loop simulation-based optimization approaches, that fully optimize solutions in simulation, but also allow few transfer experiments during the process.

### A. Reality-based optimization

As the reality gap results 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 [16] with a Khepera mobile robot, to find robust controllers that can adapt to the variations encountered by the robot during a navigation task in a maze: environment, battery lifetime, ... The optimization took more than 60 hours with about 8000 evaluations on the physical robot, while the task seems relatively simple. As a case in point, similar approaches have been implemented on Sony AIBO robots [24], [33][1] and on a nine-legged robot [59].

Pollack et al. [50] proposed an alternative that partly allows to tackle high computational cost of optimizing in reality. As part of the GOLEM project, one of whose goals consists in co-evolving morphologies and controllers, the solutions were mostly evolved in simulation and only the last generations of the optimization process were conducted on real robots. First, the robot's morphology and its controller were co-evolved with a realistic simulation. Next, an embodied evolution took place on a population of physical robots with the best morphology to overcome the reality gap. Nolfi et al. reported a similar work regarding a navigation task addressed by a mobile Khepera robot with 30000 evaluations in simulation followed by 3000 evaluations on the physical robot [47].

Such approaches assume that the optimal solutions found with the simulation model are relatively close to the true optimal ones on the real robot, i.e. that the high values of the fitness function in simulation are not too misleading. Consequently, a local search around the controllers seen as optimal in simulation should be sufficient to retrieve near optimal controllers in reality. This assumption is clearly debatable when the optimal solutions in simulation achieve high fitness values because of inaccuracies or bad modeled dynamical phenomena.

### B. Simulation-based optimization

The prohibitive computational cost of direct optimization on physical robots has led some researchers to envisage full optimization processes in simulation [53]. A first attempt to deal with the reality gap is to build more accurate simulation models. If all physical phenomena are well-modeled, there should not be any significant gap between simulation and reality. However, simulation models often are trade-offs between accuracy and computational cost: although the reality gap problem highlights the need of accurate simulators, accurate models can lead to very high computational costs, which are incompatible with optimization techniques. It has notably been underlined in [11] with visually guided robots. Besides, some kinds of robots rely on little-known dynamics like bird-sized unmanned aerial vehicles [38] or small helicopters. For such devices, perfect simulations are still out of reach.

To cope with not fully accurate simulation models, some techniques have been developed in order to evolve controllers

[1] About 1000 experiments in 3 hours in [33].

that exhibit robust enough behaviors in simulation or that are based on robust enough mechanisms to transfer well onto the real robot. Among such approaches, the most formalized one is Jakobi's minimal simulation [26]. It consists in only modeling meaningful parts of the physical system in relation to the target behavior by dropping the complex physical phenomena that are only involved in bad or unstable behaviors. The unwanted phenomena are hidden in an envelope of noise or not modeled at all so that the evolved solutions cannot 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 [28]. The robustness can also be achieved by optimizing the potential solutions with several minimal simulation models whose parameters are slightly varied from a generation to another. It has been applied to a navigation task with a Khepera mobile robot in a T-maze [27]. Similar robustness issues have been investigated in [40], also with Khepera mobile robots: by only choosing the most realistic amount of noise to add on the simulated sensor values, the transfer from simulation to reality did not lead to any performance loss. The robustness of the optimized behaviors can also be obtained by only evaluating the solutions with several simulation environments and initial conditions as in [56].

Other approaches deal with the reality gap as a variation of the environment that can be overcome online by some adaptive mechanisms. In [17], plastic neural network controllers have been used to integrate several sub-behaviors and also to overcome a gap when a solution is transferred onto the real device by online adaptation to the "new" environment. There was no clear separation between simulation and reality. The robustness is then directly linked to the mechanism of plasticity. The robot can also explicitly build an approximate model of its environment to use it as a reference and then adapt to environment variations. For instance in [21], an anticipation module allowed to build a model of the motor consequences in the simulated environment. 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.

Whether the robustness is obtained by the optimization process in simulation or by some adaptive mechanisms, all these approaches rely on the following hypothesis: the level of robustness is sufficient to overcome the reality gap. In Jakobi's methodology, it can be quite tricky to find which parameters have to be changed and from which amount to variate them. Besides, one can wonder if adaptive mechanisms are always able to retrieve the global optimum in reality from the global optimum in simulation. If the real behavior corresponding to the optimal controller in simulation differs a lot from its simulated counterpart, such mechanisms are rather likely to retrieve local optima in reality with significantly worse performances.

### *C. Robot-in-the-loop simulation-based optimization*

The robot-in-the-loop simulation-based optimization approaches also rely mostly on simulators but some transfer experiments are allowed during the optimization. In [6], Bongard et al. introduced a co-evolutionary process, 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 simulation part. This process is iterated until a good simulator is found and relevant controllers for a given task can next be built on it. These simulators allow to speed up the evaluation of the controllers, while being upgraded by conducting some meaningful transfer experiments on the real device. Moreover, resorting to an update heuristic based on a disagreement measure allows to reduce the number of experiments required to explore efficiently the solution space. This approach has been successfully implemented with a four-legged robot [8]. A similar method based on multi-objective evaluation of the solutions has been applied to a stabilization task with a simulated quadrotor helicopter [34].

Also based on co-evolution between simulators and controllers, the Back-to-Reality algorithm [58] does not resort to a disagreement measure, but tries to reduce the fitness variation observed between simulation and reality. Once the controllers have sufficiently converged to 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 co-evolution process ends when a good simulator and a good controller are found. The approach has successfully been applied to a ball-kicking task with a Sony AIBO robot.

However, such co-evolutionary methods rely on the assumption that the simulation model can become accurate enough to allow perfect transfers with only few experiments. It is plausible when modeling simple dynamics or simply adjusting a few parameters, but debatable for optimizations on a wider search space.

The optimization process can itself directly rely on a so-called surrogate model by evaluating the individuals with a simple model of the fitness function instead of building an entire simulation model. The surrogate model has to be upgraded during the optimization process by conducting some test experiments depending on a given update heuristic. As a case in point, such an approach has successfully been applied to fast humanoid locomotion [23]. Without relying on EA, similar approaches have been applied to reality gap problems in the field of reinforcement learning. Abbeel et al. notably applied such techniques to aerobatic helicopter flight [2]. From several trajectories previously made by a pilot, they identified an approximate local model of the helicopter dynamics before learning the optimal flight policy which was next transferred onto the physical device. If the policy did not work in reality, the corresponding data obtained in reality were used to upgrade the local dynamic model and the policy optimization took place again. The process was iterated until the optimal policy worked in reality. A similar method was applied in [1] to the autorotation of a remote control helicopter in case of an engine failure. The results obtained for these applications are quite impressive. However, it can only be

applied when a human pilot/operator is able to mimic the task to solve in order to identify the dynamic model.

### D. Concluding thoughts

This state-of-the-art on the reality gap problem leads us to five main thoughts:

1. Optimizing on the physical robot is an appealing way, but it leads to slow optimization processes and some risky behaviors can be transferred;
2. Completing the optimization process by some evaluations onto the real robot is only meaningful if the optimal solutions in simulation are close to the optimal ones in reality, that is if the reality gap is small enough;
3. There is no guarantee that a robust controller only optimized in simulation will be robust enough to transfer well in reality and such a robustness is hardly assessable;
4. Adaptive mechanisms inside the controller structure may not efficiently re-adapt to optimal behaviors in reality if the gap is too strong;
5. To our opinion, the robot-in-the-loop simulation-based optimization approaches are currently the most promising ones, but building a perfect simulator or a meaningful surrogate model is arguably difficult, especially if it is built from scratch or improved during the optimization process as is often the case.

One of the most pivotal point is the use of simulation models: is it necessary to build it from scratch or to improve it as is often the case in the robot-in-the-loop simulation-based optimization approaches? For all practical purposes, simulation models are often available when working on robotic applications and while a simulation model can lead to reality gap problems, it is also designed to properly describe the dynamics of a given system: it probably contains both accurate parts and inaccurate ones. Our main idea is to base the optimization on a simulation model that remains fixed during the whole process. The approach then looks for the most efficient controllers whose behaviors are sufficiently based on the realistic parts of the simulation model to transfer well onto the real robot. Consequently, we do not build a simulation model from scratch nor modify it, but we rather exploit an already available simulator where it mimics the reality at most.

## III. The Transferability Approach

### A. Principles

The Transferability approach fits into the robot-in-the-loop simulation-based optimization approaches. The optimization process relies on a simulator designed once and not improved afterwards. Our first hypothesis is that, despite reality gap problems, this simulation model is locally reliable with some parts accurate enough to ensure good transfers to reality as illustrated on Fig. 1. However, the gradient provided by the fitness in simulation does not guide the search in the same direction as the real gradient: the best solutions found in simulation are not transferable to reality and behave significantly worse on the robot. The Transferability approach aims at finding efficient solutions that mostly exploit these well-modeled parts of the simulator. To evaluate the quality of a given controller's transfer from simulation to reality, we rely on a transferability measure that compares a simulated behavior with its counterpart in reality and quantitatively reflects their closeness. We secondly hypothesize that the reality gap mainly stems from a conflict between two aspects: the efficiency of solutions in simulation and the transferability of those solutions from simulation to reality. It leads to a multi-objective formulation of ER in which two main objectives are optimized via a Pareto-based MOEA: (1) the task-dependent fitness; (2) the transferability objective.

The transferability measure cannot be obtained for each solution as it means many transfer experiments on the robot. We claim that if the value of this function is known for a few selected solutions, that is if a few solutions are transferred during the optimization, a transferability function can be approximated for all the other solutions by interpolation. This interpolated transferability objective allows to guide the evolutionary search towards good compromise solutions, both efficient in simulation and transferable from simulation to reality. The whole process is pictured on Fig. 2.

In this work, the transferability of a given controller is assessed by a simulation-to-reality disparity (STR disparity) measure, which estimates the disparities between the corresponding behaviors respectively observed in simulation and on the physical robot: the higher the STR disparity, the worse the transferability. As the STR disparity cannot be computed for each potential solution, we rely on a surrogate model to approximate this second objective. The surrogate model is interpolated thanks to a few transfer experiments conducted during the optimization, according to an update heuristic, that allows to periodically select which controllers are the most meaningful to transfer regarding the current surrogate model.

As the Transferability approach aims at finding solutions both efficient in simulation and transferable from simulation to reality, it does not always find the optimal solutions in reality, but rather good compromises between efficiency in simulation and transferability. If the optimal solutions in reality indeed rely on unrealistic parts of the simulation, as illustrated on the Fig. 3, the approach will consequently avoid them, because they are not transferable. The Transferability approach is therefore based on the hypothesis, that the realistic parts of the simulation include behaviors, which are sufficiently relevant to efficiently address the task. We hypothesize that this situation is unlikely to occur for realistic robotic applications, because the mechanical models used as simulations are designed to model the most important phenomena regarding the task to solve.

Another case can arguably be problematic. As pictured on the Fig. 4, the optimal solution in simulation may also be optimal in reality, while corresponding to a non-transferable behavior. As the Transferability approach looks for transferable zones in the simulation, this optimal solution is avoided. However, this case can easily be detected, as the fitness value in reality obtained with the best solution found with the Transferability approach should be lower than the fitness value in reality of the most efficient solution in simulation. Based on

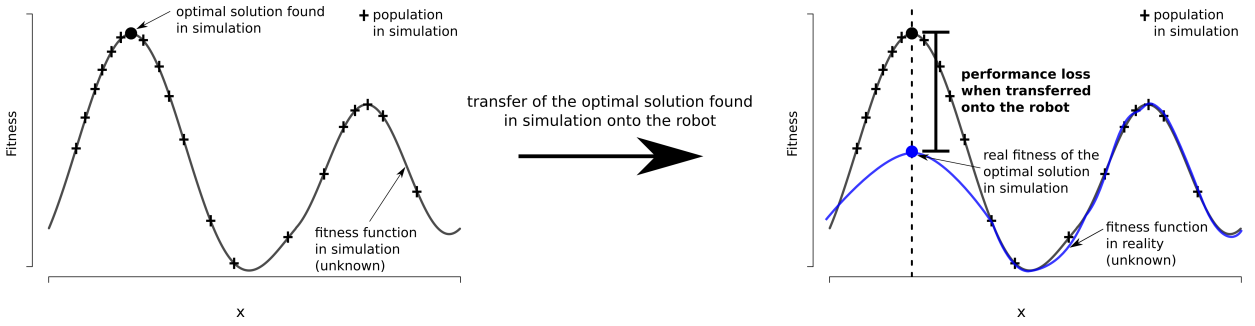


Fig. 1. Illustration of the reality gap problem on a fictional 1-dimensional optimization problem. While the simulation model is locally reliable, optimizing the fitness objective lead to non-transferable solutions: there is a significant performance loss when they are transferred onto the robot.

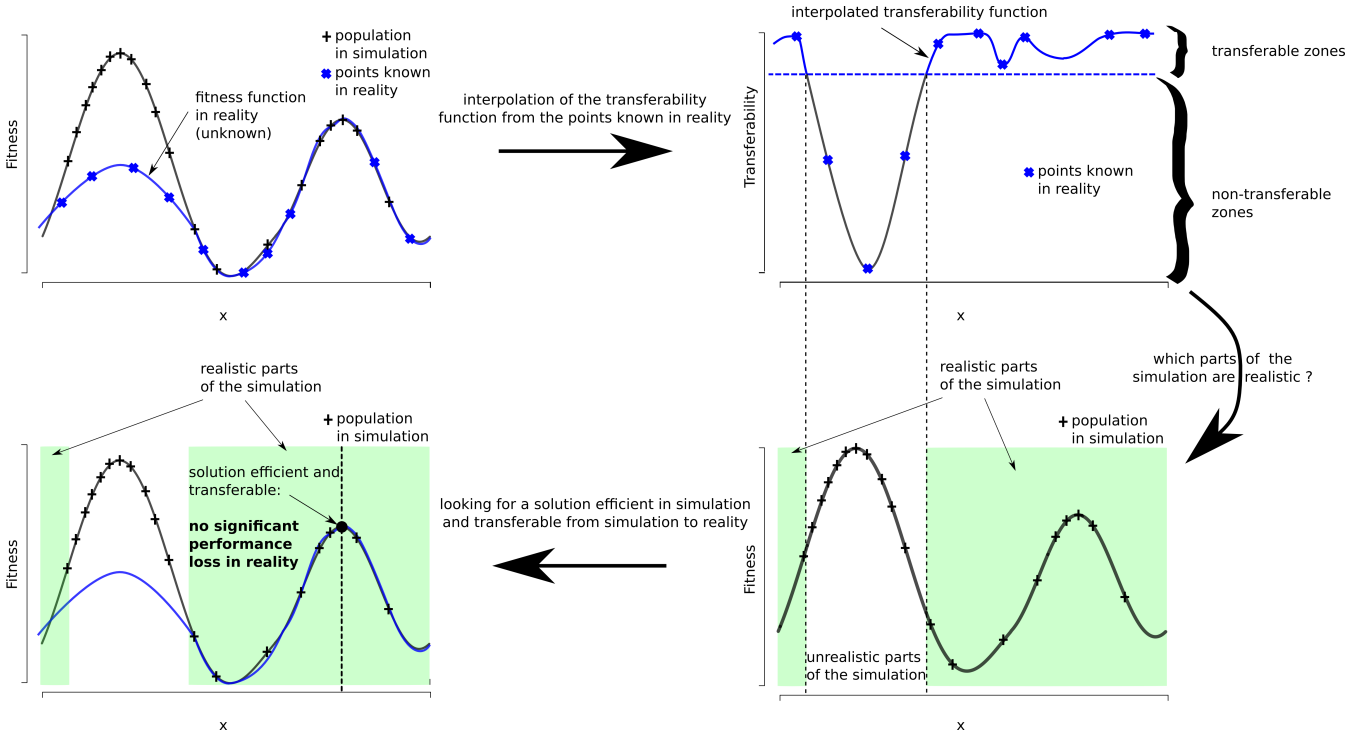


Fig. 2. Illustration of the Transferability approach. If some points are known in reality, it allows to interpolate an approximation of the transferability function. According to this approximate function, well-modeled parts of the simulation can be distinguished from the unrealistic ones: it is then looked for solutions, which are efficient in simulation and transferable from simulation to reality.

this observation, this case was never encountered in the two experiments studied in this paper.

### *B. From an exact STR disparity to a surrogate model*

The STR disparity function $D^*$ links, for any possible controller $c$, the corresponding behavior in simulation $b(c)$ in the behavior space $\mathfrak{B}$, to its exact STR disparity value $D^*(b(c))$. By an abuse of notation, for a given controller $c$, the corresponding exact STR disparity value is noted $D^*(c)$. Such a STR disparity function cannot be used directly as an objective, because it would mean that each potential solution in the population is transferred in reality while being evaluated. Both real and simulated behaviors are indeed needed to compute the corresponding exact STR disparity value. Consequently, we rely on a so-called surrogate model to approximate the STR disparity function during the optimization process.

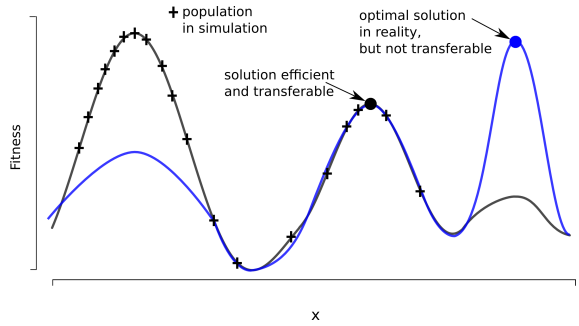


Fig. 3. The optimal solution in reality may correspond to a non-transferable behavior in simulation. In this case, as the Transferability approach looks for transferable solutions, it may not find the optimal solutions in reality. We hypothesize that this situation does not happen for realistic robotic applications, because the mechanical models used as simulations are designed to model the most important phenomena regarding the task to solve.

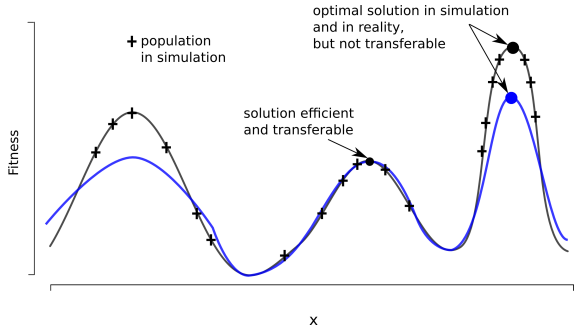


Fig. 4. The optimal solution in simulation may correspond to the optimal solution in reality, despite a significant performance loss when transferring it on the robot. The Transferability approach therefore may avoid the optimal zones in simulation and miss the optimal solutions in reality. However, this case can be detected, as the fitness value in reality obtained with the best solution found with the Transferability approach should be lower than the fitness value in reality of the most efficient solution in simulation. This case was never observed in the two experiments presented in the paper.

Surrogate models [23], [31], [57] are usually resorted to in real engineering problems when evaluating an individual on the target system means very high computation costs or too long experiments. Then, instead of a direct evaluation on the physical system, solutions are optimized via an approximate model of it. For instance, in EA, a surrogate model can be used to approximate the fitness objective. This model has also to be updated during the optimization by conducting some pertinent experiments on the physical system.

*Building the surrogate model:* In order to build a surrogate model of the STR disparity function during the evolutionary optimization, some transfer experiments have to be performed during the run to obtain some exact STR disparity values. We want this process to verify the three following constraints: (C1) The number of experiments remains small; (C2) close behaviors in simulation (regarding a given behavioral distance) should have close STR disparity values; (C3) the experiments are iteratively or periodically generated. This last constraint is necessary when using EA, because we usually do not have a good sampling of behaviors at first in the initial population.

There are many approximation techniques that can be used to build a surrogate model. Among them, interpolation methods try to find an approximate model whose error on the known data points is null. Some of the most used interpolation methods are: 1) Radial Basis Function [30]; 2) Inverse Distance Weighting model [54]; 3) Kriging model [32]. Such interpolation methods rely on a distance function to compare solutions: the value predicted for a given solution mostly depends on the exact values of solutions that are close to it. In the Transferability approach, the distance function, noted $b_{dist}$, is defined on the behavioral space $\mathfrak{B}$ in simulation. It ensures that solutions related to close behaviors in simulation according to $b_{dist}$ correspond to close STR disparity values (constraint C2). In the rest of the article, this behavioral distance function $b_{dist}$ is called *in silico metric*, as it only compares simulated behaviors. It has to be distinguished from the STR disparity function $D^*$ which compares simulated and real behaviors, but both can be computed on similar behavior definitions.

Radial Basis Function (RBF) interpolation [30] builds an approximation of the target function by a weighted sum of some predefined radial basis functions. As additional RBFs can easily be incorporated into the approximation when new data points are available, it is consistent with online interpolation. Nevertheless, such a method has several degrees of freedom: the types of RBFs, the number of RBFs and the learning method to find the weights. Moreover, new data points globally affect the whole interpolated function, because the weights of all the RBFs are modified at the same time, while local modifications should be more appropriate [18].

The Inverse Distance Weighting method (IDW) [54] is much simpler to use. Each term of the interpolating function is linked to one data point for which the exact value of the function to approximate is known. Such a method only depends on one power parameter $k$ that defines the smoothness of the interpolating function. The main drawback is that the predicted value always lies between the maximal exact value and the minimal exact value used for the interpolation.

Kriging is a group of popular interpolation geostatistical methods [23], [32], somewhat similar to IDW interpolation. For the IDW method, the weights assigned to each interpolated point while predicting the value of an unknown point only depends on the distance function $b_{dist}$. Kriging additionally assumes a correlation between the interpolated data. Consequently, the weights are based not only on the distance function, but also on the correlation function between the interpolated points (directions, trends, ...). To model such a correlation, an empirical variogram has to be built that links the observed variance among the interpolated points to the distance from each other. As building an empirical variogram requires a lot of preliminary experiments, classic correlation functions are usually used for optimization problems [31] in place of an ad-hoc function derived from an empirical variogram.

While Kriging is a very popular method to build surrogate models, some works show that Kriging interpolation leads to good results only if the underlying spatial arrangement is sufficiently known [37]. If not, Kriging methods lead to similar or even worse results than less computationally expensive approaches (it was compared to the IDW method in [37] and [44], for instance). Moreover, Kriging models include several parameters, which implies many experiments to initialize the model. For instance, the ordinary Kriging method introduces 1 + 2 * n parameters, if n is the size of the input space, which means at least 1 + 2 * n preliminary experiments. If n is large, that is in the case of many input parameters, it leads to numerous preliminary experiments before the surrogate model can be used.

In this work, we choose the Inverse Distance Weighting interpolation method to build the surrogate model of the STR disparity function, as it does not clash with any of the three constraints while being easy and fast to implement.

*Choice of an in silico metric:* For each controller evaluated in simulation, the corresponding behavior is summed up by $n$ values, called behavioral features. Once computed, these fea-

tures allow to define a behavioral distance between individuals in simulation, the *in silico metric*. Let $\mathbf{b}^{(1)}$ and $\mathbf{b}^{(2)}$ be the vectors of the $n$ behavioral features computed in simulation and corresponding to the controllers $c^{(1)}$ and $c^{(2)}$, the in silico metric $b_{dist}$ between these controllers is:

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

This in silico metric allows to compare controllers in a simple and fast manner without any dependence or assumption on controllers' genotype and phenotype.

*Surrogate model of the STR disparity:* If some controllers have already been transferred onto the real robot (at least once) and the corresponding exact STR disparity values have been computed, a surrogate model $\hat{D}$ of the STR disparity function can be interpolated by Inverse Distance Weighting (IDW) from these values. Let $\mathfrak{C}$ be the set of all possible controllers, let $\mathfrak{C}_T \subset \mathfrak{C}$ be the set of the already transferred controllers and $D^*(c_i)$ the exact STR disparity value corresponding to each controller $c_i \in \mathfrak{C}_T$. The surrogate model of the STR disparity $\hat{D}$ is built as follows[2]:

$$\forall c \in \mathfrak{C}, \hat{D}(c) = \frac{\sum\limits_{c_i \in \mathfrak{C}_T} D^*(c_i)\ b_{dist}(c_i, c)^{-2}}{\sum\limits_{c_i \in \mathfrak{C}_T} b_{dist}(c_i, c)^{-2}}.$$

The use of a surrogate model implies the choice of an update heuristic that selects which test experiments have to be conducted on the physical device in order to pertinently upgrade the model. We choose a relatively simple heuristic that promote controllers whose minimal behavioral distance in simulation to the already transferred controllers is higher than a pre-defined threshold $\tau_{div}$. It should ensure that the surrogate model is not build on a too localized part of the behavior space.

### C. Optimization scheme

*Evaluation objectives:* Each controller is evaluated by three objectives:

1. the task-dependent fitness, to find good controllers;
2. the corresponding approximated STR disparity computed with the surrogate model, to find transferable controllers;
3. the behavioral diversity objective.

This last objective allows to maintain behavioral diversity among the population, which efficiently enhances exploration of the controller state space [15], [42].

*Behavioral diversity:* To quantify the diversity of a controller from the already transferred ones, we define a behavioral diversity value as follows. Let $\mathfrak{C}_T$ be the set of the already transferred controllers and $b_{dist}$ the in silico metric, the behavioral diversity value $diversity(c)$ for a given controller $c$ is:

$$diversity(c) = \min_{c_i \in \mathfrak{C}_T} b_{dist}(c,\ c_i)$$

This diversity value does not directly depend on the genotype nor on the phenotype of the controller to evaluate, as it is only derived from the behavior of the robot. It promotes solutions that show the most different behaviors from those of the already transferred controllers according to the in silico metric $b_{dist}$. In such a context, the update heuristic defined earlier boils down to randomly selecting one individual among those whose diversity value is higher than the diversity threshold $\tau_{div}$. It ensures that any new experiment selected by the update heuristic is meaningful.

### D. Algorithm outline

To initialize the surrogate model of the STR disparity, we assume that a controller $c_0$ has already been transferred onto the real system at the beginning of each optimization process. The corresponding exact STR disparity value $D^*(c_0)$ and the behavioral features in simulation are computed.

Each generation of the algorithm takes place as follows (Fig. 5):

A. evaluation of each controller:
   - A1. computation of the behavioral features and the task-dependent fitness objective in simulation;
   - A2. evaluation of the two other objectives: approximated STR disparity and diversity objective, according to the distance $b_{dist}$ to the already transferred controllers;

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

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

The transfer step B occurs at each generation according to the update heuristic: 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 the number of experiments, the update heuristic relies on a *diversity threshold* $\tau_{div}$: one controller, randomly selected among those in the current population whose behavioral diversity value is greater than $\tau_{div}$, is transferred.

The diversity threshold is designed by hand to achieve a given number of transfer experiments on average during the whole optimization process. When no controller has a sufficiently high diversity value at a given generation, there is no transfer. When a controller is transferred, it is added to the set of the already transferred controllers $\mathfrak{C}_T$ and the corresponding exact STR disparity value is recorded. It is next used to update the surrogate model of the STR disparity function.

### E. Best solution of a run

We assume at first that a threshold $D^*_{threshold}$ on the STR disparity values can be empirically chosen in such a way

[2]We select the power parameter value $k = 2$.

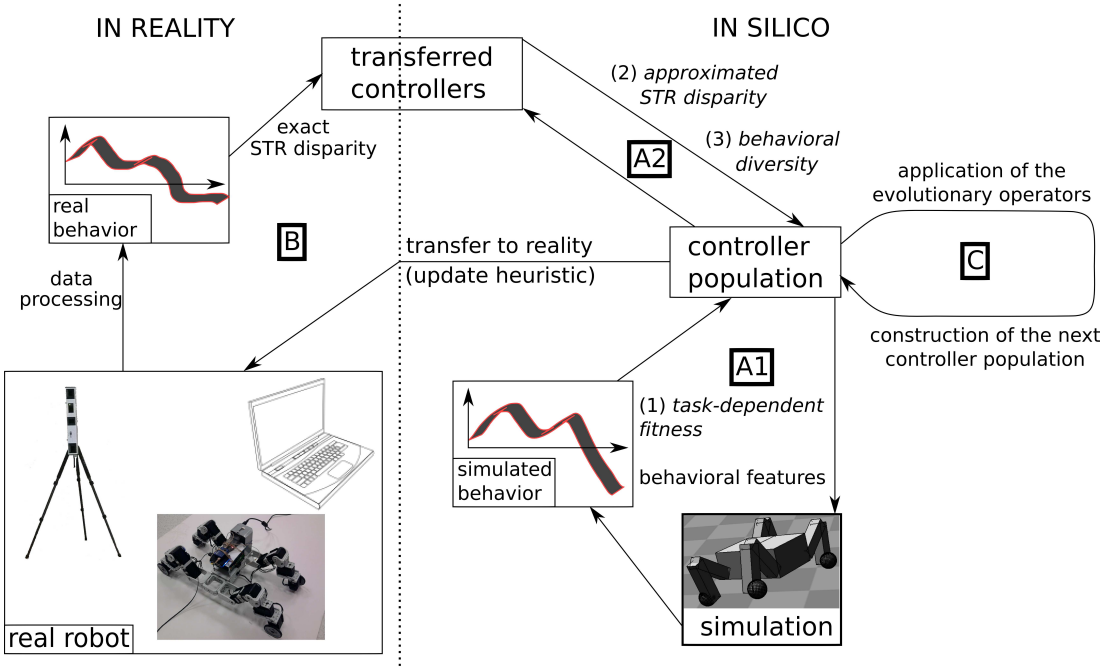


Fig. 5. Steps of the proposed algorithm at each generation – A1. The behavioral features and the task-dependent fitness are evaluated for each controller in simulation. A2. The simulated behavior of a given controller allows to compute the corresponding approximated STR disparity value as predicted by the surrogate model along with the diversity value based on the distance $b_{dist}$ to the already transferred controllers. B. If this behavioral diversity value is high enough for some controllers, one among them is randomly picked up to be transferred onto the real system and the corresponding exact STR disparity value 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.

that STR disparity values greater than $D^*_{threshold}$ empirically means bad transfers.

One can discern two main classes of applications: (A1) the optimality of any solution is known once the corresponding behavior is evaluated: the controller is optimal if it solves the task even if its fitness is not maximal; (A2) there is no clear criteria that allows to distinguish optimal solutions from non-optimal ones: the fitness is simply maximized. For the class A1, the goal of the optimization process boils down to find an optimal individual in reality, that is a controller which solves the task. Therefore, as some transfer experiments are conducted during the optimization when using the Transferability approach, an optimal individual in reality can be found before the end of the whole process. In this case, the process can be stopped as soon as such an individual is observed and this individual is the best solution of the run.

If no optimal individual in reality is observed during the optimization for an application of the class A1, some criteria have to be defined to select the best solution. The same criteria are used for applications of the class A2. At the end of any evolutionary multi-objective run, there is a set of best solutions instead of a single one: the set of non-dominated solutions or non-dominated set. We call *transferable non-dominated set* the part of the non-dominated set that corresponds to STR disparities lower than $D^*_{threshold}$. 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 choose a best compromise solution in this transferable set.

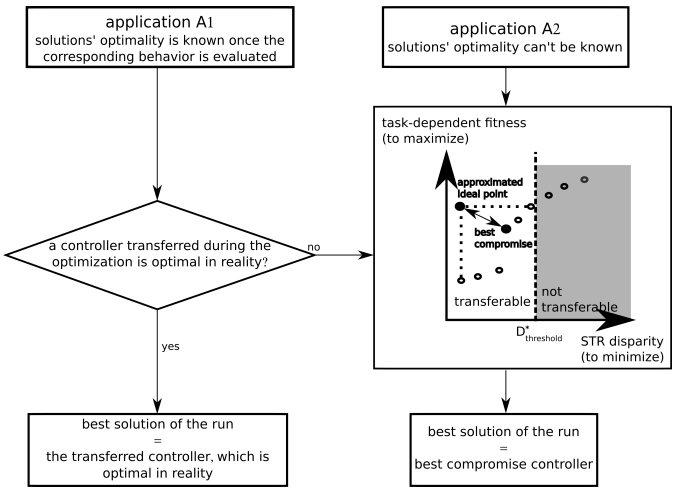


Fig. 6. How to select the best solution: two classes of applications.

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 Fig. 6.

### F. Validation on two robotic applications

Our approach has been validated with an e-puck robot on one of Jakobi's early experiments on the reality gap problem (class A1, [26]). This application allowed us to compare our approach with a noise-based approach inspired from Jakobi's

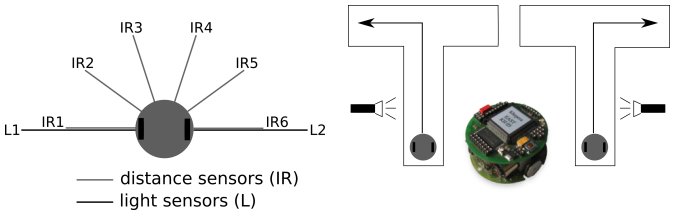


Fig. 7. Left, diagram of the sensors used with the Khepera mobile robot in the original set-up. Right, both test cases of the reproduced Jakobi's experiment pictured with the Khepera mobile robot.

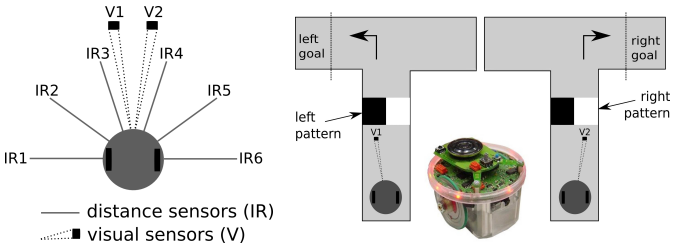


Fig. 8. Left, diagram of the sensors used with the e-puck robot. Right, our experimental set-up pictured with the e-puck robot.

one, which is actually the most formalized methodology dedicated to the reality gap problem. Another set-up dealing with walking gait optimization is next investigated with an 8-DOF wheeled-legged quadrupedal robot (class A2, Fig. 13). For this second application, preliminary results have been published in [35]. Additionally, our approach is compared to two reality-based optimization methods and two simulation-based optimization methods on both applications.

## IV. Application I: e-puck in a T-maze

### A. Experimental set-up

Our first application aims to reproduce one of Jakobi's experiments on the reality gap problem [26], [27], notably to compare our approach with Jakobi's one that nowadays remains the most formalized methodology dedicated to this problem. In the original set-up, a two-wheeled robot has to turn at the junction of a T-maze according to the position of a previously encountered light. The robot uses six front infrared distance sensors and one light sensor on each side. The experiment is made of two test cases. When the light is on the right (resp. the left), the robot has to turn on the right (resp. on the left) at the junction. This simple set-up is illustrated on the Fig. 7.

Instead of the Khepera mobile robot, the similar *e-puck robot* is employed [41]. Nevertheless, the light sensors of the e-puck robot appear not to be reliable enough and our experimental set-up is slightly different from the original one. The light detection has been replaced by a color detection using the camera of the e-puck.[3] During the left test case, the left visual sensor $V1$ is 1 only if the bottom left pixel of the camera is black and the right one $V2$ is always 0. During the right test case, the right visual sensor $V2$ is 1 only if the bottom right pixel of the camera is white and the left one $V1$ is always 0. Our set-up is illustrated on the Fig. 8.

According to Jakobi's methodology, a minimal simulation of this experimental set-up has to be built by splitting the task in two sub-parts: first, the robot moves in a corridor where it can detect the color patterns; next, when the robot has traveled a given distance, it pops in a second corridor where it has to turn in the right direction. Regarding the sensing capabilities of an e-puck robot, this simulation model differs from a T-maze because it does not model the junction. When the robot pops in the second corridor, there should not be a wall behind it. Consequently, Jakobi argues that if the robot

[3]The camera resolution is 40 × 40.

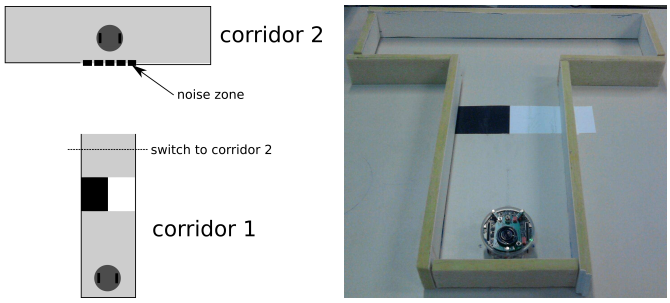


Fig. 9. Left, representation of the T-maze used as minimal simulation; right, photography of the real T-maze with both color patterns

detects this wall in simulation, its sensors have to be noised so that the optimal behaviors found in simulation cannot exploit it and fail when transferred onto the real device (see Fig. 9). The implementation of the noise zone is detailed in the next section.

No slippage or friction are modeled in the simulator. The variation of robot's orientation during a step is equal to the difference between the distance covered by the two wheels divided by the diameter of the robot (about 75 mm).

The distance sensors of the robot are modeled as follows: depending on the position of the robot in the simulated maze, the distance to the walls are computed in the 6 sensor directions. The model between the distance $d$ in millimeters and the sensor value $s$[4] has been built on the basis of sensor data from the e-puck robot used during the experiments. The full sensor model is defined as follows[5]:

$$s = \begin{cases} 3300, & \text{if } d < 6 \\ 3500 * F(d), & \text{if } 6 \leq d \leq 60 \\ 20, & \text{if } d > 60 \end{cases}$$

The function F has been obtained by polynomial regression between the logarithm of the sensor values and the distance to the walls. A degree 2 polynomial function was found sufficient to fit the relation, which led to:

$$F(d) = exp(0.601 - 0.118 * d + 7.22 \cdot 10^{-4} * d^2)$$

The sensor values are noised with uniformly distributed random deviates in the range $[-15, 15]$ and normalized by 3500 when used as inputs for the controller.

[4]E-puck robot's infrared sensor values lie in the interval $[0, 3500]$.

[5]As we do not use the same type of robot as in [26], the model sensor is different. Moreover, it is not detailed in the original work, how the sensor model was built.

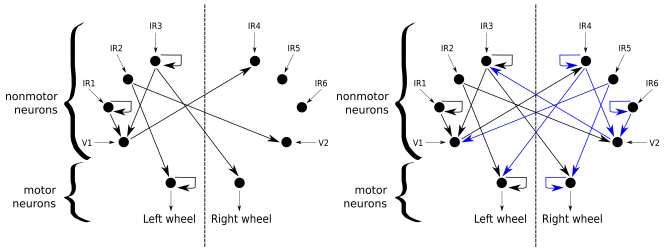


Fig. 10. Left, valid connections encoded in the genotype. Right, full network developed from the genotype.

We use exactly the same structure of recurrent neural network controllers as in the original set-up [26]. Each network contains 10 neurons: 8 non-motor neurons, each one receiving information from one sensor, and 2 motor neurons computing the speeds of the wheels. Besides, each neuron has at most 3 outgoing connections. In order to obtain bilaterally symmetric neural networks, the genotype only encodes half the network: the 5 neurons corresponding to the left sensors and the left wheel. To develop the full network, the half-network is reflected across the midline as illustrated on the Fig. 10.

The activation function is the same as in the original set-up. Let $T_j$ be the threshold of the $j^{th}$ neuron and let $w_{ij}$ be the weight of the connection from the $i^{th}$ neuron to the $j^{th}$ neuron, the activation $A_j$ of the $j^{th}$ non-motor neurons is computed as follows:

$$A_j = \begin{cases} 0, \text{ if } \sum A_i w_{ij} < T_j \\ 1, \text{ if } \sum A_i w_{ij} \geq T_j \end{cases}$$

The motor neurons have a different activation function:

$$A_j = \begin{cases} 0, & \text{if } \sum A_i w_{ij} - T_j < -1 \\ \sum A_i w_{ij} - T_j, & \text{if } -1 < \sum A_i w_{ij} - T_j \leq 1 \\ 1, & \text{if } \sum A_i w_{ij} - T_j > 1 \end{cases}$$

The genotype encodes 7 parameters for each of the 5 "left" neurons: 1) its threshold value (16 values regularly spaced from -1 to 1); 2) the destination neuron of its 3 possible outgoing connections (integer values from 1 to 16)[6]; 3) the 3 weights corresponding to these 3 connections (16 values regularly spaced from -2 to 2). Each neuron depends on 7 parameters and the full genotype contains 35 parameters. The genotype is encoded as a string of 140 bits (4 bits per parameter).

During the evaluation step, the behavior of an individual is simulated during 150 steps from a fixed initial position in each test case. At each step, the neural network is updated and the outputs of the motor neurons are multiplied by the maximal wheel speed and applied to the wheels. For each test case, the fitness value is computed as the covered Manhattan distance (in mm) from the initial position to the final position of the simulated robot plus a bonus of 1000 mm if it turns in the right direction at the junction. Let c be the controller to evaluate, if the robot has traveled $d_x$ mm along the x-axis and $d_y$ mm along the y-axis, the corresponding fitness value is computed as follows:

$$fitness(c) = d_x + d_y + \begin{cases} 1000, \text{ if right way at junction} \\ 0, \quad \text{if wrong way at junction} \end{cases}$$

The fitness values are averaged on both test cases to compute the global fitness value of an individual. The maximal fitness value is about 1700 mm. In reality, the e-puck robot is controlled through a Bluetooth connection by a laptop that sends new wheel speeds at each 0.4 seconds during 150 steps. A test case then lasts 1 minute and a full experiment lasts around 2 minutes (two test cases). The maximal wheel speed is fixed to 2 cm per second [7].

For the experiments in reality, robot's 2D trajectory is recorded with two CODA cx1 scanners (Charnwood Dynamics Ltd, UK). To track the motion of the robot, we rely on 1 marker on its top.

### *B. Problems encountered when implementing Jakobi's approach*

In order to obtain controllers that transfer well from simulation to reality, Jakobi argues that if we look for robust enough individuals in simulation, they should transfer well onto the real device and also be robust in reality. Here, we only consider the reality gap problem and the concerns on robustness in reality are not especially evaluated.

In the T-maze problem, the discrepancies between the simulation and the reality are mainly a result of the sensors of the e-puck robot. The infrared sensor values can indeed dramatically deviate from an experiment to another, as well as the duration of the color pattern detection by the camera. To cope with these potential discrepancies, some parameters of the simulation model are varied during the optimization, in order to prevent over-fitting of the solutions to a specific context. In Jakobi's set-up, each individual is then evaluated in ten simulations whose parameters change from one generation to another. Nevertheless, three important points of the original approach are not detailed in [26] and [27]:

- the amount by which each parameter has to be varied from trial to trial is not specified;
- the implementation of the noise zone at the end of the first corridor is sparsely described: "sometimes [the infrared sensors] returned maximum values, sometimes low values, sometimes totally random values" [27];
- there is no criterion to select the best-of-run individual.

Moreover, all parameters appear to be independently changed from individual to individual from generation to generation, which can lead to significant effects on the fitness: fitness obtained in large mazes can be higher than fitness obtained in small mazes. Preliminary experiments with such optimization schemes did not lead to individuals with high fitness or high robustness in simulation with the budget of evaluations fixed for the set-up.

[6]As there only are 10 neurons in the network, fictive neuron numbers (from 11 to 16) are used to encode non-existing connections so that the number of outgoing connections by neuron can be changed.

[7]The maximal wheel speed in Jakobi's original setup was 8 cm per second and the simulation was updated 10 times per second. As the Bluetooth connection with the e-puck robot only allows 2.5 updates per second, we decided to divide the maximal wheel speed by 4.

### C. Approaches

*1) Noise-based approach inspired from Jakobi's one:* In order to obtain individuals with good robustness abilities in simulation with the desired budget of evaluations (about $2 \cdot 10^5$ evaluations), we made the following choices in response to the three points raised above:

- we define for each parameter a set of discrete values it can take and all the individuals are evaluated in the same ten simulations at each generation;
- the sensors, which detect the noise zone, return values generated at random in $[0, 3500]$;
- at the last generation, the best solution is the one whose fitness value in simulation is maximal on the ten current simulations, although it is not always the one with the best robustness abilities.

In our set-up, the varying parameters are: 1) the maze size (4 values regularly distributed from 500 mm to 650 mm); 2) the initial orientation of the robot (9 values regularly distributed from -20 to 20); 3) the length of the color patterns (5 values regularly distributed from 20 mm to 120 mm). The evolved individuals have to achieve the task in 10 chosen simulations among the 180 possible ones at each generation. One of these simulations approximately corresponds to the real values of the parameters: 550 mm for the map size, 0 for the initial orientation and 70 mm for the length of the color patterns. At the beginning of the generation, the value of each parameter of the ten simulations is randomly selected in the corresponding discrete set.

*2) Transferability approach:* The *Transferability approach* relies on multi-objective optimization with a Pareto-based ranking scheme. The individuals are optimized in the simulation with the real parameter values. The two following entities have to be defined at first: the exact STR disparity measure and the behavioral features used for the in silico metric.

We define the STR disparity measure on a given controller as the sum of two nMSE respectively computed on the x-values and on the y-values between the corresponding trajectories observed in reality and in simulation during the 150 steps. Let $S_t = \{x_S^t, y_S^t\}$ and $R_t = \{x_R^t, y_R^t\}$ be the horizontal position of the robot respectively recorded in simulation and in reality, let $\overline{x_S}$ (resp. $\overline{y_S}$, $\overline{x_R}$, $\overline{y_R}$) be the mean of $x_S^t$ (resp. $y_S^t$, $x_R^t$, $y_R^t$), the exact STR disparity $D^*(c)$ of the controller $c$ is:

$$D^*(c) = \sum_{i=1}^{150} \frac{(x_S^i - x_R^i)^2}{\overline{x_S}\ \overline{x_R}} + \sum_{i=1}^{150} \frac{(y_S^i - y_R^i)^2}{\overline{y_S}\ \overline{y_R}}$$

We empirically choose a threshold on the STR disparity value equal to $D^*_{threshold} = 0.1$.

The in silico metric is based on a 6-dimensional behavior (3 for the "left" test case, 3 for the "right" one): 1) the covered Manhattan distance obtained during the corresponding test case; 2) the minimal distance to the left wall in the first corridor; 3) the minimal distance to the right wall in the first corridor. Before computing the behavioral distance $b_{dist}$ between two individuals in simulation, the corresponding behavioral features are normalized by their upper bounds in the simulation: $\{800, 100, 100, 800, 100, 100\}$.

The diversity threshold is empirically fixed at $\tau_{div} = 0.25$ in order to conduct around 15 transfer experiments in reality during a whole optimization. Nevertheless, we clearly know in this experiment when an individual is optimal in simulation or in reality (class A1 on the Fig. 6), when the corresponding behavior is evaluated. Consequently, we can stop the optimization process as soon as an optimal individual in simulation appears to be also optimal when transferred onto the e-puck robot with a sufficiently low STR disparity value. Moreover, in order to transfer as few individuals as possible, only individuals that are optimal in simulation are transferred onto the e-puck robot during the optimization process. This constraint does not concern the individual transferred at the beginning of the process to initialize the surrogate model, which is randomly generated.

*3) Evolution on the physical robot:* For comparison, we conduct a simple evolution directly on the real robot. Each individual is evaluated by the fitness on the physical robot and a diversity objective. The diversity value is computed as the average Hamming distance based on the binary genotype (string of 140 bits) to the rest of the population. The size of the population is 4 and the number of generations is 5, which implies 20 experiments on the robot by run. Because this approach relies on much more experiments in reality than the other approaches, it has only been repeated 3 times to have the same amount of experiments in reality in total (about 60 experiments for each approach).

*4) Surrogate modelling of the fitness:* A classic way to optimize controllers with expensive fitness functions comes down to directly building a surrogate model of the fitness in reality [31], instead of relying on a simulation model: the surrogate model tries to approximate the relation between the control parameters and the real fitness. Once such a model is available, the best solution is the controller that maximizes the approximate fitness function according to the model.

To build the surrogate model, a given number of selected controllers have to be transferred onto the physical robot to record the corresponding fitness values in reality. The controllers to be transferred can be randomly generated, but they are usually selected with an update heuristic, which explores the zones where the model can be at most improved.

For the T-maze problem, the surrogate model can hardly rely on Kriging interpolation with our budget of evaluations on the robot: the controller depends on 35 parameters, which implies at least 71 experiments to initialize the Kriging model. Consequently, we use the Inverse Distance Weighting (IDW) interpolation to build the surrogate model.

Our implementation relies on the same update heuristic as in the Transferability approach. The generation of the controller to be transferred is included in a multi-objective evolutionary optimization process. A population of 200 individuals is evolved during 1000 generations based on two objectives to be maximized: the fitness approximated by the surrogate model and a diversity objective. It allows to guide the search for new

transfer experiments towards seemingly efficient solutions or towards the most different solutions from the already transferred ones. At each generation, a controller is transferred onto the physical robot, among those whose diversity value is above a pre-defined threshold. If no controller has a sufficiently high diversity value, no controller is transferred.

The diversity objective is computed as the minimal Hamming distance based on the binary genotype[8] to the already transferred controllers. Let $\mathfrak{C}_T$ be the set of the already transferred controllers, the diversity value $diversity(c)$ corresponding to the controller $c$ is computed as follows:

$$diversity(c) = \min_{c_i \in \mathfrak{C}_T} Hamming(c,\ c_i)$$

The diversity threshold is set to $\tau_{div} = 0.55$, which allows about 6 transfer experiments onto the robot during a run. The best solution of a run is the transferred controller with the highest fitness value.

*5) Control approach:* For the *Control approach*, each controller is evaluated by its global fitness in the simulation with the real parameter values. There is no transfer during the optimization and the best solution of the run is the controller whose fitness value is maximal.

*6) Control approach + Local search:* Concerning the last implemented approach, the optimization process follows the same scheme as in the *Control approach*. Once the best solution in simulation is selected, it is used as the starting point of a local search to maximize the fitness on the physical robot. The following process is iterated 5 times at the end of a run[9]: the best solution is mutated and if the mutated individual achieves a better fitness on the robot, it becomes the new best solution. If the best solution obtained at first in simulation is already optimal in reality, this solution is kept and no local search is conducted

### D. Results

Table I sums up the location of the evaluation step (simulation, reality or both) along with the number of experiments done on the physical robot by run for each approach.

All the approaches have been implemented using the state-of-the-art MOEA NSGA-II [14], based on non-dominated sorting and elitist tournament selection.[10] For single-objective optimization schemes (Control approaches), NSGA-II is equivalent to an elitist tournament-based EA. Two operators are defined on the genotype described above: recombination crossover and bitwise mutation with probability 0.7 (a mutated individual undergoes about 2 bit changes). Except for the evolution on the physical robot (4 individuals, 5 generations, repeated 3 times), a population of 200 individuals is evolved during 1000 generations and each approach is run 10 times.

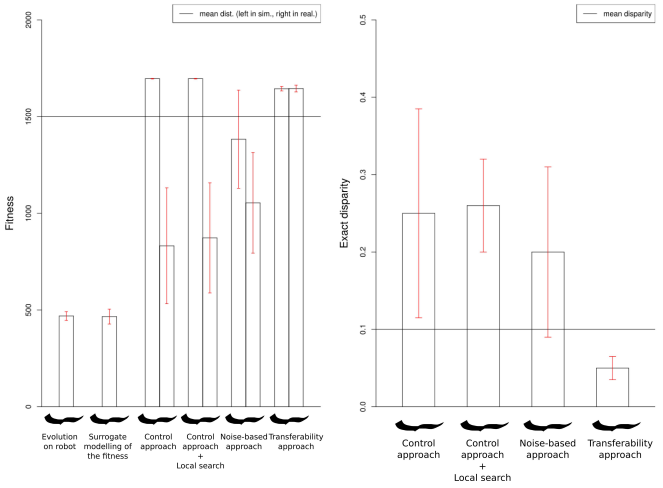


Fig. 11. Left, fitness of the best solutions found with all the approaches. Right, exact disparity values computed for these individuals. Means and standard deviations are computed on 10 runs, except for the evolution on the robot with 3 runs. Error bars indicate one unit of standard deviation. The number of experiments done on the physical robot by run for each approach is indicated in Table I. The *Transferability approach* behaves at best and always finds best solutions that are both transferable (disparity values lower than $D^*_{threshold} = 0.1$) and optimal in reality on the e-puck robot.

Quantitative results are shown on Fig. 11 and in table II. The *Control approach* behaves well for 3 runs out of 10 by finding best solutions both optimal in simulation and in reality. On average, the best solutions achieve a 1696 mm fitness value in simulation (sd = 3 mm) and a 832 mm fitness value on the physical robot (sd = 599 mm) with 0.25 disparity (sd = 0.27). It demonstrates that there is a reality gap problem between our minimal simulation and the real set-up, but also that the minimal simulation is relatively realistic, as it sometimes allows to find optimal solutions in reality. A typical behavior obtained in reality with the *Control approach* is pictured on Fig. 12.

For the *Control approach + Local search*, the best solutions after the local search achieve 873 mm on average in reality (sd = 569 mm) with 0.26 disparity (sd = 0.12 mm) compared to the best solution found at first in simulation. Although the local search allows to upgrade the fitness values by 59 mm on average (sd = 38 mm), which is significantly greater than 0 (Welch's t-test p-value = $3 \cdot 10^{-3}$), none of the improved solutions are optimal on the robot. A local search is not able to retrieve as high fitness values on the physical robot as in simulation.

The two reality-based optimization approaches, *evolution on the physical robot* and *surrogate modelling of the real fitness*, achieve clearly worse results, with respective average fitness values of 469 mm (sd = 45 mm) and 466 mm (sd = 77 mm). The best solutions for these two approaches go straight to the end of the first corridor without taking into account the color patterns. It highlights that only few experiments on the robot are not sufficient to find the optimal behaviors from scratch, notably because of the wide control space ($1.4 \cdot 10^{42}$ possible

[8]The Hamming distance between two binary genotypes is the number of bits different between the bitstrings.

[9]The number of iterations is derived from the number of transfer experiments that revealed to be enough on average for the *Transferability approach* (cf. table I).

[10]This work has been implemented within the Sferes$_{v2}$ framework [43]. The source code is available at: http://www.isir.fr/evorob_db

| Approach | Running time | Success rate | Percentage of solved simulations |
|---|---|---|---|
| Control app. | $\sim$ 4h | 30% | 27% |
| Control app. + Local search | $\sim$ 4.25h | 30% | 27% |
| Noise-based app. | $\sim$ 12h | 30% | 83% |
| Transferability app. | $\sim$ 0.75h | 100% | 19% |

TABLE II
RUNNING TIME AND SUCCESS RATE ONTO THE PHYSICAL ROBOT FOR ALL THE APPROACHES COMPUTED ON 10 RUNS. THE PERCENTAGE OF SOLVED SIMULATIONS INDICATES HOW THE BEST SOLUTIONS FOUND WITH EACH APPROACH BEHAVE ON THE 180 POSSIBLE SIMULATIONS USED WITH THE NOISE-BASED SET-UP.

individuals).

Concerning the *noise-based approach*, the original results obtained in [26] are not reproduced: only 3 runs out of 10 lead to optimal solutions in reality, while the method always worked in the original set-up. On average, the best solutions achieve 1383 mm in simulation (sd = 507 mm) and 1054 mm in reality (sd = 520 mm) with 0.20 disparity (sd = 0.22). It is not very surprising, because there are a lot of missing details to accurately re-implement Jakobi's approach and some of our choices can have led to worse results. However, the noise-based approach behaves better than both Control approaches on average. In addition, recent results on a similar experimental set-up suggest that the noise-based approach needs a larger budget of evaluations to converge [49].

The *Transferability approach* works well for all the 10 runs with low disparity values and no significant gap between simulation and reality regarding the fitness values. On average, the best solutions achieve 1644 mm in simulation (sd = 23 mm) and 1645 mm in reality (sd = 35 mm) with 0.05 disparity (sd = 0.03)[11]. A typical behavior obtained in reality with the *Transferability approach* is pictured on Fig. 12. The approach also clearly outperforms the method based on surrogate modelling of the real fitness. Building a reliable surrogate model of the real fitness function from scratch would probably require more experiments on the robot for this application.

Another interesting result (cf. table II) concerns the running time: while conducting few experiments in reality should slow down the optimization process, it allows to solve the task in very few generations (52 generations on average, less than 1 hour) with 6 transfers onto the e-puck robot by run on average (cf. table I). Indeed, as the optimality of the individuals is directly known once the corresponding behavior is evaluated in the T-maze, such an algorithm can be stopped as soon as an optimal solution is found. On the contrary, the two other approaches have to be run during a fixed number of generations before any transfer, successful or not, onto the real device.

The percentage of solved simulations in table II corresponds to the proportion of the 180 possible simulations in the noise-based set-up that are solved on average by the best solutions found with each approach. For *noise-based approach*, the high value (83%) proves that the optimization process looks for the most robust solutions in simulations while no one solves all the test cases. Anyway, it appears that there is no clear link between this "robustness" value and the transferability from simulation to reality.

The values are much lower for the *Control approach* and the *Transferability approach*, respectively 27% and 19%. It notably means that the best solutions found with the *Transferability approach* would not have been selected with *noise-based approach*. In other words for this experimental set-up, robustness in simulation does not always mean transferable controllers and transferable controllers does not lead to particularly robust behaviors in simulation.

In conclusion, the Transferability approach clearly outperforms the reality-based optimization methods, along with our implementation of the noise-based approach. It appears to be a simple methodology to cross the reality gap for the T-maze problem.

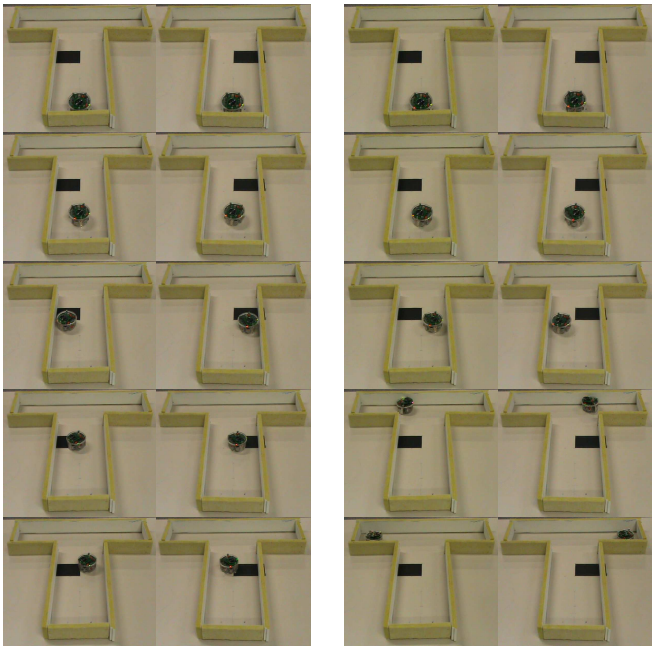

Fig. 12. Left, typical behavior evolved with the Control approach: the transferred behavior is inefficient and the robot does not solve the task. Right, typical behavior evolved with the Transferability approach: the robot solves the task in both test cases. For each approach, the left column (resp. right) of figures corresponds to the case with the color pattern on the left (resp. right).

## V. APPLICATION II: QUADRUPEDAL WALKING ROBOT

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 do not need the same level of accuracy to be correctly modeled in simulation, they are expected to achieve different transferability performances on the real device. To exploit such a gait variety, our second application consists in finding as fast as possible transferable walking gaits on a quadrupedal 8-DOF robot (Fig. 13). The fitness is the distance covered by the robot during a fixed time. Contrary to the previous application, the optimality of a given solution cannot be directly derived from

[11]Videos of typical behaviors obtained with the Control approach and the Transferability approach are available at http://people.isir.upmc.fr/koos.

| Approach | Evaluation | | Number of experiments |
|---|---|---|---|
| | simulation | reality | on the physical robot |
| Evolution on robot | | x | 20 |
| Surrogate modelling of the fitness IDW | | x | 6 (mean, sd = 1) |
| Control app. | x | | 1 |
| Control app. + Local search | x | x | 6 |
| Noise-based app. | x | | 0 |
| Transferability app. | x | x | 6 (mean, sd = 2) |

TABLE I
LOCATION OF THE EVALUATION STEP: FULLY IN SIMULATION, FULLY IN REALITY OR PARTLY IN BOTH. AVERAGE NUMBER OF EXPERIMENTS ON THE PHYSICAL ROBOT DURING A RUN FOR EACH APPROACH.

the corresponding behavior whether in simulation or on the real robot (class A2 on the Fig. 6), as the maximal robot speed is unknown.

### A. Robot and experimental set-up

The physical robot is made from a Bioloid Kit and has been built after the wheeled-legged robot Hylos [20] designed for autonomous planetary/volcanic exploration. The reduced-scale robot reproduces Hylos' degrees of freedom and its global geometry. 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. The speed of the servos is automatically fixed by the built-in controller and only depends on the position error (see Dynamixel documentation for more details). The maximal speed value is fixed at 1.75 rad/s. 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 [5]. All the parameters are set to the default values proposed by the library. The ground is simulated with non-zero friction. The engine uses the Projected Gauss Seidel constraint solver for handling collision and joint constraints with a maximal number of iterations set to 200. (see the Bullet Documentation for more details: http://bulletphysics.org).

The reality gap problem with this simulation model springs from two main reasons:

- the slippage observed on the physical ground is not accurately modeled;
- some unstable behaviors lead to unrealistically high fitness values in simulation, notably because of leg contacts.

For our application, 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, servos' built-in controller (according to the Dynamixel documentation). The simulated robot is made of 14 rigid bodies and 8 hinge constraints to model joints. Jakobi's methodology can hardly be envisaged for such an application, as it is difficult to define a set of relevant parameters in simulation whose variations would lead to robust controllers.

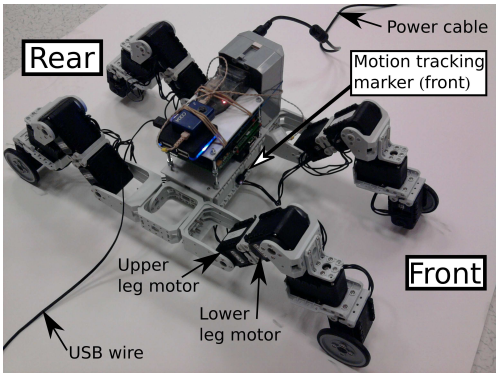


Fig. 13. Quadrupedal walking robot used in our experimental set-up.

Consequently, we did not try to apply Jakobi's approach to this application.

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 3-dimensional 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 horizontal 2D 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 Fig. 13). The trajectory of the geometric center is then obtained by averaging the 2 markers' positions.

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 $\alpha^d$ of the motor $i$ at time t is obtained by:

$$\alpha^d(i,t) = \frac{5\pi}{12} * dir(i) * p_1 - \frac{5\pi}{12} * p_2 * sin(2\pi t - \phi(i))$$ [12]

[12] Angular positions of the Dynamixel AX-12+ Robot Actuators are constrained in $[-\frac{5\pi}{12}, \frac{5\pi}{12}]$.

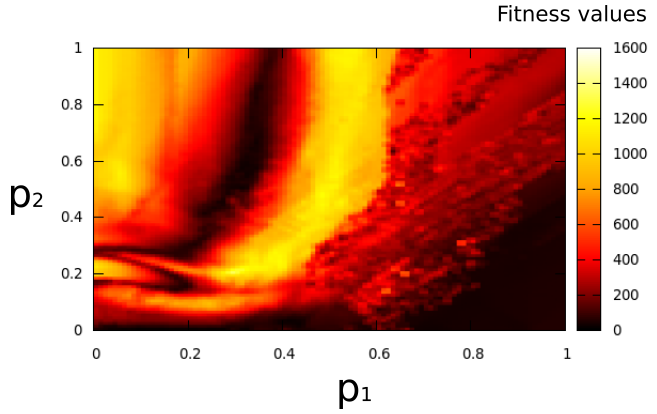


Fig. 14. Fitness landscape in simulation of the application II (covered distance, mm). This picture shows the covered distances in simulation computed for all the possible individuals on the whole control space (about $2.6 \cdot 10^5$ individuals). The control parameters $p_1$ and $p_2$ respectively are in x-coordinates and in y-coordinates. The color denotes the strength of the fitness value.

$dir(i)$ is 1 for both motors of the front-right leg and for both motors of the rear-left leg, -1 otherwise (see Fig. 13 for orientation). The phase angle $\phi(i)$ is 0 for the upper leg motors of each leg and $\pi/2$ for the lower leg motors of each leg. Both motors of one leg consequently have the same control signal with different phases.

The fitness is the distance covered by the robot in 10 seconds. Although there are only two parameters to optimize, the fitness landscape in simulation is complex as shown on Figure 14. Such fitness variations highlight a highly non-linear relation between genotype and phenotype in this application.

### B. Approaches

*1) Transferability approach:* The exact STR disparity measure is based on the real and simulated distances from the origin $\sqrt{x^2+y^2}$ of the robot's geometric center that are computed respectively from the recorded real and simulated trajectories for each sampled data point. These distances are used to evaluate the *exact STR disparity* of the transferred controller. 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.

The in silico metric is based on the following 3 behavioral features computed in simulation to sum up the behavior of each evaluated controller:

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

At each controller evaluation made within the simulator, these behavioral features are computed and normalized with the values $\{1.5, 0.2, 3.14\}$ before any use. It corresponds to their upper bound values in the simulation. We only allow 10 transfers on average for each run with a diversity threshold value $\tau_{div} = 0.1$.

*2) Evolution on the physical robot:* For comparison, we conduct a simple evolution directly on the real robot. Each individual is evaluated by the covered distance in reality and a diversity objective. The diversity value is computed as the average Euclidean distance based on the vectors of control parameters ($p_1$, $p_2$) to the rest of the population. The size of the population is 4 and the number of generations is 5, which implies 20 experiments on the robot by run. Because this approach relies on much more experiments in reality than the other approaches, it has only been repeated 5 times to have the same amount of experiments in reality in total (about 100 experiments for each approach).

*3) Surrogate modelling of the fitness:* A classic way to optimize controllers with expensive fitness functions comes down to directly build a surrogate model of the fitness in reality [31], instead of relying on a simulation model: the surrogate model tries to approximate the relation between the control parameters and the real fitness. Once such a model is available, the best solution is the controller that maximizes the approximate fitness function according to the model.

To build the surrogate model, a given number of selected controllers have to be transferred onto the physical robot to record the corresponding fitness values in reality. The controllers to be transferred can be randomly generated, but they are usually selected with an update heuristic, which explores the zones where the model can be at most improved.

For the quadrupedal locomotion problem, the use of Kriging interpolation is consistent with our budget of evaluations on the robot: the controller depends on 2 parameters, which implies only 5 experiments to initialize the Kriging model. Consequently, we implement two variants of this method depending on which interpolation techniques is used to build the surrogate model: Inverse Distance Weighting (IDW) or Kriging.

*a) Using Inverse Distance Weighting:* For this variant, our implementation relies on the same update heuristic as in the Transferability approach. The generation of the controller to be transferred is included in a multi-objective evolutionary optimization process. A population of 40 individuals is evolved during 100 generations based on two objectives to be maximized: the fitness approximated by the surrogate model and a diversity objective. It allows to guide the search for new transfer experiments towards seemingly efficient solutions or towards the most different solutions from the already transferred ones. At each generation, a controller is transferred onto the physical robot, among those whose diversity value is above a pre-defined threshold. If no controller has a sufficiently high diversity value, no controller is transferred.

The diversity objective is computed as the minimal Euclidean distance based on the genotype to the already transferred controllers. Let $\mathfrak{C}_T$ be the set of the already transferred controllers, the diversity value $diversity(c)$ corresponding to the controller $c$ is computed as follows:

$$diversity(c) = \min_{c_i \in \mathfrak{C}_T} Euclidean(c,\ c_i)$$

The diversity threshold is set to $\tau_{div} = 0.075$, which allows about 10 transfer experiments onto the robot during a run. The best solution of a run is the transferred controller with the highest fitness value.

*b) Using Kriging:* As there are only two control parameters for this application, we also implement a variant of this approach by using Kriging interpolation to build the surrogate model of the real covered distance. Our implementation is closely based on the method described in [23] and does not rely on evolutionary optimization. As the surrogate model builds the relation between the two control parameters and the covered distance in reality, five preliminary experiments are needed to initialize the Kriging model. In order to limit the number of experiments conducted onto the robot, the model is always initialized with the same five experiments, which correspond to the following vectors of control parameters: (0, 0), (1, 0), (0, 1), (1, 1), (0.5, 0.5).

The update heuristic proposed by [23] follows 3 steps:

- an optimization process takes place to find both the maximizer $c_{maxfit}$ of the approximated fitness and the maximizer $c_{maxerr}$ of the expected mean square error according to the current Kriging model;
- if $c_{maxfit}$ is not too close to an already transferred controller, it is transferred; otherwise, $c_{maxerr}$ is transferred;
- the surrogate model is updated with the new data points.

The whole heuristic is repeated 10 times, to allow 10 transfer experiments by run. As the search space is relatively small, we randomly generate $10^4$ individuals to find both maximizers $c_{maxfit}$ and $c_{maxerr}$ without optimization: the controller with the maximal approximated fitness is $c_{maxfit}$ and the one with the maximal expected mean square error is $c_{maxerr}$. The maximizer $c_{maxfit}$ is too close to an already transferred controller, if their Euclidean distance is lower than a pre-defined threshold $\epsilon$. The threshold $\epsilon$ is set to 0.025.

The Kriging-based approach has been run 10 times. The best solution of a run is the transferred controller with the highest real fitness value. The implementation and the use of the Kriging model rely on the DACE Matlab toolbox [36].

*4) Control approach:* For the *Control approach*, 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 Transferability approach.

*5) Control approach + Local search:* As for the previous application, this approach is the same as the *Control approach*, followed by a local optimization on the physical robot. Once the best solution in simulation is selected, it is used as the starting point of a local search to maximize the fitness on the physical robot. The following process is iterated 10 times[13]: the best found solution is mutated and if the mutated individual achieves a better covered distance on the robot, it becomes the new best found solution.

*6) Control approach + Diversity:* The last implemented approach optimizes two objectives: (1) the covered distance in simulation; (2) the behavioral diversity objective. Let $C_S$ be the archive of the already selected controllers and $b_{dist}$ the in silico metric, the behavioral diversity value $diversity(c)$ for a given controller $c$ is:

$$diversity(c) = \min_{c_i \in \mathfrak{C}_S} b_{dist}(c,\ c_i)$$

The archive $C_S$ is built as follows: at each generation of the optimization process, if some controllers have a diversity value greater than the diversity threshold $\tau_{div}$, one among them is randomly selected and added to $C_S$.

### C. Results

The Table III sums up the location of the evaluation step (simulation, reality or both) along with the number of experiments done on the physical robot by run for each approach.

As for the previous application, all the approaches have been implemented using the MOEA NSGA-II [14][14]. The evolved genotypes contain the two parameters of the controllers $(p_1, p_2) \in [0, 1]^2$. Two operators are defined on the genotype: Gaussian point mutation[15] and recombination crossover. The mutation probability is 0.5. Exception for the evolution on the physical robot (4 individuals, 5 generations, repeated 5 times) and the Kriging-based method, each approach has been run 10 times with a population of 40 individuals evolved during 100 generations.

*1) Results with the Control approaches:* The best solutions obtained with the *Control approach* achieve 1294 mm on average (sd = 55 mm) in the simulation and 411 mm on average (sd = 425 mm) on the real robot. The corresponding STR disparity values are 4.90 on average with standard deviation 8.78. A typical gait obtained in reality with the *Control approach* is pictured on Fig. 18. The gap problem is a little bit complex because among the best solutions found in 10 runs, 3 controllers out of 10 transfer well on the physical robot with high fitness and 7 do not. Two conclusions can be drawn from these results:

- there is a reality gap problem between the simulation model and the real robot;
- however, the simulator is worthwhile because it allows *sometimes* to find good solutions that transfer well on the real robot.

The *Control approach + Local search* achieves better results regarding the covered distance in reality with 557 mm on

[13]The number of iterations is derived from the number of transfer experiments done on average with the *Transferability approach* (cf. table III).

[14]This work has been implemented within the Sferes$_{v2}$ framework [43]. The source code is available at: http://www.isir.fr/evorob_db

[15]Mutation parameters: 0 mean, 0.2 standard deviation

average (sd = 489 mm) and with 2.53 disparity on average (sd = 3.00), compared to the initial best solution found in simulation. The fitness in reality is improved by 146 mm on average (sd = 90 mm). Although such an improvement is significantly greater than 0 (Welch's t-test p-value = $3 \cdot 10^{-4}$), the solutions improved from non-efficient controllers are not efficient in reality either. Consequently, such a local search does not allow to retrieve similar fitness values than in simulation and is not sufficient to cross the reality gap in this set-up.

One can argue that the *Control approach* is not able to find transferable controllers, because it does not find the true optimal solutions in simulation. Then, with an additional diversity objective, the *Control approach + Diversity* [16] should lead to better individuals in simulation and then better performances in reality. Nevertheless, the results shown on the Fig. 16 tends towards the opposite conclusion: the *Control approach + Diversity* indeed finds better solutions in simulation on average (1379 mm, sd = 54 mm), but these individuals are significantly less transferable to reality than with the *Control approach* with a 3.20 disparity on average (sd = 3.22) and less efficient with a 385 mm covered distance on the real robot on average (sd = 279 mm). Moreover, among the best solutions found in 10 runs with the *Control approach + Diversity*, none are both efficient and transferable, although 3 out of 10 are with the *Control approach*. Consequently, the *Control approach* leads to better results, possibly because it *does not always find* the true optimal solutions in simulation, which is quite in agreement with the antagonism we hypothesize between efficiency and transferability.

In order to study more in details the reality gap problem for this application, the best individuals found with the *Control approach + Diversity* have been transferred onto the physical robot every 5 generations. The Figure 15 sums up the fitness of these individuals in simulation and in reality (average and standard deviation). Two conclusions can be drawn from the graph: 1) the individuals randomly generated at the initial generation are neither transferable nor efficient in reality; 2) the solutions become less efficient in reality and less transferable during optimization, which highlights some over-fitting effects. It notably means that undoing over-fitting is not sufficient to retrieve efficient solutions in reality in this application.

*2) Results without the simulation model:* Contrary to the previous application, the three reality-based optimization approaches, evolution on the physical robot and both methods based on the surrogate modelling of the real fitness, behave relatively well and significantly better than the Control approaches regarding the obtained covered distance in reality. There is no significant difference between the performances of these three approaches and the use of Kriging does not lead to better results than with IDW interpolation. (Welch's t-test p-values $> 0.73$). Such results show that the real fitness landscape of this application is likely to be simple

[16]We select the same diversity threshold $\tau_{div} = 0.1$ as with the *Transferability approach*, which leads to archives $C_S$ of 11 individuals on average (sd = 1).

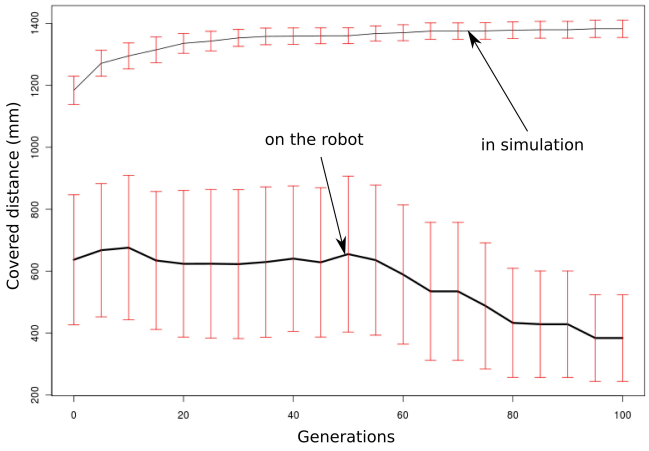


Fig. 15. Fitness of the best individuals found with the *Control approach + Diversity* plotted every 5 generations in simulation (top) and in reality (bottom). Error bars indicate one unit of standard deviation.

contrary to the landscape observed in simulation (Fig. 14), as optimization with few experiments lead to good results. However, we emphasize that Kriging methods are hardly adaptable to experiments with many parameters to be optimized due to numerous preliminary transfer tests on the test to initialize the surrogate model.

*3) Results with the Transferability approach:* The observed numbers of transfers are presented in table III and validate the choice of $\tau_{div} = 0.1$. The *Transferability approach* behaves clearly better than the 3 Control approaches (see Fig. 16) with better STR disparity values (Welch's t-test p-value $< 6 \cdot 10^{-3}$), also lower than $D^*_{threshold} = 1$, and finds more efficient controllers regarding the covered distance objective on the real robot (Welch's t-test p-values $< 4 \cdot 10^{-2}$) despite the very small number of transfer experiments (11 in a run on average, cf. table III). The best trade-off individual among the best solutions obtained with the Transferability approach achieves 1132 mm in the simulation and 1099 mm on the real robot with a 0.004 STR disparity value[17]. On average, the best solutions achieve 906 mm in the simulation (sd = 210 mm) and 848 mm on the real robot (sd = 239 mm) with a 0.16 disparity value (sd = 0.25). A typical gait obtained in reality with the *Transferability approach* is pictured on Fig. 12.

The Transferability approach then finds, for this application, good solutions that transfer well on the real robot with only few transfer experiments conducted onto the physical robot. It strengthens the global interest in such an approach to cross the reality gap problem.

Compared to the reality-based optimization approaches, the Transferability approach behaves quite better, although there is no strong statistical significance (Welch's t-test p-values $< 0.1$). As there only are two parameters to optimize, the real fitness landscape indeed is likely to be simple, although the relation between the landscapes observed in simulation and in reality is not. The Figure 17, which pictures the exhaus-

[17]Videos of typical behaviors obtained with the Control approach and the Transferability approach are available at http://people.isir.upmc.fr/koos.

| Approach | | Evaluation | | Number of experiments on the physical robot |
|---|---|---|---|---|
| | | simulation | reality | |
| Evolution on robot | | | x | 20 |
| Surrogate modelling of the fitness | IDW | | x | 10 (mean, sd = 1) |
| | Kriging | | x | 10 |
| Control app. | | x | | 1 |
| Control app. + Local search | | x | x | 10 |
| Control app. + Diversity | | x | | 1 |
| Transferability app. | | x | x | 11 (mean, sd = 2) |

TABLE III
LOCATION OF THE EVALUATION STEP: FULLY IN SIMULATION, FULLY IN REALITY OR PARTLY IN BOTH. AVERAGE NUMBER OF EXPERIMENTS ON THE PHYSICAL ROBOT DURING A RUN FOR EACH APPROACH.

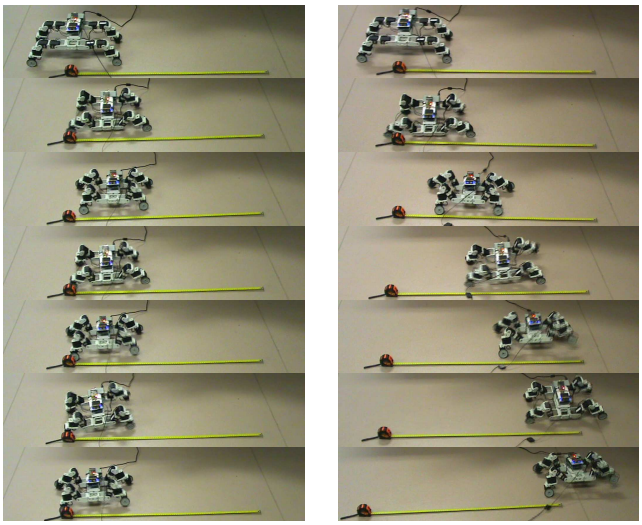

Fig. 18. Left, typical gait evolved with the Control approach: inefficient behavior due to slippage effects. Right, typical gait evolved with Transferability approaches: the behavior in reality is similarly efficient as the behavior in simulation.

tive landscape in simulation and the approximated landscape in reality, indeed is consistent with these thoughts. When evolving more complex control structures, such reality-based approaches should behave much worse, as is the case with the first application with a much larger search space.

## VI. FURTHER INVESTIGATIONS

Following these successful results, we now investigate 3 main points on the Transferability approach:

- Is the surrogate model of the STR disparity measure accurate?
- Which types of behavioral distances are appropriate as STR disparity measure and as in silico metric?
- Is the update heuristic relevant and is a diversity objective needed?

To conduct such investigations in a reasonable amount of time, we decided to resort to a fully simulated experimental set-up based on the application with the quadrupedal walking robot. In place of transfers from simulation to reality, we solve a fictive reality gap problem between a simplified simulator and the accurate simulator used in the previous section. They only differ from each other by the modeling of the servos' built-in controller. The simple simulator relies on a proportional relation between the speed and the position error, while the accurate one is based on the Dynamixel documentation. The reality gap problem is stronger in this set-up than in the original one, because optimal solutions in the simple simulator achieve unrealistic fitness values (about 14000mm) and the maximal fitness value in the accurate simulation only lies between 1300mm and 1400mm.

### A. Concerning the surrogate model

To determine if the surrogate model is accurate compared to the exact STR disparity measure, we conducted 10 runs of the Transferability approach as described in the section III with a diversity threshold $\tau_{div} = 0.05$, that corresponds to 25 transfers by run on average from the simple simulator to the accurate one (observed numbers of transfers: mean = 26, sd = 6). The graph 19 shows for all the individuals on the last non-dominated sets of each run the corresponding approximated STR disparity values and the corresponding exact STR disparity values.

According to the graph 19, the surrogate models of the STR disparity function tends to: (1) overestimate the exact function for the low disparity values; (2) underestimate the exact function for the very high disparity values. It is highly linked to the main drawback of the Inverse Distance Weighting interpolation technique: the predicted value always lies between the minimum and the maximum of the interpolated data points.

Despite this side-effect of the IDW interpolation, the Pearson's correlation coefficients between the approximated STR disparity and the exact one are relatively high, with an average of 0.76 (sd = 0.11). Similar results have been obtained on 10 runs with a diversity threshold $\tau_{div} = 0.025$ that corresponds to 45 transfer experiments by run (observed numbers of transfers: mean = 44, sd = 8): Pearson's correlation coefficient of 0.74 (sd = 0.12). It indicates that there is a strong positive monotonic relation between the approximation and the exact function. Such considerations often are sufficient to conclude that the surrogate model is of good quality [25]: the surrogate model seems to provide the evolutionary search with a good gradient.

However, in our approach, the approximated STR disparity function is not only a mean to guide the search towards transferable zones of the simulation, but is also used to

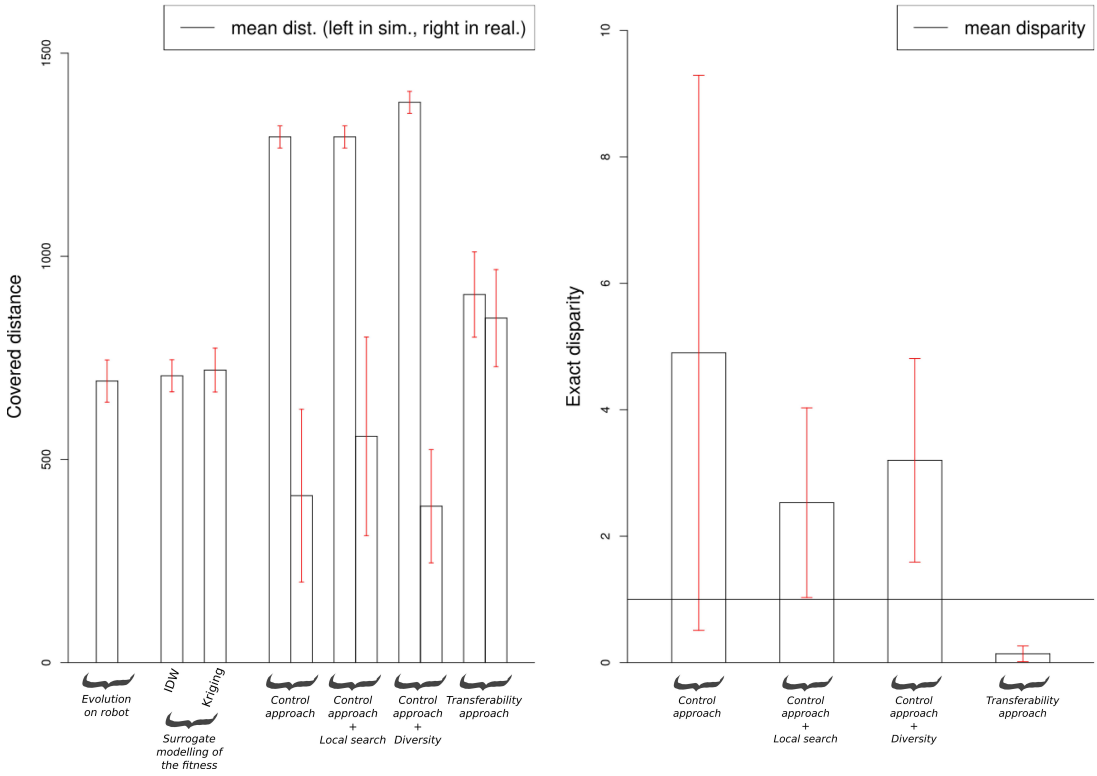


Fig. 16. Results obtained with the quadrupedal walking robot: covered distance in simulation and in reality (mm, left) and the exact disparity (right) of the best solutions obtained with each approach. Means and standard deviations are computed on 10 runs, except for the evolution on the robot with 3 runs. Error bars indicate one unit of standard deviation. The number of experiments done on the physical robot by run for each approach is indicated in Table III.

distinguish transferable controllers from non-transferable ones. The absolute quality error of the model consequently is of importance. When computing the nMSE between the approximated STR disparity and the exact one, we obtain 0.43 on average (sd = 0.29) with 25 transfer experiments and 0.38 nMSE on average (sd = 0.21) with 45 transfer experiments. Such low nMSE values indicate that the approximated STR disparity value is rather accurate, which validates the choice of the simple IDW interpolation, at least on this application.

### B. Concerning the behavioral distances

In the Transferability approach, two behavioral distances are used: (1) the transferability measure that compares simulated behaviors with real ones; (2) the in silico metric that only compares behaviors in simulation.

Let us call $d_{feat}$ the feature-based behavioral distance defined on the covered distance, the mean height of the robot and its final angular orientation[18] and $d_{traj}$ the trajectory-based behavioral distance based on the horizontal distance covered by the robot. We can now define 4 variants as summed up in the table IV. The *bFeat+DTraj* variant corresponds to the original approach.

Each of these 4 variants have been repeated 10 times with two different threshold values $\tau_{div}$ respectively corresponding to 25 transfers and 45 transfers by run on average and the results are shown on the Fig. 20. The variant *bFeat+DTraj* clearly achieves the best results. The *TrajOnly* variant also behaves well regarding the STR disparity values, but it achieves significantly lower fitness values. Both variants based on the feature-based distance as transferability measure behave much worse. The trajectory-based distance $d_{traj}$ then appears to be more efficient as a transferability measure than $d_{feat}$. The feature-based behavioral distance is probably too compact as it only takes into account 3 heterogeneous behavioral features. Building a transferability measure on such sparse informations does not provide a gradient precise enough to lead towards good trade-off solutions.

Regarding the in silico metric, there is also a significant difference between the variants *bFeat+DTraj* and *TrajOnly*. To investigate this point, we compute the nMSE between the approximated STR disparity based on the surrogate model and the exact STR disparity function on the Pareto front for each

TABLE IV
FIRST SET OF VARIANTS: IN SILICO METRIC AND STR DISPARITY.

| Variants | in silico metric $b_{dist}$ | STR disparity $D^*$ |
|---|---|---|
| *bFeat+DTraj* | $d_{feat}$ | $d_{traj}$ |
| *FeatOnly* | $d_{feat}$ | $d_{feat}$ |
| *bTraj+DFeat* | $d_{traj}$ | $d_{feat}$ |
| *TrajOnly* | $d_{traj}$ | $d_{traj}$ |

[18]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.

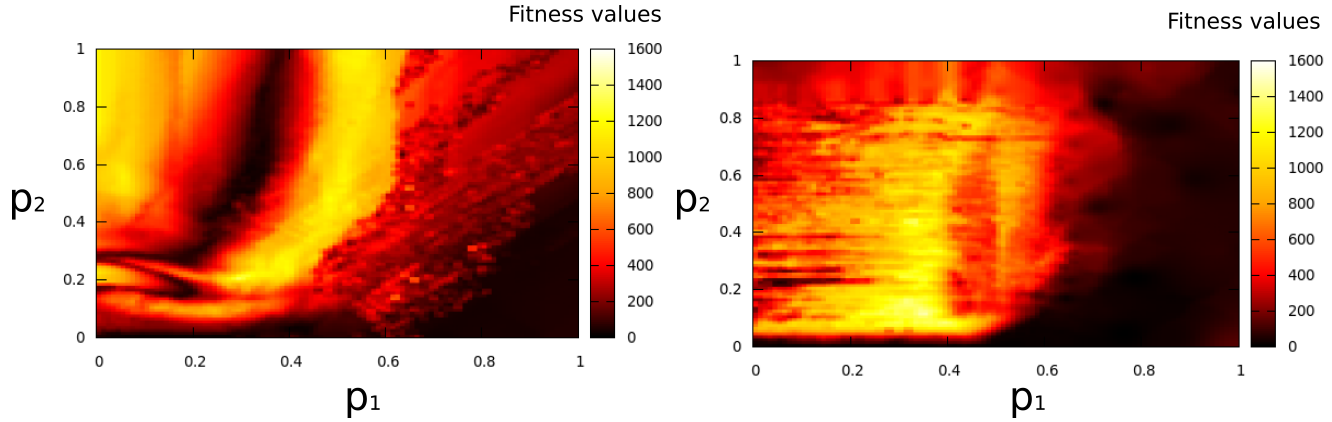


Fig. 17. Left, exhaustive fitness landscape in simulation of the application II (covered distance, mm). Right, interpolation of the fitness landscape in reality based on about 5500 transfer experiments. These experiments have mostly been conducted in the zone $p_1 < 0.6$. For higher values of $p_1$, the robot is not able to move efficiently, which systematically leads to low fitness values.

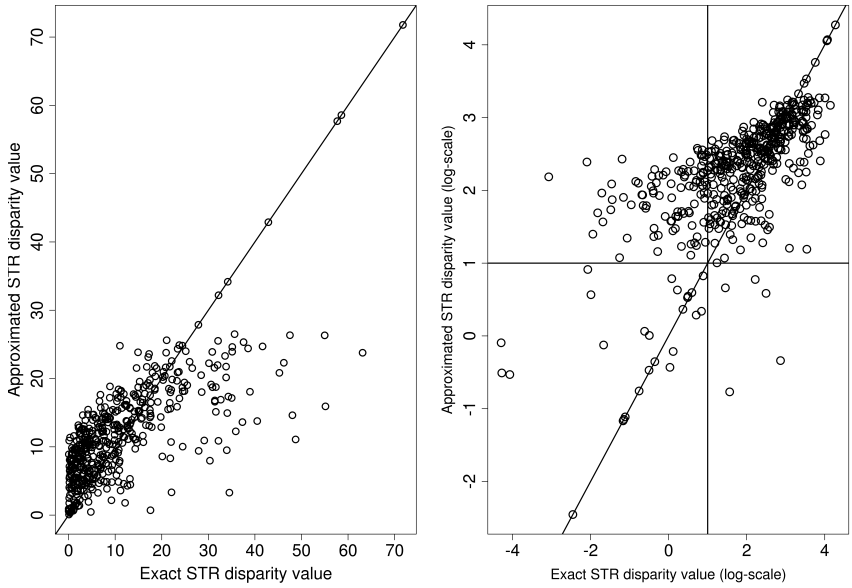


Fig. 19. Approximated STR disparity compared to exact STR disparity (right, with log-scale) for the individuals in the Pareto front obtained in 10 runs of the Transferability approach with a diversity thresholds $\tau_{div} = 0.05$ (584 individuals in all).

run. For *bFeat+DTraj*, we respectively obtain on average a 0.43 nMSE when 25 transfer experiments are allowed (sd = 0.29) and a 0.38 nMSE with 45 transfer experiments (sd = 0.21). Much worse values are observed with the *TrajOnly* variant: respectively 0.72 (sd = 0.52) and 1.13 (sd = 0.79). Consequently, the accuracy is much worse for the *TrajOnly* variant and more experiments during a run lead to even worse accuracy, which means that the trajectory-based behavioral distance $d_{traj}$ is not efficient as in silico metric. An effective in silico metric $b_{dist}$ indeed has to distinguish the different classes of behaviors observed in simulation. For instance, in the application with the quadrupedal robot, the three behavioral features separate behaviors that are efficient or not (covered distance), that make the robot overturns or do not (mean height), that are more or less stable (final orientation). Contrariwise, a distance between 2D-trajectories likely is not as discriminant as such a feature-based distance.

Such results validate our original approach with a feature-based behavioral distance as in silico metric and a trajectory-based behavioral distance as transferability measure.

### C. Concerning the update heuristic and the diversity objective

We now implement a second set of variants (table V) depending on: (1) the update heuristic used to choose the transfer experiments; (2) the presence or lack of the diversity objective. The "random" update heuristic consists in choosing at random the controller to transfer among those whose diversity value is greater than the diversity threshold $\tau_{div}$ as used in the original approach. The "max. diversity" heuristic consists in transferring the controller that maximizes the diversity value only if it is greater than $\tau_{div}$. The *RandomT & Div* variant corresponds to the original approach.

TABLE V
SECOND SET OF VARIANTS: UPDATE HEURISTIC AND DIVERSITY OBJECTIVE.

| Variants | Diversity objective | Update heuristic |
|---|---|---|
| *RandomT & Div* | × | random |
| *MaxDivT & Div* | × | max. diversity |
| *RandomT & NoDiv* | | random |

Each variant has been repeated 10 times, with two diversity threshold values $\tau_{div}$ that respectively correspond to 25 transfers and 45 transfers by run on average. All the results are shown on the Fig. 21. The variants *RandomT & Div* and *MaxDivT & Div* behave well, but the *RandomT & Div* variant shows best results, as it looks for better trade-off solutions. *MaxDivT & Div* variant finds worse trade-off solutions on average than *RandomT & Div* variant, in terms of disparity with $\tau_{div} = 0.05$ (Welch's t-test p-value=0.071) or of distance with $\tau_{div} = 0.025$ (Welch's t-test p-value=0.029). It means, counter-intuitively, that transferring the most 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 can therefore be less informative. Then, with similar numbers of transfers, the "random" update heuristic seems to be preferable to the "max. diversity" update heuristic.

The poor results obtained with *RandomT & NoDiv* show that the "random" update heuristic is not sufficient to find good trade-off solutions: the behavioral diversity objective is necessary in this set-up, as it explicitly drives the search towards individuals different from those already transferred, thus probably increasing the accuracy of the approximated STR disparity.

## VII. DISCUSSION

### A. Antagonism between efficiency and transferability

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 set-ups. Some previous hints were shown in [35] in a fully simulated set-up.

Our main result highlighting this antagonism is the difference of performances between the *Control approach* and the *Control approach + Diversity* on the application with the quadrupedal walking robot. Although the addition of a diversity objective to the fitness function allows to find more efficient solutions in simulation [15], the former approach behaves clearly better in reality than the latter one. It indicates that the most efficient solutions in simulation are not transferable from the simulation to the physical quadrupedal robot, because of the misleading fitness function in simulation. Consequently, any optimization scheme only based on this fitness function gives no guarantee that the best solutions are efficient in reality, which empirically verifies our hypothesis on the antagonism between efficiency and transferability for this application.

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 [13]; 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 [12], [13]. Handling a constraint in such a way would be equivalent to the method introduced in the present paper.

### B. Towards an on-board transferability measure

In both applications, the transferability measure relies on external information: the trajectory of the robot recorded with CODA cx1 scanners. One could argue that such external measures require heavy/costly experimental set-up which are hardly compatible with bigger robots and that on-board sensorimotor informations should be preferred to compare simulated and real behaviors.

Some recent results show that directly relying on sensor informations can be a simple way to accurately compare behaviors in simulation [15]. Nevertheless, it is only meaningful if the sensor values are accurately modeled in simulation, sometimes despite significant amounts of noise. It is also argued in [7] that accurate quantitative comparisons between two sensor time series is difficult because small initial disparities can quickly lead to very different signals, which can lead to prefer external measures for physical robots [8].

Another promising way is to exploit sensorimotor informations to obtain an accurate estimation of the trajectory by sensor integration [4], [9]. The main drawback of such a method is the drift between the real trajectory and the estimated one, which can rapidly accumulate significant estimation errors. It is sometimes pertinent to combine different types of sensors (short-range distance sensors with a camera for instance) by periodic repositioning of the estimation.

More recently, impressive results have been obtained on wheeled robots using on-board visual odometry with very accurate estimated trajectories [10], [45]. Such methods only rely on a single camera [10], sometimes coupled with classic inertial sensors [45], and appear to be robust and simple to implement. It will be investigated in our future applications.

### C. Modeling the fitness or the transferability

The use of surrogate models is increasing in robotics, most of the time to directly approximate the fitness function on the physical robot. Such approximations try to map the relation between the parameters of the controller and the fitness in reality by interpolating a global function from few experimental data with Kriging-like methods. It then assumes that close sets of parameters lead to close fitness values, which can hardly be ensured with control structures like neural networks for instance. Another issue concerns the number of parameters introduced by Kriging methods. If the number of input parameters is large, as is the case with classic control structures used in ER like neural networks, Kriging models

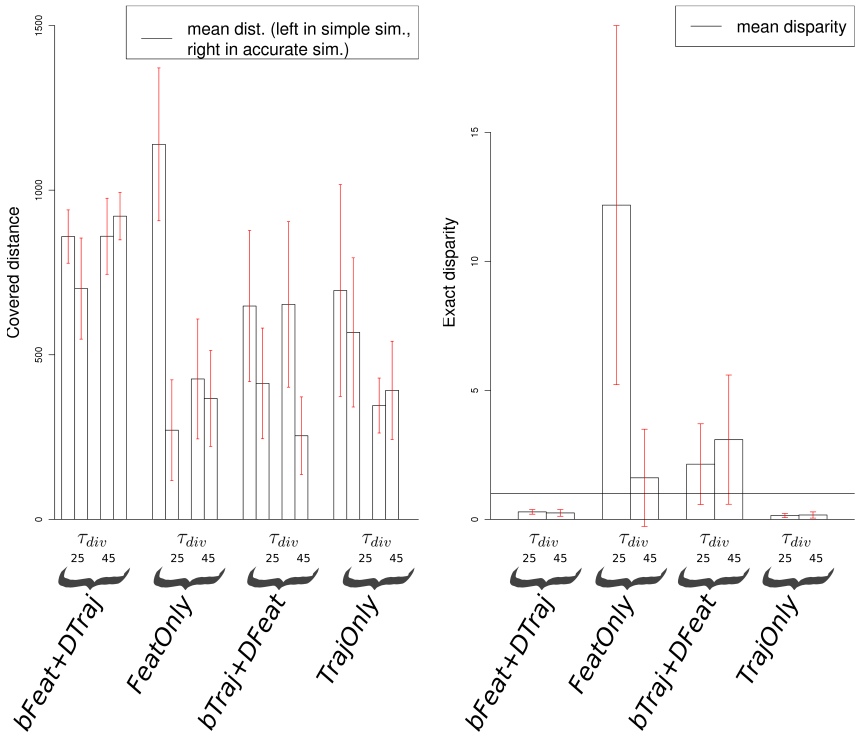


Fig. 20. Comparison between the behavioral distances: covered distance (mm, left) in the simple simulation and in the accurate one, along with the exact STR disparity values computed on trajectories (right) of the best solutions obtained with each variant. The target numbers of transfers by run on average are written above the variant names. The *bFeat+DTraj* variant behaves at best. The variant *TrajOnly* also behaves good with 25 transfers by run but finds less efficient individuals regarding the covered distance in reality (Welch's t-test p-value = 2 $10^{-4}$) with 45 transfers by run. All the other variants behave significantly worse. Error bars indicate one unit of standard deviation.

require numerous experiments on the robot to be properly initialized.

Besides, as these approaches do not rely on any prior information on the robot (contrary to simulation-based optimization), it is convenient to begin the search from an initial controller with a non-zero fitness [23], arguably to avoid typical bootstrap problems. As indicated by the results on both applications, the results obtained with a surrogate model built on the control space indeed are highly dependent on the landscape of fitness in reality. For realistic landscapes, several transfer experiments will be needed for exploring and building a sufficiently accurate surrogate model to avoid local maxima without any prior knowledge.

By modeling the transferability function rather than the fitness, the Transferability approach can exploit the simulation model by relying on comparisons between behaviors (even estimated by a simulation model) and not between controllers. The mapping between the control parameters and the corresponding behaviors can indeed be highly non-linear. Building an approximation of the transferability function in the behavior space then seems more appropriate and straightforward than in the controller space [15], which cannot be done by directly modeling the fitness. In fact, selecting one of these two approaches comes down to the availability of relevant simulation models and, in practice, simulation models are often available for robotic applications.

### D. Upgrading the simulation from the STR disparity measure

At the end of a run performed with the Transferability approach, the obtained surrogate model of the STR disparity function gives a rough landscape of which parts of the simulation are not well-modeled and which parts are realistic. It then is possible to use clustering methods to notably extract which kinds of behaviors are more or less linked to bad transferability values. Depending on the complexity of the problem, the next step, which consists to understand how the simulation model makes these behaviors non-transferable and finally to improve the model, must be conducted by interacting with experts in robotics and mechanics.

## VIII. Conclusion and further work

This paper addressed the reality gap problem in the case of controller optimization, a critical issue in Evolutionary Robotics, which often happens when resorting to simulators. We introduce the Transferability approach relying on an optimization scheme that looks for not only good controllers but also transferable ones. Controllers are evaluated by 3 main objectives in a multi-objective manner: a task-dependent fitness and a simulation-to-reality disparity that estimates controller's transferability using a surrogate model. The approach has first been compared on a T-maze experiment with the mobile e-puck robot. It has been compared to a noise-based approach inspired from Jakobi's state-of-the-art methodology and to more classic reality-based and simulation-based evolutionary

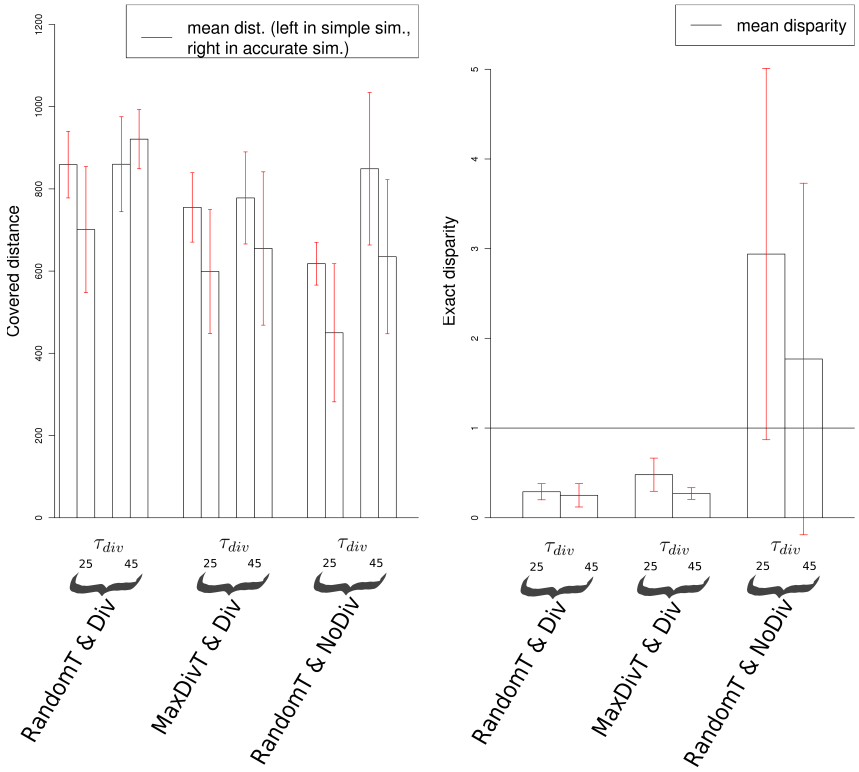


Fig. 21. Comparison between two update heuristics and the lack/presence of the diversity objective: covered distance (mm, left) in the simple simulation and in the accurate one, along with the exact STR disparity values (right) of the best solutions obtained with each variant. The target numbers of transfers by run on average are written above the variant names. For the variants *RandomT & Div* and *MaxDivT & Div*, the disparities are lower than the threshold $D^*_{treshold} = 1$: the found best solutions show good transferability properties. The *RandomT & NoDiv* variant behaves clearly worse. Error bars indicate one unit of standard deviation.

approaches. Better results were achieved by the Transferability approach regarding both exact STR disparity and covered distance in reality with very few transfer experiments during the optimization. A second application to an 8-DOF quadrupedal walking robot has also been investigated and our approach again finds controllers that are relevant regarding a walking task and that transfer well to reality. Based on these successful results, the approach appears to be a simple and relevant method to efficiently cross the reality gap in Evolutionary Robotics.

Our further prospects deal with the case where multiple simulation models are available for a single application. Each simulation model is a compromise between accuracy and speed. In the case of very accurate and slow simulation models, evolutionary computation is not competitive unless a simplified simulator is designed. Nevertheless, evolving only on the simplified simulator leads to higher reality gaps, despite a faster optimization. One wonders if the Transferability Approach can take advantage of such a set-up: an accurate but slow simulator, a faster but inaccurate simulator and the real system. There are several points to take into account for a given controller: 1) is the simplified simulator accurate enough to approximate its fitness well?; 2) otherwise, can an approximation of its fitness be inferred by conducting few experiments in the accurate simulation in the same way than with the approximated disparity value in the second approach?; 3) does it transfer well from simulation(s) to reality? The main addition in comparison to the approach presented here consists in relying on a transferability measure between the two simulators to determine which behaviors are well-modeled in the simplified simulator and to approximate more accurate fitness values for those which are not. In practice, the approach could be applied to a similar set-up presented in [55] which deals with control of an autonomous four-wheeled robot during aggressive maneuvers at high velocity to assess the Transferability Approach on a realistic robotic application.

## REFERENCES

[1] P. Abbeel, A. Coates, T. Hunter, and A. Ng. Autonomous autorotation of an RC helicopter. In *Experimental Robotics*, pages 385–394. Springer, 2009.

[2] P. Abbeel, A. Coates, M. Quigley, and A. Ng. An application of reinforcement learning to aerobatic helicopter flight. In *Proceedings of NIPS*, page 1. The MIT Press, 2007.

[3] P. Adigbli, C. Grand, J.-B. Mouret, and S. Doncieux. Nonlinear attitude and position control of a micro quadrotor using sliding mode and backstepping techniques. In *7th European Micro Air Vehicle Conference (MAV07)*, Toulouse, France, 2007.

[4] B. Barshan and H. Durrant-Whyte. Inertial navigation systems for mobile robots. *IEEE Transactions on Robotics and Automation*, 11(3):328–342, 1995.

[5] A. Boeing and T. Bräunl. Evaluation of real-time physics simulation systems. In *Proceedings 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.

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

[7] J. Bongard and H. Lipson. Nonlinear system identification using coevolution of models and tests. *IEEE Transactions on Evolutionary Computation*, 9(4):361–384, 2005.

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

[9] J. Borenstein, H. Everett, L. Feng, and D. Wehe. Mobile robot positioning: Sensors and techniques. *Journal of robotic systems*, 14(4):231–249, 1997.

[10] J. Campbell, R. Sukthankar, I. Nourbakhsh, and A. Pahwa. A robust visual odometry and precipice detection system using consumer-grade monocular vision. In *Proceedings of ICRA*, pages 3421–3427. IEEE, 2005.

[11] D. Cliff, P. Husbands, I. Harvey, et al. Evolving visually guided robots. *From animals to animats*, 2:374–383, 1993.

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

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

[14] 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.

[15] S. Doncieux and J.-B. Mouret. Behavioral diversity measures for evolutionary robotics. In *Proceedings of IEEE CEC*, pages 1303–1310, 2010.

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

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

[18] A. Gomes, I. Voiculescu, J. Jorge, B. Wyvill, and C. Galbraith. *Implicit curves and surfaces: mathematics, data structures and algorithms*. Springer Verlag, 2009.

[19] M. Gongora, B. Passow, and A. Hopgood. Robustness analysis of evolutionary controller tuning using real systems. In *Proceedings of IEEE CEC*, pages 606–613. Institute of Electrical and Electronics Engineers Inc., The, 2009.

[20] C. Grand, F. BenAmar, F. Plumet, and P. Bidaud. Decoupled control of posture and trajectory of the hybrid wheel-legged robot Hylos. In *Proceedings of IEEE ICRA*, volume 5, pages 5111–5116, 2004.

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

[22] I. Harvey, P. Husbands, and D. Cliff. Issues in evolutionary robotics. In *Proceedings of SAB*. MIT Press Bradford Books, 1992.

[23] T. Hemker, H. Sakamoto, M. Stelzer, and O. von Stryk. Hardware-in-the-loop optimization of the walking speed of a humanoid robot. In *Proceedings of CLAWAR*, pages 614–623, 2006.

[24] G. Hornby, S. Takamura, J. Yokono, O. Hanagata, T. Yamamoto, and M. Fujita. Evolving robust gaits with AIBO. In *Proceedings of IEEE ICRA*, volume 3, pages 3040–3045, 2002.

[25] M. Hüsken, Y. Jin, and B. Sendhoff. Structure optimization of neural networks for evolutionary design optimization. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 9(1):21–28, 2005.

[26] N. Jakobi. Evolutionary robotics and the radical envelope-of-noise hypothesis. *Adaptive behavior*, 6(2):325, 1997.

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

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

[29] N. Jakobi, P. Husbands, and I. Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. *Proceedings of ECAL*, pages 704–720, 1995.

[30] S. Jakobsson, B. Andersson, and F. Edelvik. Rational radial basis function interpolation with applications to antenna design. *Journal of computational and applied mathematics*, 233(4):889–904, 2009.

[31] Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 9(1):3–12, 2005.

[32] D. Jones, M. Schonlau, and W. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[33] N. Kohl and P. Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proceedings of ICRA*, volume 3, pages 2619–2624. IEEE, 2004.

[34] S. Koos, J.-B. Mouret, and S. Doncieux. Automatic system identification based on coevolution of models and tests. In *Proceedings of IEEE CEC*, pages 119–126, 2009.

[35] S. Koos, J.-B. Mouret, and S. Doncieux. Crossing the reality gap in evolutionary robotics by promoting transferable controllers. In *Proceedings of GECCO*, pages 560–567. ACM, 2010.

[36] S. Lophaven, H. Nielsen, and J. Sondergaard. DACE – A Matlab Kriging Toolbox. Technical report, IMM-TR-2002-12, IMM, 2002.

[37] T. Malvic and M. Durekovic. Application of methods: Inverse distance weighting, ordinary kriging and collocated cokriging in porosity evaluation, and comparison of results on the Benicanci and Stari Gradac fields in Croatia. *Nafta*, 54(9):331–340, 2003.

[38] E. Margerie, J.-B. Mouret, S. Doncieux, and J.-A. Meyer. Artificial evolution of the morphology and kinematics in a flapping-wing mini-UAV. *Bioinspiration & Biomimetics*, 2:65, 2007.

[39] J.-A. Meyer, P. Husbands, and I. Harvey. Evolutionary Robotics: A Survey of Applications and Problems. *Lecture Notes in Computer Science*, 1468:1–21, 1998.

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

[41] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J. Zufferey, D. Floreano, and A. Martinoli. The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th conference on autonomous robot systems and competitions*, volume 1, pages 59–65, 2009.

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

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

[44] T. Mueller, N. Pusuluri, K. Mathias, P. Cornelius, R. Barnhisel, and S. Shearer. Map quality for ordinary kriging and inverse distance weighted interpolation. *Soil Science Society of America Journal*, 68(6):2042, 2004.

[45] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3–20, 2006.

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

[47] S. Nolfi, D. Floreano, O. Miglino, and F. Mondada. How to evolve autonomous robots: Different approaches in evolutionary robotics. In *Proceedings of Artificial Life IV*, pages 190–197. Rodney Brooks and Pattie Maes, 1994.

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

[49] T. Pinville, S. Koos, J.-B. Mouret, and S. Doncieux. How to Promote Generalisation in Evolutionary Robotics: the ProGAb Approach. In *Proceedings of GECCO (to be published)*, 2011.

[50] J. Pollack, H. Lipson, S. Ficici, P. Funes, G. Hornby, and R. Watson. Evolutionary techniques in physical robotics. In *Proceedings of ICES*, pages 175–186. Springer Verlag, 2000.

[51] A. Prieto, F. Bellas, J. Becerra, B. Priego, and R. Duro. Self-organizing robot teams using asynchronous situated co-evolution. In *Proceedings of SAB*, volume 6226 of *Lecture Notes in Computer Science*, pages 565–574. Springer Berlin / Heidelberg, 2010.

[52] W. Regan, F. van Breugel, and H. Lipson. Towards evolvable hovering flight on a physical ornithopter. In *Proceedings of Artificial Life X*, volume 100, page 241, 2006.

[53] F. Saunders, E. Golden, R. White, and J. Rife. Experimental verification of soft-robot gaits evolved using a lumped dynamic model. *Robotica*, 1(-1):1–8, 2006.

[54] D. Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, pages 517–524. ACM, 1968.

[55] A. Terekhov, J.-B. Mouret, and C. Grand. Stochastic optimization of a neural network-based controller for aggressive maneuvers on loose surfaces. In *Proceedings of IEEE IROS*, 2010.

[56] 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.

[57] I. Voutchkov and A. Keane. *Multiobjective optimization using surrogates*. Springer, 2006.

[58] 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.

[59] V. Zykov, J. Bongard, and H. Lipson. Evolving dynamic gaits on a physical robot. In *Proceedings of GECCO*, volume 4, 2004.

## APPENDIX A
## PARAMETERS OF THE TRANSFERABILITY APPROACH

*Behavioral features $b^i$*

- sparse description of the behavior in simulation used to compute the in silico metric $b_{dist}$ between simulated behaviors;
- should allow to distinguish the different types of behaviors observed in simulation;
- could be inferred by conducting a principal component analysis on numerous behavioral descriptors for a large set of simulated behaviors.

*Exact STR disparity function $D^*$*

- distance computed between a simulated behavior and a real one corresponding to the same controller;
- should allow to distinguish transferable behavior from non-transferable behavior;
- based on data easy to obtain on the real robot (trajectory or sensorimotor informations).

*STR disparity threshold $D^*_{threshold}$*

- threshold on the STR disparity function $D^*$;
- disparity values lower than $D^*_{threshold}$ mean that the corresponding behaviors are transferable;
- should be fixed empirically depending on the STR disparity function used.

*Diversity threshold $\tau_{div}$*

- threshold on the diversity value in simulation used by the update heuristic of the surrogate model;
- controllers can be transferred onto the real robot if their diversity value is greater than $\tau_{div}$;
- should be roughly fixed in simulation to ensure an approximate number of transfer experiments in a run.