The Effect of Fitness Function Design on Performance in Evolutionary Robotics: The Influence of a priori Knowledge

Mohammad Divband Soorati and Heiko Hamann
Heinz Nixdorf Institute, Department of Computer Science,
University of Paderborn,
Fürstenallee 11, 33102 Paderborn, Germany
divband@mail.upb.de, heiko.hamann@uni-paderborn.de

July 18, 2015

Abstract

Fitness function design is known to be a critical feature of the evolutionary-robotics approach. Potentially, the complexity of evolving a successful controller for a given task can be reduced by integrating a priori knowledge into the fitness function which complicates the comparability of studies in evolutionary robotics. Still, there are only few publications that study the actual effects of different fitness functions on the robot's performance. In this paper, we follow the fitness function classification of Nelson et al. (2009) and investigate a selection of four classes of fitness functions that require different degrees of a priori knowledge. The robot controllers are evolved in simulation using NEAT and we investigate different tasks including obstacle avoidance and (periodic) goal homing. The best evolved controllers were then post-evaluated by examining their potential for adaptation, determining their convergence rates, and using cross-comparisons based on the different fitness function classes. The results confirm that the integration of more a priori knowledge can simplify a task and show that more attention should be paid to fitness function classes when comparing different studies.

1 Introduction

Evolutionary robotics [16, 1] emerged with great early success and is meanwhile struggling to scale to more complex tasks. Recent reports on new methods such as novelty search [9, 10] and multi-objective evolution combined with methods generating behavioral diversity [12, 13, 3] seem to have high potential. Also the new emerging field of embodied online onboard evolution [5, 21, 8], that focuses on implementing evolutionary robotics as an embodied distributed system

directly operating on a group of robots, is encouraging. Still, in all of these approaches the overall challenge of maximizing task complexity while minimizing necessary a priori knowledge persists. The idea of using little a priori knowledge in the design of either a behavioral distance measure or of a fitness function is motivated by a key advantage of evolutionary computation—it is a black-box optimizer [4]. As such, evolutionary computation is particularly suited for problems that are not well understood. The problem of defining a robot controller that generates a desired, reliable, and adaptive behavior is of that kind. However, in research of evolutionary robotics it seems to be tempting to put an effort in fitness function design. In preliminary experiments the researcher learns about specific features of the considered task, that is gaining a priori knowledge, and might consciously or unconsciously incorporate that knowledge into the fitness function. However, in an application of evolutionary robotics we would like to minimize the need for doing preliminary experiments and having to interpret them.

Nelson et al. [14] focus on that problem. They define the quality of an evolutionary-robotics approach as twofold: "the success of research is measured in the difficulty of tasks investigated, and the amount of a priori information needed to generate successful evolution of controllers capable of performing those tasks." While there is definitely a competition for more complex tasks in the community, the latter criterion of minimizing a priori knowledge seems neglected sometimes. In particular, a competition of minimizing a priori knowledge for similar tasks seems to be absent in evolutionary robotics literature except for a few directly related publications [17, 15] and a few other works that study fitness function design with strong focus on special domains [7, 6, 11]. It is also suggested by Nelson et al. [14]: "one particular ER effort might be considered an improvement over an earlier work if the later work required the use of much less a priori knowledge on the part of the researchers to evolve controllers for a similar task." In this paper, we report early results to fill this gap. An obvious challenge in this endeavor is to measure a priori knowledge. We follow the classification of fitness functions of Nelson et al. [14] to solve that issue. They define seven fitness function classes that are sorted according to the incorporated a priori knowledge. Here we focus on four of them: behavioral fitness functions (a priori knowledge incorporated: high), functional incremental fitness functions (moderately high), tailored fitness functions (moderate), and aggregate fitness functions (low). Unfortunately, classifying fitness functions according to this scheme is fuzzy and to some extent subjective. Hence, the challenge remains to define an objective measure of a priori knowledge, which is not solved within this work. However, the following study still allows for a structured approach. In alternative to the fitness function classification of Nelson et al. [14], there is also the classification of Nolfi and Floreano [16]. They define a 3-dimensional space along three axes: functional-behavioral, explicit-implicit, and external-internal. Relevant here is mostly the functional-behavioral aspect that classifies whether the fitness function defines detailed features (functional) or whether the fitness function defines abstract features (behavioral). Also the dimension explicit-implicit is of interest as it classifies basically the complexity of the fitness

function (e.g., number of components). We prefer to use the approach of Nelson et al. [14] because it allows for a more detailed classification.

In this paper, we investigate a selection tasks and fitness functions. Over a month of computational time for the simulated evolutionary runs was invested. The results indicate a clear trend across the four investigated fitness function classes. Nonetheless, our findings are not easily generalized to all possible scenario of evolutionary robotics which cannot be achieved by giving a finite number of examples. However, a reliable mathematical formalism, which would allow to prove the generality of any result in evolutionary robotics, is also out of reach. Hence, our objective is not to naively generalize from our results. Instead we want to point to the significance of a priori knowledge and we want to show that more research has to be done to fill the benchmarking gap in the literature on this issue.

This paper is structured as follows. First, the fitness function classes are introduced in Sec. 2. Then, for each experimental task these fitness function classes are instantiated and explained (Sec. 3). After designing the experiments and simulation, in Sec. 4 the performance of the robot is investigated based on post-evaluation metrics, and the final ranking of the fitness functions is presented. We conclude with a short discussion and options for future work.

2 Fitness function classes following Nelson et al.

Our study is based on the fitness function classes as defined by Nelson et al. [14]. The idea of this classification is to sort fitness functions according to the a priori knowledge that is involved in their design. As mentioned above, an objective measure of a priori knowledge is missing which complicates the classification of particular fitness functions. Still, it seems a viable approach based on the best methods that have been published so far. The classes in increasing order of required a priori knowledge are: aggregate, competitive and co-competitive, environmental incremental, tailored, functional incremental, behavioral, and training data fitness functions. In the following, we focus on the four classes only: aggregate fitness functions, behavioral fitness functions, tailored fitness functions, and functional incremental fitness functions.

Aggregate fitness functions (AFF) have the lowest degree of a priori knowledge and evaluate only *what* is achieved, that is, the completion of the task as the metric to select the best robots. The procedure of how the robot accomplishes a task is irrelevant. For example, the locomotion task of a bipedal robot can be described by an AFF that relies on traveled distance only. Such fitness functions are easy to design because they require little a priori knowledge and they are of low complexity. The drawback is obviously that there may be bootstrapping problems and there is no guidance for evolution through intermediate solutions.

Behavioral fitness functions (BFF) measure *how* the task is solved. De-

signing BFF hence requires a priori knowledge about options how the task can be solved. This way BFF also predetermine potential solutions and possibly exclude counterintuitive options. The design process for this type of fitness functions is relatively complex because it either requires good knowledge about effective solutions to the task or preliminary experiments that help to determine potential solutions.

Tailored fitness functions (TFF) combine elements of behavioral and aggregate fitness functions, that is, they combine BFF and AFF. Tailored fitness functions have a higher degree of complexity than AFF and BFF as they have several components that are typically multiplied or summed.

Functional incremental fitness functions (FIFF) are sets of fitness functions that are applied separately and gradually. The functions are applied in a series of distinct stages. The whole evolutionary process is divided into multiple intervals of generations in which only one fitness function holds. Therefore the number of fitness functions determines the number of such stages as well.

For the following experiments we need to define a particular fitness function for each task and class. All results are hence task-dependent and depending on the particular, chosen fitness function. We are interested in general conclusions, of course, and try to maximize the potential for generalizability by investigating three different tasks. Still, one could argue that our selections of fitness functions are beneficial for a certain fitness function class in individual cases and, hence, introduce a bias. To maximize the comparability of our examples, we define a structured and elaborated design strategy. First, the BFF are designed, second, the AFF. Then combinations of AFF and BFF define by design pattern the TFF. TFF combine AFF and BFF by multiplication. We make individual design decisions only for FIFF, whether the first stage starts with AFF followed by a phase with BFF or vice versa. Hence, the results within each task are easily comparable because they rely on the same fitness function components.

Besides the degree of a priori knowledge, the fitness functions can also be ordered according to their complexity. We do that in a simplistic approach. AFF have the simplest design process by considering task completion only. In contrast, BFF are complex to design. The remaining two classes are even more complex because they contain AFF and BFF as components (assuming an application scenario where we would need to define TFF and FIFF from scratch without using predefined AFF and BFF). FIFF requires the decision about the sequence of the components in addition to the component selection. Hence, it is considered as the most complex fitness function here. The order of complexity, starting with the lowest, is: AFF, BFF, TFF, and FIFF.

3 Implementation and task-specific fitness functions

We use the open-source simulator *Player/Stage*[1] to simulate a robot with differential drive that has 8 proximity sensors (6 to the front and 2 to the back) and 2 ambient light sensors (one at the front and one at the back). Our implementation is based on MultiNEAT [2] which is a portable software library implementing NEAT, a popular approach to optimize and complixify artificial neural networks [19, 20]. Using NEAT we start with simple network topologies that are then complexified during the evolutionary run. This process of complexification simplifies the approach to evolve controllers for tasks of different complexity. We evolve neural networks with 8 input neurons for the first task (obstacle avoidance) and 10 inputs for the two other tasks: 8 inputs s_i representing the input from the proximity sensors and two inputs (I_f and I_b) from the light sensors. There is a variable number of neurons on the hidden layer based on NEAT and two output neurons (speed of left and right wheel).

The total computation time was more than 112 days (30 generations $\times$ population of 30 $\times$ 3 evaluations with different initial positions $\times$ 3 repetitions each $\times$ 3 tasks $\times$ 4 fitness functions $\times$ 10 evolutionary runs $\times$ about 10 seconds for each evaluation) on evolving the best controllers even with a small number of generations. Therefore, we could run the experiments for 30 generations with a population size of 30, see Table 1 for all NEAT parameters. The controllers are evaluated for three different initial positions with different rotations. The simulation is non-deterministic due to variant time intervals between sensor readings in Player/Stage. Thus, the robot controller is simulated three times for each of these initial positions, totaling in nine evaluations. The assigned fitness is the average of these nine evaluations. All tasks use the same simulated rectangular arena of dimensions $2m \times 2m$, see Fig. 1a. The three initial positions are depicted with blue circles.

The tasks can be classified into three categories: movement (e.g., locomotion with obstacle avoidance), homing (e.g., goal homing), and object manipulation (e.g., object pushing) [18]. In the following experiments we focus on two of these categories (i.e., movement and homing) and instantiate three simple tasks: obstacle avoidance from movement category, goal homing, and periodic goal homing from the homing category. Our selection was motivated by requiring a coherent order of complexity among the tasks. This is guaranteed because the tasks of higher complexity contain the tasks of lower complexity as subtasks. Collision avoidance is part of the two other tasks. In the case of periodic goal homing, the robot needs to avoid the obstacles to reach a certain goal, hence, containing both other tasks. Next, we define the fitness function for each class and task based on the above mentioned design strategy.

[1] http://playerstage.sourceforge.net

Parameter	Value
PopulationSize	30
CrossoverRate	0.75
DynamicCompatibility	True
MutateWeightsProb	0.9
CompatTreshold	2.0
WeightMutationMaxPower	5.0
YoungAgeTreshold	15
YoungAgeFitnessBoost	1.1
OverallMutationRate	0.33
WeightReplacementMaxPower	5.0
OldAgeTreshold	31
MutateWeightsSevereProb	0.5
MinSpecies	5
WeightMutationRate	0.75
MaxSpecies	25
MaxWeight	20
SurvivalRate	0.25
MutateAddNeuronProb	0.05
RouletteWheelSelection	True
MutateAddLinkProb	0.05
RecurrentProb	0.1
MutateRemoveLinkProb	0.05

Table 1: Used NEAT parameters.

3.1 Obstacle avoidance (OA)

One of the simplest tasks in evolutionary robotics is to evolve a collision avoidance behavior [16]. Avoiding collisions is of particular relevance in embodied evolution [22]. The idea is that the robot covers as much distance as possible (otherwise staying stopped is an optimal collision avoidance behavior) while minimizing collisions with walls and objects.

For this task, BFF is based on the speed of both wheels and the inputs from the proximity sensors. This type of fitness function rewards behavioral features, that is, how the task is solved. First, the faster the robot moves the better it accomplishes the task. The first component is the average of the two wheel speeds averaged over time: $\bar{v}_l = \frac{1}{T}\sum_t v_l(t)$ and similarly for $\bar{v}_r$. Furthermore, maximal distance is covered by moving straight, that is, by avoiding rotations. The difference between the two speeds is minimized by defining $(1 - \sqrt{|\bar{v}_l - \bar{v}_r|})$ (square root gave best results in preliminary tests). The last component is $\frac{1}{\theta}\min_t(\min_i(s_i(t)))$ and rewards to stay far from obstacles. If any sensor detects an obstacle to be closer than an allowed distance, that is interpreted as a collision which needs to be penalized heavily. The threshold θ returns an extreme penalty of $\theta = 10^3$ if at least one collision occurred during the evaluation. Otherwise, we have $\theta = 1$. The behavioral fitness function for collision avoidance is defined

by

$$F_{\text{bff}}^{\text{collAv}} = \left(\frac{\bar{v}_l + \bar{v}_r}{2}\right)(1 - \sqrt{|\bar{v}_l - \bar{v}_r|})\frac{1}{\theta}\min_t(\min_i(s_i(t))). \quad (1)$$

For AFF, the focus is on task completion. Here we simply define the aggregate fitness function by the distance δ that the robot traveled during evaluation,

$$F_{\text{aff}}^{\text{collAv}} = \delta. \quad (2)$$

In the case of FIFF, a set of fitness functions is required. Following our design strategy we reuse the behavioral and aggregate fitness functions (eqs. 1 and 2), divide the evolution into two phases, and define the functional incremental fitness functions as a 2-tuple by

$$F_{\text{fiff}}^{\text{collAv}} = \{F_{\text{bff}}^{\text{collAv}}, F_{\text{aff}}^{\text{collAv}}\}. \quad (3)$$

We define TFF by the product of the behavioral and aggregate fitness function

$$F_{\text{fiff}}^{\text{collAv}} = F_{\text{bff}}^{\text{collAv}} F_{\text{aff}}^{\text{collAv}}. \quad (4)$$

3.2 Goal Homing (GH)

Goal homing is also a frequently investigated task in evolutionary robotics [14]. While in the collision avoidance task the robot's direction is not specified, the robot is now required to move as close as possible to a light source that is located in the middle of the arena. Finding the best path to the highest light intensity is the goal in this task. In simulation the light intensity is proportional to the inverse of the squared distance d between the robot's position and the arena center

$$I \propto \frac{1}{d^2}. \quad (5)$$

For all the following fitness functions, collisions are penalized by introducing a discount to the accumulated light intensity reward. For the time steps after the collision the light intensity is rewarded only by one half of the intensity.

In this task, the robot should approach the light quickly and then stay close to the light source. Hence, the robot should move fast, when it is far from the light, and it should stop moving, when it is close to the light. Hence, we reward the accumulated difference between intensity and traveled distance over all simulation steps. Based on empirical experience we add a weight $c = 5$ to increase the impact of light intensity relative to traveled distance. The behavioral fitness function is defined by

$$F_{\text{bff}}^{\text{gh}} = \sum_t \left|cI(t) - \Delta d(t)\right|, \quad (6)$$

whereas $I(t) = \frac{1}{2}(I_f + I_b)$ is the average of the detected light intensity in simulation step t and $\Delta d(t)$ is the robot's current speed. For the case of AFF, a simple function based on accumulated light intensity is defined by

$$F_{\text{aff}}^{\text{gh}} = \sum_t I(t). \quad (7)$$

For FIFF, we define two objectives: first approach the light and second avoid collisions. We reuse function $F_{\text{aff}}^{\text{collAv}} = \delta$ from eq. 2 (δ is traveled distance).

$$F_{\text{fiff}}^{\text{gh},1} = \sum_t I(t), \tag{8a}$$

$$F_{\text{fiff}}^{\text{gh},2} = F_{\text{aff}}^{\text{collAv}} = \delta \tag{8b}$$

The tailored fitness function is the product of behavioral and aggregate fitness function, defined by

$$F_{\text{tff}}^{\text{gh}} = \left(\sum_t I(t)\right)\left(\sum_t \left|cI(t) - \Delta d(t)\right|\right). \tag{9}$$

3.3 Periodic Goal Homing (PGH)

In periodic goal homing, the robot not only has to approach the light, but has to maximize the distance to the light again once it has been reached, and has to repeat that periodically. The robot should increase and then decrease the light intensity as much as possible. We divide the arena into two areas. A ring-shaped area A_1 (see Fig. 1c), containing all positions with intensities of $0.3 < I(t) < 0.7$ and an area A_2 that contains the remaining parts of the arena ($I(t) > 0.7,\ I(t) < 0.3$). The robot should traverse area A_1 as fast as possible. In addition, obstacle avoidance is imposed because the evaluation is stopped whenever the robot hits an obstacle. We define the behavioral fitness function

$$F_{\text{bff}}^{\text{pgh}} = \sum_t \varphi(t)\Delta d(t), \tag{10}$$

with $\varphi(t) = -1$ for $0.3 < I(t) < 0.7$ and $\varphi(t) = +1$ otherwise. φ penalizes spending time on A_1 and $\Delta d(t)$ is the robot's speed at time t, hence rewards moving fast on A_2.

For the case of AFF, we need a global feature that allows to determine whether the robot accomplishes its task well. From a number of options ranging from frequency analysis to measuring wavelengths, we pick the easily implemented option of the standard deviation σ that is directly calculated from the data acquired during the evaluation. Here, σ is a statistical function that gets the set of all light intensities $I = \{I(t)|\text{for all time steps } t\}$ as input and returns the standard deviation. A big standard deviation indicates a good periodic goal homing behavior. We define the aggregate fitness function

$$F_{\text{aff}}^{\text{pgh}} = \sigma(I). \tag{11}$$

We select two fitness functions to define the FIFF. Following our design strategy we would use the BFF and AFF but experiments showed that for this task that approach is ineffective (data not shown). Instead, we choose two AFFs of previous tasks (goal homing and collision avoidance), one rewards maximizing

light intensity the other rewards movement.

$$F_{\text{fiff}}^{\text{pgh},1} = F_{\text{aff}}^{\text{gh},1} = \sum_t I(t), \tag{12a}$$

$$F_{\text{fiff}}^{\text{pgh},2} = F_{\text{aff}}^{\text{collAv}} = \delta. \tag{12b}$$

This way, the robot is assumed to move toward the light, but should keep moving. This is a rather rough way of defining the task but we follow our design strategy of reusing fitness functions for FIFF and not defining new components.

The tailored fitness function is the product of the behavioral and aggregate fitness functions, defined by

$$F_{\text{tff}}^{\text{pgh}} = F_{\text{bff}}^{\text{pgh}} F_{\text{aff}}^{\text{pgh}}. \tag{13}$$

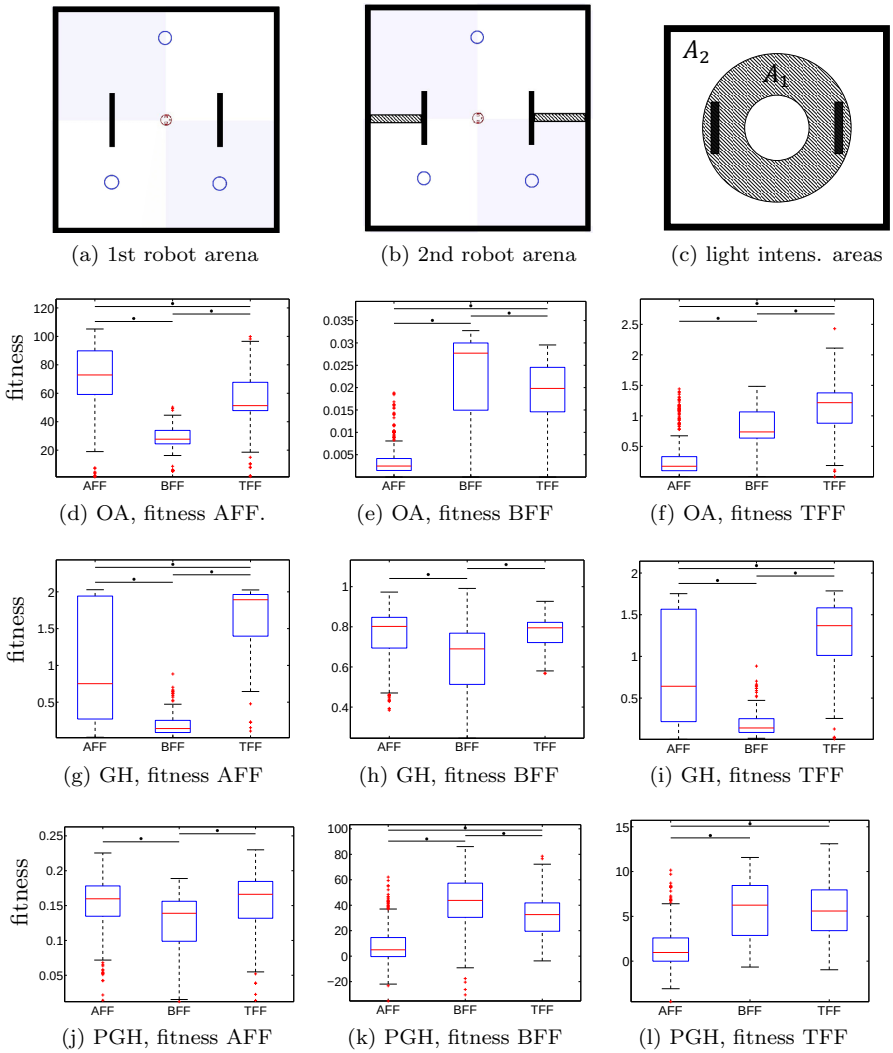
These twelve fitness functions were used to evolve robot controllers for these three tasks. Our observations of the robots' behaviors indicate that the obtained best controllers were able to achieve a periodic behavior.

4 Post-evaluation and readaptat-ion tests

In the evolutionary runs with the above described twelve fitness functions we obtain a best controller from each evolutionary run. The evolutionary algorithm is effective and hence they are best relative to their fitness function for which they have been evolved. However, the question that we have to answer is how well they perform according to an objective performance measure because we want to know the best fitness function for each task. This is in itself a difficult problem, in fact, a self-referential problem because this objective performance measure is itself, of course, a fitness function. Therefore, we do a cross-comparison of each controller with each of the three fitness functions and for each task. Our aim is to come up with a conclusion about the effectiveness of our different fitness functions and in the end of the different fitness function classes. Three types of post-evaluations and adaptation tests were done. First, the best controllers were post-evaluated 30 times with each of the fitness functions (Sec. 4.1). Second, for the last populations the evolution was resumed with each of the fitness functions for a few generations to test for evolutionary adaption (Sec. 4.2). Third, for the last populations the evolution was resumed for each of the fitness functions in a new environment to test re-adaptation (walls added, Sec. 4.3).

4.1 Post-evaluations in cross-comparison

We have three tasks and four classes of fitness functions that help in evolving best controllers that accomplish each of these tasks. For each task, the best controllers are post-evaluated with the same environment again, but this time we evaluate them with all fitness functions. The idea is to do a full cross-comparison and to find the most effective fitness function class. FIFF cannot be investigated



(a) 1st robot arena (b) 2nd robot arena (c) light intens. areas

(d) OA, fitness AFF. (e) OA, fitness BFF (f) OA, fitness TFF

(g) GH, fitness AFF (h) GH, fitness BFF (i) GH, fitness TFF

(j) PGH, fitness AFF (k) PGH, fitness BFF (l) PGH, fitness TFF

Figure 1: Simulated robot arenas, logically defined light intensity areas for Periodic Goal Homing (PGH), fitness of best controllers for Obstacle Avoidance (OA, 2nd row), Goal Homing (GH, 3rd row), and PGH (4th row) obtained by evolving with Aggregate Fitness Function (AFF, 1st boxplot in each figure), Behavioral Fitness Function (BFF, 2nd boxplot), and Tailored Fitness Function (TFF, 3rd boxplot) but post-evaluated with AFF (1st column), BFF (2nd column), and TFF (3rd column).

here because that would require to resume the evolution of populations to allow for a useful comparison, which is left for Sec. 4.2 and Sec. 4.3.

The results of our cross-comparisons between all tasks are shown in Fig. 1 (rows 2, 3, and 4). For each task (rows) we take the best controllers evolved using AFF (1st column), BFF (2nd column), and TFF (3rd column) and post-evaluate them against AFF (1st boxplot in each diagram), BFF (2nd boxplot), and TFF (3rd boxplot). For example, Fig. 1d shows the results for the best controllers evolved with AFF which are post-evaluated against AFF, BFF, and TFF. We compare the three fitness function classes based on their rank in each test. In order to rank the results, that were achieved by the different fitness functions, we applied significance tests based on the Wilcoxon rank sum test. Significantly different results ($p < 0.05$) receive different ranks while insignificant results receive the same rank. In the same example (Fig. 1d), lines with asterisks denote the significances in the data. All pairs of fitness functions are significantly different in this case. AFF is ranked the highest followed by TFF, while BFF is ranked the lowest. The first column in Tab. 2 represents the ranking in Fig. 1d. The second and third columns belong to the comparisons in Fig. 1e and Fig. 1f, respectively. After summing the ranks, the forth column denotes the overall ranking of the three fitness functions for obstacle avoidance. Tab. 3 shows the overall ranks for all of the tasks. By summing the ranks we see that TFF in total is ranked best. AFF and BFF compete basically on the same level but with a slight advantage for BFF (rank sum better by one).

4.2 Resuming evolution in the same environment

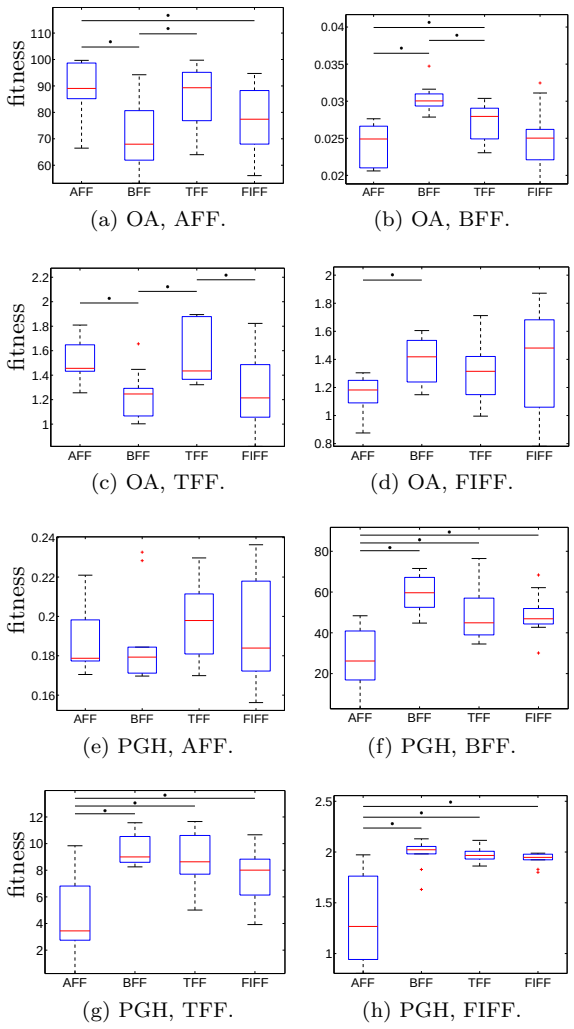
Next, we test how the populations, that were evolved with different fitness functions, adapt to new fitness functions when we resume evolution. We take the last population from each evolutionary run and resume evolution for 10 generations. This also allows to include the FIFF using both functions for 5 generations each.

For each fitness function we have a population of the last generation for each evolutionary run. We take these populations and continue to evolve them. For example, we take a population evolved with AFF and continue to evolve it for 10 generations with AFF. However, we also take a population evolved with AFF and resume evolution for 10 generations with BFF and similarly for all other cases of the cross-comparison. From each of the final populations after 10 generations we take the best controller for the following analysis.

Fig. 2 gives the results for tasks OA and PGH only. Taking into account also the results for GH (not shown), Tab. 5 is computed for all tasks and fitness functions. As shown in the overall column of this table, TFF is best followed by BFF, AFF, and FIFF.

4.3 Re-adaptation, convergence rate, and full comparison

In the last part of this study, we test the potential for re-adaptation in the evolved populations. We evolve the population of controllers for 30 generations and then resume the evolution of the populations for another 10 generations in



(a) OA, AFF. (b) OA, BFF.

(c) OA, TFF. (d) OA, FIFF.

(e) PGH, AFF. (f) PGH, BFF.

(g) PGH, TFF. (h) PGH, FIFF.

Figure 2: Best controllers for Obstacle Avoidance (OA, a-d) and Periodic Goal Homing (PGH, e-h) obtained by evolving with Aggregate Fitness Function (AFF), Behavioral Fitness Function (BFF), and Tailored Fitness Function (TFF) but then resume evolution with AFF, BFF, and TFF.

a changed environment (two walls added, shown in dashed pattern, Fig. 1b). The evolution is resumed with the same fitness function that was used for evolution before (no cross-comparison). The robot controllers have to re-adapt to accomplish their tasks. To check the ratio of re-adaptation after the change in the environment, we consider the median of the best fitness for each generation. b_g is the median of the best controllers' fitness from 10 evolutionary runs for

FF	AFF	BFF	TFF	Overall
AFF	1	3	3	3
BFF	3	1	2	2
TFF	2	2	1	1

Table 2: Detailed ranking of post-evaluation shown in Fig. 1d, e, and f only for obstacle avoidance.

FF	OA	GH	PGH	overall
AFF	3	2	2	3
BFF	2	3	1	2
TFF	1	1	1	1

Table 3: Overall ranking of post-evaluation.

generation g. To compare the maximum median values over generations before ($g \leq 30$) and after the change ($g > 30$), we define the ratio

$$R_r = \max_{30<g\leq 40} (b_g) \left(\max_{0\leq g\leq 30} (b_g) \right)^{-1}. \tag{14}$$

We are interested in finding fitness functions that score high on R and hence evolved adaptive controllers. For this test we give no boxplots but only the ranks, see Tab. 6. The overall ranks show that TFF is best in adapting to the new environment followed by BFF, AFF, and FIFF.

Finally, in a fourth test we investigate the rate of convergence because we are interested in the dependence of the optimization process speed on the fitness function classes. We define a visual evaluation of graphs to assign a value to the quality of convergence rate. For the measured functions $F_{\text{best}}(g)$ of best fitness over generations g, we measure the area below the curve and relate it to the overall area of the rectangle defined by $(0, 0)$ and $(g_{\max}, F^{\max}_{\text{best}})$, which gives $g_{\max} \times F^{\max}_{\text{best}}$. Therefore, the convergence rate is defined as

$$R_c = \frac{1}{g_{\max} \times F^{\max}_{\text{best}}} \sum_{g=1}^{g_{\max}} F_{\text{best}}(g). \tag{15}$$

The investigation of all evolutionary runs gives the information in Tab. 4 that shows AFF as the best, followed by BFF, and TFF (FIFF not considered here).

FF	OA	GH	PGH	overall
AFF	2	1	1	1
BFF	1	2	2	2
TFF	3	3	3	3

Table 4: Overall ranking of convergence rate.

FF	OA	GH	PGH	overall
AFF	2	2	4	3
BFF	2	4	1	2
TFF	1	1	2	1
FIFF	3	3	3	4

Table 5: Overall ranking of re-evolution.

FF	OA	GH	PGH	overall
TFF	1	1	3	1
BFF	3	2	2	2
AFF	4	3	1	3
FIFF	2	4	4	4

Table 6: Overall ranking of re-adaptation.

A summary of all results based on ranks from all of the tests is shown in Tab. 7. What we refer to as 'simplicity' in Tab. 7 is the inverse of complexity as defined in Sec. 2. We find that fitness function classes that incorporate a lot of a priori knowledge, such as TFF and BFF, achieve higher performance across different tasks and across several performance features (post-evaluation, adaptation). The convergence rate of AFF is good but only because they achieve lower maximal values than BFF and TFF.

#	post-eval.	resumed evolution	re-adapted	conv. rate	simplicity
1	TFF	TFF	TFF	AFF	AFF
2	BFF	BFF	BFF	BFF	BFF
3	AFF	AFF	AFF	TFF	TFF
4	-	FIFF	FIFF	-	FIFF

Table 7: Final ranking of the fitness functions.

5 Conclusion

We have presented a study on how a priori knowledge in the design of the fitness function influences the performance of the evolutionary algorithm in evolutionary robotics. Although we face the problem of fuzzy and subjective definitions of a priori knowledge and fitness function classifications [14], we still get a clear result for the proposed examples. Fitness functions that include a high degree of a priori knowledge (here TFF and BFF) influence the performance positively. They help to simplify the evolution of a successful controller and hence simplify the chosen task. However, we also know that incorporating a high degree of a priori knowledge foils the dominant idea of evolutionary computation

as a black-box optimizer. Therefore, there is a problem of comparability in evolutionary robotics. While there is a competition for more and more complex tasks in the community, more attention could possibly be paid to the amount of a priori knowledge included in the design of fitness functions. More studies on the same task with an effort on minimizing a priori knowledge could possibly be done to have a competition for minimal a priori knowledge, too. Furthermore, we require a more precise and objective measure of a priori knowledge.

Our study focuses on the standard approach of evolutionary robotics that requires the definition of a fitness function. There are also recent alternative approaches, such as novelty search [10] and multi-objective behavioral diversity [12, 13], that either substitute the fitness function with a measure for behavioral diversity or add behavioral diversity as an additional objective. Also called 'task-agnostic' approaches as discussed in [4]. However, the design of the behavioral distance measure might require an effort similar to the standard fitness function design challenge. For example, Mouret and Doncieux [13] discuss this analogy between fitness function design and the design of a measure for behavioral diversity: "More importantly, novelty search critically depends on a good behavior characterization to create a gradient. Researchers in ER used to craft fitness function to create a perfect fitness gradient; novelty search users have to craft the behavior distance to create a similar gradient. This last option may be easier for some problems but eventually some distances will be hard to define."

In future work we plan to do a similar study also for measures of behavioral diversity. In addition, we plan to extend the fitness function classification approach of Nelson et al. [14] to allow for a better motivation of chosen fitness functions. Furthermore a collection of results for a diverse set of fitness functions and measures of behavioral diversity for selected tasks, also including negative results, could prove to be helpful for the field.

Acknowledgment

Supported by EU-H2020 project 'florarobotica', no. 640959.

References

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

[2] P. Chervenski and S. Ryan. MultiNEAT, project website;. http://www.multineat.com/.

[3] S. Doncieux and J.-B. Mouret. Dynamic behavioral diversity. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation (GECCO'12)*, pages 1453–1454. ACM, 2012.

[4] S. Doncieux and J.-B. Mouret. Beyond black-box optimization: a review of selective pressures for evolutionary robotics. *Evolutionary Intelligence*, 7(2):71–93, 2014.

[5] Á. E. Eiben, E. Haasdijk, and N. Bredeche. Embodied, on-line, on-board evolution for autonomous robotics. In P. Levi and S. Kernbach, editors, *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution*, volume 7 of *Cognitive Systems Monographs*, pages 362–384. Springer, 2010.

[6] W. Fan, E. A. Fox, P. Pathak, and H. Wu. The effects of fitness functions on genetic programming-based ranking discovery for web search. *Journal of the American Society for Information Science and Technology*, 55(7):628–636, 2004.

[7] E. Fast, C. Le Goues, S. Forrest, and W. Weimer. Designing better fitness functions for automated program repair. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (GECCO'10)*, pages 965–972. ACM, 2010.

[8] E. Haasdijk and N. Bredeche. Controlling task distribution in MONEE. In *Advances In Artificial Life (ECAL'13)*, pages 671–678, 2013.

[9] J. Lehman and K. O. Stanley. Exploiting open-endedness to solve problems through the search for novelty. In S. Bullock, J. Noble, R. Watson, and M. A. Bedau, editors, *Artificial Life XI: Proceedings of the 11th International Conference on the Simulation and Synthesis of Living Systems*, pages 329–336. MIT Press, 2008.

[10] J. Lehman and K. O. Stanley. Improving evolvability through novelty search and self-adaptation. In *Proceedings of the 2011 IEEE Congress on Evolutionary Computation (CEC'11)*, pages 2693–2700. IEEE, 2011.

[11] C. López-Pujalte, V. P. Guerrero-Bote, and F. de Moya-Anegón. Order-based fitness functions for genetic algorithms applied to relevance feedback. *Journal of the American Society for Information Science and Technology*, 54(2):152–160, 2003.

[12] J.-B. Mouret and S. Doncieux. Using behavioral exploration objectives to solve deceptive problems in neuro-evolution. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (GECCO'09)*, pages 627–634. ACM, 2009.

[13] J.-B. Mouret and S. Doncieux. Encouraging behavioral diversity in evolutionary robotics: an empirical study. *Evolutionary Computation*, 20(1):91–133, 2012.

[14] A. L. Nelson, G. J. Barlow, and L. Doitsidis. Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems*, 57:345–370, 2009.

[15] A. L. Nelson and E. Grant. Using direct competition to select for competent controllers in evolutionary robotics. *Robotics and Autonomous Systems*, 54(10):840–857, 2006.

[16] S. Nolfi and D. Floreano. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. MIT Press, 2000.

[17] I. G. Sprinkhuizen-Kuyper, R. Kortmann, and E. O. Postma. Fitness functions for evolving box-pushing behaviour. In A. van den Bosch and H. Weigand, editors, *Proceedings of the 12th Belgium–Netherlands Artificial Intelligence Conference (BNAIC'00)*, pages 275–282, 2000.

[18] C. Sprong. Common tasks in evolutionary robotics, an overview. Technical report, Faculty of Sciences, University of Amsterdam, Netherlands, 2011.

[19] K. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

[20] K. O. Stanley and R. Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21(1):63–100, Jan. 2004.

[21] J. Stradner, H. Hamann, P. Zahadat, T. Schmickl, and K. Crailsheim. Online, on-board evolution of reaction-diffusion control for self-adaptation. In C. Adami, D. M. Bryson, C. Ofria, and R. T. Pennock, editors, *Alife XIII*, pages 597–598. MIT Press, 2012.

[22] M. Wahby and H. Hamann. On the tradeoff between hardware protection and optimization success: A case study in onboard evolutionary robotics for autonomous parallel parking. In *Applications of Evolutionary Computation (EvoApplications 2015)*, volume 9028 of *Lecture Notes in Computer Science*, pages 759–770. Springer, 2015.