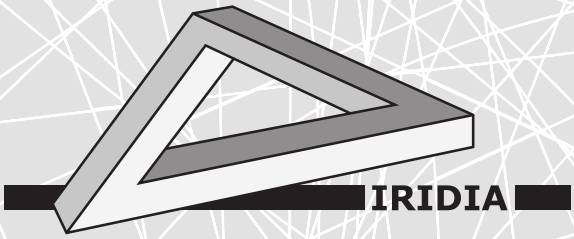

Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

Advantages of multi-objective optimisation in evolutionary robotics: survey and case studies

Vito Trianni and Manuel López-Ibáñez

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2014-014

November 2014

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:
IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2014-014

Advantages of multi-objective optimisation in evolutionary robotics: survey and case studies

Vito Trianni[1], Manuel López-Ibáñez[2],
1 Institute of Cognitive Sciences and Technologies (ISTC), National Research Council (CNR), Rome, Italy.
2 IRIDIA, CoDE, Université Libre de Bruxelles (ULB), Brussels, Belgium
∗ **E-mail: vito.trianni@istc.cnr.it**

Abstract

The application of multi-objective optimisation to evolutionary robotics has been so far relatively limited. Despite a few examples exist, the benefits of multi-objective optimisation when applied to the design of autonomous robotic systems have not been clearly spelled out and experimentally demonstrated. A survey of the literature on evolutionary robotics shows the lack of systematic studies confronting single- and multi-objective approaches. This paper fills this gap: starting from well-known results in multi-objective optimisation, we discuss how to tackle commonly recognised problems in evolutionary robotics. In particular, we show that multi-objective optimisation (i) allows evolving a more varied set of behaviours by exploring multiple trade-offs of the objectives to optimise, (ii) supports the evolution of the desired behaviour through the introduction of objectives as proxies, (iii) avoids the premature convergence to local optima possibly introduced by multi-component fitness functions, and (iv) solves the bootstrap problem exploiting ancillary objectives to guide evolution in the early phases. We present an experimental demonstration of these benefits in the context of standard case studies in robotics: maze navigation in a single robot domain, flocking in a swarm robotics context, and a strictly collaborative task in collective robotics.

1 Introduction

Artificial evolution is a powerful tool for designing the behaviour of autonomous robots, as largely demonstrated in the evolutionary robotics literature [1–3]. The advantages reside in the possibility of exploiting the sensorimotor coordination resulting from the interactions between the robot's brain (i.e., the control software), its body (i.e., the embodiment including sensors and actuators) and the environment [4]. Through evolutionary design, the designer is exempted from a detailed modelling of the brain-body-environment interactions, and solutions can be obtained that match the specificities and statistical regularities of the problem at hand. However, a suitable engineering methodology to support the fundamental design choices in evolutionary robotics is currently missing. Indeed, most of the studies in evolutionary robotics strongly rely on the expertise of the designer, who assembles the evolutionary system following his personal intuition. Only few attempts have been made to propose an engineering methodology for the evolutionary design of robotic controllers [5, 6]. Among the several design choices required to devise an evolutionary robotics experiment, the fitness function is particularly important because it explicitly determines the selective pressures that drive the evolutionary search. However, the definition of the fitness function is not always straightforward [3, 7, 8].

First of all, the features of the desired behaviour must be encoded in a measurable form, but often there is no definite and measurable way of expressing either the dynamical aspects of the robots' behaviour or the desired outcome. Hence, it is common to find in the literature fitness functions composed of multiple *behavioural terms* that contribute to the one or the other feature (e.g., move fast, avoid obstacles, approach target) [8]. That is, the design problem in evolutionary robotics is intrinsically characterised by multiple objectives, but often tackled as a single-objective problem by means of an a priori aggregation (i.e., scalarization) of the various objectives. However, finding the correct trade-off between possibly

conflicting terms is not easy. In this case, a multi-objective approach may provide with a set of solutions that explore different trade-offs, so that a principled choice can be made a posteriori.

Secondly, the fitness function must support the *evolvability* of the system, that is, the possibility to progressively synthesise better solutions through random exploration [9]. That is, even if a single-objective (fitness) function—or a scalarization of multiple objectives—is available for the desired behaviour, this function might be difficult to optimise by evolution, because it may present many local optima or suffer from the *bootstrap problem*, which is defined as the absence of selective pressures among randomly initialised individuals at the beginning of the evolutionary optimisation [8]. Hence, it may be preferable to adopt a multi-objective formulation of the fitness [10] and approximate the corresponding Pareto front (finding the actual Pareto front is typically infeasible in evolutionary robotics).

In this paper, we discuss the advantages of multi-objective optimisation (MOO) methods in evolutionary robotics, as opposed to single-objective optimisation (SOO) methods. We name *SOO approaches* those algorithms that only try to find a single solution by either optimising a single fitness function or scalarizing multiple objectives a priori, i.e., before running the algorithm. Even though the study of the (a priori) scalarization of MOO problems is a subject studied in the MOO literature [11], the algorithms that actually tackle the scalarized problems, in the context of evolutionary robotics, do not differ substantially from those used to tackle genuinely SOO problems. By contrast, we name *MOO approaches* those algorithms that aim to approximate as best as possible the Pareto-optimal set. From this approximation set, the designer can choose, a posteriori, one behaviour as the final solution. This is a simplified view of MOO for the sake of comparison with the traditional SOO approaches used in evolutionary robotics. MOO in general includes a priori, a posteriori and interactive approaches [10–14].

In the last two decades, evolutionary MOO approaches have shown their ability to explore multiple trade-offs in the objective space and to avoid premature convergence to poor solutions [12, 15]. It is therefore surprising that the application of MOO to evolutionary robotics has been so far relatively limited. A possible reason is that existing studies on MOO applied to evolutionary robotics do not systematically compare to SOO approaches and do not clearly spell out how MOO overcomes the problems faced by designers as outlined above. In Section 2, we fill this gap by casting well-known benefits of MOO approaches in terms of the problems they solve in evolutionary robotics. In Section 3, we provide a detailed survey of the literature of MOO applications to evolutionary robotics, focusing particularly on papers that, often indirectly, demonstrate the benefits of MOO approaches. In the rest of the paper, we substantiate this discussion by providing a comparative analysis between SOO and MOO approaches in three case studies taken from the literature.

The first case study (Section 4) concerns the evolution of a navigation behaviour in a looping maze, following one of the pioneer studies in evolutionary robotics [16]. In this case, we show that MOO allows the evolution of a varied set of behaviours by exploring multiple trade-offs among the available behavioural terms. The second case study (Section 5) concerns another classic task: coordinated motion (flocking) with robots having only local perception of their neighbourhood. In this case, we show how MOO avoids the convergence to local optima induced by a multiple-components fitness function. Section 6 is dedicated to the third case study concerning a strictly collaborative task designed after a well-known experiment in collective robotics [17]. The problem requires multiple robots to simultaneously process objects scattered in the environment, and the performance of the system is measured as the number of objects processed. Collaboration is strictly required to perform this task. Consequently, a bootstrap problem may arise: a null fitness is assigned to behaviours that do not result in collaboration, which leads to the absence of selective pressures for randomly generated solutions. We show how the introduction of ancillary objectives bypasses the bootstrap problem and leads to the systematic evolution of satisfactory solutions. Finally, the paper is concluded with a discussion of the results and of the future research directions.

2 MOO and evolutionary robotics

Multi-objective optimisation can be exploited in evolutionary robotics in many different ways, according to the specific characteristics of the task and of the corresponding robotic behaviour to be synthesised (i.e., the design problem). We refer mainly to three modes of problem solving with multiple objectives [10]: a) the design problem is a genuine multi-objective problem, and should be treated as such; b) the design problem is tackled through a multi-objective approximation by proxies; or c) a multi-objectivisation is performed to transform an originally single-objective problem into a multi-objective one. We discuss these three modes in the following.

Genuinely multi-objective problems The desired behaviour for the robotic system may entail multiple requirements, which may also be conflicting (e.g., they may represent the costs and benefits of applying a certain behavioural strategy). Therefore, a good trade-off must be found. In practice, the problem to be solved is a genuinely multi-objective optimisation problem. The classical approach in evolutionary robotics consists in the definition of a tailored fitness function that scalarizes, a priori, the different objectives (behavioural terms) to obtain a single-valued function [8], which can be optimised by means of well-known SOO algorithms. However, this scalarization—often a weighted sum—is somewhat arbitrary. The advantage of MOO is that it requires no such choice, and lets the evolutionary process free to explore different trade-offs between the objectives, allowing to choose a specific trade-off *a posteriori* on the basis of the analysis of the obtained solutions (see also Section 7 for a discussion about the selection of the best trade-off in relation to the evolutionary robotics domain). Most importantly, in evolutionary robotics different trade-offs may correspond to radically different behavioural strategies, which may not be attainable through single-objective evolution. Therefore, MOO leads to a large exploration of the possible solutions to a given robotics problem, and allows generating a wide set of behaviours, all optimising the trade-off between the given objectives. A simple MOO approach would be to optimise different scalarizations for a number of a priori defined weights, using SOO algorithms, and return the best solution found for each, in order to approximate the Pareto front. However, scalarizations based on weighted sums have well-known theoretical limitations [18]: (i) solutions that do not lay in the convex hull of the Pareto front cannot be found by any scalarization, and (ii) an even distribution of weights does not ensure an even distribution of solutions in the Pareto front. These limitations might not be crucial in practice for evolutionary robotics, since the Pareto front is often convex, the obtained solutions are just approximations to the optimal, and designers do not seek an even spread of solutions in the Pareto front. Nonetheless, we show later in the paper that, even for the typical scenarios of evolutionary robotics, such a simple MOO approach based on SOO algorithms is still inferior in practice to an MOO approach not relying on scalarizations.

Multi-objective approximation by proxies In evolutionary robotics, it is often the case that a suitable fitness function to optimise is not available a priori, i.e., the task does not come with a detailed performance metric. Indeed, the real objective is the behaviour of the robotic system, and the fitness function represents just a means to obtain this behaviour. Evolutionary robotics represents a prototypical case in which "*it is the solution that has primacy, and the objectives are only a means of orienting the search in order to discover this solution*" [10]. In such conditions, the design problem can be faced by introducing multiple objectives that are proxies for the true "unavailable" objective. These proxies can be complementary, each covering some particular aspect of the behaviour to be obtained. "*Thus, it should be expected that the desired solution(s) will score relatively highly under all of the 'proxy' objectives, and an MOO approach may therefore be suitable*" [10].

Multi-objectivisation Even when a single-objective function is available for measuring the performance in the given robotics problem, it may not be suitable for evolutionary optimisation because it may not

provide evolvability to the system. Generally speaking, the evolvability issue is related to the search landscape of the optimisation problem, which can be either very rugged, presenting many local optima, or on the contrary largely flat, offering no gradient for the evolutionary search. The latter case gives rise to the bootstrap problem, which is the absence of selective pressures among randomly initialised individuals at the beginning of the evolutionary optimisation [8]. Multi-objectivisation is a technique that allows transforming a difficult SOO problem into a more tractable MOO problem [10]. In the first case, the single-objective function is decomposed in multiple functions in order to disentangle the concurrent aspects giving rise to local optima. In the second case, ancillary objectives can be introduced to kick-start or guide the evolutionary search. These ancillary objectives may just be relevant for guidance, and the final solution(s) may largely ignore them, focusing on the primary objective alone.

3 Review of the State of the Art

Applications of MOO in evolutionary robotics have been proposed in the last decade, often providing limited or no comparison with SOO approaches. Only very recently, studies have addressed the topic with a rigorous methodological approach [19, 20]. One of the first applications of MOO in evolutionary robotics concerned the incremental evolution of controllers for an unmanned aerial vehicle (UAV) through multi-objective genetic programming. In this case, the objectives corresponded to the ability of the UAV to locate, reach and track a radar source [21]. More recently, MOO was applied to the evolution of virtual creatures displaying complex abilities such as legged locomotion and object grasping [22]. These and other studies demonstrate that MOO can be successfully applied in an evolutionary robotics context, but they do not provide insights on the advantages and limits of the methodology.

More related to the present paper are those studies that demonstrate the properties of the Pareto-dominance and their relationship to the evolvability of desired behaviours. We can identify two main lines of research. In the first, one objective is the usual fitness function for the specific evolutionary robotics problem, while the second objective is used to select some relevant feature of the controller, e.g., minimise the number of hidden units in a neural network [23–25], or promote the efficient usage of state variables in a controller evolved trough genetic programming [26]. These studies demonstrate how MOO can be beneficial in identifying a good trade-off between the evolvability of the desired behaviour and the complexity of the controller. The latter should be high enough—to support the evolution of the desired behaviour—but not higher—to reduce the dimensionality of the search space and ensure convergence. Teo and Abbass [23] present an extensive comparison between SOO and MOO approaches in which MOO approaches optimise both the behaviour and the controller structure, whereas SOO approaches fix the controller structure a priori and only evolve the behaviour. This comparison is, however, limited to one specific aspect of the advantages of MOO and does not address the problem of defining the correct fitness function to obtain a desired behaviour. Indeed, in the above studies the selective pressure to evolve the desired behaviour is provided by a single objective: if this objective is ill-defined or presents a low evolvability *per se*, no controller structure can help.

In the second line of research, MOO is exploited solely to define the selective pressures to obtain a desired behaviour for the robotic system. A first comparison between single- and multi-objective evolution was performed by evolving a robot controller for two conflicting tasks: protecting another robot by remaining close to it and collecting objects scattered in the environment [25]. A genuinely MOO approach was compared against multiple scalarized problems obtained as the weighted average of the performance in the two sub-tasks, and solved using a SOO algorithm. The results did not reveal significant differences in the performance achieved by the two approaches, but noted the higher computational complexity of running the SOO algorithm multiple times to produce different trade-offs between the two tasks [25]. Another relevant study [27] tackles a single-robot navigation problem inspired by the same pioneer study we refer to in this paper [16]. In this case, the MOO problem was not obtained through multi-objectivisation, but rather an additional component was introduced to reward the passage through pre-defined waypoints,

in order to favour a looping behaviour. Mouret and Doncieux [28] use multi-objective evolution to tackle the bootstrap problem in evolving controllers for complex tasks. In particular, they focus on incremental tasks, that is, problems that require the completion of multiple sub-tasks to achieve the goal task. In similar conditions, it is possible to define a MOO problem in which the different objectives correspond to the achievement of the individual sub-tasks. This approach is closely related to multi-objectivisation to guide evolution in conditions of lacking fitness gradient [10], as discussed above. The only limiting factor of this otherwise well grounded work resides in the experimental scenario, since the existence of clearly defined sub-tasks that can be characterised as conflicting objectives explicitly favours MOO over SOO approaches. In this paper, we provide further evidence on the relevance of MOO to tackle the bootstrap problem in a task that intrinsically leads to low evolvability, but that does not prevent SOO to find a solution. In this way, we extend the demonstration of the benefit of MOO as a general approach for evolutionary design methods.

Other approaches are worth mentioning, which address the bootstrap problem without focusing solely on the selective pressures. In [29], rather than adding sub-objectives to guide the evolutionary search in the early phases, MOO is used to create a seed population of individuals with some basic capabilities, which will then be used to evolve the controllers for the goal task following a SOO approach. Recently, MOO was exploited in conjunction with techniques to maximise behavioural diversity among the evolved solutions, to promote a better exploration of the search space and tackle the bootstrap problem [19]. In this case, one objective was dedicated to the goal task fitness, while a second objective corresponded to a diversity measure. The study proposes an extensive comparison of SOO and MOO approaches for encouraging behavioural diversity, varying also the controller structures and the diversity measures, showing that MOO performs best. Finally, a recent study investigated MOO to solve the problem of transferring evolved behaviours from simulation to real robots [20]. The rationale behind this study is that the optimal behaviour in hardware may not be the one obtained in simulation due to limitations in the simulated model of the robots. To find the optimum, a MOO approach is proposed in which one objective represents the ability of the given solution to cross the reality gap. This study demonstrates that such a MOO approach can achieve better results than conventional approaches based on noisy simulations or on local search [20]. In both [19] and [20], an additional objective is introduced to improve the evolutionary search. This objective is not task-specific, but promotes a wider exploration of the search space in conjunction with a task-specific fitness function. Hence, these studies can be seen as methodological improvements over a standard evolutionary robotics approach. Differently, in this paper we propose the usage of MOO with task-specific objectives, in order to improve the evolutionary search and to obtain at the same time a larger behavioural diversity and a better exploration of the search space, also avoiding bootstrap problems. We argue that these properties are delivered by MOO *per se*, and we provide an experimental demonstration by conducting a systematic comparison between single- and multi-objective evolution in three case studies, as discussed in the following sections.

4 Case Study: Navigation in a Maze

The first case study corresponds to the evolutionary design of a navigation behaviour for a single robot to explore a looping maze. We choose this case study as it represents a necessary benchmark to test the suitability of a design methodology, it is simple enough to be fully analysed and understood, and it has been widely studied in the past. We start from a classic study in evolutionary robotics [16], which proposed a simple setup to perform the evolutionary optimisation of a neural network controller directly on the real robotic hardware. The physical setup puts several constraints on the information available to compute the performance, as absolute positions are not available to the robot. The fitness function has been therefore defined exploiting solely the information accessible to the robot, that is, the infra-red (IR) proximity sensors to account for the distance from obstacles, and the wheels' encoders to account for the robot motion speed. In our work, despite experiments are performed in simulation, we adopt the same

rationale as if the fitness was computed on the physical robot (i.e., the *marXbot* [30], see Figure S1 in the supplementary material) to keep the setup coherent with the original study [16].

The desired behaviour for the robot is exploration and obstacle avoidance in a closed circuit. In this work, the circuit has been designed to replicate the main features of the experimental arena used in the original study [16], which is characterised by left and right turns, as well as wide and narrow corners (see Figure S2 in the supplementary material). A good exploration behaviour would efficiently negotiate corners and narrow passages, and would result in a smooth navigation to cover the whole circuit. With this goal in mind, we decided to evolve a simple reactive neural network controller. Therefore, no path planning or trajectory tracking has been used, and the navigation strategy directly derives from the evolved neural controller. The neural controller and the experimental setup is detailed in the supplementary Text S1. We use the original formulation for the fitness function, and contrast it with a two-objective formulation obtained through multi-objectivisation (Section 4.1). Additionally, we compare the MOO approach and a SOO approach based on a scalarization through weighted sum of the two objectives. The methodology for comparing single and multi-objective approaches is presented in Section 4.2, and the obtained results are discussed in Section 4.3.

4.1 Fitness function and multi-objectivisation

The original fitness function is designed to reward fast motion and obstacle avoidance on the basis of the information available to the robot. Therefore, at each control step t, the current angular speed of the wheels and the readings from the IR proximity sensors are collected, and a single-objective performance measure is computed as follows:

$$\mathcal{N}_{\mathcal{T}} = \frac{1}{T} \sum_{t} \Omega(t) \cdot \Phi(t) \tag{1}$$

where T is the total number of control steps. $\Omega(t)$ is the reward for moving straight and fast at time t, and is computed on the basis of the current angular speed of the robot wheels:

$$\Omega(t) = \frac{|\omega_l(t)| + |\omega_r(t)|}{2\omega_m} \cdot \left(1 - \sqrt{\frac{|\omega_l(t) - \omega_r(t)|}{2\omega_m}}\right), \tag{2}$$

where ω_l and ω_r are the left- and right-wheel angular speeds, and ω_m is the maximum possible speed of the differential-drive motion controller. Here, the first factor rewards fast motion because it is maximised by high absolute values of the angular speed. The second factor rewards straight motion as is maximised by small differences between the left and right angular speeds. $\Phi(t)$ is the reward for being away from any obstacle at time t:

$$\Phi(t) = 1 - \max_{i} \frac{\phi_i(t)}{\phi_m}, \tag{3}$$

where $\phi_i(t)$ is the activation of the i^{th} IR proximity sensor, and ϕ_m is the maximum possible value. The closer the obstacle, the higher the sensor activation, the smaller the reward. $\mathcal{N}_{\mathcal{T}}$ averages over time the product of these two components, therefore requiring that both components are non-null for a large amount of time. The interesting aspect of this function is that both its components are low at the same time while the robot is negotiating an obstacle. Therefore, $\mathcal{N}_{\mathcal{T}}$ is maximised by controllers that ensure a fast reaction to the perception of an obstacle that leads the robot away from it.

Starting from the above formulation, we define two objectives by decomposing the function $\mathcal{N}_{\mathcal{T}}$ as follows:

$$\mathcal{N}_{\Omega} = \frac{1}{T} \sum_{t} \Omega(t), \tag{4}$$

$$\mathcal{N}_{\Phi} = \frac{1}{T} \sum_{t} \Phi(t). \tag{5}$$

The objective $\mathcal{N}_\Omega$ encodes information about the wheels speed over time, and corresponds to the evaluation of the robot's ability to move straight and fast. On the other hand, $\mathcal{N}_\Phi$ encodes information about the proximity to obstacles and walls, and is maximised by behaviours that keep the robot away from them. The two objectives introduce an interesting trade-off, that was not considered in the original formulation. Indeed, there may be behaviours that move less but stay maximally away from obstacles, or fast-moving robots that however keep the walls closer. As we will see below, the MOO approach will explore these possible trade-offs producing several qualitatively different behaviours.

Finally, we introduce a single-objective formulation that scalarizes the two objectives introduced above by means of a weighted sum, as follows:

$$\mathcal{N}_\gamma = \gamma\mathcal{N}_\Omega + (1-\gamma)\mathcal{N}_\Phi \tag{6}$$

with varying $\gamma \in [0, 1]$. The rationale is that different trade-offs between the two objectives can be explored by varying γ, but we can still use the SOO approach to optimize Eq. (6) for each value of γ. Although such an approach has well-known theoretical limitations [18], it could be in practice more efficient in finding alternative solutions to the navigation problem than the purely MOO approach.

4.2 SOO vs. MOO

To compare the SOO and MOO approaches, we run several repetitions of the same evolutionary algorithm, changing only the way in which potential solutions are evaluated and selected for reproduction. In the SOO case, solutions are selected according to the fitness function. In the MOO case, solutions are selected according to the "hypervolume measure", which represents the "*size of the dominated space*" [31]. This type of selection is used by state-of-the-art multi-objective evolutionary algorithms [32, 33], but here we just replace the selection step of a standard SOO algorithm. A detailed description of the evolutionary algorithm is available as supplementary material. For each experimental condition, we run twenty evolutionary runs for a fixed number of generations, and we consider the solutions of the last generation for further analysis. All solutions are re-evaluated to obtain an unbiased assessment of their performance using the available performance metrics, i.e., the (scalarized) fitness and the individual objective measures. The comparison between the SOO and MOO approaches is carried out on these measures by looking at the probability that a specific solution dominates another one. All the details of the comparative approach we employ are available in the supplementary material.

4.3 Results

4.3.1 Performance analysis

Following the methodology described above, we compare the SOO and MOO approaches by looking at the solutions in the objective space given by $\mathcal{N}_\Omega$ and $\mathcal{N}_\Phi$. The comparison between the MOO approach and the SOO approach using the traditional fitness formulation $\mathcal{N}_\mathcal{T}$ is given in Figure 1a. Here, we plot the differential ability of the two approaches in attaining portions of the objective space. It is possible to observe that the solutions found by the SOO approach prevail only in the region in which both speed and distance from obstacles are maximised (see the dark areas on the right part of Figure 1a). This region is also attained by the MOO approach, but less frequently. Indeed, the MOO approach explores widely the objective space, and proves best in all the other regions in which either speed or distance from obstacles are maximised. These regions are never attained by the single-objective evolution. A very similar discussion can be done for the comparison with the weighted sum $\mathcal{N}_\gamma$, with $\gamma = 0.5$. Also in this case, the single-objective approach finds and optimises only solutions that maximise at the same time the two objectives, being deficient in the other parts of the objective space (Figure 1b). By looking at these results, we can conclude that, in this case, MOO does not provide a competitive advantage over SOO if we are only interested in the single-objective performance.

The situation is different if we vary the weighted sum of the two objectives. We tested different weights by systematically varying γ, as shown in Figure 2. In this case, the MOO approach has a large advantage on most parts of the objective space, generally dominating the weighted sum solutions. For $\gamma = 0.2$, mainly obstacle avoidance is rewarded, and single-objective evolution remains trapped in a local optimum in which only $\mathcal{N}_\Phi$ is maximised (Figure 2a). Conversely, for $\gamma = 0.6$ and $\gamma = 0.8$, SOO mainly finds the other local optimum in which only $\mathcal{N}_\Omega$ is maximised (Figure 2c-d). In all these cases, the SOO approach does not produce good solutions for the navigation problem. The weight $\gamma = 0.4$ provides better results, with the SOO approach showing a slight advantage in attaining areas of the objective space close to the central part of the Pareto front (Figure 2b), but this looks less efficient than the case for $\gamma = 0.5$ shown in Figure 1b. These results demonstrate that the choice of the right compromise between different objectives is very difficult, and even relatively small variations may lead to radically different results. This aspect is discussed further in the case study presented in Section 5.

4.3.2 Behaviour diversity

As expected, individual single-objective runs do not explore different trade-offs. With $\mathcal{N}_\mathcal{T}$, the selective pressure is focused on the region of the objective space where $\mathcal{N}_\Omega \approx \mathcal{N}_\Phi$. In these conditions, the evolved solutions produce behaviours that present at the same time high values of $\Omega(t)$ and $\Phi(t)$. This happens when a robot moves as fast as possible and as far as possible from obstacles. When a corner must be negotiated, the evolved strategy makes the robot quickly rotate on the spot. In particular, we observed that the avoidance action is performed by turning always in the same direction, notwithstanding the maze requiring a left or a right turn. This leads to a fast and efficient collision-avoidance action, because the robot does not need to determine whether the turn is on the left or on the right. However, despite optimal with respect to the single-objective function, this strategy trades the avoidance speed with the exploration abilities in the maze. In fact, the robot remains trapped in some parts of the circuit because it is unable to correctly negotiate both left and right turns. Evolving the neural network controllers with $\mathcal{N}_\mathcal{T}$ produces only behaviours of this kind, which we will refer to as *quick-avoidance* behaviours. An example of this behavioural strategy is provided by Video S1 in the supplementary material.

Multiple scalarizations with $\mathcal{N}_\gamma$ provide an approximated Pareto set, but are not able to cover the whole front [18]. On the contrary, the MOO approach explores the objective space widely and provides a better approximation of the Pareto set, therefore producing behaviours of different kind that optimise different trade-offs of $\mathcal{N}_\Omega$ and $\mathcal{N}_\Phi$ (see also Figure 3a). By observing the behaviour corresponding to the different solutions found, we can identify mainly four categories:

(i) the solutions that maximise $\mathcal{N}_\Omega$ only, which correspond to controllers that make the robot move fast, but do not present any obstacle avoidance ability;

(ii) the solutions that maximise $\mathcal{N}_\Phi$ only, which correspond to controllers that keep the robot away from obstacles, but do not present any navigation ability;

(iii) the solutions that mainly maximise $\mathcal{N}_\Omega$, possibly at the cost of a lower $\mathcal{N}_\Phi$, which correspond to controllers producing the quick-avoidance behaviour described above;

(iv) the solutions that mainly maximise $\mathcal{N}_\Phi$ at the cost of a lower $\mathcal{N}_\Omega$, which correspond to robots that move following the closest wall, either on the left or on the right. This *wall-following* behaviour leads to a very good exploration of the maze, which can be performed at the cost of a reduced speed in order to keep a constant distance from the wall (see also Video S1 in the supplementary material).

For the engineering of good navigation in a maze, the MOO approach has the advantage of providing a varied set of possible solutions in each evolutionary run. Some of the evolved behaviours are not useful for navigation purposes, such as those belonging to the categories (i) and (ii) described above. However, the *wall-following* behaviour clearly outperforms the *quick-avoidance* behaviour in terms of exploration of

the maze. This gives an advantage over SOO approaches that can be quantified. To this purpose, we compute an exploration performance value that measures the exploration abilities of the best evolved solutions by looking at the percentage of the maze that is visited by the robot in a sufficient amount of time to cover the full length of the circuit. Figure 3a shows the exploration performance for all the solutions obtained with MOO and SOO. Note that we consider all the solution obtained with $\mathcal{N}_\gamma$ as a unique experimental condition, in the attempt to approximate the Pareto set by systematically varying γ. The best controllers have been chosen in the MOO case from the approximated Pareto set of the different evolutionary runs, while in the SOO case they where selected as the solutions with the highest average performance.[1] Figure 3a shows that all the best solutions of the SOO approach with $\mathcal{N}_\mathcal{T}$ present poor exploration, as they all belong to the quick-avoidance category. Similarly for the case of SOO with $\mathcal{N}_\gamma$, even though a few solutions with good exploration can be observed. Instead, in the MOO case, a large fraction of solutions provide a good exploration, suggesting that the wall-avoidance behaviour was often discovered.

Given that the MOO approach produces much more solutions than the SOO approach, we can compare their efficiency by looking at the probability of obtaining a solution presenting a good exploration behaviour. We estimate this probability by computing the fraction of evolved solutions that present an exploration performance higher than a threshold τ, at varying τ. We plot this probability as an empirical tail distribution (i.e., the complementary of the cumulative distribution function) in Figure 3b. We note that $\mathcal{N}_\mathcal{T}$ produces only solutions that visit less than 20% of the maze, which corresponds to poor exploration abilities. For larger τ, the probability of obtaining good navigation behaviours is higher for the MOO approach than for $\mathcal{N}_\gamma$, which only occasionally produces the wall-following behaviour. Thus, we conclude that MOO delivers better results for what concerns the diversity of behaviours evolved, even when compared against a scalarization approach exploring various values of γ. In this particular case, better exploration also corresponds to the discovery of desired solutions that could not be easily obtained with the SOO approach.

5 Case Study: Flocking

Flocking is a standard behaviour extensively studied in swarm intelligence and swarm robotics. It requires that independent, autonomous agents in a group move coordinately on the basis of local information only. By observing the dynamics of bird flocks and fish schools, several theoretical models have been developed to describe the individual rules underlying coordinated motion [34, 35]. Generally speaking, three simple rules are sufficient: (i) collision avoidance, (ii) flock centering and (iii) velocity matching. The first two rules provide the means to achieve and maintain *cohesion* in the group, because agents are attracted to each other while maintaining a safety distance. Velocity matching instead makes an agent orient itself in the average direction of the neighbours, eventually leading to the alignment of all individuals, which is necessary for efficient group *motion*. The execution of these rules is possible on the basis of local information only—distance, bearing and heading of close neighbours—and is sufficient for the establishment and maintenance of coordinated motion.

The evolution of a coordinated motion behaviour represents a prototypical case in which the desired behaviour of the group is well understood, but a fitness function to evolve it is not available (see for instance [36, 37] for two alternative approaches). The description of the flocking behaviour naturally yields to a multi-objective formulation: the group must move as far as possible (maximise motion) while keeping coherence (maximise cohesion). These can be considered as "proxies" to guide the evolutionary search toward desired solutions. We therefore decided to contrast a MOO approach using these two metrics with the SOO approach derived from scalarization through weighted average of the two proxies. The details

[1] This corresponds to the choice one would make *a priori* when selecting or discarding solutions from the last population, as described in the supplementary material.

are given in Section 5.1, while the obtained results are discussed in Section 5.2. A complete description of the experimental setup for this case study is available as supplementary material in Text S1.

5.1 Motion and cohesion objectives and fitness function

In order to reward motion and cohesion in a group, the simplest way is to rely on the absolute positions of the robots, which are available in our simulation environment. At the beginning of a trial, robots are placed within a circle of 1 m radius, choosing their position and orientation at random. To reward group motion, it is sufficient to look at the displacement of the centre of mass of the group during a fixed-duration trial:

$$\mathcal{F}_m = \frac{\|\hat{\mathbf{X}}(T) - \hat{\mathbf{X}}(0)\|}{D_m(T)}, \tag{7}$$

where $\hat{\mathbf{X}}(t)$ is the position of the centre of mass at time t, $D_m(t)$ is the maximum distance a single robot can travel in t seconds, and T is the length of a trial. Cohesion instead is maximised when the average distance of the robots from the centre of mass of the group is minimised:

$$\mathcal{F}_c = \max\left(0, 1 - \frac{1}{N}\sum_i \frac{\|\mathbf{X}_i(T) - \hat{\mathbf{X}}(T)\|}{d_m}\right), \tag{8}$$

where $\mathbf{X}_i(t)$ is the position of robot i at time t, N is the total number of robots, and d_m is the maximum tolerated distance. Cohesion is computed at the end of the trial, assuming that the group must remain aggregated for the whole duration of the trial (and possibly beyond it).

Also in this case, we can obtain a single-objective formulation from the two above objectives through a scalarization:

$$\mathcal{F}_\gamma = \gamma\mathcal{F}_m + (1-\gamma)\mathcal{F}_c, \tag{9}$$

with $\gamma \in [0, 1]$. To properly select the weight γ, we would need to know *a priori* the relative importance of the two components of the fitness. Since this relative importance is generally unknown, the standard approach is therefore to systematically vary γ. Consistently with the previous case study, we provide here the results for $\gamma \in \{0.2, 0.4, 0.5, 0.6, 0.8\}$.

5.2 Results

We again followed the methodology described in Section 4.2 and detailed in the supplementary Text S1. The results of the comparison between the different approaches is presented in Figure 4. Generally speaking, the MOO approach outperforms the SOO approach with $\mathcal{F}_\gamma$, presenting a higher probability of attaining all regions of the objective space (Figure 4a). If we look at the comparison with specific values of γ as shown in Figure 4b-f, only for $\gamma \geq 0.6$ the differences fall below 60% in favour of the MOO approach for most of the objective space. By contrast, the choice of $\gamma = 0.5$ (the natural choice one would make without any a priori knowledge) results in poor performance, with most of the objective space attained by the MOO approach with at least 60% more probability than the SOO approach. The reason is that low values of γ bias the search towards solutions presenting high cohesion and no motion. These solutions behave as local optima from which it is difficult to escape, even when more importance is given to the motion objective $\mathcal{F}_m$ with higher values of γ. Indeed, solutions providing only cohesion and no motion are obtained also with $\gamma = 0.8$.

The performance argument alone gives already a point in favour of the MOO approach, as it proves capable of attaining more frequently all areas of the objective space. However, not all possible trade-offs correspond to good coordinated motion behaviours. From an engineering perspective, those solutions providing low motion or low cohesion should not be retained: the former correspond to packed groups that do not move, the latter correspond to groups that split by moving in multiple directions. It is

therefore useful to look at the probability of producing acceptable solutions by using the SOO and MOO approaches. To this purpose, we examine the attainment surfaces of both approaches as they give an absolute indication about the algorithm ability to produce solutions with a given trade-off between the objectives (Figure 5). Here, we compare the MOO approach with the SOO approach aggregating all values of γ. While the best attainment surfaces are comparable, the probability of producing solutions that maximise both objectives at the same time is lower for the single-objective approach. With the MOO approach, even the worst attainment surface contains solutions that feature reasonably good cohesion and motion. This indicates that the MOO approach can produce good solutions in every run, thanks to a wide exploration of the objective space, which is prevented by a single-objective approach. Additionally, also in this case it is possible to observe different kinds of behaviours evolved with the MOO approach, similarly to what was shown in Section 4 (refer also to [38]). Some of the evolved behaviours are available as supplementary material in Video S2.

6 Case Study: a Strictly Collaborative Task

This last case study is dedicated to a strictly collaborative task, that is, a task in which collaboration among multiple robots is strictly necessary. In other words, there are no other possibilities than success—when a fruitful collaboration is established—or failure. The notion of strictly collaborative tasks has been introduced by Alcherio Martinoli and collaborators in the context of the "stick-pulling experiment" [17]. In this experiment, robots had to pull a long stick out of the ground, and the collaboration of two robots was necessary as the stick was too long to be extracted by a single robot. They designed a behaviour-based controller and studied the performance of the system in terms of the number of pulled-out sticks after a fixed amount of time. Their proposed solution is based on a timeout mechanism: robots explore the arena and wait for teammates when they encounter a stick to be extracted; if no teammate arrives within a given time interval, a timeout is triggered so that the robot abandons the stick and resumes exploration. By appropriately tuning the waiting time, the performance of the system can be optimised. Inspired by this experiment, we present a strictly collaborative task in which we use beacon lights to be switched off instead of sticks to be pulled out from the ground. Beacons are made by red LEDs positioned over cylindrical obstacles and are perceived by the robots through their omnidirectional camera (see Figure S3 in the supplementary material). A beacon is switched off automatically when at least n robots are close enough at the same time, therefore enforcing the need for collaboration. The details of the experimental setup are available as supplementary material in Text S1.

The evolution of collective behaviours for strictly collaborative tasks is not trivial. The complexity of such tasks depends on the ratio between the number M of objects to be processed (sticks or beacons) and the number N of robots in the group, and non-trivial strategies are required when $N/M \ll 1$. In this study, we introduce an additional parameter to control the task complexity, that is, the number of robots n required in a single collaboration, which we refer to as the collaboration level. In this way, we can vary the task complexity and observe its bearing on the evolvability of collaborative strategies.

A delicate aspect of strictly collaborative tasks is that the intrinsic measure of performance—the number of pulled-out sticks or switched-off beacons—does not offer much gradient for evolutionary optimisation as long as some form of collaboration is not in place. Indeed, either robots know how to collaborate, and therefore their performance reflects their ability, or they do not, and in this case they would score a null fitness. This means that the evolution of controllers for strictly collaborative tasks may be affected by the bootstrap problem, because it is very likely that randomly generated solutions do not correspond to any collaboration ability. In this section, we demonstrate that it is possible to bypass the bootstrap problem and provide a performance gradient for evolutionary optimisation by resorting to a MOO approach obtained by introducing some ancillary objective that can guide evolution in the early phases. In Section 6.1, we introduce the main and ancillary objectives for our strictly collaborative task, while the obtained results are presented in Section 6.2.

6.1 Main and ancillary objectives in strictly collaborative tasks

As mentioned above, the task we have defined requires that $N = 6$ robots collaborate to switch off $M = 18$ beacons, randomly scattered in a circular arena (radius: 6 m). The beacons are automatically switched off when at least n robots are within a radius $r_v = 25$ cm from the beacon. The intrinsic performance metric for this task corresponds to the fraction of switched-off beacons at the end of a trial:

$$\mathcal{S}_{c,n} = \frac{|B_0(T)|}{M}, \tag{10}$$

$$B_0(t) = \{b,\ s_b(t) = 0, b = 1, \ldots, M\}, \tag{11}$$

where $B_0(t)$ represents the set of beacons b whose status $s_b(t)$ is off at time t, and T is the length of a trial. The state of a beacon is always on ($s_b = 1$) unless at least n robots are close enough at the same time:

$$s_b(t) = 0 \Leftrightarrow \begin{cases} \exists A' \subset A, |A'| = n \\ \exists t' \leq t\ \forall i \in A'\ d_{i,b}(t') \leq r_v \end{cases}, \tag{12}$$

where A' is a subset of cardinality n of the set of robots A, and $d_{i,b}(t) = \|\mathbf{X}_i(t) - \mathbf{X}_b(t)\|$ is the euclidean distance between robot i and beacon b. Similarly to [17], $\mathcal{S}_{c,n}$ represents the collaboration rate within a trial of length T. Behaviours in [17] were designed by hand. Here, we use $\mathcal{S}_{c,n}$ as the fitness function to evolve behaviours in the SOO approach, and we test it with two collaboration levels (i.e., $n = 2$ and $n = 3$) to explore different task complexities.

Single-objective evolution with $\mathcal{S}_{c,n}$ is affected by the bootstrap problem, as will be shown in Section 6.2. To overcome this problem, we resort to multi-objectivisation by adding an ancillary objective that is somehow related to the task. To this purpose, we note that collaborations can be obtained if robots are capable of visiting multiple beacons during a trial. Indeed, if a robot stops at the first encountered beacon and never leaves, it is very likely that the group will end in a deadlock condition in which all robots are waiting at different beacons and no collaboration takes place. Therefore, a robot must be capable of visiting multiple beacons during a trial. We compute this exploration performance as follows:

$$\mathcal{S}_v = \frac{1}{N}\sum_i \frac{|V_i(T)|}{M}, \tag{13}$$

$$V_i(t) = \{b,\ \min_{t' \leq t} d_{i,b}(t') \leq r_v\}, \tag{14}$$

where $V_i(t)$ is the set of beacons visited by robot i at time t. A beacon b is considered visited by robot i if the minimum distance achieved from the beacon is less than r_v. By visiting multiple beacons, the robots maximise their exploration abilities. However, a robot needs to remain close to a beacon for some time in order to establish a collaboration with its teammates. Therefore, a tradeoff is present between the maximisation of the exploration ability and the waiting time for collaboration. The MOO approach is most suited to explore this tradeoff and eventually provide solutions capable of maximising $\mathcal{S}_{c,n}$, which remains our main goal.

6.2 Results

As mentioned above, we study two different task complexities by setting the collaboration level to $n = 2$ and $n = 3$. We run single-objective evolution using $\mathcal{S}_{c,n}$ as fitness function. For each evolutionary run, we first observe the trend of the fitness of the best individual in the population during the evolutionary optimisation, as shown in Figure 6. We notice that the bootstrap problem only mildly affects evolution with $n = 2$. Several evolutionary runs do show a flat fitness surface for many generations, which indicates that the whole population scored a null fitness. However, by random drift, some solutions with non-null fitness appeared and optimisation rapidly started. This indicates that the bootstrap problem exists, but is

not too severe. On the contrary, when $n = 3$ the situation is worse, as can be seen in the bottom part of Figure 6. Most of the evolutionary runs suffered of the bootstrap problem, and present a very flat fitness surface. Only a minority of runs were able to discover a suitable strategy that was optimised through the generations. This also indicates that there exist possible solutions to the proposed problem, but they are not easily obtained by a SOO approach.

Once confirmed the existence of the bootstrap problem, in both a mild and severe version, it is worth comparing the performance of the MOO approach with respect to the SOO one. We are interested in studying whether multi-objective evolution is capable of producing solutions as efficient as those evolved in the single-objective case (especially for $n = 2$) and whether the bootstrap problem can be bypassed by systematically evolving good solutions in every evolutionary run (especially for $n = 3$). The results of the comparison for $n = 2$ are shown in Figure 7a-b. By looking at the difference between the EAFs of the MOO and SOO approaches (Figure 7a), we can observe that the second objective does not prevent to evolve good quality solutions. Indeed, both approaches similarly attain the regions of the objective space where $\mathcal{S}_{c,2}$ is maximised. Additionally, the MOO approach also finds solutions that provide good exploration abilities to the robots by maximising $\mathcal{S}_v$, which cannot be achieved by the SOO approach. This also confirms the existence of a tradeoff between $\mathcal{S}_{c,n}$ and $\mathcal{S}_v$ and therefore supports the usage of a MOO approach instead of a scalarization of the two objectives, which may result in poor solutions, as we have discussed in the previous case studies. The similarity in performance between the MOO and SOO approaches is also confirmed by looking at Figure 7b, which shows the performance of the best evolved controllers for each run computed over 500 different trials. We can therefore conclude that the mild bootstrap problem observed when $n = 2$ can be efficiently overcome by both approaches, and no substantial difference in performance can be observed between the two with respect to $\mathcal{S}_{c,2}$.

The situation is completely different when the bootstrap problem becomes severe, as is the case for $n = 3$. The comparison between MOO and SOO shown in Figure 7c-d demonstrates that the ancillary objective $\mathcal{S}_v$ is definitely helpful. Indeed, the EAF differences shown in Figure 7c demonstrate a strong advantage of the MOO approach in every area of the objective space. Similarly, the performance comparison among the best evolved individuals shown in Figure 7d indicates that suitable solutions are obtained in every evolutionary run with MOO, which means that the bootstrap problem is bypassed thanks to the introduction of the second objective. Indeed, $\mathcal{S}_v$ allows guiding the evolutionary optimisation when there is no fitness gradient, because exploration is a beneficial ability to promote collaborations. In this case, random drift alone is not sufficient to produce good solutions, at least not systematically in every run. We can conclude that MOO is a more reliable approach to produce satisfactory behaviours in every evolutionary run, thanks to the ability to bypass the bootstrap problem. Some examples of the evolved behaviours for both $n = 2$ and $n = 3$ are available as supplementary material in Video S3.

7 Discussions and Conclusions

The studies presented in this paper clearly demonstrate how beneficial can be the use of MOO in an evolutionary robotics context. From the experiments we performed, the overall conclusion is that MOO is the choice to be made whenever no a priori knowledge is available about the problem to be solved, because MOO allows a wider search of the space of possible solutions, provides higher evolvability and removes the constraints introduced by custom-tailored formulations of the fitness function. We do not claim that MOO should be used in any case. Indeed, we also show that single-objective evolution is more powerful when a good fitness function is available a priori, because the search effort is focused on the maximisation of the solution performance in a one-dimensional objective space. Instead, MOO diffuses the search effort to adequately approximate the whole Pareto front in a high-dimensional objective space. Consequently, MOO may require more generations to match the quality of the SOO approach with respect to an a priori fitness function. This explains the slight advantage of SOO over MOO that was detected in certain conditions of the experiments presented in this paper. We predict that this advantage would fade by

running the evolutionary algorithms for more generations. However, in evolutionary robotics is often the behaviour that has primacy over the performance score, and we have shown that the MOO approach can produce a varied set of strategies to solve a given problem. A similar variety of strategies could be achieved with the SOO approach by exploring multiple trade-offs through several independent scalarizations, even though this approach has several limitations especially when the Pareto set is concave [18]. However, our experiments have shown that, in the context of evolutionary robotics, the MOO approach produces a better approximation to the Pareto front at a lower computational cost.

Furthermore, the availability of a good fitness function is rather exceptional in evolutionary robotics. It is fair to admit that much of the past research in evolutionary robotics concealed the laborious process of finding the right fitness function and experimental design before obtaining good results. We claim that this burden can be very much alleviated by resorting to MOO, because it disentangles several problematic issues:

- With SOO, it is difficult to find the correct trade-off between multiple behavioural terms that represent the desired behaviour of the robotic system, which may have different and non-comparable scales (e.g., varying non-linearly). Trade-offs are not an issue in MOO, because all of them are naturally explored.

- With SOO, premature convergence to local optima must be carefully dealt with. In MOO, local optima are less problematic because there are several possible evolutionary paths to be exploited. Different evolutionary paths correspond to the emergence of behavioural capabilities that can be attained even at the expense of other capabilities, therefore allowing the evolution of diverse behavioural strategies.

- With SOO, the bootstrap problem can be a severe limitation in the usage of simple fitness functions related to task performance, above all when reward is sporadic or conditioned to the existence of preliminary behavioural capabilities, such as the ability to communicate and collaborate. In MOO, task performance can be optimised exploiting the guidance from ancillary objectives that contribute to the evolution of the behavioural pre-requisites necessary for task achievement.

From an engineering perspective, MOO represents a valid methodology to synthesise a wide set of potential solutions to a given robotics problem. A common criticism to MOO approaches is that eventually a single solution must be selected, therefore part of the design process is anyway arbitrarily driven by the designer. This is actually a feeble argument, because the supposedly arbitrary choice can be made on a limited set of Pareto-optimal solutions and can exploit the available information on the trade-offs they optimise. Additionally, in evolutionary robotics, the detailed analysis of several obtained solutions is common practice even within the SOO approach. The individuals produced during an evolutionary run are evaluated for the ability to generalise to a wide range of environmental conditions, which are loosely sampled during the optimisation process. More than being a generalisation test, this is actually the verification that the evolved solutions meet the design requirements. A similar verification process is necessary also with the MOO approach, but in this case one can limit the analysis to the Pareto set.

To conclude, we strongly believe that the usage of MOO in evolutionary robotics has more advantages than drawbacks, and should therefore be promoted whenever the need of multiple objectives arises. Future work should be focused on collecting more results on the advantages of MOO over SOO in several test-cases, therefore leading to the establishment of a multi-objective methodology for evolutionary robotics. We tested a relatively simple MOO algorithm that differs only from its SOO counterpart on the selection step, and a more detailed experimental analysis would be required to establish good guidelines for the choice of the MOO algorithm and its parameters. Additionally, the interaction of MOO with other design choices providing selective pressures must be considered [3]. For instance, important design choices correspond to the definition of the robot configuration or of the genotype-to-phenotype mapping [6, 38]. Some aspects of these choices can be delegated to evolutionary optimisation in the form of additional objectives, and studies going in this direction have been mentioned in Section 3. Obviously, adding an arbitrary number of objectives would hinder the quality of the obtained results simply by the fact that increasing the number

of conflicting objectives leads to a higher fraction of the objective space becoming Pareto-optimal, and the difficulty then lies on choosing a solution rather than on finding it [33, 39]. However, a more systematic study of the different design choices in a multi-objective perspective is required.

Finally, some effort must be dedicated to extend the methodology to realistic, application-driven scenarios. In this respect, the possibility to address also the simulation-to-reality gap within a multi-objective approach [20] represents a further demonstration of the potentials of MOO for pushing forward evolutionary robotics research. Another under-explored aspect is interactive MOO [13, 14]. Some research on interactive evolution in robotics has been performed to date [40–42]. However, this is a very human-intensive task that requires robotic designers to explicitly select the genotypes that would reproduce. By contrast, interactive MOO allows designers to progressively articulate their preferences with respect to the evolving behaviour by examining (some) Pareto-optimal solutions produced while running a multi-objective evolutionary algorithm, in order to focus the search in the most interesting regions of the objective space. Hence, it could be possible to include within the design process the intrinsic subjectivity of the evolution of robotics behaviours.

Acknowledgments

Vito Trianni acknowledges support from the H^{2}Swarm project, founded within the EUROCORES Programme "EuroBioSAS" of the European Science Foundation. The project is partially supported by funds from the Italian CNR and the Belgian F.R.S.-FNRS. We thank Manuele Brambilla and Guido De Croon for proofreading an earlier version of the paper.

References

1. Nolfi S, Floreano D (2000) Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-organizing Machines. MIT Press, Cambridge, MA.

2. Floreano D, Keller L (2010) Evolution of adaptive behaviour in robots by means of Darwinian selection. PLoS Biology 8: e1000292.

3. Doncieux S, Mouret JB (2014) Beyond black-box optimization: a review of selective pressures for evolutionary robotics. Evolutionary Intelligence 7: 71–93.

4. Pfeifer R, Lungarella M, Iida F (2007) Self-organization, embodiment, and biologically inspired robotics. Science 318: 1088–1093.

5. Lipson H (2005) Evolutionary design and open-ended design automation. In: Bar-Cohen Y, editor, Biomimetics: Biologically Inspired Technologies, Taylor & Francis/CRC Press,, chapter 4. pp. 129–155.

6. Trianni V, Nolfi S (2011) Engineering the evolution of self-organizing behaviors in swarm robotics: A case study. Artificial Life 17: 183–202.

7. Floreano D, Urzelai J (2000) Evolutionary robots with on-line self-organization and behavioral fitness. Neural Networks 13: 431–443.

8. Nelson A, Barlow G, Doitsidis L (2009) Fitness functions in evolutionary robotics: A survey and analysis. Robotics and Autonomous Systems 57: 345–370.

9. Wagner G, Altenberg L (1996) Perspective: Complex adaptations and the evolution of evolvability. Evolution : 967–976.

10. Handl J, Knowles JD (2008) Modes of problem solving with multiple objectives: Implications for interpreting the Pareto set and for decision making. In: Knowles JD, Corne D, Deb K, Chair DR, editors, Multiobjective Problem Solving from Nature, Springer. pp. 131–151. doi: 10.1007/978-3-540-72964-8_7.

11. Marler RT, Arora JS (2004) Survey of multi-objective optimization methods for engineering. Structural and Multidisciplinary Optimization 26: 369–395.

12. Deb K (2001) Multi-Objective Optimization Using Evolutionary Algorithms. Chichester, UK: Wiley.

13. Branke J, Deb K, Miettinen K, Słowiński R, editors (2008) Multi-objective Optimization: Interactive and Evolutionary Approaches, volume 5252 of *LNCS*. Springer.

14. Jaszkiewicz A, Branke J (2008) Interactive multiobjective evolutionary algorithms. In: Branke et al. [13], pp. 179–193. doi:10.1007/978-3-540-88908-3_7.

15. Coello Coello CA, Lamont GB, Van Veldhuizen DA (2007) Evolutionary Algorithms for Solving Multi-Objective Problems. Springer, New York, NY.

16. Floreano D, Mondada F (1994) Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot. From animals to animats 3: 421–430.

17. Ijspeert A, Martinoli A, Billard A, Gambardella L (2001) Collaboration through the exploitation of local interactions in autonomous collective robotics: The stick pulling experiment. Autonomous Robots 11.

18. Das I, Dennis JE (1997) A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. Structural Optimization 14: 63–69.

19. Mouret JB, Doncieux S (2012) Encouraging behavioral diversity in evolutionary robotics: An empirical study. Evolutionary computation 20: 91–133.

20. Koos S, Mouret JB, Doncieux S (2012) The Transferability Approach: Crossing the Reality Gap in Evolutionary Robotics. IEEE Transactions on Evolutionary Computation : 1.

21. Barlow G, Oh C, Grant E (2004) Incremental evolution of autonomous controllers for unmanned aerial vehicles using multi-objective genetic programming. In: IEEE Conference on Cybernetics and Intelligent Systems. IEEE Press, Pyscataway, NJ, volume 2, pp. 689–694.

22. Bongard J (2011) Innocent until proven guilty: Reducing robot shaping from polynomial to linear time. IEEE Transactions on Evolutionary Computation : 1–15.

23. Teo J, Abbass HA (2004) Automatic generation of controllers for embodied legged organisms: A Pareto evolutionary multi-objective approach. Evolutionary Computation 12: 355–394.

24. Teo J, Abbass H (2005) Multiobjectivity and complexity in embodied cognition. IEEE Transactions on Evolutionary Computation 9: 337–360.

25. Capi G (2007) Multiobjective evolution of neural controllers and task complexity. IEEE Transactions on Robotics 23: 1225–1234.

26. Agapitos A, Togelius J, Lucas SM (2007) Multiobjective techniques for the use of state in genetic programming applied to simulated car racing. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2007). IEEE Press, Pyscataway, NJ, pp. 1562–1569.

27. Moshaiov A, Ashram A (2009) Multi-objective evolution of robot neuro-controllers. In: Proceedings of the 2009 IEEE Congress on Evolutionary Computation (CEC'09). IEEE Press, Pyscataway, NJ, pp. 1093–1100.

28. Mouret JB, Doncieux S (2008) Incremental evolution of animats' behaviors as a multi-objective optimization. In: SAB '08: Proceedings of the 10th international conference on Simulation of Adaptive Behavior: From Animals to Animats. Springer Verlag, Berling, Germany.

29. Israel S, Moshaiov A (2012) Bootstrapping aggregate fitness selection with evolutionary multi-objective optimization. In: Parallel Problem Solving from Nature-PPSN XII. pp. 52–61.

30. Bonani M, Longchamp V, Magnenat S, Rétornaz P, Burnier D, et al. (2010) The MarXbot, a miniature mobile robot opening new perspectives for the collective-robotic research. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2010. Piscataway, NJ: IEEE Press, pp. 4187–4193.

31. Zitzler E, Thiele L (1998) Multiobjective optimization using evolutionary algorithms - A comparative case study. In: Eiben AE, Bäck T, Schoenauer M, Schwefel HP, editors, Parallel Problem Solving from Nature, PPSN V. Springer, volume 1498 of *LNCS*, pp. 292–301.

32. Beume N, Naujoks B, Emmerich M (2007) SMS-EMOA: Multiobjective selection based on dominated hypervolume. European Journal of Operational Research 181: 1653–1669.

33. Bader J, Zitzler E (2011) HypE: An algorithm for fast hypervolume-based many-objective optimization. Evolutionary Computation 19: 45–76.

34. Reynolds CW (1987) Flocks, herds and schools: A distributed behavioral model. In: Proceedings of the 14th annual conference on Computer graphics and interactive techniques. New York, NY, USA: ACM, SIGGRAPH '87, pp. 25–34.

35. Vicsek T, Zafeiris A (2012) Collective motion. Physics Reports .

36. Quinn M, Smith L, Mayley G, Husbands P (2003) Evolving controllers for a homogeneous system of physical robots: Structured cooperation with minimal sensors. Philosophical Transactions of the Royal Society of London, Series A: Mathematical, Physical and Engineering Sciences 361: 2321–2343.

37. Baldassarre G, Nolfi S, Parisi D (2013) Evolving mobile robots able to display collective behaviors. Artificial Life 9: 255–267.

38. Fehérvári I, Trianni V, Elmenreich W (2013) On the effects of the robot configuration on evolving coordinated motion behaviors. In: Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC 2013). IEEE Press, Pyscataway, NJ, pp. 1209–1216.

39. Ishibuchi H, Tsukamoto N, Nojima Y (2009) Evolutionary many-objective optimization: A short review. In: IEEE CEC. Piscataway, NJ: IEEE Press, pp. 2419-2426.

40. Gruau F, Quatramaran K (1997) Cellular encoding for interactive evolutionary robotics. In: Husbands P, Harvey I, editors, Proceedings of the Fourth European Conference on Artificial Life (ECAL 1997). The MIT Press, Cambridge, MA, pp. 368–377.

41. Takagi H (2001) Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation. Proceedings of the IEEE 89: 1275–1296.

42. Miglino O, Gigliotta O, Ponticorvo M, Lund H (2008) Human breeders for evolving robots. Artificial Life and Robotics 13: 1–4.

43. Grunert da Fonseca V, Fonseca CM, Hall AO (2001) Inferential performance assessment of stochastic optimisers and the attainment function. In: Zitzler E, Deb K, Thiele L, Coello Coello CA, Corne D, editors, Evolutionary Multi-criterion Optimization, EMO 2001, Springer, volume 1993 of *LNCS*. pp. 213–225.

44. Grunert da Fonseca V, Fonseca CM (2010) The attainment-function approach to stochastic multi-objective optimizer assessment and comparison. In: Bartz-Beielstein T, Chiarandini M, Paquete L, Preuss M, editors, Experimental Methods for the Analysis of Optimization Algorithms, Berlin, Germany: Springer. pp. 103–130.

45. López-Ibáñez M, Paquete L, Stützle T (2010) Exploratory analysis of stochastic local search algorithms in biobjective optimization. In: Bartz-Beielstein T, Chiarandini M, Paquete L, Preuss M, editors, Experimental Methods for the Analysis of Optimization Algorithms, Berlin, Germany: Springer. pp. 209–222. doi:10.1007/978-3-642-02538-9_9.

Figure Legends

Figure 1. Navigation experiment. Differences between the MOO and the SOO approach with (a) the traditional fitness function $\mathcal{N}_{\mathcal{T}}$ and (b) the fitness $\mathcal{N}_\gamma$, with $\gamma = 0.5$. The comparison is carried out by means of *empirical attainment functions* (EAFs) [43–45], which give the (estimated) probability that a single run of an optimiser *attains* (dominates or equals) a specific point in the objective space (for details, see Text S1 in the supplementary material). The difference between EAFs is reported in the figure, where grey-levels represent the magnitude of the difference: darker colours indicate larger differences. For example, a black point on the left indicates that the difference between the EAF of the left optimiser minus the EAF of the right optimiser is at least 0.8, that is, the left optimiser has attained that point in at least 80% more runs than the right one.

Figure 2. Navigation experiment. Differences between the EAFs of the multi-objective approach and the single objective approach with weighted fitness $\mathcal{N}_\gamma$ and varying weight γ between components.

Figure 3. Navigation experiment. (a) The exploration ability of the evolved controllers with the MOO and the SOO approaches. Each dot represents a solution in the objective space, and the darkness of the circle corresponds to the exploration performance, measured as the fraction of the maze that was effectively covered by the robot. (Left) Solutions obtained with MOO. (Centre) Solutions obtained with SOO using $\mathcal{N}_{\mathcal{T}}$. (Right) Solutions obtained with SOO using $\mathcal{N}_\gamma$ for every $\gamma \in \{0.2, 0.4, 0.5, 0.6, 0.8\}$. (b) Empirical tail distribution of exploration performance, that is, probability that the exploration performance is larger than a given performance threshold τ.

Figure 4. Flocking experiment. Differences between the EAFs of the MOO approach and the SOO approach with weighted fitness $\mathcal{F}_\gamma$. (a) Comparison with the combined results of all scalarizations. (b-f) Comparison for specific values of the weight γ between components.

Figure 5. Flocking experiment. Attainment surfaces of (a) the MOO approach and (b) the SOO approach for all values of γ. The k%-attainment surfaces shown in the figure allow visualising the EAF over the objective space [45]. As explained earlier, the EAF indicates the (estimated) probability of dominating a point in the objective space in a single run of an algorithm. A k%-attainment surface denotes the Pareto front of the points that have been attained by (dominated by or equal to) at least k% of the runs, that is, the points that have a value of at least $k/100$ of the EAF. Hence, the worst attainment surface indicates the Pareto front of the points that have been attained in 100% of the runs. Conversely, the best attainment surface indicates the Pareto front of the points that have been attained by at least one run. More details on the EAFs are available as supplementary material in Text S1.

Figure 6. Strictly Collaborative Task. Fitness of the best individual across generation for all evolutionary runs, for $n = 2$ (top) and $n = 3$ (bottom). Note that the maximum fitness achievable with $n = 3$ is lower than with $n = 2$, due to the increased complexity of the task.

Figure 7. Strictly Collaborative Task. Left: Differences between the EAFs of the MOO and the SOO approaches. Right: performance of the best individuals from the different evolutionary runs with respect to $\mathcal{S}_{c,n}$, ordered by decreasing median and mean performance.

Supporting Information Legends

Text S1: Experimental methods. The document describe in detail the experimental procedures for comparing MOO and SOO approaches. Additionally, the experimental setup is fully described with details about the robots, the simulation framework and the evolutionary experiments.

Figure S1: The marXbot robots. The robots are provided with a differetial drive motion system composed of both tracks and wheels. They feature several sensory systems, and have the ability to emit coloured signals through RGB LEDs, which can be perceived locally by the onboard omnidirectional camera.

Figure S2: The looping maze used for navigation experiments. The black rectangles and circles represent walls and rounded corners. The line in the middle of the corridor is the reference length of the looping maze, and represents also the line around which the robot is initialised.

Figure S3: A view of the arena for the strictly collaborative task. Robots and beacons are visible. Some beacons have been already switched off, while it is possible to see that two robots are waiting for collaboration from a third one that is approaching.

Video S1: Navigation experiment. This video refers to the first case study on navigation in a maze, and shows the differences between the wall-following and quick-avoidance strategies. Size: 2.8 MB, MPEG-4 (H.264 codec).

Video S2: Flocking experiment. This video refers to the flocking case study, and shows different types of flocking behaviour evolved. Size: 4.4 MB, MPEG-4 (H.264 codec).

Video S3: Strictrly collaborative task. This video refers to the case study on the strictly collaborative task, and shows the evolved behaviour for two different collaboration levels. Size: 15.7 MB, MPEG-4 (H.264 codec).