Distributed Learning of Cooperative Robotic Behaviors using Particle Swarm Optimization

Ezequiel Di Mario, Iñaki Navarro, and Alcherio Martinoli*

Distributed Intelligent Systems and Algorithms Laboratory,
School of Architecture, Civil and Environmental Engineering,
École Polytechnique Fédérale de Lausanne
{ezequiel.dimario, inaki.navarro, alcherio.martinoli}@epfl.ch

Abstract. In this paper we study the automatic synthesis of robotic controllers for the coordinated movement of multiple mobile robots. The algorithm used to learn the controllers is a noise-resistant version of Particle Swarm Optimization, which is applied in two different settings: centralized and distributed learning. In centralized learning, every robot runs the same controller and the performance is evaluated with a global metric. In the distributed learning, robots run different controllers and the performance is evaluated independently on each robot with a local metric. Our results from learning in simulation show that it is possible to learn a cooperative task in a fully distributed way employing a local metric, and we validate the simulations with real robot experiments where the best solutions from distributed and centralized learning achieve similar performances.

1 Introduction

This paper deals with the synthesis of simple controllers for cooperative tasks performed by resource-constrained robots. Under these settings, evaluative machine learning techniques are an interesting tool for human designers that may be able to fully exploit the platforms' limited sensing capabilities as well as deal with noise in the performance evaluations [1–4].

The cooperative task chosen for this study is a loosely-coordinated collective movement or flocking [5–9], in which a set of robots move together as a group. Previous works have shown that it is feasible to use learning to generate cooperative behaviors [2, 3]. However, in these cases learning has been done in a centralized manner, using homogeneous controllers and a global performance metric. The goal of this paper is to distribute the learning process, which increases robustness to failure of individual agents and may also speed up the learning process by testing several candidate solutions at the same time [10]. In order to achieve this goal, we aim to design a local or individual performance metric that can be evaluated by each robot but also leads to the desired cooperative behavior.

* This research was supported by the Swiss National Science Foundation through the National Center of Competence in Research Robotics.

It should be noted that the task as implemented in this article is harder than those from previous work in that the robots are not physically connected to each other [2], they are required not only to aggregate but also move together[3], and there is no environmental template or goal to guide their movement [1].

Both the local and global performance metrics used in this article mimic in their components two of the flocking Reynolds' rules [11]: avoiding collisions and attraction to neighboring flock-mates. The alignment or velocity matching rule is not directly reflected in the performance metric in order to simplify the implementation on real robots. In [12], Q-learning is used to generate flocking behaviors of virtual agents (not robots) in the presence of a predator, where the agents individually learn discrete actions similar to Reynolds' rules.

Some researchers have used different optimization techniques to improve the performance of human designed flocking controllers [13–15]. Our approach in this article differs in that our behaviors are generated from a general non-recurrent artificial neural network and not a specific flocking controller.

The remainder of this article is organized as follows. Section 2 describes the learning algorithms, performance metrics and controller used. Section 3 describes the experimental setup and the different experiments performed. In Section 4 we present and discuss the results obtained both in simulation and with real robots. Finally, in Section 5 we summarize the findings of this article and discuss the limitations of the approach to be addressed in future work.

2 Methodology

In this article, a version of Particle Swarm Optimization (PSO) [16] is used in order to learn a coordinated collective movement behavior. The learning problem for PSO is choosing a set of parameters of an underlying robotic controller such that a given performance metric is maximized.

2.1 Learning Algorithm

The PSO algorithm used is a noise-resistant variation introduced by Pugh et al. [17], which operates by re-evaluating personal best positions and aggregating them with the previous evaluations (in our case a regular average performed at each iteration of the algorithm). The pseudocode for the algorithm is shown in Figure 1.

The position of each particle represents a set of parameters of a controller. The movement of particle i in dimension j depends on three components: the velocity at the previous step weighted by an inertia coefficient w, a randomized attraction to its personal best $x^*_{i,j}$ weighted by w_p, and a randomized attraction to the neighborhood's best $x^*_{i',j}$ weighted by w_n (Eq. 1). $rand()$ is a random number drawn from a uniform distribution between 0 and 1.

$$v_{i,j} = w \cdot v_{i,j} + w_p \cdot rand() \cdot (x^*_{i,j} - x_{i,j}) + w_n \cdot rand() \cdot (x^*_{i',j} - x_{i,j}) \qquad (1)$$

```
1: Intialize particles
2: for $N_i$ iterations do
3:     for $\lceil N_p/N_{rob} \rceil$ particles do
4:         Update particle position
5:         Evaluate particle
6:         Re-evaluate personal best
7:         Aggregate with previous best
8:         Share personal best
9:     end for
10: end for
```

Fig. 1. Noise-resistant PSO algorithm.

The PSO neighborhood presents a ring topology with one neighbor on each side. Particles' positions and velocities are initialized randomly with a uniform distribution in the $[-20, 20]$ interval, and their maximum velocity is also limited to that interval. The PSO algorithmic parameters are set following the guidelines for limited-time adaptation we presented in our previous work [18] and are shown in Table 1. Since the dimension of the search space is 26, we round up to 28 particles in order to have exactly seven particles per robot in the distributed implementation.

Table 1. PSO parameter values

Parameter	Value
Number of robots N_{rob}	4
Population size N_p	28
Iterations N_i	50
Evaluation span t_e	4x45 s
Re-evaluations N_{re}	1
Personal weight pw	2.0
Neighborhood weight nw	2.0
Dimension D	26
Inertia w	0.8
V_{max}	20

Using the PSO algorithm we explore two different learning schemes, in relation to how the particles are distributed among the robots and how the fitness function is defined. The first, global homogeneous, copies the same candidate solution (or set of weights) to every robot, and uses a global fitness function that evaluates the group behavior. The second, local heterogeneous, distributes a different candidate solution (or set of weights) to each robot, and uses a local fitness function that is evaluated independently and individually on each robot. The distributed version allows to speed up the evaluations by a factor equal to the number of robots, yet it makes the learning harder, especially when the local and global performance metrics are not trivially aligned (e.g., the global performance can be represented by a linear combination of local performances).

2.2 Performance Functions

This Section gives the mathematical definition of the performance metrics used for centralized and distributed learning. The way inputs are measured during the experiments in simulation and reality is described in Section 3.

Both global and local performance functions have three factors: movement, compactness, and collision avoidance. These factors reward robots that move as far as possible from their initial positions, stay close to each other, and avoid collisions between them. The factors are all normalized to the interval [0, 1].

The movement factor of the global performance metric (f_{1g}) is the normalized distance between the initial and the final positions of the center of mass of the group of robots. The normalization factor is the maximum distance that a robots can travel in one evaluation, i.e., the robot's maximum speed multiplied by the evaluation time.

$$f_{1g} = \frac{|\boldsymbol{x_c}(t_f) - \boldsymbol{x_c}(t_0)|}{D_{max}} \tag{2}$$

The global compactness factor (f_{2g}) is the average over the evaluation time and over each pair of robots of the inter-robot fitness. We define the inter-robot fitness between two robots as a function of the distance between them, as shown in Fig. 2. The fitness is maximum at $0\,m$, and it is zero when the robots are further apart than $0.7\,m$. At each time step, we calculate the inter-robot fitness for each pair of robots, and then average across all pairs:

$$f_{2g} = \frac{1}{N_{eval}} \sum_{k=1}^{N_{eval}} \left(\frac{1}{N_{pairs}} \sum_{j=1}^{N_{pairs}} fit\_inter_{j,k}\right) \tag{3}$$

where N_{eval} is the number of time steps in the evaluation period, N_{pairs} is number of inter-robot pairs and $fit\_inter_{j,k}$ is the inter-robot fitness for inter-robot pair j at time step k.

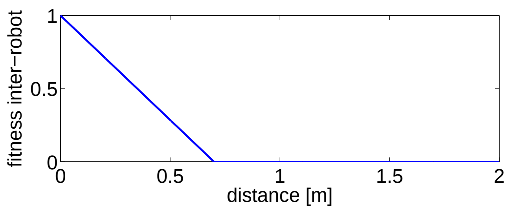


Fig. 2. Inter-robot fitness as a function of the distance between two robots.

The global collision-avoidance factor (f_{3g}) is the average for every robot and over the evaluation time of the maximum value of the proximity sensors at each time step:

$$f_{3g} = \frac{1}{N_{robots}} \sum_{j=1}^{N_{robots}} (\frac{1}{N_{eval}} \sum_{k=1}^{N_{eval}} i_{max,j,k}) \tag{4}$$

where $i_{max,j,k}$ is the normalized proximity sensor activation value of the most active sensor at time step k for robot j, and N_{robots} is the number of robots.

The local performance metric is calculated individually by each robot, using exclusively on-board resources. The local movement factor (f_{1l}) is the normalized distance travelled by the robot, based on the final position, which is calculated with odometry using the wheel encoders.

$$f_{1l} = \frac{|\boldsymbol{x_i}(t_f) - \boldsymbol{x_i}(t_0)|}{D_{max}} \tag{5}$$

The local compactness factor (f_{2l}) is also based on the inter-robot fitness as defined in Figure 2 and used in Equation 3. However, in the local case the number of pairs N_{pairs} in Equation 3 is modified so that each robot only measures the distance to the other three using an on-board range and bearing module, and then averages the inter-robot fitness only for those other three robots, as opposed to averaging across all pairs of robots. Another difference worth noting between the local and global compactness factors is that the local inter-robot distance measurements are affected by occlusion, while the global ones are not.

Finally, the local collision-avoidance factor (f_{3l}) is the single robot version of the global factor:

$$f_{3l} = \frac{1}{N_{eval}} \sum_{k=1}^{N_{eval}} i_{max,j,k} \tag{6}$$

Both global and local fitness are obtained by aggregating the three corresponding factors using the generalized aggregation functions described by Zhang et al. [19]:

$$F = \left(\frac{\omega_1 f_1^s + \omega_2 f_2^s + \omega_3 f_3^s}{\omega_1 + \omega_2 + \omega_3} \right)^{\frac{1}{s}} \tag{7}$$

where f_i are the individual fitness factors, ω_i their corresponding aggregation weights, and s is the degree of compensation. For all experiments in this article we set the degree of compensation s equal to zero, simplifying Eq. 7 to:

$$F = (f_1^{\omega_1} f_2^{\omega_2} f_3^{\omega_3})^{\frac{1}{\omega_1+\omega_2+\omega_3}} \tag{8}$$

Since the three factors (f_i) are in the interval [0, 1], the fitness function F will also be in the same range. The different combinations of aggregation weights explored in this article are as follows: $\{\omega_1 = 0.25, \omega_2 = 0.5, \omega_3 = 0.25\}$, $\{1/3, 1/3, 1/3\}$, and $\{0.1, 0.8, 0.1\}$.

In our previous work [20], we showed that the fitness evaluations for learning a simpler robotic task had a large standard deviation, and that performing re-evaluations was an effective way of dealing with this challenge in the learning.

Given the more complex behavior to be learned in this article and the difficulties encountered while doing so, we decided to perform multiple internal evaluations of the fitness and average them in order to make the learning more robust. Concretely, for each particle evaluation of the PSO algorithm, we perform 4 complete experimental runs of $45\,s$ and average them arithmetically ($F' = \frac{1}{4}\sum_{i=1}^{4} F_i$).

2.3 Controller Architecture

The controller is a non-recurrent artificial neural network of two units which uses only local, on-board measurements regardless of the performance metric. Its inputs are the range and bearing measurements and the infrared proximity sensors, and it outputs the two wheel speeds. Each neuron has 13 input connections: 4 corresponding to the infrared proximity sensors, 8 corresponding to the range and bearing sensor, and one constant bias speed, resulting in 26 weight parameters (w_k) in total. The outputs of the neurons define the wheel speeds $\{v_l, v_r\}$ as given by Equations 9 and 10. $f(\cdot)$ represents the sigmoidal activation function.

$$v_l = f(w_1 + \sum_{k=1}^{4} i_k \cdot w_{k+1} + \sum_{k=1}^{8} rb_k \cdot w_{k+5}) \tag{9}$$

$$v_r = f(w_{14} + \sum_{k=1}^{4} i_k \cdot w_{k+14} + \sum_{k=1}^{8} rb_k \cdot w_{k+18}) \tag{10}$$

Instead of using the robot's nine proximity sensors as inputs, the neural network inputs use four virtual sensors ir_k (front-left, front-right, back-left and back-right) obtained from averaging in pairs and normalizing the sensor values of eight sensors and discarding the central sensor in the back part. This grouping allows us to reduce the number of weight parameters while still being able to detect and avoid obstacles [21].

The eight range and bearing inputs rb_k are obtained by dividing the bearing into eight sectors, and calculating the activation of each sector by taking the maximum range value measured in that sector and dividing it by the maximum possible range, which is approximately 3 meters.

3 Experimental Setup

Our experimental platform is the Khepera III mobile robot, a differential wheeled vehicle with a diameter of 12 cm (Fig. 3a). It is equipped with nine infra-red sensors for short range obstacle detection, which constitute one of the external inputs for the controller. The other inputs are the distances to neighboring robots. This information is obtained through a relative positioning system [22], which calculates range and bearing to nearby robots based on the strength of

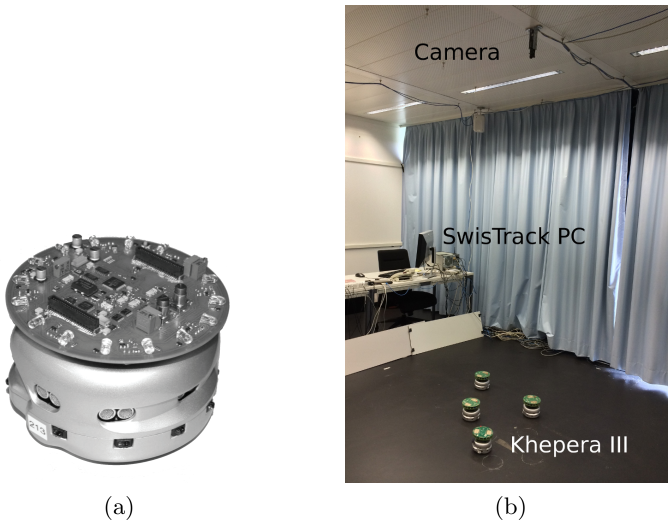


Fig. 3. (a) A Khepera III robot with a range and bearing module attached. (b) Experimental setup in the arena.

the infrared signal. The Khepera III has two wheel encoders, which are used to estimate the trajectory followed by the robots for the fitness movement factor calculations.

Since the response of the Khepera III proximity sensors is not a linear function of the distance to the obstacle, the proximity values are inverted and normalized using measurements of the real robot sensor's response as a function of distance. This inversion and normalization results in a proximity value of 1 when touching an obstacle, and a value of 0 when the distance to the obstacle is equal to or larger than 10 cm.

Simulations are performed in Webots [23], a realistic physics-based submicroscopic simulator that models dynamical effects such as friction and inertia. In this context, by submicroscopic we mean that it provides a higher level of detail than usual microscopic models, faithfully reproducing intra-robot modules (e.g., individual sensors and actuators). The simulator has a built-in relative positioning system that gives information about the distance and direction to neighboring robots within line-of-sight, mimicking the one used in the real robots.

The learning process is performed completely in simulation. Each evaluation during the learning process has a duration of $45\,s$ and takes place in an unbounded arena. Four robots are placed forming a square of side length equal to two robot diameters with random orientations. In order to calculate the local fitness function, robots only use their internal measurements (simulated range and bearing sensor and wheel encoders, both with added noise). The global fitness function is calculated using the robots' global positions with no errors provided by the simulator.

After the learning process is finished, the performance of the solution from each of the 20 learning runs is evaluated systematically in simulation, running 20 experiments of $45\,s$ for each solution.

Based on the results of these tests in simulation, the best solution for global homogeneous learning and the best solution for local heterogeneous learning are selected for systematic tests with real robots. We run 20 experiments for each solution. In these experiments, the global positions is monitored using an overhead camera connected to a computer running SwisTrack [24] (see Figure 3b). The initial positions and number of robots are the same as used for learning in simulation, but the evaluation time is reduced to $10\,s$ in order to be able to keep track of the robots' positions during the whole evaluation due to the limited field of view of the fixed overhead camera and the ideal unbounded arena.

Following the same scheme as done in simulation, the local fitness function is computed on each robot using only its on-board resources, while the global fitness is computed externally given the information provided by the overhead camera and complemented with the local measures for the avoidance factor obtained from the robots.

The two selected best controllers are also re-evaluated in simulation using the reduced time of $10\,s$ in order to perform valid quantitative comparisons and validate our models.

4 Experiments and Results

As mentioned in Section 3 the learning is conducted in simulation with PSO, which is a stochastic optimization method. Therefore, for statistical significance, we perform 20 optimization runs for global homogeneous (centralized) learning and another 20 runs for local heterogeneous (distributed) learning. Figure 4 shows the progress of the best solution found at each iteration for the two different learning approaches. The curves show the average of the 20 runs, and the error bars represent one standard deviation.

Comparing Figure 4b with Figure 4a we can notice that the performance of the local heterogeneous learning measured with the global metric is not as high as the global homogeneous one measured with the same metric. However, it should be noted that in homogeneous learning each iteration uses four times the number of evaluations as heterogeneous learning, as in homogeneous learning each candidate solution must be copied to all robots while in heterogeneous learning each robot tests a different candidate solution at the same time.

In addition to the global metric, Figure 4b shows the progress of the local performance metric for local heterogeneous learning. The global and local metrics are aligned, in the sense that learning with the local one leads to an improvement in the global one.

After the learning is finished, each of the 20 solutions found in the learning runs is tested in simulation for 20 evaluations of 45 seconds. Figure 5 shows the performance measured using the global metric obtained in this testing. The solutions from homogeneous learning outperform the ones from heterogeneous learn-

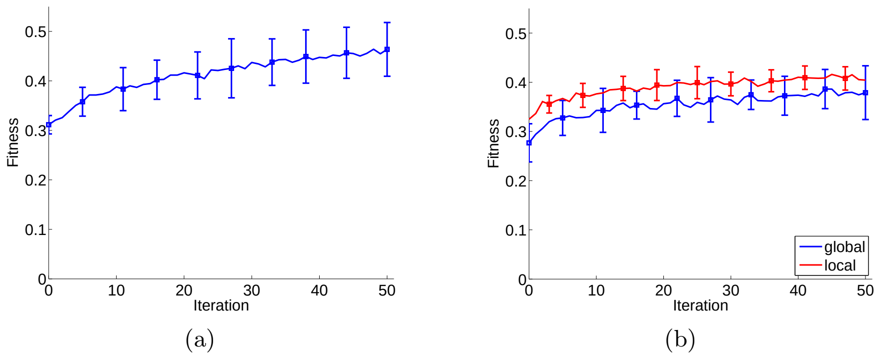


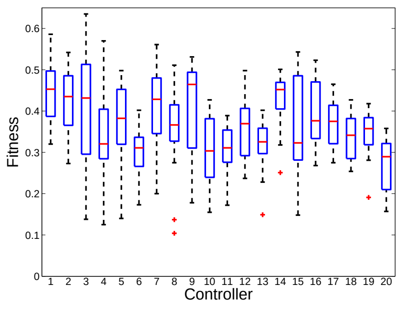
Fig. 4. (a) Learning progress measured using the global metric for global homogeneous (centralized) learning . (b) Learning progress measured using the global (blue) and local (red) metrics for local heterogeneous (distributed) learning.

ing in average. However, the best solution from heterogeneous learning (number 15 in Figure 5b) has the highest performance over all.

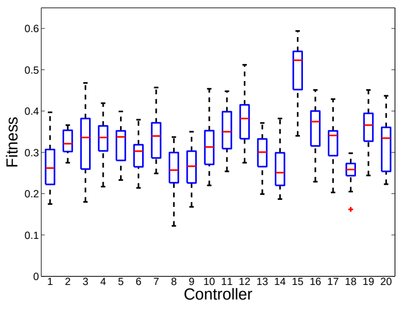
All solutions present a high variation between evaluations (note that in this case the boxplots represent the variation in the performance of each solution during the 20 evaluation runs, which is different from the variation in the learning shown in Figure 4). This variation between individual evaluations implies that the controllers are sensitive to the initial conditions, i.e. the initial random orientations of the robots. The initial orientations affect the time it takes for the robots to find a common direction of movement, and therefore the total distance that the center of mass is able to travel in the 45 seconds. When robots fail to find a common direction of movement, they either aggregate close to their initial positions in a very compact group or split in smaller groups and go in separate directions. Figure 6 shows two example trajectories where a common direction of movement was found relatively quickly, allowing the robots to travel a large distance while staying close to each other.

It should be noted that the weights used in the fitness aggregation function also have a significant effect on the behavior of the resulting controllers. Before choosing the final values of $\{0.25, 0.5, 0.25\}$ for movement, compactness, and avoidance respectively, preliminary tests were conducted in simulation with two other set of weights: $\{1/3, 1/3, 1/3\}$ and $\{0.1, 0.8, 0.1\}$. Figure 7 shows the effect of these fitness aggregation weights on the resulting behaviors. Figure 7a has a low compactness weight, causing the robots to spread out, while Figure 7b has a high compactness weight, causing robots to stay together without moving far. In order to keep the plots clear and avoid clutter, only the initial $10\,s$ of the trajectories are shown.

Based on the results from the tests in simulation, the solutions with the highest medians were chosen to be tested on real robots (number 9 in Figure 5a, homogeneous learning, and number 15 in Figure 5b, heterogeneous learning). We conducted 20 evaluation runs of 10 seconds for each solution, both in simulation

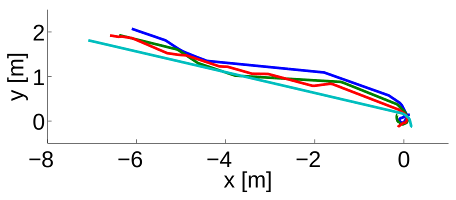


(a)

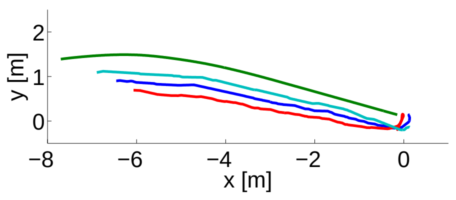


(b)

Fig. 5. Performance measured with the global metric in simulation for the 20 solutions found with (a) global homogeneous (centralized) learning and (b) local heterogeneous (distributed) learning.



(a)



(b)

Fig. 6. Example of trajectories from simulation for successful runs with (a) local heterogeneous and (b) global homogeneous learning.

and reality. The evaluation time was reduced to 10 seconds to keep the robots in the overhead camera's field of view. The quantitative performance measured in these evaluations is presented in Figure 8, which shows a good correspondence between simulation and reality.

Qualitatively, the behaviors observed in reality were similar to the ones obtained in simulation. Figure 9 depicts two example trajectories from these 10 s evaluations. Note that the trajectories shown here for real robots last 10 s, representing only a fraction of the 45 s from those in Figure 6, but the initial steps are very similar. During the real robot experiments, it became evident that for the heterogeneous solution the robot in front of the group was always the same, while the other three robots followed, meaning that heterogeneous learning led to specialized roles. On the other hand, for the homogeneous solution, the robot in front changed every time based on the initial random orientations.

5 Conclusion and Future Work

Our results have shown that it is feasible to learn a cooperative task in a fully distributed way with a local performance metric measured using local, on-board,

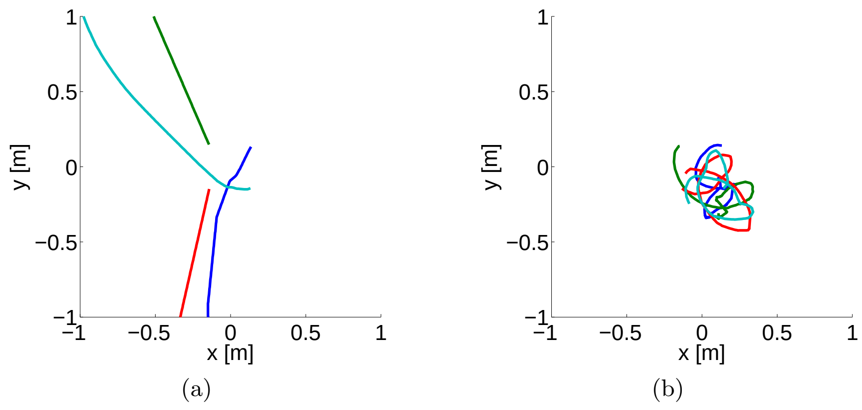


Fig. 7. Example of trajectories for different fitness aggregation weights: (a) $\{1/3, 1/3, 1/3\}$ and (b) $\{0.1, 0.8, 0.1\}$, for compactness, movement, and avoidance, respectively.

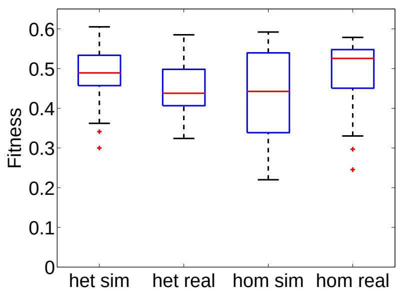
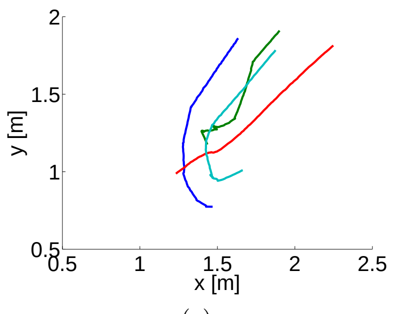


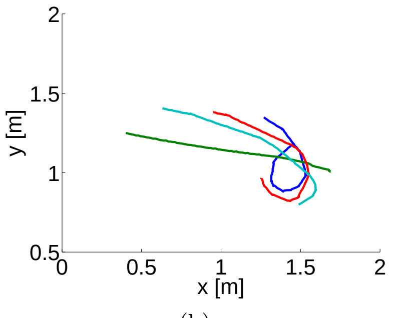
Fig. 8. Performance measured with the global metric in simulation and reality for the best controllers found with global homogeneous and local heterogeneous learning.

noisy sensors. On average, the performance of the solutions found with the distributed approach measured with the global metric was not as high as the ones from centralized learning. This difference was not only due to the metric chosen for learning but also to the increased difficulty in coordination arising from heterogeneous controllers. However, the best solutions found for centralized and distributed learning performed similarly, both in simulation and in experiments with real robots. Additionally, the best solution from distributed learning exhibited specialized roles in which one robot consistently led the group while the others followed.

We have also seen that regardless of the learning method, the coordinated motion task was very sensitive to the initial configuration of the robots, and therefore the performance evaluations were noisy. We addressed this issue in the learning by using different initial orientations for each evaluation and averaging their performances.



(a)



(b)

Fig. 9. Example of trajectories for successful runs with (a) local heterogeneous and (b) global homogeneous learning tested on the real robots.

As future work, we intend to make the solutions for the coordinated motion task more consistent and robust. In order to achieve this, we will explore increasing the complexity of the controller by using the relative velocity or the relative orientation to other robots as inputs, as well as a adding a corresponding alignment term in the local and global learning metrics. In addition, we would like to explore learning in the presence of obstacles in order to generate obstacle avoidance at the group level. Finally, we would like to test the learned controllers in larger flocks of robots by replicating the sets of controllers. We are specially interested in seeing how the heterogeneous solutions perform.

References

1. Floreano, D., Mondada, F.: Evolution of homing navigation in a real mobile robot. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics **26**(3) (1996) 396–407
2. Baldassarre, G., Trianni, V., Bonani, M., Mondada, F., Dorigo, M., Nolfi, S.: Self-organized coordinated motion in groups of physically connected robots. IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society **37**(1) (February 2007) 224–39
3. Gauci, M., Chen, J., Dodd, T., Groß, R.: Evolving Aggregation Behaviors in Multi-Robot Systems with Binary Sensors. In: International Symposium on Distributed Autonomous Robotic Systems. (2012)
4. Jin, Y., Branke, J.: Evolutionary Optimization in Uncertain EnvironmentsA Survey. IEEE Transactions on Evolutionary Computation **9**(3) (June 2005) 303–317
5. Balch, T., Arkin, R.: Behavior-based formation control for multi-robot teams. IEEE Transactions on Robotics and Automation **14**(6) (1998) 926–939
6. Fredslund, J., Mataric, M.J.: A general algorithm for robot formations using local sensing and minimal communication. IEEE Transactions on Robotics and Automation, Special Issue on Multi Robot Systems **18(5)** (October 2002) 837–846
7. Olfati-Saber, R.: Flocking for multi-agent dynamic systems: Algorithms and theory. IEEE Transactions on Automatic Control **51** (2006) 401–420

8. Antonelli, G., Arrichiello, F., Chiaverini, S.: Flocking for multi-robot systems via the null-space-based behavioral control. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. (2008)
9. Navarro, I., Matía, F.: A framework for collective movement of mobile robots based on distributed decisions. Robotics and Autonomous Systems **59**(10) (October 2011) 685–697
10. Di Mario, E., Martinoli, A.: Distributed Particle Swarm Optimization for Limited Time Adaptation with Real Robots. Robotica **32**(02) (2014) 193–208
11. Reynolds, C.W.: Flocks, herds, and schools: A distributed behavioral model. Computer Graphics **21**(4) (1987) 25–34
12. Morihiro, K., Isokawa, T., Nishimura, H., Matsui, N.: Emergence of flocking behavior based on reinforcement learning. In Gabrys, B., Howlett, R., Jain, L., eds.: Knowledge-Based Intelligent Information and Engineering Systems. Volume 4253 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2006) 699–706
13. Lee, S.M., Myung, H.: Particle swarm optimization-based distributed control scheme for flocking robots. In Kim, J.H., Matson, E.T., Myung, H., Xu, P., eds.: Robot Intelligence Technology and Applications 2012. Volume 208 of Advances in Intelligent Systems and Computing. Springer Berlin Heidelberg (2013) 517–524
14. Vatankhah, R., Etemadi, S., Honarvar, M., Alasty, A., Boroushaki, M., Vossoughi, G.: Online velocity optimization of robotic swarm flocking using particle swarm optimization (pso) method. In: ISMA '09. 6th International Symposium on Mechatronics and its Applications. (March 2009) 1–6
15. Celikkanat, H.: Optimization of self-organized flocking of a robot swarm via evolutionary strategies. In: Computer and Information Sciences, 2008. ISCIS '08. 23rd International Symposium on. (Oct 2008) 1–4
16. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE International Conference on Neural Networks. (1995) 1942 – 1948 vol.4
17. Pugh, J., Zhang, Y., Martinoli, A.: Particle swarm optimization for unsupervised robotic learning. In: IEEE Swarm Intelligence Symposium. (2005) 92–99
18. Di Mario, E., Martinoli, A.: Distributed Particle Swarm Optimization for Limited Time Adaptation in Autonomous Robots. In: International Symposium on Distributed Autonomous Robotic Systems 2012. (Springer Tracts in Advanced Robotics 2014 (to appear))
19. Zhang, Y., Antonsson, E., Martinoli, A.: Evolutionary engineering design synthesis of on-board traffic monitoring sensors. Research in Engineering Design **19**(2) (2008) 113–125
20. Di Mario, E., Navarro, I., Martinoli, A.: Analysis of fitness noise in particle swarm optimization: From robotic learning to benchmark functions. In: 2014 IEEE Congress on Evolutionary Computation. (to appear)
21. Di Mario, E., Navarro, I.n., Martinoli, A.: The Role of Environmental and Controller Complexity in the Distributed Optimization of Multi-Robot Obstacle Avoidance. In: IEEE International Conference on Robotics and Automation. (2014)
22. Pugh, J., Raemy, X., Favre, C., Falconi, R., Martinoli, A.: A fast on-board relative positioning module for multi-robot systems. IEEE/ASME Transactions on Mechatronics, Focused Section on Mechatronics in Multi Robot Systems (2009)
23. Michel, O.: Webots: Professional Mobile Robot Simulation. Advanced Robotic Systems **1**(1) (2004) 39–42
24. Lochmatter, T., Roduit, P., Cianci, C., Correll, N., Jacot, J., Martinoli, A.: SwisTrack - a flexible open source tracking software for multi-agent systems. In: IEEE/RSJ 2008 International Conference on Intelligent Robots and Systems (IROS 2008), IEEE (2008) 4004–4010