Swarm Intelligence manuscript No.
(will be inserted by the editor)

Distributed scalable multi-robot learning using particle swarm optimization

Jim Pugh and Alcherio Martinoli

Received: date / Accepted: date

Abstract Designing effective behavioral controllers for mobile robots can be difficult and tedious; this process can be circumvented by using online learning techniques which allow robots to generate their own controllers online in an automated fashion. In multi-robot systems, robots operating in parallel can potentially learn at a much faster rate by sharing information amongst themselves. In this work, we use an adapted version of the Particle Swarm Optimization algorithm in order to accomplish distributed online robotic learning in groups of robots with access to only local information. The effectiveness of the learning technique on a benchmark task (generating high-performance obstacle avoidance behavior) is evaluated for robot groups of various sizes, with the maximum group size allowing each robot to individually contain and manage a single PSO particle. To increase the realism of the technique, different PSO neighborhoods based on limitations of real robotic communication are tested and compared in this scenario. We explore the effect of varying communication power for one of these communication-based PSO neighborhoods. To validate the effectiveness of these learning techniques, fully distributed online learning experiments are run using a group of 10 real robots, generating results which support the findings from our simulations.

Keywords multi-robot systems · robotic learning · particle swarm optimization

1 Introduction

Designing even simple behaviors for mobile robots that are both efficient and robust can be very difficult for humans; it is often not hard to implement a rudimentary controller that accomplishes the task, but achieving near-optimal performance can be very challenging. This is especially true for miniature mobile robots as it can be quite difficult to conceptualize the environment from the robot's point of view. *Online robotic learning* allows for automated design of efficient, robust controllers, which saves much design time and effort (see Section 2 for a complete definition of online robotic

Distributed Intelligent Systems and Algorithms Group
École Polytechnique Fédérale de Lausanne
1015 Lausanne, Switzerland
{jim.pugh,alcherio.martinoli}@epfl.ch

learning). Learning is also essential for allowing robots to adapt to situations where the task or environment is unknown beforehand or is constantly changing.

Particle Swarm Optimization (PSO) is an optimization technique which models a set of candidate problem solutions as a swarm of particles moving about in a virtual search space. The method was inspired by the movement of flocking birds and their interactions with their neighbors in the group. PSO achieves optimization using three primary principles: evaluation, where quantitative performance can be determined for some particle location; comparison, where the best performer out of multiple particles can be selected; and imitation, where the qualities of better particles are mimicked by others. By coupling it with an Artificial Neural Network (ANN), the algorithm can be used to optimize parameters for robot controllers in order to accomplish online robotic learning.

In the PSO algorithm, groups of virtual agents interact in order to achieve optimization. In distributed robotics, groups of robots interact to accomplish their goals. It may therefore be possible to implement these algorithms in a parallel distributed fashion for learning in multi-robot systems. Each robot would be responsible for one or several virtual agents, which it evaluates at each iteration of the algorithm. After each set of evaluations, the robots would communicate to share the performance information needed to progress to the next iteration. By running the algorithm in this fashion, we would need no external supervisor to oversee the learning process, and the speed of learning could be significantly improved, as many robots evaluating in parallel increase the rate of candidate solution evaluations and therefore decrease the total learning time.

In the local neighborhood version of PSO, each particle only needs to be aware of the state of a small subset of particles in the population in order to update itself at each iteration. It may therefore be possible to implement PSO in a distributed manner where communication from any given node would only be necessary with several other nodes, making it a very scalable, parallel approach. This could allow the speed and quality of learning to be improved by simply introducing more robots into the group.

In this paper, we explore the effectiveness of using a modified version of PSO on groups of robots performing distributed online learning. Our case study is the generation of high-performance obstacle avoidance behavior, a fundamental building block for more complex behaviors and a common benchmark for online robotic learning techniques. At the maximum robot team size, the number of robots is set equal to the number of particles in the PSO population, allowing each robot in the group to manage a single particle. We test how the performance is affected if we adapt the standard PSO neighborhood structure to more closely model what is possible for robots with limited communication abilities. Simulated experiments are validated by replicating them on a group of 10 real robots learning in parallel. Section 2 of this work provides some background on PSO, online robotic learning, and multi-robot learning. In Section 3, we examine how the effectiveness of distributed online learning is affected by the number of robots in the group. Section 4 analyzes how the learning performance is affected by different neighborhood structures based on the limitations of robotic communication when each robot contains a single particle. Section 5 focuses on one such neighborhood structure and tests the effect of varying the communication range of the robots. In Section 6, we perform distributed learning experiments on a group of 10 real Khepera III robots in order to validate the previously obtained results from our simulations. Section 7 discusses the implications of the results and concludes the paper.

2 Background

The original PSO method was developed by Kennedy and Eberhart (Eberhart & Kennedy, 1995; Kennedy & Eberhart, 1995). Every particle in the population begins with a randomized position $\bar{x_i} = (x_{i,1}, ..., x_{i,j}, ..., x_{i,n})$ and randomized velocity $\bar{v_i} = (v_{i,1}, ..., v_{i,j}, ..., v_{i,n})$ in the n-dimensional search space, where i represents the particle index and j represents the dimension in the search space. Candidate solutions are optimized by flying the particles through the virtual space, with attraction to positions in the space that yielded the best results. Each particle remembers the position at which it achieved its highest performance ($\bar{x_i^*}$). Each particle is also a member of some neighborhood of particles, and remembers which particle achieved the best overall position in that neighborhood (given by the index i'). This neighborhood can either be a subset of the particles (local neighborhood), or all the particles (global neighborhood). For local neighborhoods, the standard method is to set neighbors in a pre-defined way (such as using as neighbors particles with the closest array indices modulo the size of the population, henceforth known as a "ring topology") regardless of the particles' positions in the search space. The equations executed by PSO at each step of the algorithm are

$$\begin{aligned} v_{i,j} &= w \cdot v_{i,j} + pw \cdot rand() \cdot (x^*_{i,j} - x_{i,j}) \\ &\quad + nw \cdot rand() \cdot (x^*_{i',j} - x_{i,j}) \\ x_{i,j} &= x_{i,j} + v_{i,j} \end{aligned}$$

where w is the inertia coefficient in $(0,1)$ that slows velocity over time, pw is the weight given to the attraction to the previous best location of the current particle and nw is the weight given to the attraction to the previous best location of the particle neighborhood. $rand()$ is a sample of a random variable uniformly-distributed in $[0,1)$. The PSO algorithm can be used for the optimization of many different systems, and there has been a growing interest in its application within the field of robotics (for example, as a model for distributed robotic foraging (DiChio & DiChio, 2007) or distributed odor localization (Jatmiko et al., 2006)). By coupling the algorithm with a parameterized robotic controller (such as an ANN), PSO optimization can be translated into robotic learning.

Two different general approaches can be used to accomplish robotic learning. **Offline learning** refers to techniques where a robot behavior is generated outside of the actual application. This is the case for controllers created via demonstration or based on pre-collected data. In contrast, **online learning**[1] occurs when a robot adjusts its behavior in real-time based on feedback it receives during operation. Online learning is necessary for creating effective controllers in unknown or dynamic scenarios, as offline learning requires a priori knowledge of the environment. There has been extensive work in the past on both online and offline robotic learning (see Franklin et al., 1996). Common approaches to online learning include regression-based techniques (Atkeson et al., 1997) and evaluative techniques such as Reinforcement Learning (Mahadevan &

[1] In robotic applications, online vs. offline can also refer to whether operation occurs on a real robot vs. in a simulation of the same scenario. The terms **supervised** and **unsupervised learning** may also be used to describe offline and online learning in robotic applications, respectively.

Connell, 1991; Smart & Kaelbling, 2002), Genetic Algorithms (Floreano & Mondada, 1996), and more recently Particle Swarm Optimization (Pugh et al., 2005).

Multi-robot learning has been used and explored in various ways; surveys of work (including learning on non-robotic multi-agent systems) can be found in (Stone & Veloso, 2000; Panait & Luke, 2005). Matarić studied mechanisms to encourage individual agents in a group to act in ways to help the group performance (Matarić, 1994). Parker proposed an architecture, L-ALLIANCE, which allows for efficient adaptation in a robot team (Parker, 1997). Multi-robot learning using several methods in a wide variety of scenarios has been explored in (Balch, 1998; Stone, 1998). Many studies have used Genetic Algorithms (GAs) for homogeneous learning on a team of robots (for example, Dorigo et al., 2004). Multi-robot learning has also been used for groups of robots engaged in adversarial tasks (Bowling & Veloso, 2003). Specialization in multi-agent systems was studied using reinforcement learning (Murciano et al., 1997) and adaptive line-search (Li et al., 2004). Techniques for increasing individual learning speed via multi-robot learning were studied by Kelly & Keating, 1998 and Matarić, 2001. A modified version of GA has been embedded onto a 2-robot system to allow for parallel learning (Nehmzow, 2002). Watson developed the technique "embodied evolution", which exploits the parallelism of scalable multi-robot systems for fast distributed learning (Watson et al., 2002). Pugh and Martinoli found that both GA and PSO could be used for effective distributed parallel multi-robot learning in simulation (Pugh & Martinoli, 2006) and applied to simulated heterogeneous robot groups (Pugh & Martinoli, 2009).

While GA has been used for online robotic learning far more frequently in the past, some recent studies (Pugh et al., 2005 and Pugh & Martinoli, 2006) have found that PSO can actually achieve superior results. In this work, we expand on previous research using PSO for parallel distributed multi-robot learning with simulated robots (Pugh & Martinoli, 2006) and validate results using real-robot experiments.

3 Varying the Robotic Group Size

In previous work (Pugh et al., 2005), online learning was used to teach robots obstacle avoidance behavior for both a single robot and two robots co-learning. We wish to expand this to test online learning on larger robotic groups, where the developing candidate solutions are distributed throughout the group. While using a larger robotic group allows for faster learning, it may also increase the noise in performance evaluations by the robots, which could make learning more difficult.

3.1 Experimental Setup

A major challenge in mobile robotic learning is overcoming the performance noise innate to controller evaluation. Because robots only have a partial perception of their surroundings, random environmental factors may influence results, and sensors and actuators are intrinsically noisy components, the perceived performance over a brief evaluation time span may vary significantly from the robots true performance level in different and unpredictable ways. In order to partially overcome this problem, we use the noise-resistant PSO algorithm from Pugh et al., 2005. This technique uses a local neighborhood in a ring topology with one neighbor on each side (*lbest* topology). At every iteration, the previous best locations of particles are reevaluated; by averaging the new performance value with previous ones, we can partially filter out the noise to get a more accurate measure of the actual performance. The longer a previous best location remains in the algorithm, the more times it will be evaluated, and the more accurate the performance measure will become. Although this requires twice as many performance evaluations at each iteration (normal evaluation plus re-evaluations) as standard PSO, this technique prevents noisy performance evaluations from severely disrupting the learning process and gives significantly better results given the same amount of computational time. A flowchart of the modified algorithm can be seen in Fig. 1.

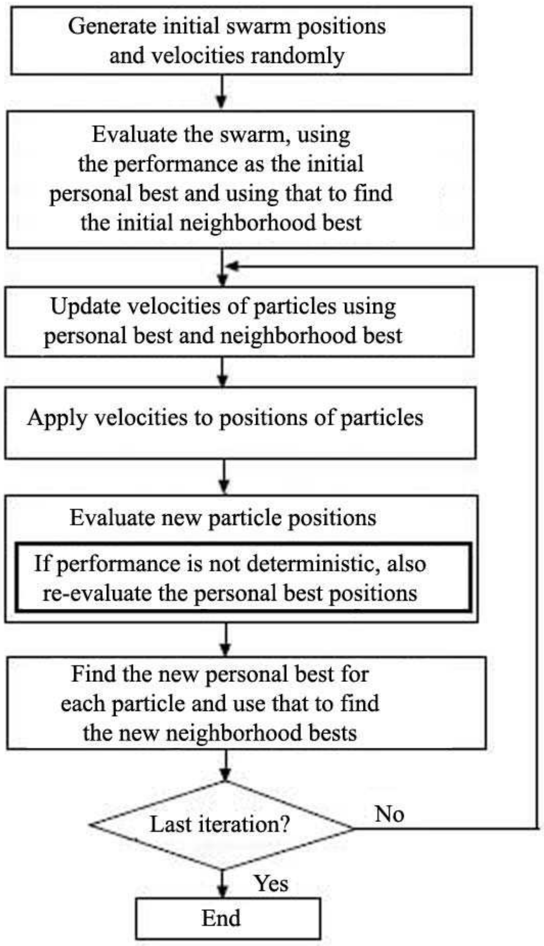

Generate initial swarm positions and velocities randomly
Evaluate the swarm, using the performance as the initial personal best and using that to find the initial neighborhood best
Update velocities of particles using personal best and neighborhood best
Apply velocities to positions of particles
Evaluate new particle positions
If performance is not deterministic, also re-evaluate the personal best positions
Find the new personal best for each particle and use that to find the new neighborhood bests
Last iteration?
No
Yes
End

Fig. 1 Optimization loop used by noise-resistant PSO

The parameters for the PSO algorithms are given in Table 1. While a swarm (or population) size of 10 particles would be very small for other multi-agent optimization techniques such as Genetic Algorithms, we have found that it offers good performance with the PSO algorithm. Initial particle positions are randomly generated in the range $[-20, 20]$ but are allowed to change to any value during optimization. Velocity in PSO is also randomly initialized in the range $[-20, 20]$ but prevented from ever going outside this range.

Table 1 PSO parameters for online learning

Swarm Size	10
pw	2.0
nw	2.0
w	0.8

For all experiments in this work (both in simulation and the real world), we use the Khepera III robotic platform (see Fig. 2), produced by K-Team Corporation with development assistance from the Distributed Intelligent Systems and Algorithms Laboratory at EPFL. The robot has a diameter of 12 cm, making it appropriate for indoor experimentation. Locomotion is accomplished via a differential drive system using two independent motors. The Khepera III uses the Korebot platform, running a standard embedded Linux operating system on an Intel XSCALE PXA-255 processor running at 400 MHz. A stackable expansion bus allows for the addition of custom robot modules. Belts of both ultrasonic and infrared proximity sensors surround the robot, allowing for detection of both close and medium range objects. The robot can be endowed with IEEE 802.11 wireless ability by using an appropriate card with the built-in Compact-Flash slot.

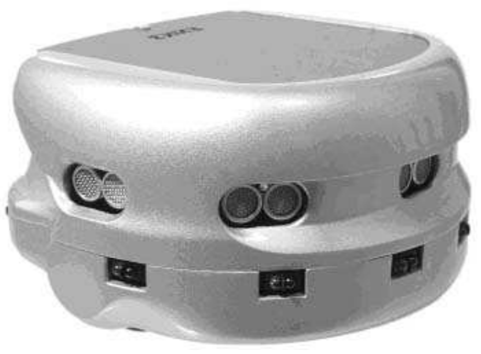
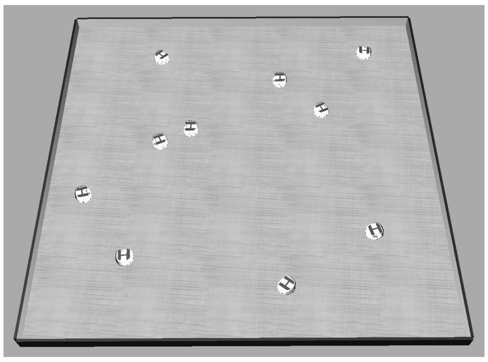


Fig. 2 Khepera III robot

For our experiments in this section of the paper, we use Webots, a realistic simulator (Michel, 2004). This simulator can run experiments much faster than real-world execution (often by a factor greater than 100), allowing many more experiments to be performed than would otherwise be possible. Henceforth, when discussing the length of evaluations and total learning time in simulation, we always refer to simulation time (i.e., how long it would have taken to run the experiment in reality) rather than real-world time (i.e., how long it took to run the simulation).



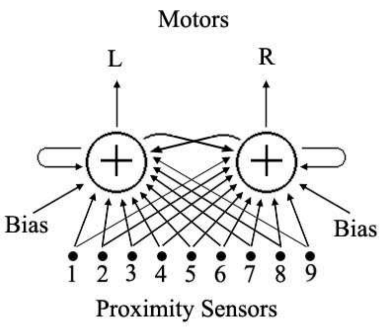


Fig. 3 *Left:* Virtual Khepera III robots in their simulated arena in Webots. *Right:* Depiction of the artificial neural network used for the robot controller. Curved arrows are recurrent connections and lateral inhibitions.

The simulated Khepera III robot operates in a 3.0 m $\times$ 3.0 m square arena (see Fig. 3 left). The robotic controller is a single-layer discrete-time ANN of two neurons, one for each wheel speed, with sigmoidal output functions. The inputs are the nine infrared proximity sensors (approximately equally spaced around the robot), as well as a recursive connection from the previous output of the neuron, lateral inhibitions and bias values (see Fig. 3 right), giving us 24 weights in total. Each of these weights is mapped to a separate dimension in the PSO search space, creating a 24-dimensional space in which particles move about (in other words, $w_j = x_{i,j}$ with $1 \leq j \leq 24$ for some set of ANN weights $\bar{w}$ mapping to some particle position $\bar{x_i}$). Coupling the PSO optimization process with robot performance evaluations allows the ANN weights to be tuned for improved performance as optimization proceeds, which results in the robot learning the desired behavior.

Robot proximity sensors have a maximum range of 10.0 cm, and sensor output varies linearly from 0.0 at maximum range to 1.0 at minimum range (0.0 cm). To increase the realism of the sensors, we add uniform noise of $\pm$ 0.05 at maximum range which scales linearly down to $\pm$ 0.0 at minimum range; this is approximately equivalent to the average noise observed on Khepera III proximity sensors. The maximum speed of the robots is 25 cm/s. Slip noise of 10% is applied to wheel speeds. The time step for neural updates is 64 ms.

The benchmark study for the learning techniques explored here is to generate high-performing controllers for obstacle avoidance behavior in an online fashion. To achieve this, we use the same performance function as was used by Floreano & Mondada, 1996, given by:

$$
\begin{aligned}
&F = V \cdot (1 - \sqrt{\Delta v}) \cdot (1 - i) \\
&0 \leq V \leq 1 \\
&0 \leq \Delta v \leq 1 \\
&0 \leq i \leq 1
\end{aligned}
$$

where V is the average absolute wheel speed of both wheels, Δv is the average of the difference between the wheel speeds, and i is the average activation value of the most active proximity sensor over the evaluation period. These factors reward robots that move quickly (V), turn as little as possible ($1 - \sqrt{\Delta v}$), and spend little time near obstacles ($1 - i$). The terms are normalized to give a maximum performance of 1. Between each performance test, the position and bearing of the robots are randomly set by the simulator to ensure the randomness of the next evaluation.

For the choice of swarm size, number of PSO iterations, and performance evaluation time span for these experiments, we opted for a swarm of 10 particles to be optimized for 50 iterations with an evaluation time of 30 seconds per performance measurement. These choices were guided by empirical tests that indicated that this set of parameters results in good performance given a limited amount of learning time. After the specified number of iterations have been run, learning is halted, regardless of the performance of the swarm at that point.

We test for team sizes of 1, 2, 5, and 10 robots. To distribute work amongst the robots, different robots are assigned to "manage" different particles, by which we mean each robot is responsible for evaluating the performance of the same set of particles at each iteration. With one robot, that robot is responsible for managing all ten particles. With two robots, each robot manages five particles, with particles assigned arbitrarily to each robot at the start of learning. With five robots, the number of managed particles per robot decreases to two, and with ten robots, each robot only manages a single particle (again assigned arbitrarily when learning starts).

Because the number of iterations is fixed, the "speed" of learning is determined by the time required for each iteration. By running performance evaluations in parallel on the different robots, we can decrease this time and therefore reduce the total time required for learning. In this way, going from 1 to 10 robots has a significant effect on the total learning time needed - 8 hours 20 minutes for 1 robot compared to 50 minutes for 10 robots.

At the end of an optimization run, all candidate controllers are tested for five runs of 30 seconds each, and the controller with the highest performance is selected as the best solution. In order to fairly evaluate this best solution, it is tested both on a single robot running by itself and on ten robots running simultaneously; this measures its ability to successfully avoid walls and its ability to avoid both walls and other robots, respectively. Intuition tells us that controllers generated using small robot group sizes would be better suited for the single robot scenario, while controllers generated using large robot group sizes would be better suited for the ten robot scenario, as controllers adapt to best fit their learning environment. For both of these scenarios, the controller is evaluated for 40 runs of 30 seconds each to determine its average performance.

3.2 Results

A comparison of the average performances of the best solutions over 100 runs can be seen in Fig. 4. For all group sizes, single robot evaluation gives a higher performance than ten robot evaluation; this is to be expected, since a robot will encounter less obstacles if it is the only one in the arena. Also as expected, one robot learning by itself achieves better performance for single robot evaluation than larger group sizes. However, it also achieves better performance for ten robot evaluation. This suggests that the added noise from having larger groups of robots increases the difficulty in learning, resulting in worse controllers. However, the decrease in performance is not particularly high (a 5% decrease going from one to ten robots) and may be considered well worthwhile in order to obtain a learning speed-up factor of 10.

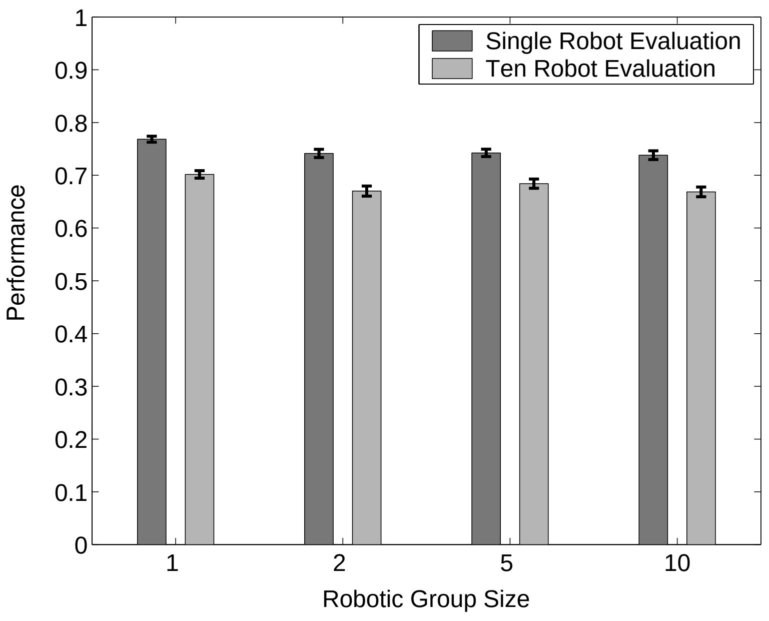


Fig. 4 Average of final best performances over 100 runs with different robotic group sizes. Error bars represent standard error across runs.

One concern in the learning process is overfitting, where generated controllers are highly specific to their exact learning environment and achieve poor performance in other scenarios. Observing Fig. 4, we can discern that generated behaviors are at least somewhat flexible with respect to the obstacle density of the environment; controllers generated with a single robot perform well with 10 robots present, and controllers generated with 10 robots perform well on a lone robot. This suggests that the solutions are not suffering from over-fitting and allow for some generalization.

4 Communication-based Neighborhoods

In multi-robot scenarios, communication range is often limited. Untethered robots have a very limited amount of available energy at their disposal, and it is important to conserve this by restricting transmission power. Also, if communication range is too large, interference between signals can decrease the rate at which data can be sent. If we distribute particles in a PSO population between robots and use the standard PSO local neighborhood model, robots may be required to share information with other robots that are far from their position. Therefore, to realistically model a scalable multi-robot system, particle neighborhoods should be set in such a way that robots are not required to communicate with other robots outside of some close proximity.

4.1 Experimental Setup

We propose two such models for PSO neighborhoods to emulate realistic robot communication.

Model A: Each robot contains one particle. At the end of each performance evaluation, the robot selects the two other robots closest to it, and uses their particles as its neighborhood for the next iteration of the algorithm. This maintains the same number of particles in the neighborhood, but allows for the neighbors to change over the course of the learning process. As the physical location of the robots is independent of the particle indices, this should be roughly equivalent to randomly choosing two neighbors at each iteration of the algorithm, especially since obstacle avoidance behavior should result in a uniformly random distribution of robots within the environment.

Model B: Each robot contains one particle. At the end of each performance evaluation, the robot selects all robots within a fixed radius r, and uses their particles as its neighborhood for the next iteration of the algorithm; this emulates a broadcast transmission from every robot, detectable by any other robot within range. This results in a variable number of neighbors, as the robot may be close to very few or very many robots. However, it is perhaps more realistic than Model A, since for very sparse robot distributions, there may be fewer than two other robots in close proximity at times. If no other robot is within range, the robot uses its own particle as the neighborhood best.

We compare the performance of the original *lbest* neighborhood topology to the two new models, using $r = 1.2$ m, for a group of 10 robots. We use the setup previously described.

4.2 Results

A comparison of the average performances over 100 runs is shown in Fig. 5. The new neighborhood models do not show any decrease in performance, and in fact achieve higher performance than the standard *lbest* topology, though not significantly so (a Mann-Whitney U-test between Model A and *lbest* yields a p-value of 0.583, and a U-test between Model B and *lbest* yields a p-value of 0.110). This does, however, show that random neighborhood selection at each iteration does not result in a performance decrease as compared to the static *lbest* topology. The good performance of Model B indicates that the effectiveness of learning is not tied to keeping strictly two neighbors at each iteration. The success of these models shows that we can accomplish distributed online learning in a realistic multi-robot system.

5 Varying the Communication Range

We now explore the effects of varying the communication range used in Model B. This could be accomplished in a real robotic system by varying the output power of the transmission. It is useful to know the trade-off between output power and learning performance.

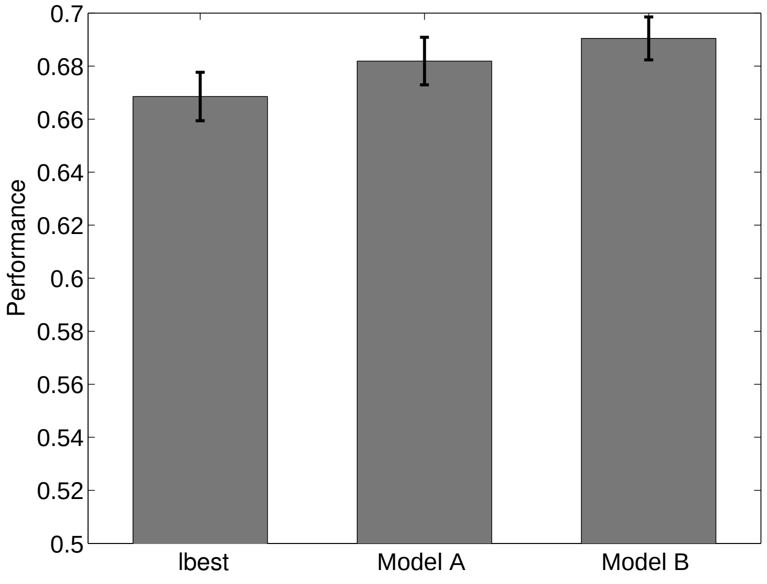


Fig. 5 Average of final best performances over 100 runs for different neighborhood models. Error bars represent standard error across runs.

5.1 Experimental Setup

We use communication ranges of 0.3 m, 0.8 m, 1.0 m, 1.2 m, 3.3 m, and 5.0 m. Table 2 gives the expected number of robots within communication range, assuming a uniformly random distribution of robots within the arena. We therefore go from little interparticle communication to full interparticle communication.

Table 2 Expected number of neighboring robots

r **(m)**	**Expected Number of Neighbors**
0.3	0.3
0.8	1.8
1.0	2.3
1.2	3.1
3.3	8.9
5.0	9.0

These results were obtained by decomposing the arena space into a two-dimensional grid, calculating the likelihood that some robot A is within range of some robot B for all grid positions of both A and B, and scaling this value by the total number of other robots present (in this case nine, for a team of 10 robots). The resolution of the grid was made sufficiently small so that the results are accurate to the level of significance presented. However, the assumption of uniform distribution may not hold for robots learning obstacle avoidance behavior, which could cause the true expected number of neighbors to be either higher or lower or to vary during the learning process.

5.2 Results

The average performances of the best particles over 100 runs for different communication ranges can be seen in Fig. 6. The best results are obtained for intermediate ranges. The progress of the average swarm performance throughout the learning process for 0.3 m, 1.2 m, and 3.3 m can be seen in Fig. 7. The low communication range again performs poorly here, but instead of observing a low performance for the high communication range, both the intermediate and high range appear to perform well.

The discrepancy between our best particle performances in Fig. 6 and average swarm performances in Fig. 7 may be caused by variations in swarm diversity. A lower communication range would mean less information sharing among robots, which could result in greater differences among the robot controllers; a higher communication range would mean more information sharing and possibly more similar controllers. We would expect a swarm with greater diversity to have a larger gap between the performance of the best solution and the average performance of the swarm, while a swarm with less diversity would have a smaller gap. This could explain why the high communication range appears to perform relatively better when using average swarm performance rather than best particle performance as the evaluation metric.

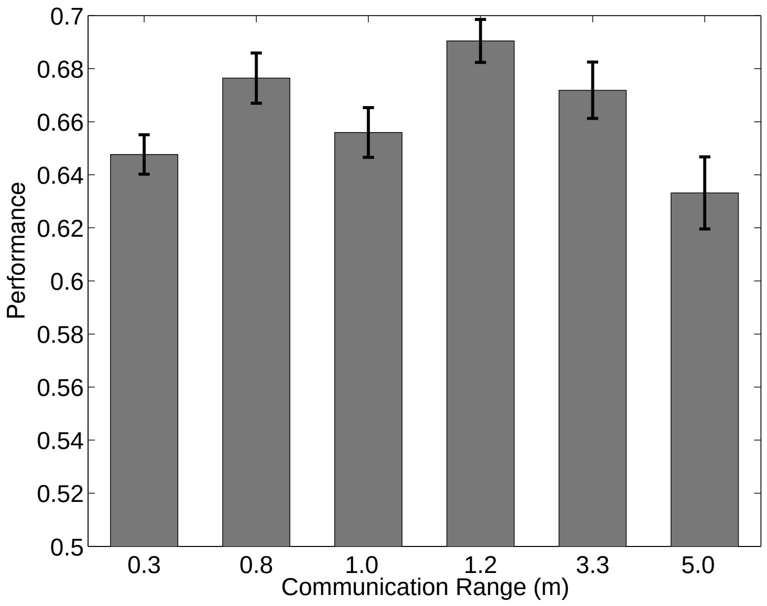


Fig. 6 Average of final best performances over 100 runs for different communication ranges in Model B. Error bars represent standard error across runs.

Decreased performance for low communication range is due to not enough information being exchanged between particles; particles end up mostly relying on their own

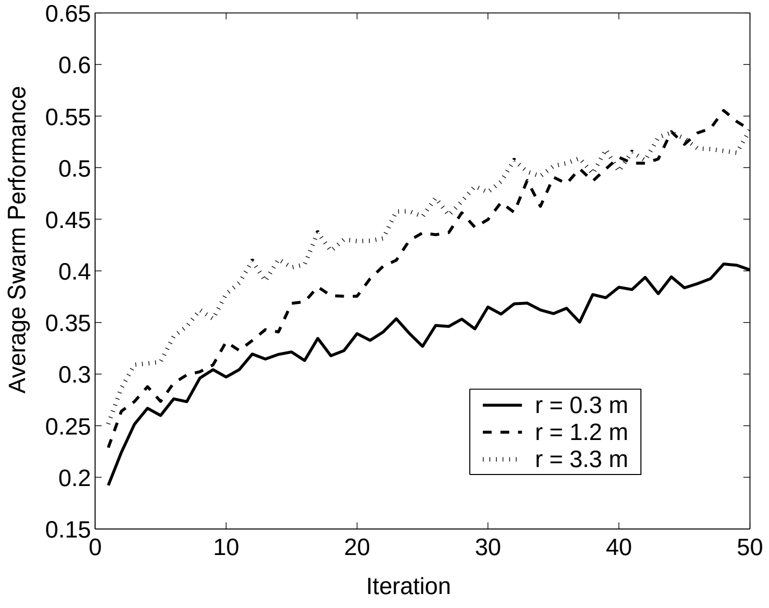


Fig. 7 Average swarm performance over 100 runs for 0.3 m, 1.2 m, and 3.3 m communication range in Model B.

personal best position for learning, which causes slow convergence. In the case of very high communication range, the initial convergence of the population was faster than with the shorter communication ranges, but it would sometimes prematurely converge on a solution which did not have particularly high performance. This indicates that a global neighborhood can actually be detrimental to finding very good solutions, and we therefore gain no benefit whatsoever by expanding our communication range beyond a certain point.

Referring back to Fig. 6, we observe that there is an unexpected drop in the performance for a communication range of 1.0 m as compared to the performances for 0.8 m and 1.2 m. We have not been able to ascertain a specific cause of this variation. One possible explanation is that it may simply be due to noise in the results, although the size of the error bars suggests that this is quite unlikely. The same holds true with the relatively large gap in performance between the 3.3 meter and 5.0 meter ranges, despite the fairly minor difference in expected communication connectivity (8.9 expected robots in range vs. 9.0). An alternative explanation for this discrepancy is that the true expected number of neighbors differs from the results of our numerical calculations, making the gap in effective connectivity much larger than previously expected.

We can further explore the variations in performance for different communication ranges by observing the diversity of the PSO swarm throughout the optimization process. For our diversity metric, we choose simple Euclidean distance in the virtual search space. The pairwise diversity between two particles i and k is therefore given by:

$$d(\bar{x_i}, \bar{x_k}) = \sqrt{\sum_j (x_{i,j} - y_{k,j})^2}$$

The diversity of the entire swarm is given by the average pairwise diversity of all pairs of particles, or:

$$D(\bar{x_1}, ..., \bar{x_n}) = \frac{1}{n(n-1)} \sum_{\bar{x_i}} \left[\sum_{\bar{x_k} \neq \bar{x_i}} d(\bar{x_i}, \bar{x_k}) \right]$$

where n is the total size of the swarm.

Using this metric, we calculate the diversity of the particle swarms throughout the learning process averaged over 100 runs for communication ranges of 0.3 m, 1.2 m, and 3.3 m (see Fig. 8). As predicted, a lower communication range results in higher swarm diversity, while a higher communication range results in a lower swarm diversity. We observe that for the low range of 0.3 m, diversity increases throughout optimization and reaches more than twice its initial value by the end; this suggests that very little convergence is taking place, which prevents the algorithm from finding a good solution in the virtual search space. For the high communication range of 3.3 m, diversity immediately decreases at the start of optimization and remains below its initial value throughout; this indicates rapid convergence, which could prematurely select a robot behavior which gives only mediocre performance. For the intermediate communication range of 1.2 m, diversity increases in the early stages of optimization and then levels off; this may offer a good balance between exploration and exploitation of successful robot behaviors, ultimately achieving the best performance.

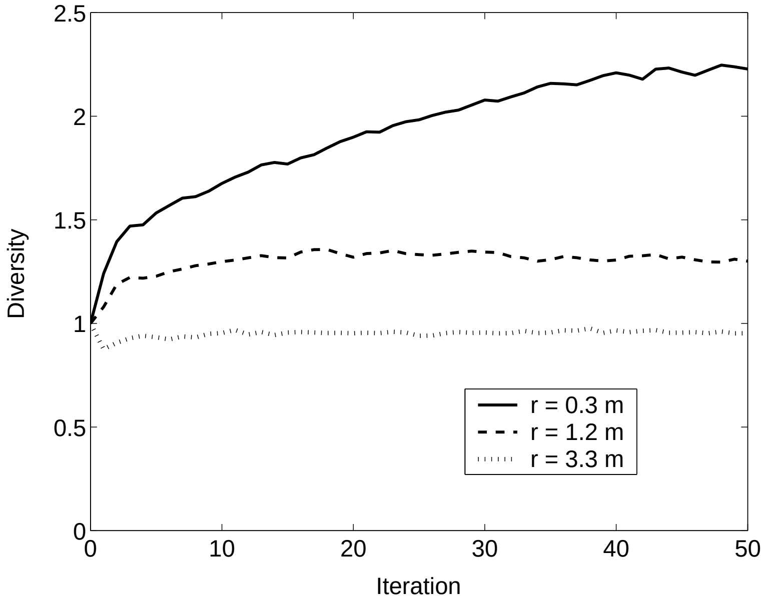


Fig. 8 Average population diversity over 100 runs for 0.3 m, 1.2 m, and 3.3 m communication range in Model B, normalized to a starting value of 1.0.

While the intermediate communication range had the best performances in Model B, all ranges achieved fairly high performance. The success of all these suggests that the effectiveness of the algorithm is not highly dependent on choosing an exact neighborhood size, making the choice of algorithm parameters quite robust. This is an important feature, as the communication range with real robots can vary due to obstruction and environmental effects.

6 Distributed Learning with Real Robots

While realistic simulation may give us valuable insight into the effectiveness of different robotic controllers and learning strategies, it is important to validate these approaches using actual robots in order to ensure that our virtual scenario is well-aligned with the real world. We therefore run parallel robotic learning experiments using real robots.

In the past, it has been quite rare for online robotic learning to be run completely on real robots. This was due to the excessive time required for full online learning experiments, which were often infeasible on robots with very limited battery life. A major exception to this was the work of Floreano & Mondada, 1996, where a tethered robot learned obstacle avoidance behavior after more than 60 hours of optimization. More often, hybrid learning is used, where most of the learning is done in realistic simulation, and then transferred onto real robots for the final portion (for example, see Miglino et al., 1995). This can offer a very good compromise between controller quality and learning time, but requires an accurate model of the environment to exist. Some offline learning techniques (such as apprenticeship learning or learning by imitation studied by Abbeel et al., 2008 and Billard & Matarić, 2001) can offer fast performance as well, but also require some sort of teacher and may not be appropriate for unknown/dynamic environments. By exploiting the speed-up from parallelism of a multi-robot system, however, full online robotic learning becomes feasible. This was accomplished in previous work using embodied evolution with 8 robots learning in parallel (Watson et al., 2002). We now use distributed PSO to do full robotic learning using 10 real Khepera III robots.

6.1 Experimental Setup

For our real robot experiments, we use the same environmental setup as was used in simulation; ten Khepera III robots operate in a 3.0 m $\times$ 3.0 m square arena (see Fig. 9). Robots are able to measure the performance of their own obstacle avoidance behavior on board by recording their wheel speeds and proximity sensor detections throughout an evaluation and plugging those values into the performance equation. Because robots are regularly bumping into walls and each other throughout the learning process, a foam bumper ring was attached around the periphery of each one (at a position where it did not interfere with the infrared proximity sensors) to dampen the impact from collisions and prevent permanent damage.

In our virtual simulations, after every evaluation, robots would be moved to random locations within the arena. This is clearly impossible with real-world robots. To approximate this effect, between each evaluation, a random speed is applied to each wheel of each robot for a duration of three seconds. While this is a poor approximation of a completely random new position, it should help somewhat to prevent bias on the next robot evaluation.

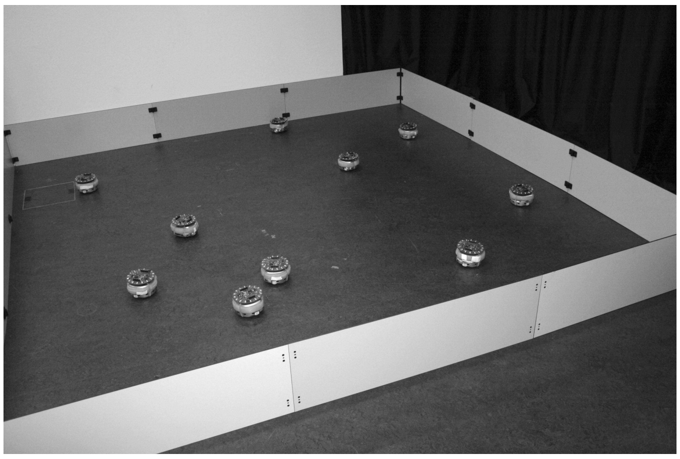


Fig. 9 Real Khepera III robots with relative positioning boards and bumper rings in their arena.

For communication-based PSO neighborhoods, particle neighbors are determined by distances between robots. In order to measure this, we utilize an on-board relative positioning system (see Pugh et al., 2009). This is a board which is connected atop the robot and regularly emits modulated infrared packets (at approximately 15 Hz). These packets can be detected by the relative positioning systems of other robots on several different receivers, and the Received Signal Strength Indication (RSSI) of each can be used to calculate both the range and bearing of the transmitting robot relative to the receiving robot's location. This allows robots to estimate the distance between themselves, with an error level of approximately 10% of the actual range. However, because infrared is used for positioning, this method is sensitive to occlusion. Therefore, robots will not detect each other if some other robot is in the way (this differs from our simulation, where communication could not be occluded).

For simulated robotic learning, robots in the group were considered to always be completely synchronized. In real-world multi-robot systems, this may not be a valid assumption, since robots may not begin running simultaneously or may not have a precise internal timer. Synchronicity is necessary in distributed PSO robotic learning, since robots must exchange particle information between evaluations. In order to achieve this, we exploit the three second random movement between evaluations as a period to share particle information with other robots via wireless communication (and range information via relative positioning). Using this approach, it is still necessary for robots to be loosely synchronized (to within one or two seconds of each other), but this is relatively easy to achieve with the speed of wireless communication.

On a Khepera III robot using a relative positioning board, the battery lifetime is approximately 50 minutes. While this is theoretically enough time to complete a full learning run, it is not enough to select the best solution and evaluate its performance. Additionally, variation in battery quality results in some batteries lasting for less time than others. For these reasons, it was necessary to design the distributed PSO robotic learning algorithm so that the swarm state (particle parameters and performance) was periodically saved on all robots. If a robot prematurely ran out of energy (or crashed for some other reason), the batteries could be replaced and the robots reset to resume optimization from that saved state.

6.2 Communication-Based Neighborhoods with Real Robots

We test the effectiveness of the standard neighborhood and communication-based neighborhoods described in Section 4 using real robots. Five trials are run using the *lbest* topology, Model A, and Model B with a communication radius of 1.0 m; the performances averaged over 5 runs are shown in Fig. 10. We see that all three approaches are able to achieve very good performance; in every single run, an effective obstacle avoidance approach is discovered. In most experiments, the generated behavior would be for the robot to move forward at full speed until it detected an obstacle with its front sensors, at which point it would consistently turn either right or left. This approach worked very well most of the time, but would occasionally cause two robots to become stuck together if they happened to collide at a certain angle. A second less common approach used the recursive nature of the ANN to achieve state switching; the robot would move forward at full speed until it detected an obstacle with its front sensors, at which point it would reverse its movement direction and go backwards at full speed, possibly turning slightly during the switch. This approach actually yielded higher performance, since by using both the front and back sensors for avoidance, the most active sensor has lower activation than if the robot always moves in the same direction. Additionally, reversing direction prevented robots from becoming stuck against each other nearly as often.

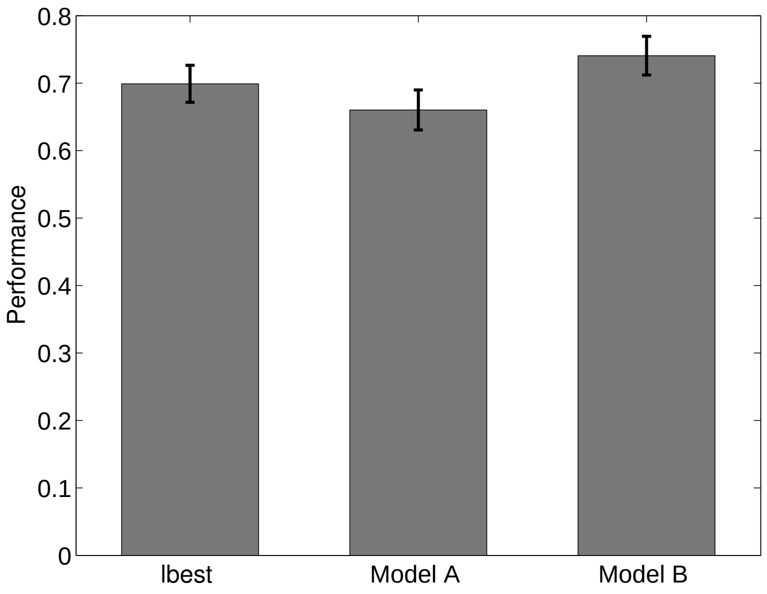


Fig. 10 Average of final best performances over 5 runs for different neighborhood models on real robots. Error bars represent standard error across runs.

When comparing the results of the different neighborhood topologies, we see that while Model B still outperforms the standard topology, Model A performs more poorly. One possible explanation for this is the difference in how robots are randomly moved between evaluations; in simulation, the random re-positioning allowed information to be spread more quickly throughout the swarm, while the small shift resulting from three seconds of movement with the real robots is not sufficient for robot neighbors to change. Model B still performs well because the communication radius is large enough that information is spread even with the small position shift. However, it is difficult to make definitive comparisons among the real results, as the number of runs is too low for small differences in performance to be significant.

6.3 Varying Communication Range with Real Robots

We also explore how different communication ranges affect the learning performance on real robots. We run experiments using the Model B particle neighborhood with communication ranges of 0.3 m, 1.0 m, and 3.3 m. The results of 5 runs can be seen in Fig. 11. Although we cannot state with confidence that the performance for any range is higher than any other, we do see the same trend observed in simulation (Fig. 6): performance is higher for an intermediate communication range than either a very low or very high one. The average performance remains high for all communication ranges.

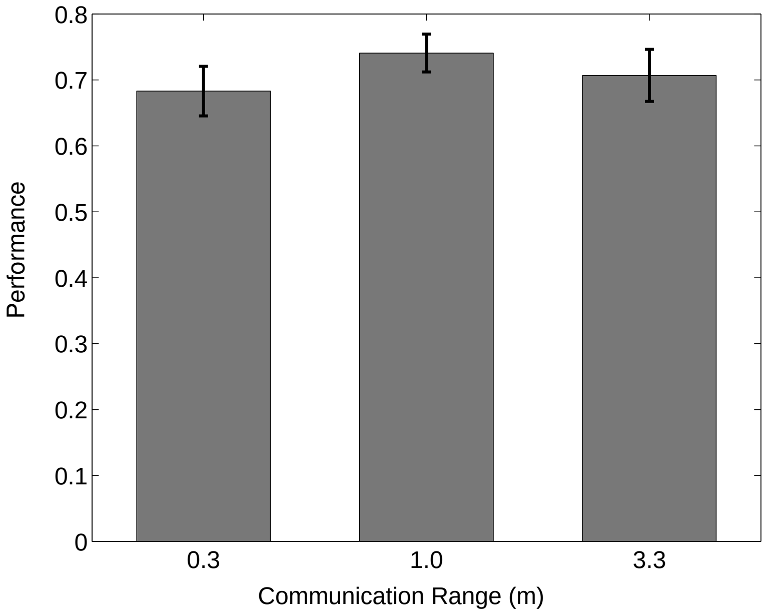


Fig. 11 Average of final best performances over 5 runs for different communication ranges in Model B with real robots. Error bars represent standard error across runs.

7 Conclusions

We have presented a distributed implementation of PSO for fast parallel learning in multi-robot systems. The technique allows the speed of online learning to be increased by using larger groups of robots, with only a marginal drop in learning performance. Using a particle neighborhood based on the limitations of robot communication, we achieve performance as good as (and possibly better than) that achieved with the standard approach. The communication-based neighborhood provides maximum performance when the communication range is limited, allowing robots to potentially save power. The efficacy of the distributed PSO robotic learning technique is clearly shown through multiple experiments using 10 real robots operating in parallel; these experiments show similar trends to those observed in simulation.

Perhaps the biggest difference between our simulated and real-world experiments was the random repositioning of robots after each evaluation. To get the most fair test of a controller, starting conditions should not be biased by previous actions, which is why robots were positioned completely randomly in simulation. With real robots, completely random positioning is not possible, and therefore the random motor speeds were applied as a limited approximation. Comparing the results of simulations with real experiments, we see that there was no penalty to the performance of the generated controllers for using this approximation (with the possible exception of the Model A neighborhood results). This suggests the learning process does not require strictly unbiased evaluations to achieve good results; it is possible that the technique would continue to be effective even if no random motion was executed between evaluation runs, something which might be a necessary constraint for certain applications. It would also be informative to recreate the random motor speed movement in simulation and systematically compare the results of the two approaches to quantitatively evaluate the impact of the change.

By increasing the robot group size from one to ten robots, we were able to decrease the behavior learning time with our distributed PSO algorithm by a factor of ten. However, using a particle population size of ten, there is no easy way to further decrease the time while maintaining only local interactions. It may be possible to use twenty robots to simultaneously evaluate the ten new particles and reevaluate the ten previous best particles, but this would require a global supervisor to manage the assignments of candidate solutions to different robots, as each robot is no longer fully in charge of a particle. Therefore, further increasing the number of robots using only local interactions would only allow us to increase the size of the population. It has yet to be explored how increasing the swarm size could affect the learning rate, and what type of neighborhood type would be most appropriate as the population continues to grow.

In this work, our robotic learning technique was run for a specific number of iterations to ultimately generate an effective controller; after the learning process, the best performing controller was selected and used indefinitely by all robots. This approach is only valid if the robots' environment is static. If the environment changes over time, any generated controller will likely become obsolete, as it will not make the necessary adjustments needed to maintain good performance. An alternate learning approach is to keep the learning technique running indefinitely and never select a single controller to be used by all robots. This would allow robots to optimize their behavior to the environment and automatically re-optimize whenever that environment changes. Preliminary results suggest that this may be an effective technique to achieve adaptive, high-performance behavior (Pugh & Martinoli, 2008), and further study is warranted.

While generating obstacle avoidance behavior is often used as a benchmark for robotic learning techniques (e.g., Floreano & Mondada, 1996; Kaelbling, 1993; Michels et al., 2005; Nordin & Bahnzaf, 1997; Ye et al., 2003), it is a single, relatively simple task; good performance in obstacle avoidance does not guarantee that the technique will allow robots to effectively learn other behaviors, particularly more complex ones. Previous work has shown that the technique is also effective for learning aggregation (Pugh & Martinoli, 2006) and target localization (Pugh & Martinoli, 2008) behavior. However, both of these behaviors are also quite simple, and it is likely that were the technique applied to more complex task (e.g., one requiring a much larger ANN), it would have difficulty finding the best solutions, simply because the high parameter dimensionality would result in a search space that is too large to explore effectively. This is a known limitation for many learning algorithms, and it would be useful to evaluate how well the distributed PSO technique is able to handle the challenge and whether traditional work-arounds (such as behavior decomposition) would be effective.

For obstacle avoidance behavior, using a communication-based particle neighborhood yielded good performance with the distributed PSO learning technique. It is not evident whether this result can be generalized to other behaviors; the geographical distribution of robots is highly dependent upon how they respond to their environment. A previous study (Pugh & Martinoli, 2006) showed that communication-based neighborhoods also yield high-performance results in aggregation behavior, indicating flexibility in the application of the approach. However, full generality should not be assumed until further studies have shown similar success on a variety of scenarios.

Our distributed online robotic learning technique achieves homogeneous learning in a robot team, with all robots collaborating to create a single, high-performing controller. Depending upon the scenario, heterogeneous behavior may be desirable, where different robots behave differently to achieve some beneficial synergy. While it does not result directly from our approach, the optimization of heterogeneous controllers could be accomplished with only minor adjustments to the approach. One possibility would be to divide the robot team into smaller sub-teams and run the learning algorithm on each sub-team separately. The disadvantage of this approach is that the heterogeneity of the system would need to be pre-established, which could limit its flexibility. An alternative would be for all robots to simultaneously optimize multiple controllers, with some (possibly adaptive) probability of selecting each of the behaviors at the start of operation (for example, robots could learn the weights of two different ANNs; at the start of each evaluation, they would select the first ANN with probability p and the second with probability $1 - p$ and use that ANN for the entire evaluation). However, it is not immediately evident whether the same information sharing approach would be as effective in this case. Both of these approaches would require the optimization of a much greater number of parameters, resulting in slower learning (which might offset the benefit of a heterogeneous team), and their viability should be explored through future experimentation.

8 Acknowledgements

The authors would like to thank the referees for all their helpful feedback. This work has been supported by a Swiss National Science Foundation grant (contract Nr. PP002-116913).

References

Abbeel, P., Dolgov, D., Ng, A., & Thrun, S. (2008). Apprenticeship Learning for Motion Planning with Application to Parking Lot Navigation. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* (pp. 1083-1090). Piscataway, NJ: IEEE Press.

Atkeson, C. G., Moore, A. W., & Schaal, S. (1997). Locally Weighted Learning for Control. *Artificial Intelligence Review, 11,* 75-113.

Balch, T. (1998). *Behavioral diversity in learning robot teams.* PhD Thesis, College of Computing, Georgia Institute of Technology, Atlanta, GA.

Billard, A. & Matarić, M. J. (2001). Learning human arm movements by imitation: Evaluation of a biologically-inspired connectionist architecture. *Robotics and Autonomous Systems, 37,* 145-160.

Bowling, M. & Veloso, M. (2003). Simultaneous adversarial multi-robot learning. In *Proc. of the Int. Joint Conf. on Artificial Intelligence* (pp. 699-704). Mahwah, NJ: Lawrence Erlbaum.

Di Chio, C. & Di Chio, P. (2007). EcoPS - a Model of Group-Foraging with Particle Swarm Systems. In *Proc. of the Euro. Conf. on Artificial Life, LNCS 4648* (pp. 685-695). Berlin: Springer.

Dorigo, M., Trianni, V., Şahin, E.,Groβ, R., Labella, T. H., Baldassarre, G., Nolfi, S., Deneubourg, J.-L., Mondada, F., Floreano, D., & Gambardella, L. M. (2004). Evolving Self-Organizing Behaviors for a Swarm-Bot. *Autonomous Robots, 17,* 223-245.

Eberhart, R. & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proc. of the Int. Symp. on Micro Machine and Human Science* (pp. 39-43). Piscataway, NJ: IEEE Press.

Floreano, D. & Mondada, F. (1996). Evolution of Homing Navigation in a Real Mobile Robot. *IEEE Trans. on Systems, Man and Cybernetics, Part B, 26,* 396-407.

Franklin, J. A., Mitchell, T. M., & Thrun, S. (1996). *Recent Advances in Robot Learning.* Boston, MA: Kluwer Academic Publishers.

Jatmiko, W., Sekiyama, K., & Fukuda, T. (2006). A PSO-based Mobile Sensor Network for Odor Source Localization in Dynamic Environment: Theory, Simulation and Measurement. In *Proc. of the IEEE Congress on Evolutionary Computation* (pp. 1036-1043). Los Alamitos, CA: IEEE Computer Society.

Kaelbling, L. P. (1993). *Learning in embedded systems.* Cambridge, MA: The MIT Press.

Kelly, I. D. & Keating, D. A. (1998). Faster learning of control parameters through sharing experiences of autonomous mobile robots. *Int. Journal of System Science, 29,* 783-793.

Kennedy, J. & Eberhart, R. (1995). Particle swarm optimization. In *Proc. of the IEEE Int. Conf. on Neural Networks* (pp. 1942-1948). Piscataway, NJ: IEEE Press.

Li, L., Martinoli, A., & Abu-Mostafa, Y. (2004). Learning and Measuring Specialization in Collaborative Swarm Systems. *Adaptive Behavior, Special issue on Mathematics and Algorithms of Social Interactions, 12,* 199-212.

Mahadevan, S. & Connell, J. (1991). Automatic Programming of Behavior-based Robots using Reinforcement Learning. In *Proc. of the Natl. Conf. on Artificial Intelligence* (pp. 768-773). San Francisco, CA: Morgan Kaufmann.

Matarić, M. J. (1994). Learning to Behave Socially. In *Proc. of the Int. Conf. on the Simulation of Adaptive Behavior* (pp. 453-462). Cambridge, MA: MIT Press.

Matarić, M. J. (2001). Learning in behavior-based multi-robot systems: Policies, models, and other agents. *Cognitive Systems Research, Special Issue on Multi-Disciplinary Studies of Multi-Agent Learning, 2,* 81-93.

Michel, O. (2004). Webots: Professional Mobile Robot Simulation. *Int. Journal of Advanced Robotic Systems, 1,* 39-42.

Michels, J., Saxena, A., & Ng, A. Y. (2005). High speed obstacle avoidance using monocular vision and reinforcement learning. In *Proc. of the Int. Conf. on Machine Learning* (pp. 593-600). New York, NY: ACM.

Miglino, O., Lund, H. H., & Nolfi, S. (1995). Evolving Mobile Robots in Simulated and Real Environments. *Artificial Life, 2,* 417-434.

Murciano, A., Millán, J. R., & Zamora, J. (1997). Specialization in multi-agent systems through learning. *Behavioral Cybernetics, 76,* 375-382.

Nehmzow, U. (2002). Learning in multi-robot scenarios through physically embedded genetic algorithms. In *Proc. of the Int. Conf. on the Simulation of Adaptive Behavior* (pp. 391-392). Cambridge, MA: MIT Press.

Nordin, P. & Bahnzaf, W. (1997). An On-Line Method to Evolve Behavior and to Control a Miniature Robot in Real Time with Genetic Programming. *Adaptive Behavior, 5,* 107-140.

Panait, L. & Luke, S. (2005). Cooperative Multi-Agent Learning: The State of the Art. *Autonomous Agents and Multi-Agent Systems, 11,* 387-434.

Parker, L. E. (1997). L-ALLIANCE: Task-Oriented Multi-Robot Learning in Behavior-Based Systems. *Advanced Robotics, 11,* 305-322.

Pugh, J., Zhang, Y., & Martinoli, A. (2005). Particle swarm optimization for unsupervised robotic learning. In *Proc. of the IEEE Swarm Intelligence Symposium* (pp. 92-99). Piscataway, NJ: IEEE Press.

Pugh, J. & Martinoli, A. (2006). Multi-Robot Learning with Particle Swarm Optimization. In *Proc. of the Int. Conf. on Autonomous Agents and Multiagent Systems* (pp. 441-448). New York, NY: ACM.

Pugh, J. & Martinoli, A. (2008). Distributed Adaptation in Multi-Robot Search using Particle Swarm Optimization. In *Proc. of the Int. Conf. on the Simulation of Adaptive Behavior, LNCS 5040* (pp. 393-402). Berlin: Springer.

Pugh, J. & Martinoli, A. (2009). An Exploration of Online Parallel Learning in Heterogeneous Multi-Robot Swarms. *Design and Control of Intelligent Robotic Systems, SCI 177* Chapter 7. (pp. 145-165). Berlin: Springer.

Pugh, J., Raemy, X., Favre, C., Falconi, R., & Martinoli, A. (2009). A Fast On-Board Relative Positioning Module for Multi-Robot Systems. *IEEE/ASME Transactions on Mechatronics, Focused Section on Mechatronics in Multi Robot Systems,* to appear.

Smart, W. D. & Kaelbling, L. P. (2002). Effective reinforcement learning for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation* (pp. 3404-3410). Piscataway, NJ: IEEE Press.

Stone, P. (1998). *Layered Learning in Multi-Agent Systems.* PhD Thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Stone, P. & Veloso, M. (2000). Multiagent Systems: A Survey from a Machine Learning Perspective. *Autonomous Robots, 8,* 345-383.

Watson, R. A., Ficici, S. G., & Pollack, J. B. (2002). Embodied Evolution: Distributing an Evolutionary Algorithm in a Population of Robots. *Robotics and Autonomous Systems, 39,* 1-18.

Ye, C., Yung, N. H. C., & Wang, D. (2003). A fuzzy controller with supervised learning assisted reinforcement learning algorithm for obstacle avoidance. *IEEE Trans. on Systems, Man, and Cybernetics, Part B, 33,* 17-27.