Multi-Robot Learning with Particle Swarm Optimization

Jim Pugh and Alcherio Martinoli
Swarm-Intelligent Systems Group
École Polytechnique Fédérale de Lausanne
1015 Lausanne, Switzerland
{jim.pugh,alcherio.martinoli}@epfl.ch

ABSTRACT

We apply an adapted version of Particle Swarm Optimization to distributed unsupervised robotic learning in groups of robots with only local information. The performance of the learning technique for a simple task is compared across robot groups of various sizes, with the maximum group size allowing each robot to individually contain and manage a single PSO particle. Different PSO neighborhoods based on limitations of real robotic communication are tested in this scenario, and the effect of varying communication power is explored. The algorithms are then applied to a group learning scenario to explore their susceptibility to the credit assignment problem. Results are discussed and future work is proposed.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics—*Autonomous vehicles*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Algorithms, Experimentation

Keywords

particle swarm optimization, unsupervised learning, multi-robot systems

1. INTRODUCTION

Designing even simple behaviors for robots that are 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 optimal performance can be very challenging. Unsupervised robotic learning allows for automated design of efficient, robust controllers, which saves much design time and effort. Unsupervised learning is also useful for allowing robots to adapt to situations where the task/environment is unknown beforehand or is constantly changing.

Genetic Algorithms (GAs) are a very common method of accomplishing machine learning and optimization. Candidate solutions to a problem are modeled as members of a population, and breeding (selection and crossover) and mutation are applied to "parents" (high performing solutions) in the population to generate "children" (new candidate solutions). GA can be used to shape an Artificial Neural Network (ANN) controller by using the parameter set as the weights, and the evaluative function as a measure of the performance of a desired robot behavior.

Particle Swarm Optimization (PSO) is a promising new optimization technique which models a set of potential 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 can also be used to evolve ANN robotic controllers.

Both GA and PSO use groups of interacting virtual agents in order to achieve their optimization. In collective 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 several virtual agents, which it would need to evaluate at each iteration. After each set of evaluations, the robots would communicate to share the fitness information needed to progress to the next iteration of the algorithm. By running the algorithms 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 would decrease the number of required controller 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. In contrast, the GA population manager must have knowledge of the entire population in order to implement standard breeding techniques, which prevents the same scalable technique from being applied.

In this paper, we explore the effectiveness of using a modified version of PSO on groups of realistically simulated robots performing distributed unsupervised learning. At the

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.

maximum group 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 unique particle. We test how the performance is affected if we adapt the standard PSO neighborhood structure to more closely model what is possible in a real robot group with limited communication abilities. Section 2 provides some background on GA, PSO, unsupervised robotic learning, and multi-robot learning. Section 3 examines how the effectiveness of distributed unsupervised 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. Section 6 applies the algorithms to a group learning task, to see how affected they are by the credit assignment problem and to see how the communication-based neighborhoods fare in different scenarios. Section 7 discusses the implications of the results and suggests future work, and Section 8 concludes.

2. BACKGROUND

Genetic algorithms were originally developed in the 1960s by John Holland. The algorithms are inspired by evolution, where the fittest members of a population tend to reproduce more often than the less fit members. Candidate solutions are modeled as a population of "chromosomes". At each iteration of the algorithm, a new population is generated from the previous one. Selection of the parents of the new generation is implemented using one or more of several schemes, such as elitism (using only the top performing members of the population), Roulette Wheel sampling (stochastically choosing parents with weight proportional to performance), and rank selection (ranking chromosomes from best to worst and stochastically choosing parents with weight proportional to the rank). After parents have been chosen, crossover between the parents can occur with some probability (each chromosome is split into two, and children use one part from one parent and the other part from the other). This allows positive aspects from different chromosomes to be merged into a single chromosome. Last, mutation is applied, where each element of the chromosome may have its value randomly changed with some probability. This provides a random local search, which allows solutions to continue to improve beyond the genetic diversity that was available in the original population ([6], [15]).

The original PSO method was developed by James Kennedy and Russel Eberhart ([9], [3]). Every particle in the population begins with a randomized position ($x_{i,j}$) and randomized velocity ($v_{i,j}$) 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 ($x^*_{i,j}$). 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 particles with the closest array indices as neighbors 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 which 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 uniformly-distributed random number in $[0, 1]$.

PSO has been shown to perform as well as or better than GA in several instances. Eberhart and Kennedy found PSO performs on par with GA on the Schaffer f6 function [3, 9]. In work by Kennedy and Spears [10], a version of PSO outperforms GA in a factorial time-series experiment. Fourie showed that PSO appears to outperform GA in optimizing several standard size and shape design problems [5].

Unsupervised learning describes learning scenarios where there is no external entity which decides upon the training set inputs for the learning agent(s). Rather, inputs are generated dynamically as the agents interact with their environment. This is as opposed to supervised learning, where the inputs are generated/collected first and then used repeatedly. In supervised learning, the accuracy of the system at each iteration is usually decided by an external "teacher" evaluating the system output. The pre-defined inputs are split into two separate sets, one for training the system and the other for testing the performance. Supervised learning tends to be easier than unsupervised, as the data does not change between iterations of the algorithm and can be preselected to avoid using unusual or particularly noisy data points. However, supervised learning is not possible in situations where the input data to the system depends on the current state of the learning agent; this is the case for online robotic learning, since the robot's movements affect what its sensors will perceive.

Evolutionary algorithms have been used extensively for unsupervised learning of robotic behavior. A good survey of the work is given in [12]. More specifically, standard GA has been shown to be effective in evolving simple robotic controllers [4], and modified noise-resistant versions of both GA and PSO were shown to achieve very good performance on simulated unsupervised robotic learning, outperforming the standard versions of the algorithms [18].

In collective robotics, many desired behaviors result in strong interactions between robots in the group, and the actions of one robot can significantly impact the performance of another. If a behavior is being learned and each robot is evaluating a different controller, this can give rise to the "credit assignment" problem, where robots do not know whether a good/bad fitness score was due to its own performance or to that of other robots. This effect can be particularly pronounced in cases where robots do not explicitly share their intentions through communication channels and can severely hamper the learning process. The credit assignment problem can arise in two different scenarios. The

first is when robots are learning individual behaviors, but the performance of their behavior can be impacted by the actions of other robots in the group, causing an inaccurate fitness evaluation. The second is when robots are learning a group behavior with a single collective fitness. Often, it is not easy to decompose the fitness value to know which robots positively affected the performance and which didn't. An elegant way to bypass the credit assignment problem in these cases is to use "homogenous" learning (as opposed to "heterogeneous" learning), where all the robots simultaneously evaluate the same controller and therefore contribute equally to the collective score on average. While this drastically slows the learning process, it is the only way of achieving good results in some highly stochastic scenarios, such as those investigated in Hayes et al. [7].

Multi-robot learning has been used and explored in various ways. Matarić studied mechanisms to encourage individual agents in a group to act in ways to help the group performance [11]. Multi-robot learning using several methods in a wide variety of scenarios has been explored ([2], [20]). Techniques for increasing individual learning speed via multi-robot learning were studied in [8] and [13]. A modified version of a genetic algorithm has been embedded onto a 2-robot system to allow for distributed parallel learning [17]. Particle swarm optimization has thus far not been used for learning in scalable multi-robot systems.

3. VARYING THE ROBOTIC GROUP SIZE

In Pugh et al. [18], unsupervised 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 unsupervised learning on much larger robotic groups, where the evolving candidate solutions are distributed throughout the group to achieve faster learning.

3.1 Experimental Setup

We use the noise-resistant GA and PSO algorithms from [18]. GA uses elitism to select the best half of the population as the parent set, and then applies Roulette Wheel sampling to replenish the missing chromosomes. PSO uses a local neighborhood in a ring topology with one neighbor on each side. At every iteration, these algorithms reevaluate their previous best locations and parent sets for PSO and GA, respectively, combining the new fitness value with previous ones to get a more accurate measure of the actual fitness. Although this requires twice as many fitness evaluations at each iteration as their standard counterparts, this technique prevents noisy fitness evaluations from severely disrupting the learning process and gives much better results given the same amount of computational time.

We modify the noise-resistant PSO algorithm from its original form slightly: when updating the neighborhood best particle for particle i ($x^*_{i'}$), the neighborhood best is only changed if the new neighborhood best particle ($x^*_{i''}$) has higher fitness than the previous best location of the current particle. In other words

$$x^*_{i'} = x^*_{i''} \text{ only if } fitness(x^*_{i''}) > fitness(x^*_i)$$

It should be noted that this excludes the previous best location of a particle from ever being its neighborhood best. We found that this modification dramatically improved the performance of the learning over the standard update method. Although we are not certain why this is the case, it may be that the new technique encourages diversity by ensuring that every particle will have two distinct locations to which it is attracted.

The parameters for the algorithms are given in Table 1.

Table 1: GA and PSO Parameters for Unsupervised Learning

GA		PSO	
Population Size	20	Population Size	20
Crossover Probability	0.6	pw	2.0
Mutation Probability	0.15	nw	2.0
Mutation Range	[-5.0, 5.0]	w	0.6

We use Webots, a realistic simulator, for our robotic simulations [14], using the Khepera robot model [16]. The robot(s) operate in a 2.0 m x 2.0 m square arena (see Fig. 1). The robotic controller is a single-layer discrete-time artificial neural network of two neurons, one for each wheel speed, with sigmoidal output functions. The inputs are the eight infrared proximity sensors (six in front, two in back), as well as a recursive connection from the previous output of the neuron, lateral inhibitions and bias values (see Fig. 2), giving us 22 weights total. Sensors have a maximum range of 5.0 cm, and sensor output varies linearly from 0.0 at maximum range to 5.12 at minimum range (0.0 cm) with 10% noise. Slip noise of 10% is applied to the wheel speed. The time step for neural updates is 128 ms. We use the fitness function used in [18]. The fitness function is given by:

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

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, turn as little as possible, and spend little time near obstacles, respectively. The terms are normalized to give a maximum fitness of 1. The evaluation period of the fitness tests for these experiments is 480 steps, or approximately 60 seconds. Between each fitness test, the position and bearing of the robots are randomly set by the simulator to ensure the randomness of the next evaluation.

We test for robot group sizes of 1, 2, 5, 10, and 20 for 100 iterations of each algorithm. Since learning is being done in parallel, this has a significant effect on the simulated time needed ($\sim$67 hours for 1 robot compared to $\sim$3 hours for 20). In the case of 20 robots, each robot effectively contains a single candidate solution in the algorithm population.

3.2 Results

A comparison of the average fitnesses can be seen in Fig. 3. The progress of the average population fitness throughout the learning process for a 20-robot group can be seen in Fig. 4. There is no significant change in the performances of the algorithms for different robot group sizes, indicat-

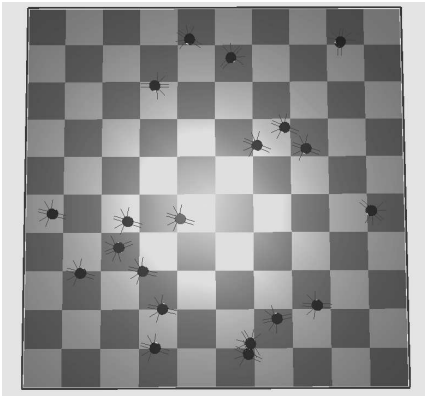

Figure 1: Robot arena with Khepera robots. Lines protruding from Kheperas represent proximity sensors.

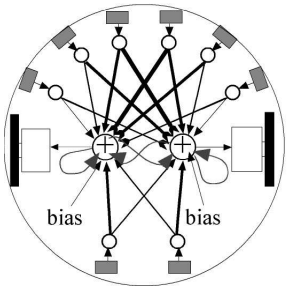

Figure 2: Depiction of the artificial neural network used for the robot controller. Grey boxes represent proximity sensor inputs and white boxes on the sides represent the motor outputs. Curved arrows are recurrent connections and lateral inhibitions.

ing that this technique is quite scalable. Although GA has initially faster convergence, the performance is noticeably lower than that of PSO for all group sizes. This was due to GA converging to poor solutions a large fraction of the time. A likely cause of this is the small population size (20 agents here as opposed to 60 in [18]), which does not provide enough genetic diversity for GA in this scenario, while PSO, though slower, is able to converge well with much smaller population sizes.

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.

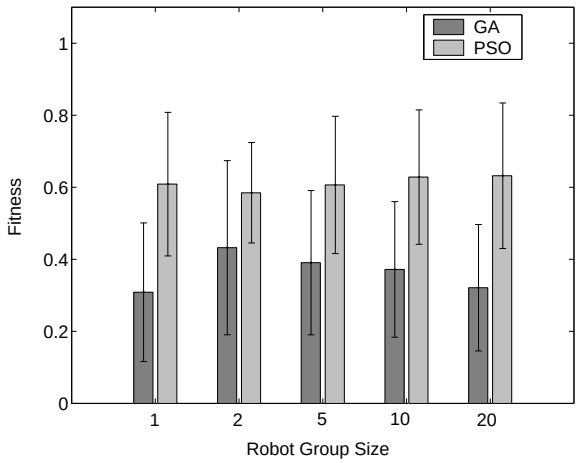

Figure 3: Average of final best performances over 20 evolutions for GA and PSO with different robotic group sizes. Error bars represent standard deviation across evolutionary runs.

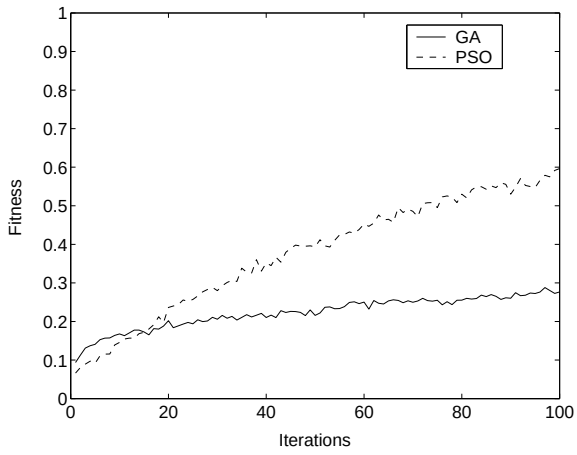

Figure 4: Average performance of population over 20 evolutions for GA and PSO with 20-robot groups.

4.1 Experimental Setup

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

Model 1: Each robot contains one particle. At the end of each fitness evaluation, the robot selects the two 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. 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 2: Each robot contains one particle. At the end of each fitness 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 results in a

variable number of neighbors, as the robot may be close to very few or very many robots randomly. However, it is perhaps more realistic than Model 1, since for very sparse robot distributions, there may be fewer than two other robots in close proximity at times.

We compare the performance of the original neighborhood topology to the two new models, using $r = 40$ cm, for a group of 20 robots. We use the setup previously described.

4.2 Results

A comparison of the average fitnesses is shown in Fig. 5. Both new neighborhood models achieve slightly better fitness than the original. This suggests that random neighborhood selection at each iteration is marginally superior to the fixed ring topology. The good performance of Model 2 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 unsupervised learning in a realistic multi-robot system.

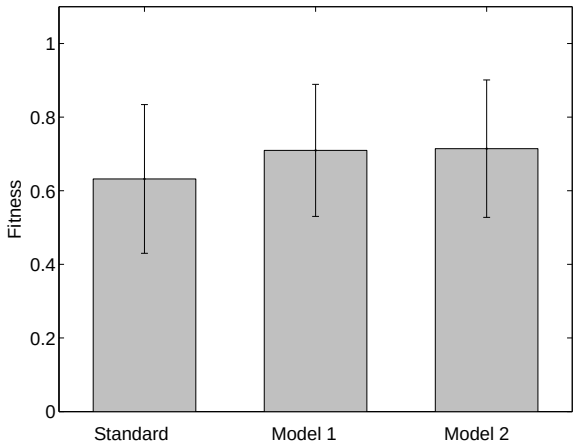

Figure 5: Average of final best performances over 20 evolutions for different neighborhood models. Error bars represent standard deviation across evolutionary runs.

5. VARYING COMMUNICATION RANGE

We now explore the effects of varying the communication range used in Model 2. 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.

5.1 Experimental Setup

We use communication ranges of 10 cm, 20 cm, 40 cm, 80 cm, and 160 cm. The expected number of robots within communication range are given in Table 2, assuming a uniformly random distribution of robots within the arena. We therefore go from almost no interparticle communication to almost full interparticle communication.

5.2 Results

The average fitnesses for different communication ranges can be seen in Fig. 6. The progress of the average population fitness throughout the learning process for 10 cm, 40 cm, and 160 cm can be seen in Fig. 7. Both very high and very low communication ranges achieve fairly poor performance, while the intermediate ranges all achieve fairly good results.

Table 2: Expected Number of Neighboring Particles

r (cm)	Expected Number of Neighbors
10	0.14
20	0.54
40	2.0
80	6.5
160	16

Failure of low communication range is due to not enough information being exchanged between particles; particles end up almost exclusively using their own personal best position for learning, which causes extremely 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 often prematurely converge on a solution which did not have particularly high performance. This indicates that a global neighborhood is actually detrimental to finding very good solutions, and we therefore gain no benefit whatsoever by expanding our communication range beyond a certain point.

Both communication ranges of 40 cm and 80 cm (corresponding to average neighborhood sizes of 2.0 and 6.5 particles respectively) achieved very high fitness. Even a communication range of 20 cm, corresponding to 0.54 neighbors on average, achieved good fitness. The success of all these suggests that the effectiveness of the algorithm is not highly dependent on choosing an exact neighborhood size, making the algorithm parameters quite flexible. This is an important feature, as the communication range with real robots can vary due to obstruction and environmental effects.

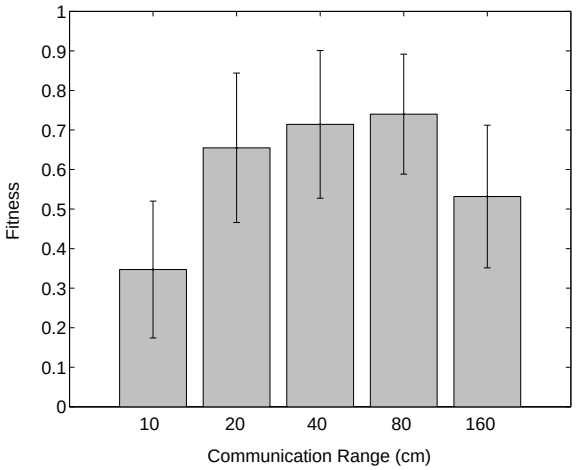

Figure 6: Average of final best performances over 20 evolutions for different communication ranges in Model 2. Error bars represent standard deviation across evolutionary runs.

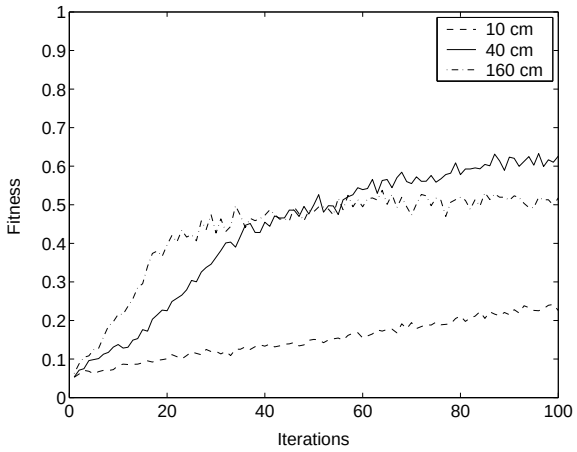

Figure 7: Average performance of population over 20 evolutions for 10 cm, 40 cm, and 160 cm communication range in Model 2.

6. GROUP LEARNING AND CREDIT ASSIGNMENT

Obstacle avoidance is a largely single-robot behavior. The observations and actions of other robots do not impact a robot's performance, except in having to avoid robots which move into its path. We wish to explore how susceptible our algorithms are to the credit assignment problem by evolving aggregation, a behavior whose success is highly dependent on the coordinated actions of many agents in the group.

6.1 Experimental Setup

We endow the Khepera robots with the ability to detect the relative positions of other nearby robots. This measurement is completely independent of any global coordinate system, and is given solely by where other robots are from the detecting robot's point of view (for example, range, the distance to the other robot, and bearing, the angular offset from the detecting robot's forward direction). This is a capability common to many robots working in collective scenarios, and can be accomplished with fairly simple systems (e.g., [19]). We add zero-mean Gaussian noise to the range and bearing, with range noise standard deviation equal to 10% of the range value and bearing noise standard deviation equal to 0.1 radians. We assume our relative positioning system is not susceptible to occlusions.

We expand the inputs to our artificial neurons to include relative positioning information. Because the number of robots within relative positioning range may vary, we use the center of mass of all detected robots as the input values. This is represented as x and y, where x is the forward-back displacement of the center of mass and y is the left-right displacement from the robot's point of view. This increases our total neural weights to 26.

The fitness value we use for this scenario is given by

$$F(i) = \frac{rob_{RP}(i)}{rob_{tot}}$$

where $F(i)$ is the fitness of robot i, $rob_{RP}(i)$ is the number of robots within relative positioning range of i, and rob_{tot} is the total number of robots. Therefore, a robot is rewarded for having the maximum number of other robots within relative proximity range at the end of a run.

The progress of evolving aggregation behavior may be susceptible to the first type of credit assignment problem described in Section 2, where robots use individual fitness values which can be impacted by the actions of other robots in the group, causing inaccurate evaluations. We therefore wish to compare the performance of our normal heterogeneous algorithms to homogenous algorithms, an established method of overcoming the credit assignment problem where all robots use the same controller at each evaluation. We generate a group fitness for each run by averaging all the individual fitness values obtained:

$$F_g = \frac{1}{rob_{tot}} \sum_i F(i)$$

Because the individual and group fitness functions are well-aligned, this allows us to compare their performances in a very fair manner. While homogenous learning will drastically slow the algorithm speed since we can no longer evaluate controllers in parallel, it will immediately provide a very noise-free estimation of the effectiveness of the solution, something which may not be available in the heterogeneous case (e.g., a robot may have a very good controller, but achieves poor performance because no other robot is aggregating well).

We use an unbounded arena with 20 Khepera robots for our setup. At each evaluative run, the Kheperas are distributed randomly in a 2m x 2m square. The evaluation lasts 10 simulated seconds, and the fitness is measured at the end. Robots are capable of sensing other robots within 80 cm of them. The arena can be seen in Fig. 8.

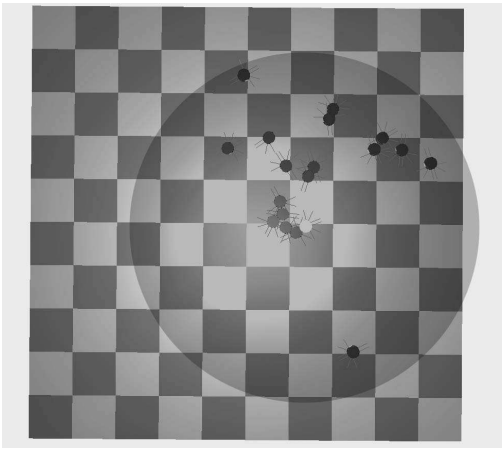

Figure 8: Aggregation arena with Khepera robots aggregating. The dimmed circle represents the relative positioning range of the white robot. All robots within this range are detectable.

We use our noise-resistant GA and PSO (with ring topology) algorithms and homogenous versions of these algorithms (HGA and HPSO, respectively), as well as the Model 1 and Model 2 neighborhood versions of the PSO algorithm. All algorithms have the same parameters used previously. Model 2 uses $r = 40$ cm. We run 100 iterations of the heterogeneous algorithms. Because we are using 20 robots, homogenous versions of the algorithms progress 20 times slower than the heterogeneous versions. We therefore run only 5 iterations

of these algorithms to match the number of evaluative runs.

6.2 Results

The final performance of all algorithms can be seen in Fig. 9. The progression of GA, PSO, HGA, and HPSO over the evolution can be seen in Fig. 10. All algorithms achieved good results in this scenario. For GA, the heterogeneous algorithm performed slightly worse than the homogenous version, while heterogeneous PSO performed as well as homogenous PSO on average, though with a higher standard deviation. This suggests that PSO may be less susceptible to the credit assignment problem than GA. However, as very few iterations were performed with the homogenous algorithms, it is likely HPSO would be able to achieve superior performance in longer runs. If we observe the progression of the algorithms throughout the learning process, GA initially improves more quickly, but levels off, while PSO continues to improve throughout. GA could therefore be preferable to PSO if we are only able to run very few iterations.

While the best final solutions from homogenous GA and PSO achieved very similar performances, the average population fitness for HGA was much higher than the average population fitness for HPSO throughout the learning process. This was observed in [18] and is likely due to the GA population containing much less variation than the PSO population, as the local neighborhood in PSO maintains diversity and no mechanism is present in GA to accomplish this. This diversity is likely what allows PSO to continue improving after the GA population converges on some solution.

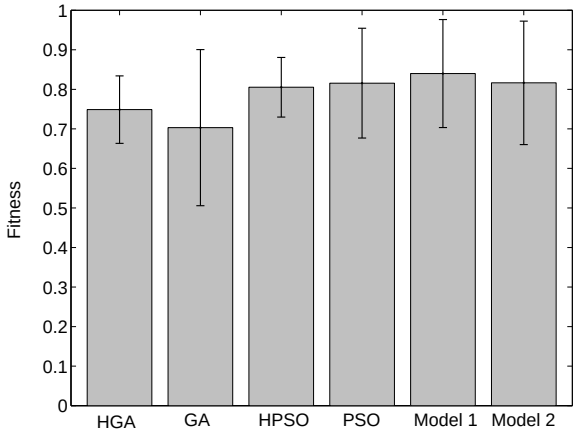

Figure 9: Average of final best performances in aggregation over 20 evolutions. HGA and HPSO are homogenous versions of the GA and PSO algorithms. Model 1 and Model 2 are PSO with neighborhoods described in Section 4. Error bars represent standard deviation across evolutionary runs.

Model 1 and Model 2 neighborhoods again achieved performances comparable to the standard ring topology, in spite of the very different distribution of robots in aggregation (high performing robots will be clustered together, while low performing robots may have traveled very far from the group center). The fact that these models continue to perform well is a good indication that neighborhoods based on limited communication capabilities of real robots can continue to function in a variety of scenarios.

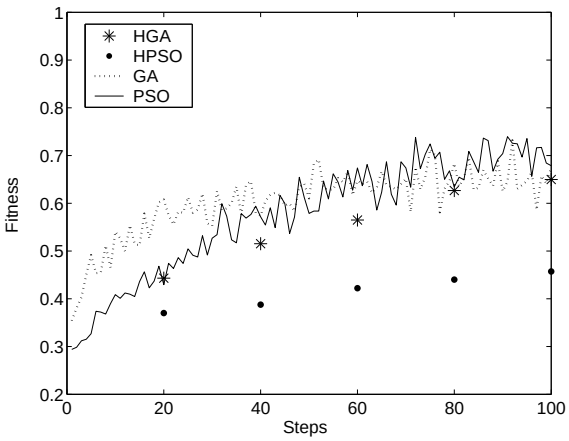

Figure 10: Average performance of population in aggregation over 20 evolutions. HGA and HPSO are the homogenous versions of the GA and PSO algorithms.

7. DISCUSSION AND OUTLOOK

Although PSO far outperformed GA in Section 2 of this paper, we suspect this is due almost exclusively to the small population size used. Indeed, the good performance of GA in Section 6 indicates that it can perform comparatively to PSO, and even better for fewer iterations. However, because of the different ways in which the population is managed, we would need to modify GA much more heavily in order to allow it to function distributedly with limited communication. Although this is certainly possible (e.g., as was done in [17]), the fundamental changes to the algorithm structure make it much more likely that we will lose the useful dynamics of the algorithm, as compared to PSO, where very few modifications are needed.

By increasing the robot group size from 1 to 20 robots, we were able to decrease the behavior learning time by a factor of 20. However, using a particle population size of 20, there is no easy to further decrease the time while maintaining only local interactions. It may be possible to use 40 robots to simultaneously evaluate the 20 new particles and reevaluate 20 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 population size could effect the convergence time.

In our model, the progress of all the robots was synchronized (i.e., fitness evaluations began and ended at the same time). In real-world multi-robot scenarios, this often isn't the case. Therefore, it would not make sense to exchange particle information only at the end of an evaluation, as the difference in time between robots could cause major delays. A simple alternative would be to exchange particle information on the previous evaluation during the evaluation itself. This is a rather minor modification to the algorithm, and we predict it will not significantly impact the performance. In fact, robots moving about during the evaluation would likely be exposed to more robots in close proximity, which

may allow them to further decrease their communication range while maintaining the same number of neighbors.

The model we use for communication in this paper is omnidirectional, immune to obstruction and error free. This corresponds to using radio transmissions in an open noiseless environment. In the real world, many other communication methods may be preferred or required. Infrared transmissions are often directional and could be blocked by other robots or by environmental obstacles. Radio could be blocked by large obstacles in some environments. Many types of communications may be susceptible to errors in noisy environments. The performance of the algorithm in these scenarios is thus far unexamined.

The scenario we used for testing susceptibility to the credit assignment problem only explored a small portion of the problem. It would be interesting to apply the algorithms to other scenarios, such those requiring specialization amongst robots or having less well-aligned group and individual fitness functions.

8. CONCLUSION

A modified version of the Particle Swarm Optimization algorithm was tested for unsupervised learning in groups of robots. The algorithm maintained good performance for groups of robots of various sizes. In the case of assigning a single unique particle to each robot, the performance was further improved by using PSO neighborhoods based on the limited communication abilities of real-world robots. Varying the communication range demonstrated that there is no benefit to communicating farther than a certain distance, and that the algorithm maintains high performance over a large variation of range. Applying the algorithm to a group learning task showed that it is able to overcome the credit assignment problem and that communication-based neighborhoods can perform well for non-uniform robot distributions. Implications of the results are examined and future research is suggested.

9. ACKNOWLEDGEMENTS

Jim Pugh and Alcherio Martinoli are currently sponsored by a Swiss NSF grant (contract Nr. PP002-68647).

10. REFERENCES

[1] Antonsson E. K, Zhang Y., & Martinoli A. "Evolving Engineering Design Trade-Offs". Proc. of the ASME Fifteenth Int. Conf. on Design Theory and Methodology, September 2003, Chicago, IL.

[2] Balch, T. *Behavioral diversity in learning robot teams.* PhD Thesis, College of Computing, Georgia Institute of Technology, 1998.

[3] Eberhart, R. & Kennedy, J. "A new optimizer using particle swarm theory" Proc. of the Sixth Int. Symposium on Micro Machine and Human Science, MHS '95, 4-6 Oct 1995, pp. 39-43.

[4] Floreano, D. & Mondada, F. "Evolution of Homing Navigation in a Real Mobile Robot" Systems, Man and Cybernetics, Part B, IEEE Transactions on, Vol. 26, No. 3, Jun 1996, pp. 396-407.

[5] Fourie, P. C. & Groenwold, A. A. "The particle swarm optimization algorithm in size and shape optimization" Struct. Multidisc. Optim., 2002, Vo. 23, pp. 259-267.

[6] Goldberg, D. E. *Genetic Algorithms in Search, Optimization & Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[7] Hayes, A. T., Martinoli, A. & Goodman, R. M. "Swarm Robotic Odor Localization: Off-Line Optimization and Validation with Real Robots", Special Issue on Biological Robots, D. McFarland, editor, Robotica, 2003, Vol. 21, pp. 427-441.

[8] Kelly, I. D. & Keating, D. A. "Faster learning of control parameters through sharing experiences of autonomous mobile robots" Int. Journal of System Science, 1998, Vol. 29, No. 7, pp. 783-793.

[9] Kennedy, J. & Eberhart, R. "Particle swarm optimization" Neural Networks, 1995. Proceedings., IEEE International Conference on, Vol.4, Iss., Nov/Dec 1995, pp. 1942-1948.

[10] Kennedy, J. & Spears, W. M. "Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator" in Proceedings of IEEE International Conference on Evolutionary Computation, Anchorage, May 1998, pp. 78-83.

[11] Matarić, M. J. "Learning to Behave Socially" In Proc. of the 3rd Int. Conf. on Simulation and Adaptive Behaviors - From animals to animats 3, 1994, pp. 453-462.

[12] Matariíc, M. J. & Cliff, D., "Challenges in evolving controllers for physical robots", Robot. and Autonomous Syst., 1996, Vol. 19, No. 1, pp. 6783.

[13] Matarić, M. J. "Learning in behavior-based multi-robot systems: Policies, models, and other agents" Special Issue on Multi-disciplinary studies of multi-agent learning, Ron Sun, editor, Cognitive Systems Research, 2001, Vol. 2, No. 1, pp. 81-93.

[14] Michel, O. "Webots: Professional Mobile Robot Simulation" Int. J. of Advanced Robotic Systems, 2004, Vo. 1, pp. 39-42.

[15] Mitchell, M. *An Introduction to Genetic Algorithms.* MIT Press, Cambridge, MA, 1996.

[16] Mondada, F., Franzi, E. & Ienne, P. "Mobile robot miniaturisation: A tool for investigation in control algorithms" Proc. of the Third Int. Symp. on Experimental Robotics, Kyoto, Japan, October, 1993, pp. 501-513.

[17] Nehmzow, U. "Learning in multi-robot scenarios through physically embedded genetic algorithms" In Proc. of the 7th Int. Conf. on the Simulation of Adaptive Behavior: From animals to animats, 2002, pp. 391-392.

[18] Pugh, J., Zhang, Y. & Martinoli, A. "Particle swarm optimization for unsupervised robotic learning" Swarm Intelligence Symposium, Pasadena, CA, June 2005, pp. 92-99.

[19] Pugh, J. & Martinoli, A. "Relative Localization and Communication Module for Small-Scale Multi-Robot Systems", Proc. of the IEEE International Conference on Robotics and Automation, Miami, Florida, USA, May 15-19, 2006.

[20] Stone, P. *Layered Learning in Multi-Agent Systems.* PhD Thesis, School of Computer Science, Carnegie Mellon University, 1998.