Distributed Adaptation in Multi-Robot Search using 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 present an adaptive strategy for a group of robots engaged in the localization of multiple targets. The robotic search algorithm is inspired by chemotaxis behavior in bacteria, and the algorithmic parameters are updated using a distributed implementation of the Particle Swarm Optimization technique. We explore the efficacy of the adaptation, the impact of using local fitness measurements to improve global fitness, and the effect of different particle neighborhood sizes on performance. The robustness of the approach in non-static environments is tested in a time-varying scenario.

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 near-optimal performance can be very challenging. Unsupervised robotic learning allows for automated design of efficient, robust controllers, which saves much design time and effort. Learning is also essential for allowing robots to adapt to situations where the task/environment is unknown beforehand or is constantly changing.

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. PSO achieves optimization using three primary principles: evaluation, where quantitative fitness can be determined for some particle location; comparison, where the best performing location for some particle can be selected out of multiple possibilities; and imitation, where the qualities of better particles are mimicked by others. The algorithm can be used to evolve parameters for robotic controllers.

In the field of robotics, locating targets within an unknown environment is a task well-suited to mobile robots. Robots can be equipped with sensors to detect targets and programmed to explore the area in search of their goals. The automated nature of this approach may save much time and effort as compared

* Jim Pugh and Alcherio Martinoli are currently sponsored by a Swiss NSF grant (contract Nr. PP002-116913).

to other search methods. Performance may be further improved by using multiple robots in parallel, which will decrease the time needed to complete the search task and increase robustness to failures of individual robots. Robotic search is especially preferable when the area is hazardous or inaccessible to humans (for example, finding victims in a disaster area [10]).

In the PSO algorithm, groups of virtual agents interact in order to achieve optimization. In collective robotics, groups of robots interact to accomplish their goals. It is therefore possible to make a one-to-one parallel between particles and robots and to implement PSO in a distributed fashion for learning in multi-robot systems. Each robot is responsible for a single particle, which it evaluates at each iteration. After each evaluation, the robots communicate to share the fitness information needed to progress to the next iteration of the algorithm. By running the algorithms in this fashion, we need no external supervisor to oversee the learning process, and the speed of learning is significantly improved, as many robots evaluating in parallel decreases the number of required controller evaluations and therefore decreases the total learning time.

2 Background

The original PSO method was developed by James Kennedy and Russell Eberhart [12]. 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

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

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 $[0, 1)$.

Unsupervised robotic learning has been studied extensively in the past, including some focus on multi-robot learning. Several multi-robot learning methods were shown to work in a wide variety of scenarios [1] [21]. Techniques for increasing individual learning speed via multi-robot learning were studied in [11] and [14]. Recently, PSO has been used for unsupervised learning of robotic behaviors by evolving weights for Artificial Neural Networks (ANNs), both in the case of single-robot learning [18] and distributed multi-robot learning [19].

Some exploration has been done in the past on multi-robot search and similar tasks. This work has been relatively disjoint thus far, with most studies focusing on a particular scenario which is not explicitly connected to other related work. The cost of using additional robots in a search task was explored and tested with simulation [7]. Detailed analysis has been done for swarms following a gradient [16]. In 2001, a contest at the International Joint Conference on Artificial Intelligence on collective robotic urban search and rescue [17] prompted some research on the topic [10]. Other publications explore multi-robot search strategies in simulation [6], for infrared tracking with simulation and real robots [7], and for odor source localization with real robots [8].

In [5], PSO was used to tune the parameters of a PSO-inspired multi-robot search strategy. Besides this, we are not aware of any previous publications on synthesis of multi-robot search behavior, and no previous studies have considered multi-robot search adaptation in changing environments.

3 Bio-Inspired Multi-Robot Search

The algorithm we use on our robot group for localizing targets in an unknown environment is inspired by the chemotaxis behavior of some types of bacteria, such as *E. coli*. By changing the rotation direction of their flagella, these bacteria can either swim in a straight line or to tumble in place. When moving, if the bacterium observes that the chemical gradient is positive, it is likely to continue to movement in the same direction. If it observes that the chemical gradient is negative, it is more likely to tumble and therefore assume a new random direction. This behavior results in overall movement in a positive gradient direction [2]. This type of chemotaxis behavior has inspired several effective robotic search strategies in the past [4] [9] [13]. However, none of these strategies used collaboration between robots in the search process.

In our algorithm, robots begin at some random locations within a bounded environment containing one or more target. We assume that robots are capable of perceiving the intensity of some emission from targets which fades non-linearly with increasing distance from the target. The robot will measure the perceived emission intensity and make a forward movement for a fixed distance. The robot will then measure the new perceived emission intensity. If the intensity is higher, the robot will maintain the same direction and make another step. If the intensity is lower, the robot will assume a new bearing and make a step in that direction. The process is then repeated. If a robot encounters any obstacles while moving

(i.e. walls or other robots), it will turn to avoid the obstacle using a reactive obstacle avoidance algorithm.

The collaboration aspect of our algorithm arises when a robot is choosing a new bearing. Robots are assumed to be capable of relative localization and communication with other nearby robots within a certain range (this could be accomplished using an on-board module such as the one described in [20]). In this scenario, relative localization is not restricted to line-of-sight, though this assumption may not hold on a real robotic platform. Using relative localization, robots continually share their current position and most recent perceived emission intensity. When choosing a new angle, if a robot detects at least one other robot in range with higher perceived intensity than its own, it will choose a bearing directly towards the robot with highest perceived intensity. If it detects no other robots with higher perceived intensity, it will uniformly select a random bearing within some arc in the approximately opposite direction that it currently faces.

This algorithm has four free parameters which can be adjusted to optimize the behavior for different environments: STEP_SIZE, RL_RANGE, CW_LIMIT, and CCW_LIMIT. STEP_SIZE is the distance robots move forward at each step of the algorithm. RL_RANGE is the maximum range of the relative localization and communication system (only robots with distance less than or equal to RL_RANGE will be perceivable by another robot). CW_LIMIT is the maximum angular offset from 180 degrees which the robot will consider when choosing a random bearing in the clockwise direction, and CCW_LIMIT is the maximum angular offset from 180 degrees which the robot will consider when choosing a random bearing in the counter-clockwise direction. The bearing is therefore uniformly randomly selected from an arc of size CW_LIMIT+CCW_LIMIT.

Algorithm parameters can be seen graphically in Fig. 1a.

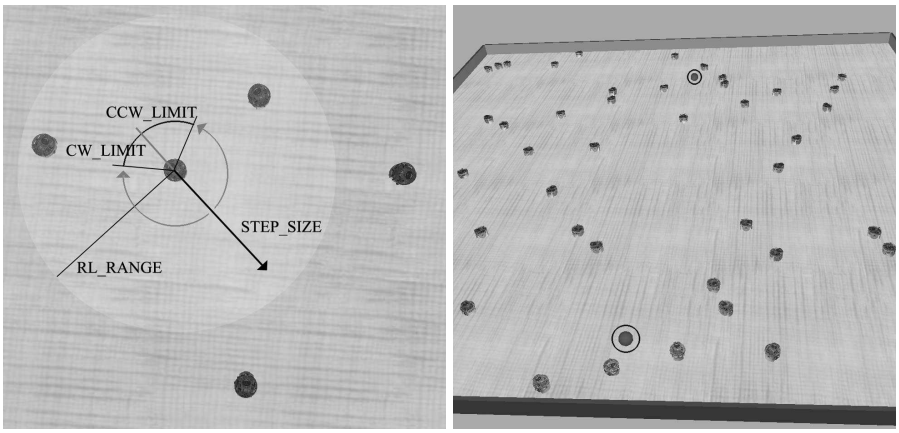

Fig. 1. (a) Graphical depiction of multi-robot search algorithm parameters with simulated e-puck robots. (b) Robot arena with e-puck robots and targets. Targets have been circled.

4 Distributed PSO for Parameter Adaptation in Multi-Robot Search

We now apply distributed Particle Swarm Optimization to adapt the free parameters of our multi-robot search algorithm in a realistically simulated environment.

4.1 Experimental Setup

For our adaptation technique, we use the noise-resistant PSO algorithm from [19]. At each iteration, the algorithm reevaluates the previous best locations, averaging 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 often gives better results given the same number of evaluations of candidate solutions.

Initial particle elements are randomly generated in the range $[0.0, 1.0]$. The elements are allowed to change to any value during evolution, but are bounded to this range and scaled appropriately during evaluation (i.e. any negative value is considered as 0.0 and any value greater than 1.0 is considered as 1.0 and then scaled during evaluation). Velocity is randomly initialized in the range $[-0.5, 0.5]$ and prevented from ever going outside this range. We use a swarm size of 50, with $pw = nw = 2.0$ and $w = 0.6$.

To explore the impact of particle neighborhood size in this scenario, we consider three different particle neighborhoods in experimentation: an *lbest* local neighborhood in a ring topology with one neighbor on each side (2 neighbors total), a *gbest* global neighborhood where all particles are neighbors (49 neighbors total), and an intermediate neighborhood in a ring topology with five neighbors on each side (10 neighbors total) which we denote *ibest*.

We use Webots, a realistic simulator, for our robotic simulations [15], using 50 e-puck[1] robots [3]. Every robot is responsible for the evaluation of a single, unique particle from the PSO swarm. The robot(s) operate in a 4.0 m x 4.0 m square arena with no additional obstacles (see Fig. 1b). Several targets are randomly placed in the environment. If a robot comes within range $r = 0.10$ m of a target, the target is considered to be "found", and it is randomly moved to a new location in the arena. The emission intensity perceived by a robot i is equal to:

$$I_i = \eta(.) + \sum_j \frac{P_j}{d_{ij}^2}$$

where P_j is the power of target j, d_{ij} is the distance between robot i and target j (in meters), and $\eta(.)$ is random background noise, given by the absolute value of a sample taken from a Gaussian probability density function with mean 0 and standard deviation $\sigma = 10$.

[1] http://www.e-puck.org

For the robot controller, we use the bio-inspired multi-robot search algorithm described previously, with parameters STEP_SIZE, RL_RANGE, CW_LIMIT, and CCW_LIMIT determined by the distributed PSO algorithm; these parameters are linearly scaled from the $[0.0, 1.0]$ range given by PSO to the ranges $[0.1, 1.0]$, $[0.0, 3.9]$, $[0, \pi]$, and $[0, \pi]$, respectively. To evaluate a candidate solution, we run the multi-robot search algorithm with the specified parameters for a span of 120 seconds. During that time, robots average their perceived emission intensity, sampling every 64 milliseconds, and use this value as their fitness (normalized to a maximum of 1.0, assuming an upper intensity limit of 255). This fitness function rewards robots who spend more time in close proximity to a target (where emission intensity is higher) over those who remain farther away.

For our initial experiments, we use three targets with emission power $P_j = 10$ for all targets. Each of the 50 robots is responsible for a single member of the PSO swarm (i.e. they must evaluate that member at each iteration and communicate the resulting fitness measure). We assume all robots are synchronized in their evaluation here, something which might not necessarily be possible on real multi-robot systems. With 100 iterations of the PSO algorithm, the adaptation requires a total simulated learning time of approximately 6 hours 40 minutes (100 iterations comprised of 2 evaluations each lasting 2 minutes).

4.2 Results

The average progress of individual robot fitness throughout adaptation for the three different particle neighborhoods can be seen in Fig. 2a. Clear improvements in fitness for all neighborhood types can be observed over the course of the algorithm, although the overall change in fitness is not dramatic (approximately a 30% increase from the initial average fitness value). There do not appear to be major differences in performance for different particle neighborhood types; using an *lbest* neighborhood seems to cause slower initial improvements, but final performances are similar for all neighborhood types, with *gbest* doing only slightly better than *ibest*, which does slightly better than *lbest*.

While individual robot fitness is useful for adaptation, it does not necessarily provide us with a good indicator of the performance of the robot swarm as a whole. To measure this, we use the number of targets "found" at every evaluation of multi-robot search as a group fitness value. The average progress of group fitness throughout adaptation can be seen in Fig. 2b. We again see clear improvements in fitness for all neighborhood types, though with a significantly larger improvement (more than a 100% increase from the initial average fitness value); this shows that our individual and group fitness measures are very well-aligned. We also notice more pronounced differences between results for different particle neighborhoods: *lbest* improves quite slowly throughout adaptation; *gbest* improves more quickly initially, but plateaus early in the adaptation process; *ibest* offers a compromise between the two, improving more slowly than *gbest* but continuing throughout adaptation to eventually achieve a higher fitness than the other two neighborhood types. This is similar to what was observed in [19], where an intermediate neighborhood size gave optimal performance.

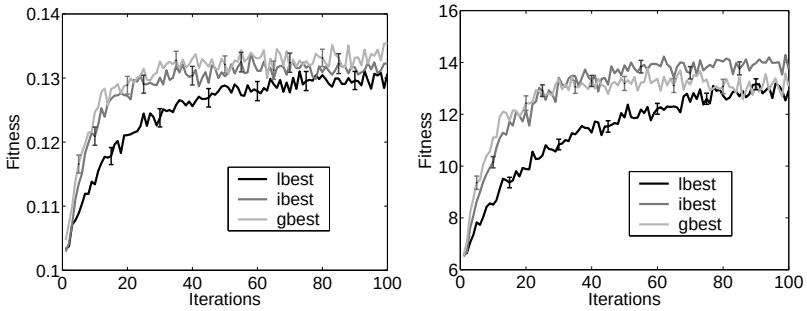

Fig. 2. Average individual fitness of particles throughout adaptation averaged over 250 runs for *lbest*, *ibest*, and *gbest* particle neighborhoods for (a) individual fitness, and (b) group fitness. Error bars represent standard error.

When using PSO for optimization, the standard approach is to run the algorithm for repeated iterations until a termination criterion is reached (e.g., a certain number of iterations have occurred) and then select the best found solution. While this is straight-forward for numerical optimization, it can be much more difficult in robotic learning. Determining which solution is actually best requires repeated evaluations to obtain low-noise fitness measures, which may require significantly more running time for robots. In addition, once a final solution is selected by all robots, the robotic swarm can no longer adapt to changing environmental parameters which may occur in its surroundings. For these reasons, there are clear advantages to running distributed PSO optimization without termination, where robots continue to adapt their parameters indefinitely. The possible drawback to this method is that the performance of robot swarm may be significantly lower than if it were to use the best found solution.

In order to assess whether continuing adaptation incurs a performance penalty, we compare the final average group fitness and the best found solution group fitness after 100 iterations of adaptation. The final average group fitness was taken as the average group fitness over the last five adaptation iterations. The best found controller was selected by evaluating the final personal best solutions for all particles five times and selecting the one with the highest average individual fitness; this solution was run on all robots to determine its group fitness. The average performance over 250 runs for different neighborhood types can be seen in Fig. 3. The final average group fitness is as high as the best found solution group fitness for all neighborhood types. This indicates that continuing adaptation indefinitely will not result in a significant performance decrease. Oddly, for *gbest*, the average group fitness is actually higher than the best found controller fitness. One possible explanation is that the *gbest* neighborhood may overfit on individual fitness, causing a slight decrease in the group fitness when an over-optimized solution is selected. We can also confirm here that the *ibest* neighborhood significantly outperforms *lbest* and *gbest* using an ANOVA test,

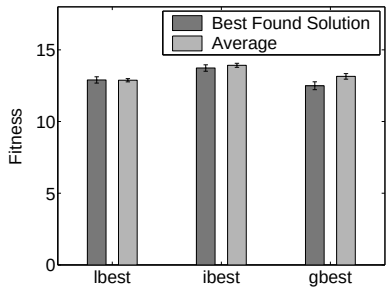

Fig. 3. Final average group fitness and best found solution group fitness averaged over 250 runs for *lbest*, *ibest*, and *gbest* particle neighborhoods. Error bars represent standard error.

both for the best found solution (P-value of 0.0084 for *lbest* and 0.0005 for *gbest*) and average group fitness (P-value of $1.6 \cdot 10^{-8}$ for *lbest* and 0.0014 for *gbest*).

5 Adaptation in Non-Static Environments

A major potential benefit to unending robotic adaptation is that robots could automatically adjust to changes that might arise in the environment. To test this possibility, we rerun our experiments in a non-static environment. We use the same simulated scenario as in the previous experiments. At the start of adaptation, we use three targets with emission power $P_j = 3$ for all targets. After 50 iterations, we switch to using ten targets with emission power $P_j = 10$ for all targets. We again run 100 iterations total using 50 robots.

The average of individual robot fitness throughout adaptation in the non-static environment can be seen in Fig. 4a (fitness in the last 50 iterations is significantly higher due to the increased power and number of targets and is shown at a reduced scale in the plot, with the scaling factor chosen to best align the data). We observe the same trend as in the previous experiment for the first 50 iterations. At this point, there is a fitness shift for all neighborhood types as the simulations switches over to the new environmental parameters and fitness scaling. In the last 50 iterations, we see a slightly larger increase in individual fitness compared to what was observed in the static environment, with similar final fitness for all neighborhood types.

The group fitness throughout adaptation in the non-static environment can be observed in Fig. 4b (fitness in the last 50 iterations is significantly higher due to the increased power and number of targets and is shown at a reduced scale, with the scaling factor chosen to best align the plots). Progress in the first 50 iterations is the same as for the static environment. In the last 50 iterations, we see continuing fitness improvements for *lbest* and *ibest* neighborhoods, with larger gains than were observed in the last 50 iterations in the static environment, particularly for *lbest*. This indicates that robots successfully adapt from the initial scenario to the new environmental parameters. The fitness for the

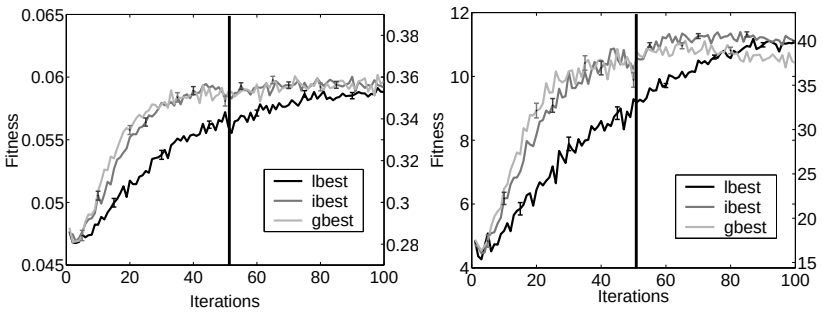

Fig. 4. Average fitness of particles throughout adaptation in a non-static environment averaged over 250 runs for *lbest*, *ibest*, and *gbest* particle neighborhoods for (a) individual fitness, and (b) group fitness. Fitness in the later iterations is scaled to match fitness in the initial iterations. Error bars represent standard error.

gbest neighborhood remains approximately constant. This could be caused by premature convergence, which would lead to lower particle diversity and prevent further adaptation. Higher diversity in the other neighborhoods allow them to continue to adapt in non-static environments, particularly in the case of *lbest*, which achieves final performance as high as *ibest* in this scenario.

6 Conclusion and Outlook

We have shown that distributed Particle Swarm Optimization can be used for adaptation in multi-robot systems, illustrated with the case study of multi-robot search. Best performance is obtained by using an intermediate particle neighborhood between *lbest* and *gbest*, which offers fast optimization without premature particle convergence in this scenario. Adaptation can be continued indefinitely without a significant performance penalty, making it possible for the robot swarm to automatically adapt in non-static environments.

In this paper, we have devoted our focus to the multi-robot adaptation process using distributed PSO and spent little time studying the multi-robot search algorithm itself. Observing the final solutions found by the adaptation process could give us insight into the impact of each of the different algorithmic parameters and how they influenced the robots' performance. The bio-inspired multi-robot search technique should also be compared to other common search strategies for similar scenarios (such as standard "optimal" search techniques or PSO-inspired search) to evaluate its overall potential.

References

1. Balch, T. (1998) *Behavioral diversity in learning robot teams.* PhD Thesis, College of Computing, Georgia Institute of Technology.
2. Berg, H. C. (2003) *E. coli in motion.* Springer-Verlag, NY.

3. Cianci, C., Raemy, X., Pugh, J., & Martinoli, A. (2007) "Communication in a swarm of miniature robots: The e-puck as an educational tool for swarm robotics" Swarm Rob. Workshop, Springer Lect. Notes in Comp. Sci., Vol. 4433, pp. 103-115.
4. Dhariwal, A., Sukhatme, G. S., & Requicha, A.A.G. (2004) "Bacterium-inspired robots for environmental monitoring", Proc. of the IEEE Intl. Conf. on Robotics and Automation, New Orleans, LA, USA, April 26 - May 1, pp. 1436-1443.
5. Doctor, S., Venayagamoorthy, G. K., & Gudise, V. G. (2004) "Optimal PSO for Collective Robotic Search Applications", Proc. of the IEEE Congress on Evolutionary Computation, June 19-23, Portland, OR, USA, pp. 1390-1395.
6. Goldsmith, S. Y. & Robinett, R. (1998) "Collective search by mobile robots using alpha-beta coordination", Collective Robotics (A. Drogoul, M. Tambe, and T. Fukuda, eds.), Springer Verlag:Berlin, pp. 136146.
7. Hayes, A. T. (2002) "How Many Robots? Group Size and Efficiency in Collective Search Tasks", Proc. of the 6th Intl. Symp. on Distributed Autonomous Robotic Systems DARS'02, Fukuoka, Japan, pp. 289-298.
8. Hayes, A. T., Martinoli, A. & Goodman, R. M. (2003) "Swarm Robotic Odor Localization: Off-Line Optimization and Validation with Real Robots", Special Issue on Biological Robots, D. McFarland, editor, Robotica, Vol. 21, pp. 427-441.
9. Holland, O. & Melhuish, C. (1996) "Some adaptive movements of animats with single symmetrical sensors", 4th Intl. Conf. on Simulation of Adaptive Behaviour, MIT Press, Cambridge, MA.
10. Kantor, G., Singh, S., Peterson, R., Rus, D., Das, A., Kumar, V., Pereira, G., & Spletzer, J. (2003) "Distributed search and rescue with robot and sensor teams", Proc. of the 4th Intl. Conf. on Field and Service Robotics, Japan.
11. Kelly, I. D. & Keating, D. A. (1998) "Faster learning of control parameters through sharing experiences of autonomous mobile robots" Intl. J. of System Science, Vol. 29, No. 7, pp. 783-793.
12. Kennedy, J. & Eberhart, R. (1995) "Particle swarm optimization" Neural Networks, 1995. Proceedings., IEEE Intl. Conf. on, Vol.4, Iss., pp. 1942-1948.
13. Marques, L., Nunes, U., & de Almeida, A. T. (2002) "Olfaction-based mobile robot navigation", Thin Solid Films, Vol. 418, pp 51-58.
14. Matarić, M. J. (2001) "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, Vol. 2, No. 1, pp. 81-93.
15. Michel, O. (2004) "Webots: Professional Mobile Robot Simulation" Int. J. of Advanced Robotic Systems, Vol. 1, pp. 39-42.
16. Ögren, P., Fiorelli, E., Leonard, N. E. (2004) "Cooperative Control of Mobile Sensor Networks: Adaptive Gradient Climbing in a Distributed Environment", IEEE Transactions on Automatic Control. Vol. 49, No. 8, pp. 1292-1302.
17. Osuka, K., Murphy, R., & Schultz, A. C. (2002) "USAR Competitions for Physically Situated Robots", IEEE Robotics and Automation Magazine, Sept, pp. 26-33.
18. Pugh, J., Zhang, Y. & Martinoli, A. (2005) "Particle swarm optimization for unsupervised robotic learning" Swarm Intelligence Symp., Pasadena, CA, pp. 92-99.
19. Pugh, J. & Martinoli, A. (2006) "Multi-Robot Learning with Particle Swarm Optimization" Intl. Conf. on Autonomous Agents and Multiagent Systems, Hakodate, Japan, May 8-12, pp. 441-448.
20. Pugh, J. & Martinoli, A. (2006) "Relative Localization and Communication Module for Small-Scale Multi-Robot Systems", Proc. of the IEEE Intl. Conf. on Robotics and Automation, Miami, Florida, USA, May 15-19, pp. 188-193.
21. Stone, P. (1998) *Layered Learning in Multi-Agent Systems.* PhD Thesis, School of Computer Science, Carnegie Mellon University.