Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007)

Inspiring and Modeling Multi-Robot Search with Particle Swarm Optimization

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

Abstract— **Within the field of multi-robot systems, multi-robot search is one area which is currently receiving a lot of research attention. One major challenge within this area is to design effective algorithms that allow a team of robots to work together to find their targets. Recently, techniques have been adopted for multi-robot search from the Particle Swarm Optimization algorithm, which uses a virtual multi-agent search to find optima in a multi-dimensional function space. We present here a multi-search algorithm inspired by Particle Swarm Optimization. Additionally, we exploit this inspiration by modifying the Particle Swarm Optimization algorithm to mimic the multi-robot search process, thereby allowing us to model at an abstracted level the effects of changing aspects and parameters of the system such as number of robots and communication range.**

I. Introduction

Locating one or more targets within an unknown environment is a task well-suited to mobile robotics. Robots can be equipped with sensors to detect targets and programmed to explore the area in search of their goal(s). The automated nature of this approach may save a lot of time and effort as compared to other search methods. Robotic search is especially preferable when the area is either hazardous or inaccessible to humans. Examples include locating mines for de-mining [1], [8], finding victims in a disaster area [10], and planetary exploration [13].

Using a swarm-intelligent robotic approach in search tasks can offer several major benefits over the single robot alternative. Searching can be done massively in parallel, significantly decreasing the time taken to locate targets and improving robustness against failure of single agents by redundancy as well as individual simplicity. The scalability of the system provides a simple method to further increase the rate and robustness by adding more agents. The system is also less prone to poor decision-making, as the swarm provides more sensory and environmental information than a single robot can. This could allow for a more informed choice, which can further increase the speed at which the swarm operates. Although search has been well-explored in the past [3], using multi-robot systems for search is a more recent development and has not yet been studied extensively.

A significant amount of time is often needed to collect experimental data with multi-robot systems. Even realistic models of the system, such as sensor- and actuator-based simulation, may require large quantities of computational time if systematic experimentation is necessary. This limitation motivates the use of abstracted models which uses approximations of details of the system which have little impact on the targeted performance metrics. Such models can be divided into two main categories: macroscopic, which model the robotic swarm as a whole, and microscopic, which model each robot separately. However, within these categories, there further exists multiple abstraction levels which may differ significantly in their computational complexity. It can be very beneficial to use a multiple abstraction level modeling methodology to allow for easy tuning of model parameters and fast experimentation (see for example, [2], [11], [14], [16]). However, developing accurate abstracted models can be a very difficult task for some multi-robot scenarios, as it may not be immediately obvious which aspects of the system can be ignored or approximated given the targeted performance metrics.

Particle Swarm Optimization (PSO) is a promising new optimization technique developed by James Kennedy and Russell Eberhart [7] [12] which models a set of potential problem solutions as a swarm of particles searching in a virtual space for good solutions. The method was inspired by the movement of flocking birds and their interactions with their neighbors in the group. Every particle in the swarm begins with a randomized position (x_i) and (possibly) randomized velocity (v_i) in the n-dimensional search space, where $x_{i,j}$ represents the location of particle index i in the j-th dimension of 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 at which position it achieved its highest performance $(x^*_{i,j})$. Every 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 swarm, henceforth known as a "ring topology") regardless of the particles' positions in the search space. Global neighborhoods tend to be favored for problems where immediate convergence is desired, while local neighborhoods are preferable for problems with local optima where a purely

1-4244-0708-7/07/$20.00 ©2007 IEEE

greedy algorithm may become stuck. 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 the velocity over time to prevent explosions of the swarm and ensure ultimate convergence, 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 sampling of a uniformly-distributed random variable in $[0, 1]$.

The parallel between the multi-agent search in the robotic scenario and the multi-agent search in the virtual optimization space has been recently explored in several instances. Distributed unsupervised robotic learning was accomplished in a robotic group by assigning each robot a unique PSO particle that represented the robot controller [19]. Adaptations of PSO have been used for multi-robot odor search in several instances [9], [15]. Particle Swarm Optimization was also applied recursively to a multi-robot search task, where the parameters of the PSO-inspired search were optimized by an external PSO algorithm [6]. The effect of including aspects of multi-robot search in PSO has been partially explored [21]. Additionally, PSO was used as an inspiration for a solution to a multi-animal foraging task [5], which could be applied to multi-robot systems as well. However, none of these applications extend the inspiration to use PSO as an effective model of the robot group performance.

Section II of this paper presents our PSO-inspired multi-robot search algorithm as well as our modifications to PSO to model it. In Section III, we analyzes the performances of the algorithm using a realistic sensor- and actuator-based model for differing numbers of robots and communication ranges; we compare these results to those obtained using the simplified modified PSO model. Section IV introduces a new PSO-inspired multi-robot search algorithm which doesn't require global positioning. This new algorithm is analyzed in Section V, and results are compared both to the respective simplified modified PSO model and to those obtained for the original PSO-inspired algorithm. In Section VI, we discuss the implications of the observed results, and Section VII concludes.

II. Techniques

By using PSO as an inspiration for multi-robot search, we hope not only to generate an effective search algorithm, but also to allow for the creation of a simplified microscopic model of the search system by modifying the PSO algorithm to include aspects of the multi-robot search. We therefore have an exchange between the two scenarios, with PSO influencing the robotic system design and aspects of mobile robotics guiding the creation of the simplified PSO model. If this simplified model is able to achieve similar results to our multi-robot search, it can be used to explore the effects of changing the parameters of the system using only a fraction of the computational resources of our realistic sensor- and actuator-based microscopic model.

A. Multi-Robot Search Using PSO

Our PSO-inspired multi-robot search algorithm is motivated by using a one-to-one matching between particles in the PSO swarm and robots in the multi-robot system. We assume that our robots can communicate amongst themselves. We also initially assume they have perfect knowledge of their location in the environment, either via GPS or some other global positioning mechanism (an implementation of PSO-inspired search without this constraints will be introduced in Section IV). Robots are therefore able to use the basic PSO equations to determine their desired velocities. However, there are some key differences between PSO and multi-robot search that require us to make some modifications to the algorithm.

1) Discrete versus Continuous Time: PSO works by having particles update their positions within the search space at every discrete iteration of the algorithm. Multi-robot search operates in continuous time, and as of yet robots are unable to teleport themselves between locations. We therefore approximate this jump by having the robot move for a fixed amount of time at the appropriate velocity towards its desired location. Iterations of the algorithm happen after each of these steps. This approach requires that the robotic swarm is synchronized so that iterations match between robots.

2) Movement Limitations: In PSO, particles can have infinite acceleration and no intrinsic limitations on velocity (if there is a velocity limit, it is often set high enough that particles can cross the region of interest of the virtual space in a single step). Because they exist in the real world, robots have limits to how quickly they can move and adjust their headings. In most multi-robot search scenarios, it would take a substantial amount of time for a robot to cross the search environment at maximum velocity. If a robot needs to go in a different direction, they typically must spend the time to rotate to face the new direction.

The velocity limit of the robots can be overcome by using a high value for the fixed time given for each robot step. However, this comes with the disadvantage of slowing down the search process. We deal with the acceleration limitation by alloting a short period of time after each step during which robots rotate to their new desired bearing.

3) Function Evaluation: We assume robots have a sensor which can detect the intensity of the target signal. This intensity is given by:

$$I(d) = \frac{P_0}{d^2} + \eta()$$

where P_0 is the source power, d is the distance between the robot and the source, and $\eta()$ is a sampling of additive Gaussian noise. These intensities represent the function evaluations of the PSO algorithm. Robots prefer higher intensities.

4) Robot Collisions: Particles in PSO are assumed to be infinitely small. This allows them to be arbitrarily close to each other without causing interference. In multi-robot systems, both robots and search targets have some volume which prevents them from clustering too densely. Some sort of collision avoidance is typically desired to prevent robots from becoming stuck on surfaces. We use Braitenberg obstacle avoidance to cause robots to veer away from possible collisions. If a robot is executing a step of the algorithm and avoids an obstacle, it will continue moving in its new direction but will not modify its internal velocity representation (i.e. at the next iteration it will re-orient to its previous heading).

5) Particle Neighborhoods: The standard neighborhood structures in PSO require particles to share information with other particles that can be anywhere in the search space. Mobile robots often have strict limitations on their maximum communication range and capacity (i.e. number of others with which they can simultaneously communicate) or may prefer shorter communication distances to conserve energy. In this context, it makes more sense to define a neighborhood structure which is based on position in the search space, where nearby robots belong to the same neighborhood. We define a robot's neighborhood as all other robots within some fixed range r which could be the maximum communication range. Because robots are constantly in motion, this means that the particle neighborhood is dynamic, with neighbors possibly changing at each iteration of the algorithm. This neighborhood topology was shown to have good performance in low-dimensional spaces in [21].

B. Adapting PSO to Model Multi-Robot Search

In order to successfully create our simplified microscopic model of multi-robot search, some aspects of mobile robotics systems must be incorporated into the PSO algorithm.

1) Discrete versus Continuous Movement and Robot Collisions: One of the primary aspects of multi-robot search which will affect the performance is the collisions and obstacle avoidance among robots and between robots and the target or walls. Using the standard PSO particle displacement at each iteration, we will be unable to detect any collisions that might occur along the path. We therefore need to approximate the continuous movement of the robots by dividing the displacement into multiple steps and checking for collisions at each.

A particle is defined to have "collided" with an object when it comes within some distance of that object, defined by the object's and particle's radii and the particle's proximity sensor detection range. When a particle collides with something, its velocity will maintain its magnitude but be redirected. The response to a collision will vary depending upon the object with which it collides:

Environment Boundary - The particle's velocity will be "reflected" off the boundary by negating the component of the velocity tangential to the boundary.

Search Target - The particle's velocity will be redirected to go directly away from the target.

Another Particle - Both particle's velocities will be redirected to go directly away from each other.

Typically obstacle avoidance in multi-robot systems requires some interaction time for the robots to adjust their headings. We approximate by ignoring this time and allowing particles to immediately change direction.

2) Low Dimensionality: Typically PSO is used for the optimization of functions with many parameters (anywhere from around ten to several thousand). This means that the PSO virtual search space is generally of high dimensionality. For our multi-robot search scenario, robots can only move about on a plane, so the number of dimensions is limited to two. This may have an impact on the optimal values for different parameters of the system.

3) Real-World Noise: Real-world robotic systems are subject to the stochasticity of their components and the environment around them. Even the most carefully tuned sensors and actuators will have some noise component in real experiments. In our scenario, we have noise on our proximity sensors and slippage of the robot wheels. We consider these effects to not have a significant effect on the performance and ignore them in our simplified model.

We replicate the position-based neighborhood we use in multi-robot search in PSO, and we use the same intensity function as is used in the multi-robot search. Particles prefer higher function evaluations (function maximization).

III. Analysis of PSO-inspired Search with Global Positioning

We now simulate our PSO-inspired multi-robot search algorithm using a realistic sensor- and actuator-based model and compare its performance to that of our simplified model. We consider two performance metrics concerned with distance from the target location.

A. Setup

For our simulation, we use the realistic robotic simulator Webots [17]. We use a square arena of 8 m x 8 m with a cylindrical target in the center. Our robots are simulations of the e-puck[1] robot [4]. Robots are initially placed within the arena with random positions, headings, and velocities. System parameters can be seen in Table I. A larger than usual inertia coefficient (w) was used to encourage robots to explore the area well.

We use a matching setup with our PSO model, with all parameters derived from the robotic scenario (see Table II).

In PSO, particles can never "discover" the optimum of the function; rather, they congregate around it and their position is used to get an estimate of its location. Similarly, we assume that robots are unable to discover the exact target, but rather we use their observations to estimate its position. This is the case for real search scenarios such as land mine detection, where the mine is not observed directly but sensed via the chemical traces it emits.

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

TABLE I
ROBOTIC PARAMETERS

Parameter	Value
Robot Radius	0.0265 m
Target Radius	0.10 m
Proximity Sensor Detection Range	0.025 m
Maximum Velocity	0.1287 m/s
Time per Iteration	9.6 s
Target Power (P_0)	1.0 m^2
Additive Noise ($\eta()$) Standard Deviation	1.0
Inertia Coefficient (w)	1.2
Personal Weight (pw)	2.0
Neighborhood Weight (nw)	2.0
Proximity Sensor Noise	3%
Wheel Slip	10%

TABLE II
PSO PARAMETERS

Parameter	Value
Space Bounds	+/- 4.0
Maximum Velocity	1.24
Inter-Robot Collision Distance	0.078
Robot-Target Collision Distance	0.152
Number of Steps per Iteration	100
Target Power (P_0)	1.0
Additive Noise ($\eta()$) Standard Deviation	1.0
Inertia Coefficient (w)	1.2
Personal Weight (pw)	2.0
Neighborhood Weight (nw)	2.0

We execute 1000 runs of the algorithm of 100 iterations each with varying numbers of robots and communication ranges r.

B. Results

To evaluate the performance of the algorithms, we initially look at the distance from the target to the location where the strongest signal was detected by the swarm. The performance averaged over all runs for varying numbers of robots with a communication range of 2.0 m can be seen in Fig. 1. There is fairly close matching between the robotic simulation and PSO. As expected, the performance increases as the number of particles/robots increases, as the swarm is better able to explore the environment. We can see the progression of the algorithm with 20 particles/robots in Fig. 2. There is a rapid improvement in the early stages, followed by gradual improvement in the latter stages.

Thus far, our performance metric has been the distance of the strongest detected signal (closest robot) from the target; this is the best estimate any individual robot has of the target location and is a useful evaluation if our goal involves moving one robot as close as possible to the target. However, the estimate is limited in its precision, as robots are not able to measure inside the target, and it may be possible to improve that estimate by combining the knowledge of the swarm. We can take the strongest detected signal location of every robot in the swarm and average this position within the search space to generate a new prediction of the target location (see Fig. 3 for example). This gives us another metric with which we can evaluate the performance of our algorithms. While no robot will be able to reach this average position, it is a useful evaluation if our goal is only to record the target location for later use.

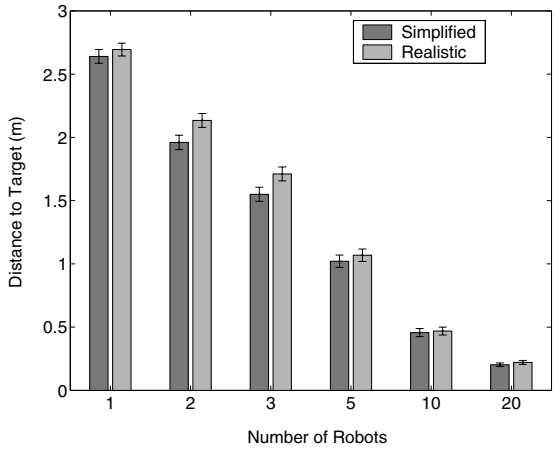

Fig. 1. Distance from target of swarm's strongest signal detection averaged over 1000 runs. Communication range $r = 2.0m$. Error bars represent standard error.

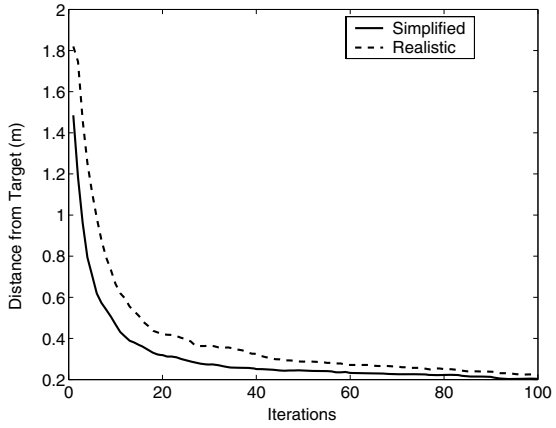

Fig. 2. Distance of swarm's strongest signal detection from target during search process for 20 robots averaged over 1000 runs. Communication range $r = 2.0m$.

The effect of varying communication ranges with 20 robots can be seen in Fig. 4 for both strongest swarm detection and the average position of the strongest detection by all robots. The matching between robots and particles is not as close here, though the same trend can be observed between the two. The simplified model achieves slightly better performance in all cases. This suggests that perhaps our modeling collision avoidance is not precisely accurate and does not cause as much dispersion as it should Counter to intuition, for the strongest swarm detection, the best performance here is for the smallest communication range. Intermediate communication ranges had worse performances, while the largest range performed slightly better. After observing several simulations, this was found to be the result of larger communication ranges causing more particle/robot clustering. This occurred when some robot made

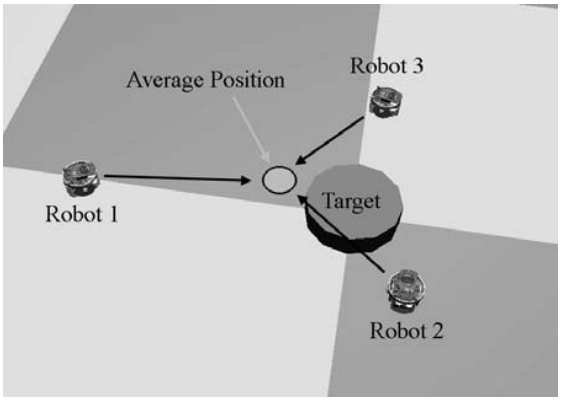

Fig. 3. Several robots near the target and their average position. This position may be closer to the target than any individual robot location.

a noisy detection which gave a falsely high result. If these clusters happened to form near the periphery of the arena, where the signal to noise ratio of the target was low, they might not observe the stronger signal near the arena center and therefore never find the target. This was improved slightly for very large communication range, which resulted in larger robotic clusters which typically were able to find the target after some time.

The average position of the strongest detection achieved fairly poor results for low communication ranges. However, performance improved dramatically as the range increased, and at the maximum communication range, the average position was closer to the target than the strongest detected signal at any communication range. This demonstates that, with the proper evaluation metric, a well-connected swarm is able to achieve superior performance working together than any individual member of the swarm.

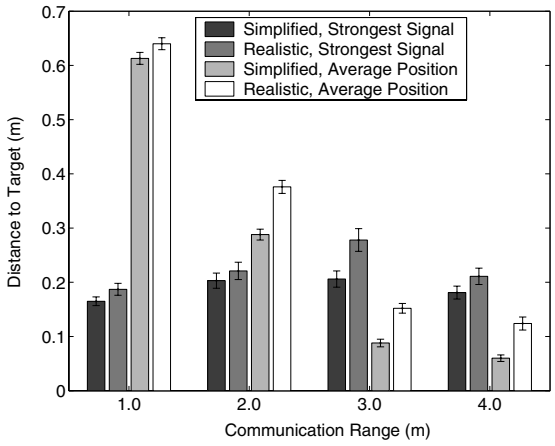

Fig. 4. Distance from the target of swarm's strongest signal detection and of average position of particles' strongest signal detections for 20 robots averaged over 1000 runs. Error bars represent standard error.

IV. PSO-inspired Search without Global Positioning

Thus far, we have assumed that robots have perfect knowledge of their position within the environment. This is often not the case in real multi-robot systems; very large numbers of robots may make it infeasible for a central positioning system to track them all, and the swarm may operate in locations where GPS and similar systems are unavailable. An alternative method for robots to determine the locations of nearby teammates is to use an on-board relative positioning system, such as the one described in [20]. This can be used to allow for effective multi-robot search techniques. However, our simplified PSO model will need some additional modifications before it can be used on robots that are not aware of their global position.

A. Short Particle Memory

A major difference between PSO-inspired search with and without global positioning is that robots may no longer able to remember which locations in the past yielded strong signal detections, as there are no global coordinates with which to store locations. Although internal odometry may be used to allow robots to retrace their path, often odometric information is very noisy, making accurate backtracking of more than a small distance unrealistic. Other GPS-free techniques may be applied, but this may result in an unreasonable computational and/or organizational overhead for the swarm. Therefore, robots with this limitation can only know their current and perhaps immediately previous locations with reasonable accuracy, giving them a very short memory.

We modify the PSO-inspired search algorithm by limiting the strongest detected signal to be either the current or immediately previous detection. The implications of this change are that if the current detection is stronger than the last, there is no personal best component to the modification of the velocity (i.e. $x^*_{i,j} - x_{i,j} = 0$). If the last detection was stronger than the current one, the robot is pulled in the direction from which it just came.

B. Sharing Information Among Robots

Because robots do not remember their previous locations, instead of sharing their strongest detection locations, they only share their current detections. Robots can then sense which other robot (if any) in the vicinity has the strongest current detection and use that as their neighborhood best with the location given by the relative positioning system.

C. Lack of Global Bearing

The standard PSO equations update every orthogonal dimension separately. However, because we no longer have a global coordinate system, there are no longer fixed dimensions. To overcome this, we have each robot use its own coordinate system relative to its current bearing: the x dimension is defined to be 45 degrees to the right of the robot's bearing and the y dimension is defined to be 45 degrees to the left (see Fig. 5). Because of the independent random component

of the velocity adjustment for each dimension, this causes any adjustments which are co-linear to the robot's heading (such as the personal best adjustment) to have a slightly random bearing, which could promote more exploration in the environment.

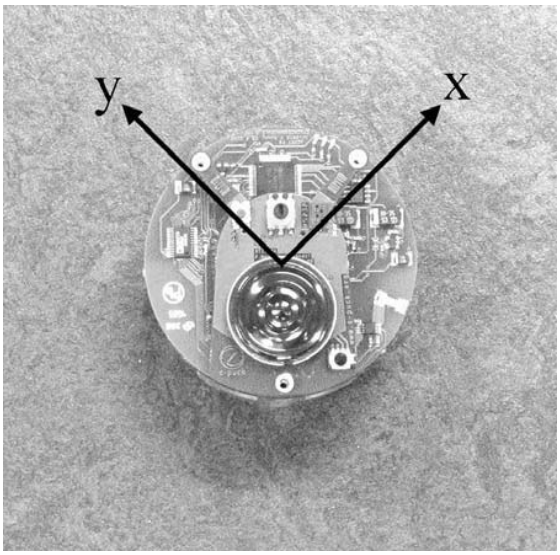

Fig. 5. e-puck robot with x and y axes relative to its heading shown

All of the changes described above can be added directly to the simplified PSO model.

V. Analysis of PSO-inspired Search without Global Positioning

We evaluate and compare PSO-inspired multi-robot search without global positioning using our realistic sensor- and actuator-based model and simplified PSO model.

A. Setup

We use the same scenario and parameters as PSO-inspired search with global positioning. However, because particles/robots no longer remember their previous detections, all results will only refer to the final positions of the robots instead of their strongest detection location.

B. Results

The strongest detection performance averaged over all runs for varying numbers of robots with communication range 2.0 m can be seen in Fig. 6. While performance again improves as the number of robots grows, the performance increase is much more abrupt than in the case with global positioning. There seems to be a critical number of robots above which the swarm is able to successfully congregate around the target. This number seems to differ slightly between the simplified and realistic models: between 10 and 20 robots in the simplified model and just over 5 in the realistic one. This leads to major discrepancies between the model and simulation in this range. This could again be due to the inaccurate obstacle avoidance in combination with a non-linear amplifying aggregation effect in the simplified model, which allows robots to cluster more and prevents exploration.

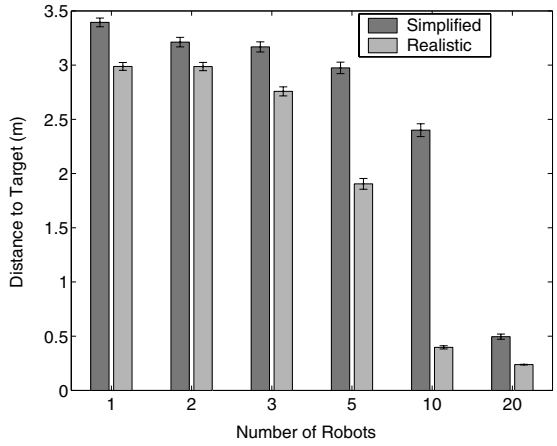

Fig. 6. Distance from target of swarm's strongest signal detection averaged over 1000 runs of PSO-inspired search without global positioning. Communication range $r = 2.0$. Error bars represent standard error.

The performance of 20 robots with varying communication ranges can be seen in Fig. 7. The matching between the realistic and simplified models is rather poor for low communication ranges but improves for higher ranges. The performance here consistently increases as the communication range increases. This can be explained by the short memory of the particles - the clusters that were formed in the previous algorithm do not form as often now because any falsely high detection will quickly be forgotten and robots will tend to migrate towards the target.

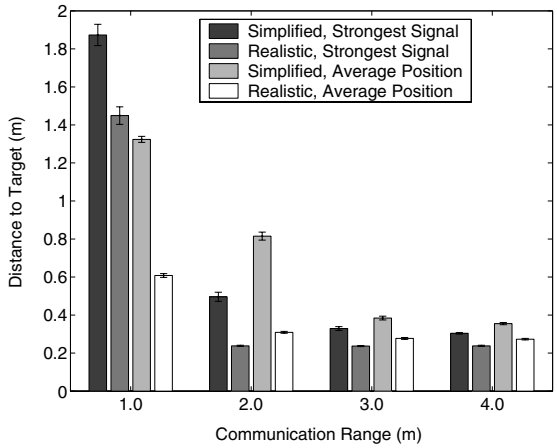

Fig. 7. Distance from the target of swarm's strongest signal detection and of average position of particles' strongest signal detections for 20 robots averaged over 1000 runs of PSO-inspired search without global positioning. Error bars represent standard error.

The distances from the target here are all farther than those of the swarm with global positioning. This is because although robots often remain near the target, they move about so that their position at any time is not guaranteed to be very close.

The algorithm could likely be improved if robots had improved odometry and could somehow store the locations of their best positions, though without actually using them in their calculations (the actions of the robots remain the same, only the data recorded changes). We can evaluate the performance with this modification. The results can be seen in Fig. 8 for 20 robots with varying communication ranges. Not only do we see an improvement here over the previous results, the distances here are all closer than even PSO-inspired search with global positioning was able to achieve. This suggests that using no global positioning but storing the best locations may increase performance as less clustering will occur in poor regions of the search space. The matching between the simplified model and realistic sensor- and actuator-based model here is again quite poor at low communication ranges and quite good for higher ranges.

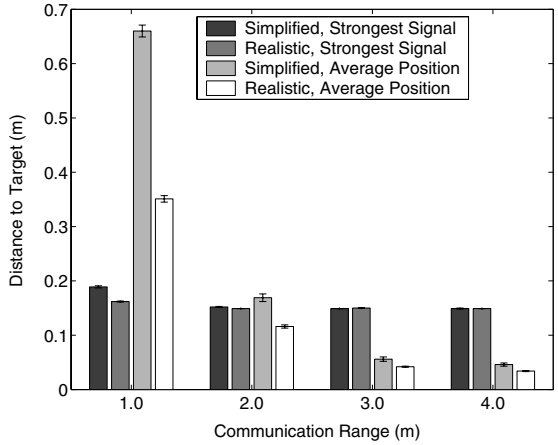

Fig. 8. Distance from the target of swarm's strongest signal detection and of average position of particles' strongest signal detections for 20 robots averaged over 1000 runs of PSO-inspired search without global positioning with remembering best positions. Error bars represent standard error.

VI. Discussion

Our simplified microscopic PSO-based model was able to replicate the results of our realistic sensor- and actuator-based model with reasonable accuracy in most cases of PSO-inspired search with global positioning. The time required to run the simplified model was several orders of magnitude less than the realistic one (less than 10 minutes for 1000 runs with the simplified model on a 2.8 GHz Pentium 4 desktop as compared to up to 20 hours for 1000 runs with the realistic model on a dual-processor 2.8 GHz Xeon server). This allows us to predict what results we may get with different search scenarios and parameters without the high time cost of lengthy systematic simulations. However, there currently seem to be discrepancies between the simplified and realistic models of PSO-inspired search without global positioning. This is likely due to the approximation of obstacle avoidance for a robot colliding with another robot or obstacle. More realistic approximations may improve the model quality at the price of a slightly higher computational cost.

While our PSO-inspired search swarm was able to congregate around the target in most cases, it did not converge to stable positions very close to the target. While this may not have had a particularly drastic effect in the case with global positioning, in the case without global positioning this resulted in much higher distances between the robots and the target at the end of the run and required marking or remembering the best positions to achieve good results. It might be possible to overcome this problem by adjusting the swarm behavior throughout the search, either as a function of the time passed or the observations made by the particles. A simple idea would be to decrease the inertia coefficient w linearly during the search, a common technique in standard PSO optimization. Another could be to have robots that detect very strong signals stop exploring in order to record that position and serve as a constant beacon for others. If the swarm dynamic were to change as it converged around the target, it might be possible to use that change to at some point declare that the target had been "found" and stop the search.

The search scenario we used here had only a single target in a non-dynamic environment. This is a very simple scenario and more complex search tasks may yield different results. Often search tasks have a large (possibly unknown) number of targets which robots must locate. Other tasks such as odor search may have a dynamic environment where the chemical concentration may vary in time as well as space. Some search algorithms are specifically tailored to locate only a single target in a non-dynamic environment and do not fare well in these scenarios. Standard PSO has been used successfully on functions which have many optima and on dynamic functions (see for example [18]), suggesting that PSO-inspired search may work well with these more complex scenarios, but it ought to be tested to determine concretely how well it is able to cope with these changes.

In standard PSO, the portion of the algorithm which typically requires the most (computational) time is the sampling by particles of the function space as opposed to the equations which modify the particle positions. While this may be the case for some instances of multi-robot search (e.g., odor search with a slow detection sensor), often the opposite is true, where it can be easy and fast to sense the source signal intensity but difficult and time-consuming to move to different locations in the environment (e.g., sound search). It might therefore be useful to further modify the PSO-inspired search algorithm to take advantage of this aspect of multi-robot search (e.g., signal detections could be made while the robot is moving towards its new point, allowing for more informed swarm decisions).

In this work, we have only focused on evaluating and comparing models of PSO-inspired multi-robot search. The results presented here ought to be compared to those of other multi-robot search algorithms to determine whether using PSO as an inspiration really does bring a significant benefit.

VII. Conclusion

We have presented a multi-robot search algorithm based on the principles of Particle Swarm Optimization and shown it

can be successful at finding a target. We have adapted PSO to model this algorithm and achieved close matching between the two. We have further presented a multi-robot search algorithm which does not require global positioning and a matching PSO model. Implications of results and relevant future work have been discussed.

VIII. Acknowledgements

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

References

[1] Acar, E. U., Choset, H., Yangang, Z., & Schervish, M., 2003. "Path Planning for Robotic Demining: Robust Sensor-based Coverage of Unstructured Environments and Probabilistic Methods", International Journal of Robotics Research, Vol. 22, No. 7-8, pp. 441-466.

[2] Agassounon, W., Martinoli, A., and Easton, K., 2004. "Macroscopic modeling of aggregation experiments using embodied agents in teams of constant and time-varying sizes", Autonomous Robots. Vol. 17, No. 2-3, pp. 163-192.

[3] Benkoski, S. J., Monticino, M. G., Weisinger, J. R., 1991. "A Survey of the Search Theory Literature", Naval Research Logistics, Vol. 38, Nr. 4, pp. 469-494.

[4] Cianci, C., Raemy, X., Pugh, J., & Martinoli, A., 2006. "Communication in a swarm of miniature robots: The e-puck as an educational tool for swarm robotics," Swarm-Robotics Workshop, Springer Lecture Notes in Computer Science. To appear.

[5] Di Chio, C., Poli, R., & Di Chio, P., 2006. "Extending the Particle Swarm Algorithm to Model Animal Foraging Behaviour", International Workshop on Ant Colony Optimization and Swarm Intelligence, Brussels, Belgium, September 4-7, pp. 514-515.

[6] Doctor, S., Venayagamoorthy, G. K., & Gudise, V. G., 2004. "Optimal PSO for Collective Robotic Search Applications", IEEE Congress on Evolutionary Computation, June 19-23, Portland, OR, pp. 1390-1395.

[7] Eberhart, R. & Kennedy, J., 1995. "A new optimizer using particle swarm theory" Micro Machine and Human Science, Proceedings of the Sixth International Symposium on, 4-6 Oct, pp. 39-43.

[8] Gage, D. W., 1995. "Many-Robot MCM Search Systems", Proc. of the Autonomous Vehicles in Mine Countermeasures Symposium. Monterey, CA, pp. 4-7.

[9] Jatmiko, W., Sekiyama, K., & Fukuda, T., 2006. "A PSO-based Mobile Sensor Network for Odor Source Localization in Dynamic Environment: Theory, Simulation and Measurement", IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, July 16-21, pp. 1036-1043.

[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] Kazadi, S., Abdul-Khaliq, A., Goodman, R., 2002. "On the Convergence of Puck Clustering Systems", Robotics and Autonomous Systems, Vol. 38, No. 2, pp. 93-117.

[12] Kennedy, J. & Eberhart, R., 1995. "Particle swarm optimization" Neural Networks, Proc. of IEEE International Conference on, Nov/Dec, pp. 1942-1948.

[13] Landis, G. A., 2003. "Robots and humans: Synergy in planetary exploration", Acta Astronaut, Vol. 55, No. 12, pp. 985-990.

[14] Lerman, K., Galstyan, A., Martinoli, A., 2001. "A Macroscopic Analytical Model of Collaboration in Distributed Robotic Systems", Artificial Life, Vol. 7, No. 4, pp. 375-393.

[15] Marques, L., Nunes, U., & de Almeida, A. T., 2006. "Particle swarm-based olfactory guided search", Autonomous Robotics, Vol. 20, pp. 277-287.

[16] Martinoli, A., Easton, K., & Agassounon, W., 2004. "Modeling Swarm Robotic Systems: A Case Study in Collaborative Distributed Manipulation", Special Issue on Experimental Robotics, Siciliano, B., editor, Int. Journal of Robotics Research, Vol. 23, No. 4, pp. 415–436.

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

[18] Parsopoulos, K. E. & Vrahatis, M. N., 2001. "Particle Swarm Optimizer in Noisy and Continuously Changing Environments" M.H. Hamza (Ed.), Artificial Intelligence and Soft Computing, IASTED/ACTA Press, pp. 289-294.

[19] Pugh, J. & Martinoli, A., 2006. "Multi-Robot Learning with Particle Swarm Optimization", International Conference 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 International Conference on Robotics and Automation, Miami, FL, May 15-19, pp. 188-193.

[21] Pugh, J., Segapelli, L., & Martinoli, A., 2006. "Applying Aspects of Multi-Robot Search to Particle Swarm Optimization", International Workshop on Ant Colony Optimization and Swarm Intelligence, Brussels, Belgium, September 4-7, pp. 506-507.