

Applying Aspects of Multi-Robot Search to Particle Swarm Optimization

Jim Pugh, Loïc Segapelli, and Alcherio Martinoli

Swarm-Intelligent Systems Group
École Polytechnique Fédérale de Lausanne
1015 Lausanne, Switzerland
Email: {jim.pugh,loic.segapelli,alcherio.martinoli}@epfl.ch

Abstract. We present a modified version of the Particle swarm Optimization algorithm in which we adjust the virtual swarm search by incorporating inter-agent dynamics native to multi-robot search scenarios. The neighborhood structure of PSO is modified to accurately represent feasible neighborhoods in multiple robot systems with limited communication in several different ways. The new algorithms are tested on several standard benchmark problems with a varying number of dimensions and are shown to offer superior performances to the standard algorithm in some cases. Further potential modifications and uses of the new algorithms are discussed.

1 Introduction

Throughout the history of research, some of the most innovative and useful discoveries have arisen from a fusion of two or more seemingly unrelated fields of study; a characteristic of some method or process is infused into a completely disjoint technique, and the resulting creation exhibits superior behavior. Some common examples include simulated annealing modeled after the annealing process in physics [10], Ant Colony Optimization modeled after the behavior of social insects [2], and the Particle Swarm Optimization algorithm modeled after the patterns of flocking birds [5], [8].

Particle swarm optimization (PSO) is a promising new optimization technique developed by James Kennedy and Russell Eberhart [5] [8] 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 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 $[0, 1]$.

Within the field of multi-robot systems, one area that has received some attention is collective robotic search, where a group of robots works together to localize one or more targets (e.g., [1], [3], [4], [6], [7], [11]). Using a collective 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 the target(s) 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 simply 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.

Both PSO and collective robotic search are instances of multi-agent search. For PSO, the search is virtual, and there are no limitations to particle movement, while multi-robot search is situated in the real world, and robots are limited by physical constraints, such as inter-robot collisions and limited communication range. However, there may be characteristics and ideas which can be shared between the two search scenarios to improve one or both; adapting the strategies of PSO particles could yield an effective search technique in multi-robot systems, and the dynamics of the collective robotic search might generate interesting effects in the PSO algorithm. There has thus far been fairly little work in this area. A one-to-one mapping from particles in PSO to robots in a collective robotic system was used for distributed unsupervised multi-robot learning in [12].

In this paper, we modify the neighborhood structure of Particle Swarm Optimization to incorporate the limited communication range common to members

of a mobile multi-robot system. Section 2 presents our modifications to the algorithm. Section 3 compares the new algorithms' performance against standard PSO on a set of benchmark problems with low dimensionality. Section 4 tests the algorithms on problems with higher dimensionality. Section 5 discusses the implications of the results, and Section 6 explores possible improvements and applications of the algorithm and concludes the paper.

2 Robotic 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. During collective robotic search, robots will often only communicate their observations with nearby neighbors. In the standard neighborhood setup of PSO, if we consider sharing information in a particle neighborhood as communication, particles may be required to communicate with other particles that are far from their position in the virtual search space. Therefore, to realistically model a multi-robot system, particle neighborhoods should be set in such a way that particles are not required to communicate with other particles outside of some close proximity.

We adopt the two neighborhood models suggested in [12] for communicating robots:

Model A: At each iteration of the algorithm, every particle selects itself and the two other particles with the smallest distance to it in the virtual search space as its neighborhood. This maintains a particle neighborhood of size three for the entire evolution, but allows the neighbors to change at every iteration. By only grouping particles that are near in the virtual space, we may significantly alter the convergence of the algorithm on different problems.

Model B: At each iteration of the algorithm, every particle selects itself and all other particles within some radius r in the virtual search space as its neighborhood. This results in a variable number of neighbors, as a particle may be close to very few or very many other particles at different times. However, it is perhaps more realistic than Model 1, since robot communication range has an upper limit, and for very sparse agent distributions, there may be fewer than two others within communication range at times. We expect that as the swarm converges, the particles will cluster together and the number of neighbors will increase.

Both of these models require the calculation of inter-particle distances in the virtual search space, which will require additional processing. However, in many PSO problems, the vast majority of processing time is spent on evaluation of the problem function rather than on the overhead required for the PSO algorithm. In these cases, the additional computation required for the robotic communication-based neighborhoods should be negligible.

3 Benchmark Evaluation with Low Dimensionality

Because collective robotic search takes place in the real world, the search space will always be either 2-dimensional (e.g., finding a target on the ground) or 3-dimensional (e.g., flying or swimming robots). To better match PSO and collective robotic search, we initially evaluate our PSO algorithm on several benchmark problems encoded in three dimensions.

3.1 Setup

We use the standard test functions from [9]. Functions can be found in Table 1, along with the number of iterations run by each algorithm. The number of iterations was chosen based on empirical evidence for how long the algorithms took to converge.

Table 1. Test Functions

Function	Function Name	Number of Iterations
f_1	Sphere	50
f_2	Generalized Rosenbrock	1000
f_3	Rastrigin	500
f_4	Griewank	500

The functions are defined as follows:

$$f_1(x) = \sum_{i=1}^n x_i^2 \quad (1)$$

$$f_2(x) = \sum_{i=1}^{n-1} [100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2] \quad (2)$$

$$f_3(x) = \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i) + 10] \quad (3)$$

$$f_4(x) = 1 + \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) \quad (4)$$

For all functions, $n = 3$, and x_i was constrained to $[-5.12, 5.12]$, the range used in [9].

For all algorithms, we used an inertia coefficient w of 0.6, with personal best and neighborhood best weights pw and nw both set to 2.0. We compare the performances of six variants of PSO: standard PSO with a local neighborhood ring topology with the nearest particle on each side assigned to be in a particle's neighborhood (PSOL), standard PSO with a global neighborhood (PSOG), PSO with Model A neighborhood (PSOA), and PSO with Model B neighborhood with radii of 1.0, 5.0, and 10.0 (PSOB1, PSOB5, and PSOB10, respectively).

3.2 Results

The final achieved values and standard deviations for all functions with all algorithms over 100 runs can be seen in Table 2.

Table 2. Performance (and Standard Deviation) of All Algorithms in Three Dimensions

	PSOL	PSOG	PSOA	PSOB1	PSOB5	PSOB10
f_1	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)	0.004 (0.022)	0.000 (0.000)	0.000 (0.000)
f_2	0.023 (0.022)	0.005 (0.020)	0.162 (0.161)	0.669 (2.047)	0.005 (0.016)	0.013 (0.084)
f_3	0.042 (0.197)	0.129 (0.365)	0.419 (0.526)	3.648 (2.302)	0.070 (0.225)	0.139 (0.347)
f_4	0.003 (0.004)	0.007 (0.005)	0.003 (0.004)	0.006 (0.005)	0.004 (0.004)	0.007 (0.005)

The progress of the best solutions on f_1 can be seen in Fig. 1, left. We see that PSO with a global neighborhood outperforms PSO with a local neighborhood, as there are no local optima in the Sphere function where the particles could become stuck. PSO with Model A neighborhood and PSO with Model B, radius 5.0 and 10.0 all do very well on this function, with PSOA converging the most quickly of any algorithm. PSOB1 performs rather poorly, likely because its small communication range doesn't allow it to communicate sufficiently with other particles.

The progress of the best solutions on f_2 can be seen in Fig. 1, right. None of the algorithms are able to consistently reach the minimum. PSOG again outperforms PSOL, though perhaps given many more iterations, PSOL might eventually surpass it. PSOA fairs quite poorly in this situation, suggesting that it may be very susceptible to becoming stuck in local minima. Both PSOG and PSOB5 achieve the best results here. PSOB10 does slightly worse, while PSOB1 again performs very poorly.

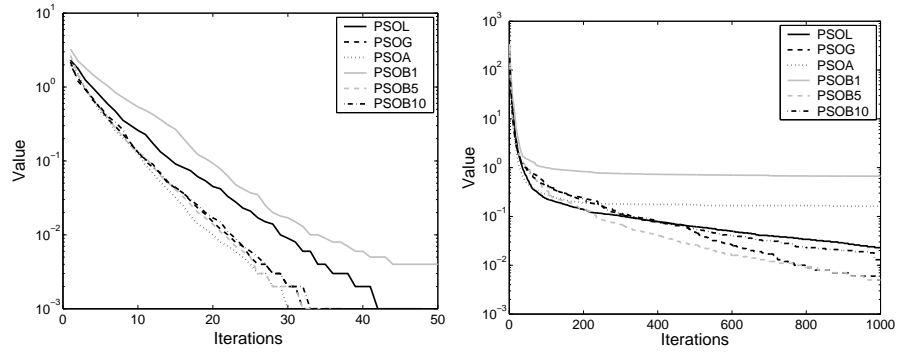


Fig. 1. Average of best solutions on f_1 (left) and f_2 (right) in three dimensions over 100 runs.

The progress of the best solutions on f_3 can be seen in Fig. 2, left. We observe a similar situation to that of f_2 ; however, in this case, PSOL is able to eventually surpass PSOG to obtain a better final performance. PSOB5 also achieves a better performance than PSOG, though not quite as low as that of PSOL. PSOB10 performs comparably to PSOG, and PSOA and PSOB1 again both do quite poorly.

The progress of the best solutions on f_4 can be seen in Fig. 2, right. PSOL quickly surpasses PSOG here, suggesting the existence of local optima. However, PSOA achieves a very good performance, which indicates that its susceptibility to becoming stuck in local optima may depend upon other aspects of the fitness landscape. PSOB5 again performs very well here.

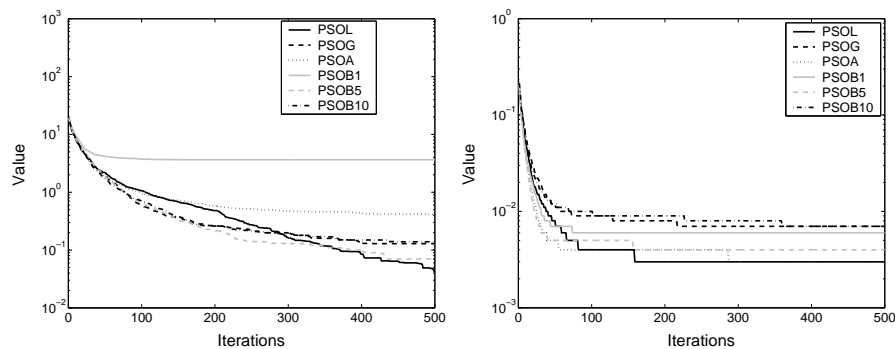


Fig. 2. Average of best solutions on f_3 (left) and f_4 (right) in three dimensions over 100 runs.

In summary, PSOG achieved superior performance to PSOL on f_1 and f_2 , while PSOL did better on f_3 and f_4 . With the Model A neighborhood, very good performance was achieved on f_1 and f_4 , while the performance on the other two problems was rather poor. The Model B neighborhood with a radius of 5.0 was one of the top performers on every single problem, making it the highest performing overall algorithm.

4 Benchmark Evaluation with High Dimensionality

We now compare the performances of the algorithms on the same benchmark problems, but with 30 dimensions instead of 3.

4.1 Setup

We use the same function set as in the previous section. Because of the higher function dimensionality, the number of iterations needed to be increased, and the new amounts can be seen in Table 1.

Table 3. Test Functions

Function	Function Name	Number of Iterations
f_1	Sphere	1000
f_2	Generalized Rosenbrock	20000
f_3	Rastrigin	10000
f_4	Griewank	10000

For all functions, $n = 30$, and x_i was still constrained to $[-5.12, 5.12]$. All algorithmic parameters remained the same as in the previous section. Because of the higher dimensionality of the virtual the space, the range in PSOB needed to be increased. We now use radii of 10.0, 30.0, and 40.0 (PSOB10, PSOB30, and PSOB40, respectively).

4.2 Results

The final achieved values and standard deviations for all functions with all algorithms over 100 runs can be seen in Table 4.

Table 4. Performance (and Standard Deviation) of All Algorithms in 30 Dimensions. -.- represents no convergence.

	PSOL	PSOG	PSOA	PSOB10	PSOB30	PSOB40
f_1	0.297 (0.486)	0.000 (0.002)	0.065 (0.120)	-.- (-.-)	0.000 (0.001)	0.000 (0.001)
f_2	8.006 (14.89)	14.31 (30.96)	147.3 (119.3)	-.- (-.-)	15.42 (22.58)	17.38 (24.44)
f_3	64.66 (16.42)	40.66 (11.41)	76.57 (17.42)	-.- (-.-)	40.89 (12.52)	41.00 (12.39)
f_4	0.002 (0.005)	0.011 (0.015)	0.012 (0.023)	-.- (-.-)	0.009 (0.011)	0.010 (0.012)

The progress of the best solutions on f_1 can be seen in Fig. 3, left. PSOG, PSOB30, and PSOB40 all converge to zero at approximately the same rate. PSOL has very slow convergence here, and doesn't manage to reach the minimum after 1000 iterations. PSOA also has difficulties, while PSOB10 doesn't converge at all.

The progress of the best solutions on f_2 can be seen in Fig. 3, right. PSOL achieves the best performance here, slowly pulling away from PSOG, PSOB30, and PSOB40 in the later stages of the evolution. PSOA again does not do very well, and PSOB10 doesn't converge.

The progress of the best solutions on f_3 can be seen in Fig. 4, left. PSOG, PSOB30, and PSOB40 all achieve the best performance here, with PSOL and PSOA doing slightly more poorly. PSOB10 again has no convergence.

The progress of the best solutions on f_4 can be seen in Fig. 4, right. PSOL clearly achieves the best results here. All others become stuck very early in the evolution except for PSOB10 which again doesn't converge. PSOB30 achieves marginally better performance than PSOG.

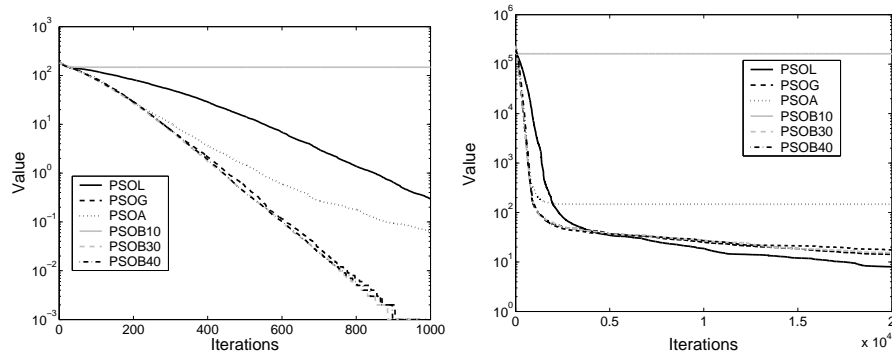


Fig. 3. Average of best solutions on f_1 (left) and f_2 (right) in 30 dimensions over 100 runs.

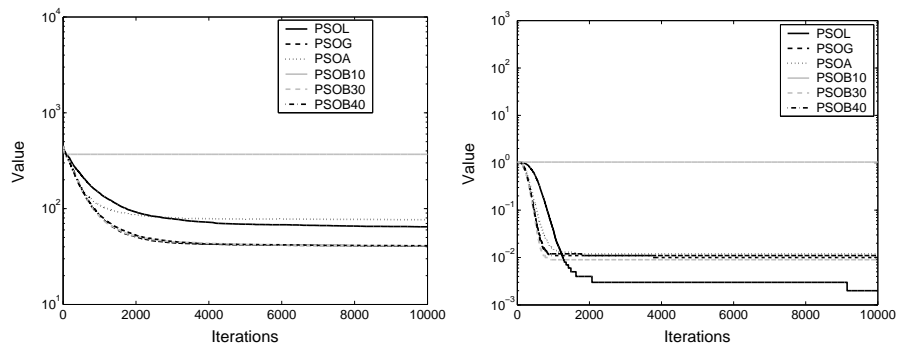


Fig. 4. Average of best solutions on f_3 (left) and f_4 (right) in 30 dimensions over 100 runs.

The performance of the robotic communication-based neighborhoods was much poorer here. The Model A neighborhood algorithm did not achieve very good performance on any of the problems. The Model B neighborhood with a communication radius of 10.0 was unable to converge in all cases. The Model B neighborhood algorithm for radii of 30.0 and 40.0 both performed approximately as well as PSOG, with PSOB30 doing perhaps slightly better on f_4 .

5 Discussion

Both neighborhood models had some success on problems with low dimensionalities. Although the Model A neighborhood did rather poorly on several problems, it outperformed all other algorithms on two of the benchmark problems, including the Sphere function. This suggests that a neighborhood of this type may be preferred in certain scenarios, including those where very fast convergence is desired. A possible explanation for this behavior is that by using only the closest

particles as neighbors, a particle is much more likely to move along the gradient in that region of the search space, which would promote faster convergence, but could limit particles' ability to escape local optima.

The Model B neighborhood with radius 5.0 did very well in low dimensionalities, achieving near optimal performance on all problems. One possible explanation for this is that in the early stages of the algorithm, particles are distributed and have rather few neighbors, which allows them to explore the space more thoroughly. Once particles begin to converge, the inter-particle distance decreases, which increases the neighborhood size and leads to more rapid convergence. This gives a good tradeoff between exploration and exploitation throughout the evolutionary process.

The reason for the lower performance of the robotic communication-based neighborhoods in higher dimensionalities is not completely clear. A strong possibility is that because we use the same number of particles in both 3 and 30 dimensions, the spatial density of the particles in 30 dimensions is so much lower that the communication-based neighborhoods no longer function nearly as effectively. To compensate for this, it might be advisable to use a different metric for calculating neighborhoods than the standard definition of distance ($\sqrt{\sum_i \delta x_i^2}$) to reduce the impact of increasing dimensions (e.g., $\sum_i \delta x_i$). The very poor performance of PSOB10 can be easily explained, as the "volume" covered by a sphere of radius 10.0 in a 30-dimensional space is less than 10^{-14} the size of a sphere of radius 30.0, which would cause particles using this neighborhood to almost never share information with any other particle. This demonstrates that the radius in Model B must be tuned much more carefully in higher dimensional spaces.

6 Conclusion and Outlook

We have presented new models for particle neighborhoods in the Particle Swarm Optimization algorithm based on the communication of robots in collective robotic search. These models offer superior performance to PSO on standard benchmark problems with low dimensionality, but do not perform as well on higher dimensional problems. Possible explanations for the algorithms behaviors have been given.

The work presented here could be expanded upon in several different ways: *Use Different Distance Metrics* - there were indications that the standard distance metric may not scale well with the dimensionality of the problem. Different distance metrics may prove to be more effective.

Incorporate Other Aspects of Collective Robotic Search - we have thus far only modified the neighborhood structure of PSO to match that of multi-robot search. Including other aspects such as obstacle avoidance or robotic dispersion might offer interesting results.

Analyze and Model Details of Multi-Agent Search - the nuances of agent dynamics is not thoroughly understood in either PSO or collective robotic search. By closely observing the behavior of the agents over time, we may be able to

to better understand the similarities and differences between the two different search scenarios, which might allow us to make more intelligent modifications to both. Ultimately, it would be very beneficial to develop an abstracted model of the systems which could be used to predict behavior in both scenarios, as well as prove properties of the swarm such as convergence and stability.

7 Acknowledgements

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

References

1. Das, A. et al. "Distributed search and rescue with robot and sensor teams", Field and Service Robotics. Japan, 2003.
2. Dorigo, M. & Di Caro, G. "The ant colony optimization meta-heuristic" In D. Corne, M. Dorigo and F. Glover, editors, *New Methods in Optimization*. McGraw-Hill, 1999.
3. Gage, D. W. "Randomized search strategies with imperfect sensors", In Proc. of SPIE Mobile Robots, Boston, Vol. 2058, 1993, pp. 270-279.
4. Gage, D. W. "Many-Robot MCM Search Systems", Proc. of the Autonomous Vehicles in Mine Countermeasures Symposium. Monterey, CA, 1995, pp. 4-7.
5. Eberhart, R. & Kennedy, J. "A new optimizer using particle swarm theory" *Micro Machine and Human Science*, 1995. MHS '95., Proceedings of the Sixth International Symposium on, Vol., Iss., 4-6 Oct 1995, pages:39-43
6. Hayes, A. T., Martinoli, A., Goodman, R. "Comparing Distributed Exploration Strategies with Simulated and Real Autonomous Robots", Proc. of the 5th International Symposium on Distributed Autonomous Robotic Systems DARS'00. Knoxville, TN, 2000, pp. 261-270.
7. Hayes, A. T., Martinoli, A., Goodman, R. "Distributed Odor Source Localization", *Special Issue on Artificial Olfaction, IEEE Sensors*, Vol. 2, Nr. 3, 2002, pp. 260-271.
8. Kennedy, J. & Eberhart, R. "Particle swarm optimization" *Neural Networks*, 1995. Proceedings., IEEE International Conference on, Vol.4, Iss., Nov/Dec 1995, pages:1942-1948 vol.4
9. Kennedy, J. & Eberhart, R. *Swarm Intelligence*, Morgan Kaufmann Academic Press, 2001.
10. Kirkpatrick, S., Gelatt Jr., C. D., & Vecchi, M. P. "Optimization by Simulated Annealing" *Science*, Vol. 220, No. 4598, 1983, pp. 671-680.
11. Miller, D. "Multiple Behavior-Controlled Micro-Robots for Planetary Surface Missions", Proc. of the IEEE International Conference on Systems, Man, and Cybernetics, Los Angeles, CA, 1990, pp. 289-292.
12. Pugh, J. & Martinoli, A. "Multi-Robot Learning with Particle Swarm Optimization" Proc. of Autonomous Agents and Multi-Agent Systems, Hakodate, Japan, 2006.