2009-05-21

An Exploration of Topologies and Communication in Large Particle Swarms

Matthew Gardner

Andrew McNabb

Kevin Seppi
Brigham Young University, kseppi@byu.edu

Original Publication Citation

Andrew McNabb, Matthew Gardner, and Kevin Seppi. "An Exploration of Topologies and Communication in Large Particle Swarms." In Proceedings of the IEEE Congress on Evolutionary Computation (CEC 29). pp. 712-719. Trondheim, Norway.

An Exploration of Topologies and Communication in Large Particle Swarms

Andrew McNabb, Matthew Gardner, and Kevin Seppi

Abstract— **Particle Swarm Optimization (PSO) has typically been used with small swarms of about 50 particles. However, PSO is more efficiently parallelized with large swarms. We formally describe existing topologies and identify variations which are better suited to large swarms in both sequential and parallel computing environments. We examine the performance of PSO for benchmark functions with respect to swarm size and topology.**

We develop and demonstrate a new PSO variant which leverages the unique strengths of large swarms. "Hearsay PSO" allows for information to flow quickly through the swarm, even with very loosely connected topologies. These loosely connected topologies are well suited to large scale parallel computing environments because they require very little communication between particles. We consider the case where function evaluations are expensive with respect to communication as well as the case where function evaluations are relatively inexpensive. We also consider a situation where local communication is inexpensive compared to external communication, such as multicore systems in a cluster.

I. Introduction

Particle Swarm Optimization (PSO) is an optimization algorithm that was inspired by experiments with simulated bird flocking [1]. This evolutionary algorithm has become popular because it is simple, requires little tuning, and has been found to be effective for a wide range of problems.

Often a function that needs to be optimized takes a long time to evaluate. A problem using web content, commercial transaction information, or bioinformatics data, for example, may involve large amounts of data and require minutes or hours for each function evaluation. Alternatively, some functions may be easy to evaluate but difficult to optimize. For example, highly dimensional problems may take many iterations to converge, and functions with deceptive local optima may converge prematurely. In all of these cases, PSO must be parallelized to fully utilize available resources.

PSO has typically been used with small swarms of 50 particles [2]. But with the widespread availability of multicore processors and even a modest number of particles per core, swarms running on a cluster could easily have thousands of particles. Some have observed that with many processors, larger swarm sizes may improve the rate of convergence [3], [4].

Section II of this paper describes the constricted PSO algorithm. Section III, building on other work in PSO topology [5], uses principles from graph theory to more formally describe particle swarm topologies. Section IV examines the performance of PSO for benchmark functions with respect to swarm size and topology and shows that large swarms can improve PSO with and without parallelism.

Section V focuses on functions which are more difficult to optimize in parallel because their evaluation is inexpensive relative to the cost of communication. This section considers ways to reduce communication by choosing appropriate topologies, including generalized ring and dynamic random. This point of view allows us to develop Hearsay PSO, a variant which leverages the unique strengths of large swarms by allowing information to flow quickly through the swarm, even with very loosely connected topologies. These loosely connected topologies are well suited to large scale parallel computing environments because they require very little communication between particles.

We also consider an approach for situations where some communication is very inexpensive, and other communication is expensive. This approach is well suited to clusters of multicore nodes, where particles on the same processor (possibly different cores) can easily communicate, but paricles on different processors require significantly more time to communicate.

II. Particle Swarm Optimization

Particle Swarm Optimization simulates the motion of particles in the domain of a function. These particles search for the optimum by evaluating the function as they move. During each iteration of the algorithm, the position and velocity of each particle are updated. Each particle is pulled toward the best position it has sampled, known as the *personal best*, and toward the best position of any particle in its neighborhood, known as the *neighborhood best* or global best. This attraction is weak enough to allow exploration but strong enough to encourage exploitation of good locations and to guarantee convergence.

Each particle's position and velocity are initialized to random values based on a function-specific feasible region. During each iteration of constricted PSO, the following equations update a particle's position $\vec{x}$ and velocity $\vec{v}$ with respect to personal best $\vec{b}_P$ and neighborhood best $\vec{b}_N$:

$$\vec{v}_{t+1} = \chi \left[\vec{v}_t + \phi_1 \, \mathrm{U}() \otimes (\vec{b}_P - \vec{x}_t) + \phi_2 \, \mathrm{U}() \otimes (\vec{b}_N - \vec{x}_t) \right] \tag{1}$$

$$\vec{x}_{t+1} = \vec{x}_t + \vec{v}_{t+1} \tag{2}$$

where ϕ_1 and ϕ_2 are usually set to 2.05, $\mathrm{U}()$ is a vector of samples drawn from a standard uniform distribution, and

Andrew McNabb, Matthew Gardner, and Kevin Seppi are with the Department of Computer Science, Brigham Young University, 3361 TMCB, Provo, UT 84602 (phone: 801-422-8717; email: {a,mjg82,k}@cs.byu.edu).

978-1-4244-2959-2/09/$25.00

$\otimes$ represents element-wise multiplication. The constriction constant χ is:

$$\chi = \frac{2}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|}$$

where $\phi = \phi_1 + \phi_2$ [6].

III. Topology Representation

The topology or sociometry of a swarm indicates how information is communicated between particles. The most typical representation of swarm topology is an undirected graph, where each vertex is a particle. The neighborhood of a particle—the set of vertices with shared edges—has two intuitive interpretations. From one perspective, it indicates which neighbors (or "informants") contribute their personal bests to a particular particle's neighborhood best. Alternatively, it indicates which neighbors a particular particle sends its personal best to. Since the graph is undirected, these are equivalent.

Swarm topologies are usually defined informally, using a combination of prose and diagrams. Although insightful, this approach is inherently imprecise and does not scale to more complex and dynamic topologies. Additionally, communication between particles, which has a significant cost in a parallel implementation of PSO, is only implicit in this representation.

To address the drawbacks of defining topologies informally, we will build on the formal definitions from graph theory. To make communication more explicit, we will use directed graphs instead of undirected graphs as suggested by Mendes as future work [5]. A *directed graph* is a pair $G = (V, E)$, where V is a nonempty finite set and E is a (possibly empty) subset of $V \times V$. The elements of V are the *vertices* of G and the elements of E are the *edges* (or arcs) of G. An edge (u, v) indicates a link from vertex u to vertex v [7].

The *topology* in iteration i is a directed graph $T_i = (P_i, E_i)$ where the vertex set $P_i = \{p_0, p_1, \ldots, p_{n-1}\}$ is a set of particles. Note that the topology in one iteration may be different from the topology in another iteration as particles and edges are added or removed. In a *static topology*, the topology is the same in all iterations, while in a *dynamic topology*, the topolgies may vary between iterations.

The *neighborhood* of a particle p in iteration i is the set $N_{i,p} = \{p_j \in P_i \,|\, (p, p_j) \in E_i\}$. In other words, it is the set of particles to which p sends its personal best. This representation is preferable to the "informants" model because it makes the communication more explicit. A particle may be a member of its own neighborhood, meaning that it considers its own personal best when updating its neighborhood best. Using the definition of a neighborhood, the edge set is constructed by combining the individual neighborhoods according to the equation $E_i = \bigcup_{p \in P} N_{i,p}$.

Many simple topologies can be more succinctly described by a neighborhood function. A *neighborhood function* ν is a function of a particle index and swarm size that gives a set of indices indicating the neighbors. The neighborhood N_{i,p_j} of the particle p_j in iteration i is related to the neighborhood function ν by the equation $N_{i,p_j} = \bigcup_{k \in \nu(j,n)} p_k$.

The complete topology K_n is a static topology of n particles where each particle sends its personal best to all other particles in the swarm. This topology is often referred to as fully connected, gbest, global topology, or star[1]. A complete topology is described by the neighborhood function:

$$\nu_K(i, n) = \{0, 1, \ldots, n-1\} \tag{3}$$

The ring topology $Ring_{n,1}$ is a static topology with n particles where each particle sends its personal best to itself and one neighbor on either side. The ring topology is also known as lbest or local topology. This topology has been generalized as $Ring_{n,k}$, where each particle sends its personal best to k neighbors on each side, although the $Ring_{n,1}$ variant is more common. The neighborhood function for $Ring_{n,k}$ is:

$$\nu_{Ring}(i, n, k) = \{(i-k) \bmod n, \ldots, i, \ldots, (i+k) \bmod n\}$$

In addition to fixed topologies, there are also many dynamic and adaptive topologies, including Randomized Directed Neighborhoods [8] and Dynamic Multi-Swarm [9]. This paper does not focus on adaptive topologies, such as TRIBES [10], because they tend to require global state, which is difficult to parallelize.

IV. Choosing a Topology and Swarm Size

The ideal topology and swarm size for Particle Swarm Optimization depend on the target function. Researchers have devised various benchmark functions and have found that the ideal topology for one function may perform very poorly for another function.

An attempt to standardize PSO found that although K_{50} converged to the global optimum more quickly than $Ring_{50,1}$ for many benchmark functions, it was also more likely to permaturely converge to local optima for other functions. The study found no significant improvement for any other swarm size between 20 and 100 and concluded with a recommendation to use $Ring_{50,1}$ as a starting point. The authors acknowledged that choosing the ideal topology requires thorough experimentation for the particular problem [2].

The Sphere, Rastrigin, and Griewank benchmark functions are representative of three different types of functions. Sphere is unimodal and smooth. It is most easily optimized when information flows between particles as quickly as possible. Although the $Ring_{n,1}$ topology will eventually converge, the K_n topology is much more efficient. Rastrigin is much less smooth, and PSO tends to prematurely converge to local minima. However, increasing communication still tends to improve performance for this function, which is best suited for the K_n topology. Griewank has even more deceptive local minima and requires less communication. Although K_n initially appears to perform well, $Ring_{n,1}$ is more effective as it is less prone to converge prematurely.

[1]The word "star" has been used historically in the PSO community for K_n. We, and others, feel that this choice is unfortunate because graph theory uses the term to denote the complete bipartite graph $K_{1,k}$.

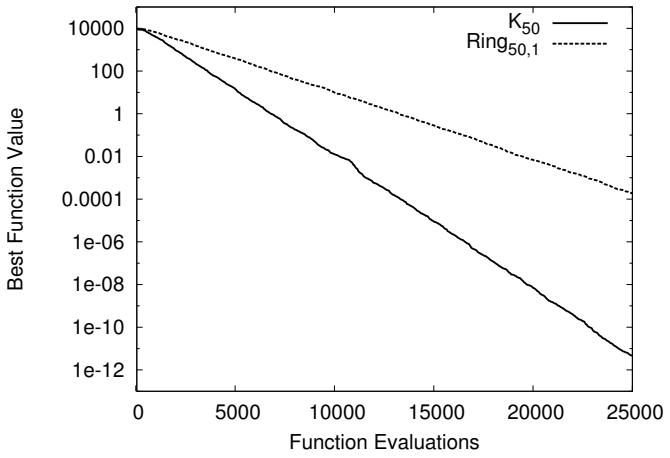

Fig. 1. PSO performance on Sphere for conventional-sized K_{50} and $Ring_{50,1}$ swarms.

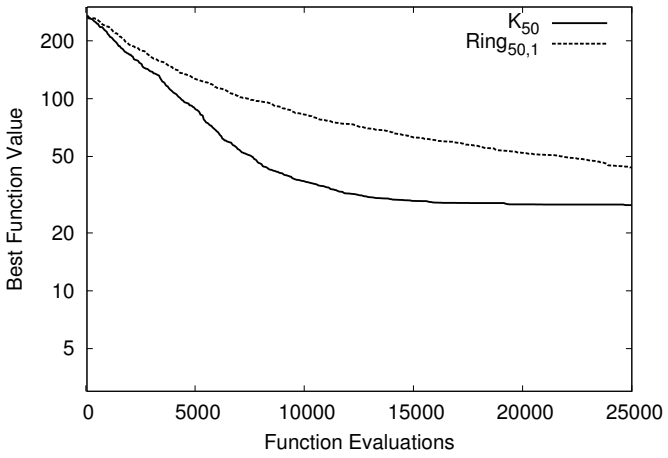

Fig. 3. PSO performance on Rastrigin for conventional-sized K_{50} and $Ring_{50,1}$ swarms.

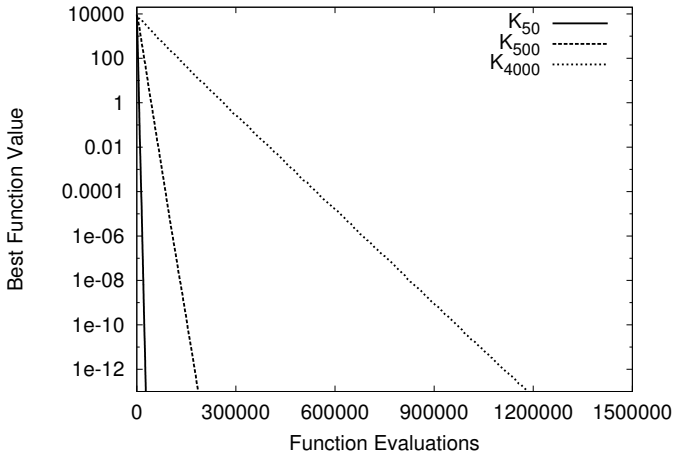

Fig. 2. PSO performance on Sphere for K_n with increasing swarm sizes.

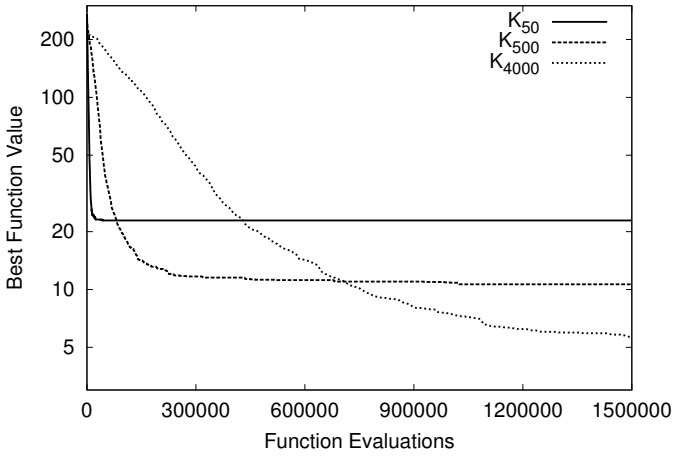

Fig. 4. PSO performance on Rastrigin for K_n with increasing swarm sizes.

The remainder of this section reviews the behavior of various sizes of swarms with the K_n and $Ring_{n,1}$ topologies. Performance is evaluated with respect to the Sphere, Rastrigin, and Griewank benchmark functions in order of increasing deceptiveness. All plots represent the average over 20 runs.

A. Sphere

The simplest benchmark function is the *Sphere* or parabola, expressed by the function $f_S(\vec{x}) = \sum_{i=1}^{D} x_i^2$. We use the 20-dimensional variant with the feasible region $[-50, 50]^{20}$. Figure 1 shows the best value in the swarm at each iteration of PSO with both K_{50} and $Ring_{50,1}$. The K_{50} swarm is more effective because its particles share information as quickly as possible, while information in $Ring_{50,1}$ takes time to propogate from one particle to the next.

Since Sphere is a smooth unimodal function, additional particles do very little to improve performance. Figure 2 shows that relative to function evaluations, PSO performs better with fewer particles. In other words, a smaller swarm with more iterations optimizes Sphere better than a larger swarm with fewer iterations.

B. Rastrigin

The more complicated *Rastrigin* function is given by $f_R(\vec{x}) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$. We use the 20-dimensional variant with the feasible region $[-5.12, 5.12]^{20}$. Figure 3 shows K_{50} and $Ring_{50,1}$ for Rastrigin. As with Sphere, the $Ring_{50,1}$ topology is less effective because information propogates slowly.

Unfortunately, the K_n swarm is prone to prematurely converge to local optima because Rastrigin is noisy. There are many variants of PSO that address the problem of premature convergence by encouraging exploration. Figure 4, which shows the performance of PSO with the K_n topology for various values of n, demonstrates that simply adding particles can increase exploration. Since a smaller swarm can do more iterations with fewer function evaluations, it initially outperforms a larger swarm. However, it converges to a local optimum and gets passed up by the larger swarm. Thus,

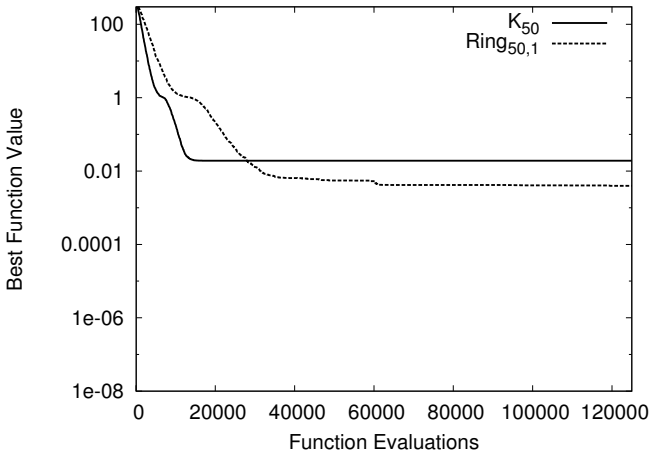

Fig. 5. PSO performance on Griewank for conventional-sized K_{50} and $Ring_{50,1}$ swarms.

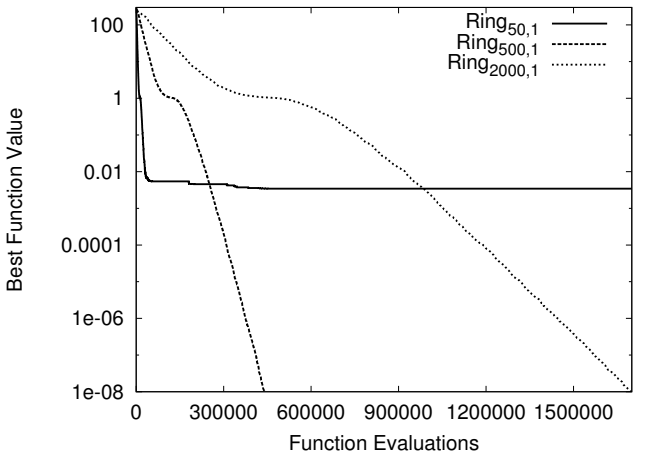

Fig. 6. PSO performance on Griewank for $Ring_{n,1}$ with increasing swarm sizes.

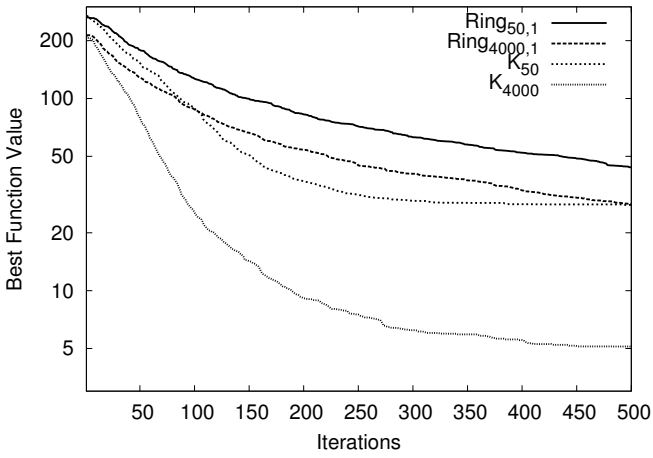

Fig. 7. PSO performance on Rastrigin with respect to iterations with various topologies.

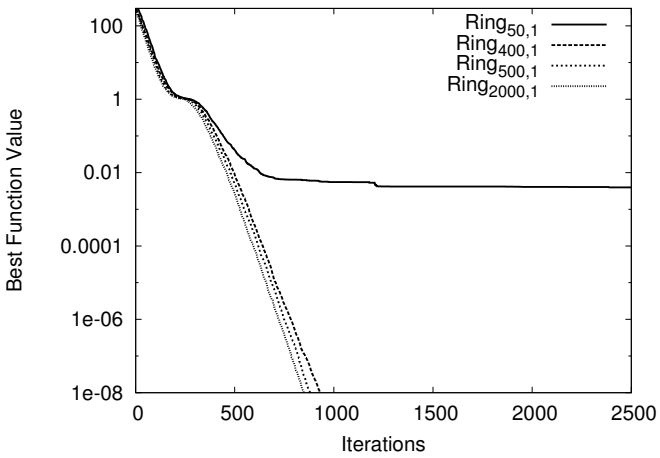

Fig. 8. PSO performance on Griewank with respect to iterations for $Ring_{n,1}$ with increasing swarm sizes.

adding particles can improve performance, even in a sequential implementation of PSO.

C. Griewank

The *Griewank* benchmark function is defined by the equation $f_G(\vec{x}) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \Pi_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$. We use the 20-dimensional variant with the feasible region $[-600, 600]^{20}$. As shown in Figure 5, the K_n topology is highly susceptible to premature convergence, and it is generally recommended to use the $Ring_{n,1}$ topology instead.

Although it performs better than K_{50}, even $Ring_{50,1}$ tends to get stuck in local minima. Figure 6 shows that adding more particles increases exploration, and $Ring_{n,1}$ does not prematurely converge for any $n \geq 500$. However, adding particles to K_n did not see the same improvement, and K_{4000} did not perform significantly better than K_{50}. Apparently information must fundamentally move slowly through the swarm for PSO to successfully optimize Griewank.

D. Parallel With Expensive Function Evaluations

Sections IV-A, IV-B, and IV-C showed how large swarm sizes can improve the convergence of PSO even with respect to the number of function evaluations. However, in a parallel implementation of PSO, it may be more appropriate to consider the number of iterations, particularly if function evaluations are expensive [11].

When judged by function evaluations, smaller swarms initially performed better for Rastrigin, but larger swarms were less prone to premature convergence (Figure 4). In contrast, Figure 7 shows that with respect to the number of iterations, adding particles always improves performance. The same is true for Griewank, as shown in Figure 8. This conventional-style plot is somewhat misleading, since it shows the average result over 20 runs. Of these 20 swarms, some find the global minimum and others get stuck at local minima, resulting in a high variance. Figure 9 addresses this as in [5] by showing the number of swarms with best values below 10^{-6}, which is

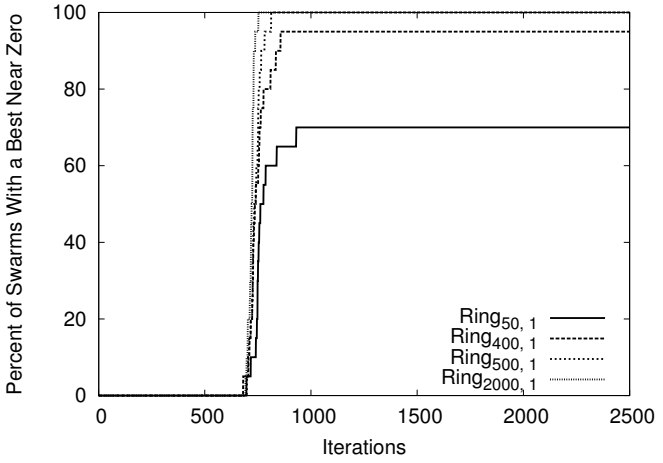

Fig. 9. Success rate on Griewank with respect to iterations for $Ring_{n,1}$ with various swarm sizes.

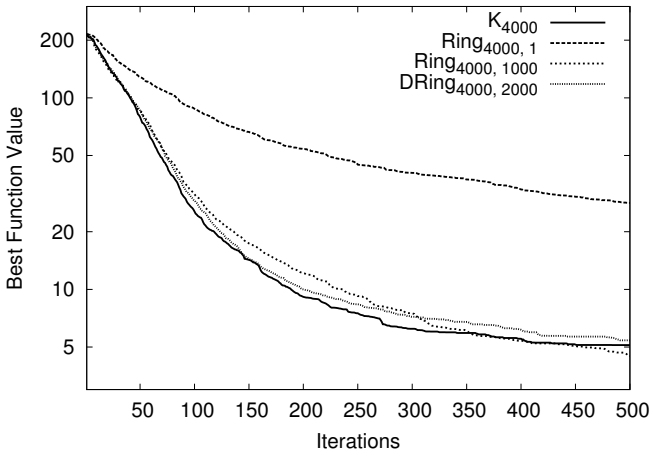

Fig. 10. PSO performance on Rastrigin with ring topologies.

below any of the local minima. The figure shows that larger swarms find the global minimum in fewer iterations and with a higher success rate.

V. Inexpensive Function Evaluations Relative to Communication

In a parallel implementation of PSO, communication overhead may significantly affect performance. For functions that take a long time to evaluate, this communication may be negligible. However, for simpler functions, limiting communication may be critical for performance [11].

Functions such as Griewank, which is best optimized with $Ring_{n,1}$, inherently demand low communication. The ring topology only needs to send 2 messages per particle for a total of $2n$ messages per iteration. Since the communication overhead is naturally low for such functions, they are naturally efficient in a parallel implementation of PSO.

Functions like Sphere and Rastrigin are best optimized when information flows quickly through the swarm. The K_n topology requires $n-1$ messages per particle for a total of n^2-n messages per iteration. In this section, we will focus on reducing communication for such functions when communication is expensive or function evaluation is inexpensive.

A. Generalized Rings

As noted in Section III, the ring topology has been generalized as $Ring_{n,k}$, where each particle sends its personal best to k neighbors on each side. Another variant of the ring topology is a one-way ring $DRing_{n,k}$, which is like $Ring_{n,k}$ except that information is only sent in one direction. This is given by the neighborhood function:

$$\nu_{DRing}(i,n,k) = \{i, (i+1) \bmod n, \ldots, (i+k) \bmod n\}$$

The $Ring_{n,k}$ topology requires $2kn$ messages per iteration, and $DRing_{n,k}$ requires kn messages per iteration. If $k = n$, then $Ring_{n,k}$ is identical to K_n. The k parameter can be tweaked to compromise the tradeoff between information flow and communication overhead. Figure 10 shows Rastrigin with a few ring topologies, demonstrating that communication can be halved without affecting performance.

B. Random Topology

The random topology $Rand_{n,k}$ is a dynamic topology where each particle randomly picks k different neighbors each iteration. Its neighborhood function is:

$$\nu_{Rand}(i,n,k) = \{i, U_1, U_2, \ldots, U_k\}$$

where U_j is a uniform random integer between 0 and $n-1$. These random numbers are drawn independently in each iteration and for each particle.

The $Rand_{n,k}$ topology has kn messages per iteration. When k is large, particles send messages to most other particles in the swarm, making the topology similar to K_n. A topology called stochastic star, which differs primarily by using the informants model in its definition, also has this similarity to K_n [12].

Recall that in a dynamic topology, a particle's neighborhood may change between iterations. The PSO algorithm is ambiguous about how to update a particle's neighborhood best in this situation. With a *dynamic neighborhood best*, the particle would set its neighborhood best by taking the best personal best in its neighborhood, throwing out the prior neighborhood best. With a *stable neighborhood best*, the particle would retain the prior neighborhood best, replacing it only if some particle in its neighborhood has a better personal best.

Figure 11 shows the performance of the $Rand_{n,\frac{n}{2}}$ topology on the Rastrigin benchmark function. With a stable neighborhood best, performance improves steadily with the number of particles. However, a swarm with a dynamic neighborhood best exhibits high variance and terrible overall performance. These results show that with a dynamic neighborhood best, a particle wastes the information from neighbors because it gets tugged toward an ever-changing neighborhood best instead of focusing on a productive part of the space.

Figure 12 shows the performance of $Ring_{n,1}$ and $Rand_{n,2}$ on the Rastrigin benchmark function. The two topologies

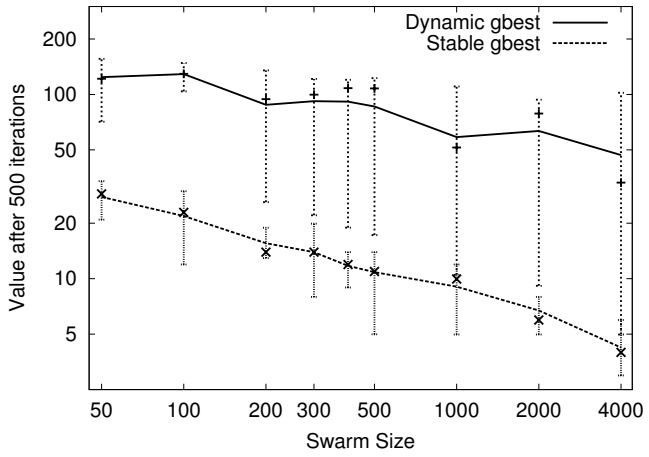

Fig. 11. PSO performance for dynamic and stable neighborhood bests on Rastrigin. The primary plots show the average performance over 20 independent runs, and the error bars show the median and the 10th and 90th percentiles.

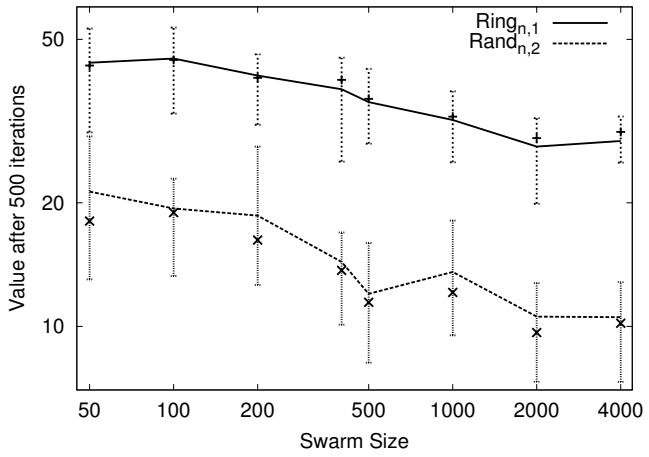

Fig. 12. PSO performance for $Ring_{n,1}$ and $Rand_{n,2}$ on Rastrigin. The primary plots show the average performance over 20 independent runs, and the error bars show the median and the 10th and 90th percentiles.

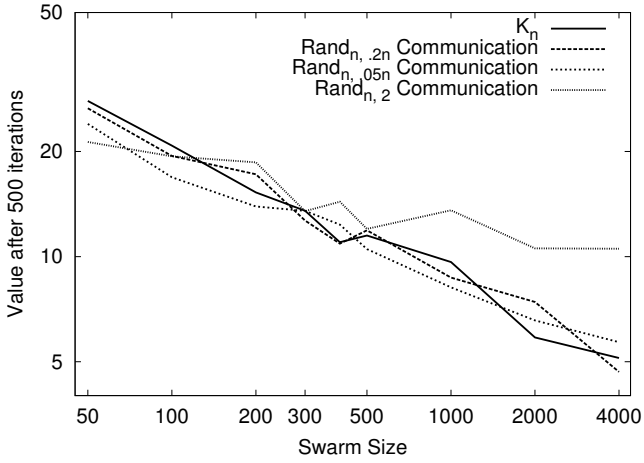

Fig. 13. PSO performance for K_n, $Rand_{n,\frac{n}{5}}$, and $Rand_{n,\frac{n}{20}}$ on Rastrigin. The primary plots show the average performance over 20 independent runs, and the error bars show the median and the 10th and 90th percentiles.

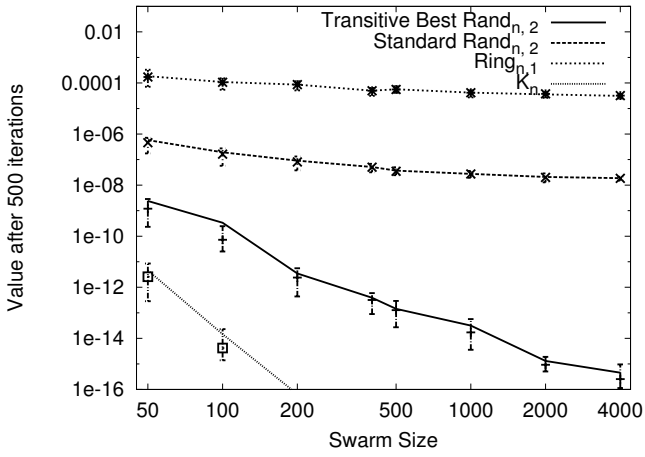

Fig. 14. PSO performance on Sphere for transitive best $Rand_{n,2}$, $Rand_{n,2}$, and $Ring_{n,2}$. The primary plots show the average performance over 20 independent runs, and the error bars show the median and the 10th and 90th percentiles.

require the exact same amount of communication as each particle sends its personal best to two neighbors. The random topology performs better, even with a small swarm, and adding additional particles for $Rand_{n,2}$ has a greater effect than for $Ring_{n,1}$.

Figure 13 compares the performance of K_n, $Rand_{n,\frac{n}{5}}$ (20% communication), $Rand_{n,\frac{n}{20}}$ (5% communication), and $Rand_{n,2}$ on the Rastrigin benchmark function. Note that with the exception of $Rand_{n,2}$ they give approximately the same results despite the random topologies requiring significantly less communication.

C. Hearsay PSO

Sphere and Rastrigin are better optimized with more shared information. Although the $Rand_{n,k}$ topology is able to reduce communication dramatically without affecting performance, information propogates too slowly through the swarm when k is small. For example, $Rand_{n,2}$ performs worse than the more communicative random topologies in Figure 13. However, a modification of the PSO algorithm restores information flow without actually increasing the communication between particles. In *Hearsay PSO*, particles convey what they have heard and not just what they have seen. Specifically, they have a "transitive best" and send their neighborhood best to neighbors rather than their personal best.

Figure 14 shows that Hearsay PSO with $Rand_{n,2}$ outperforms standard PSO on Sphere using topologies with the same amount of communication. Figure 15 makes the same comparisons for Rastrigin and shows that for large swarm sizes, Hearsay PSO using $Rand_{n,2}$ is competitive with Standard PSO using K_n, both of which slightly outperform Standard PSO with $Rand_{n,2}$. Figure 16 compares a few topologies and shows that when the number of messages is the primary cost,

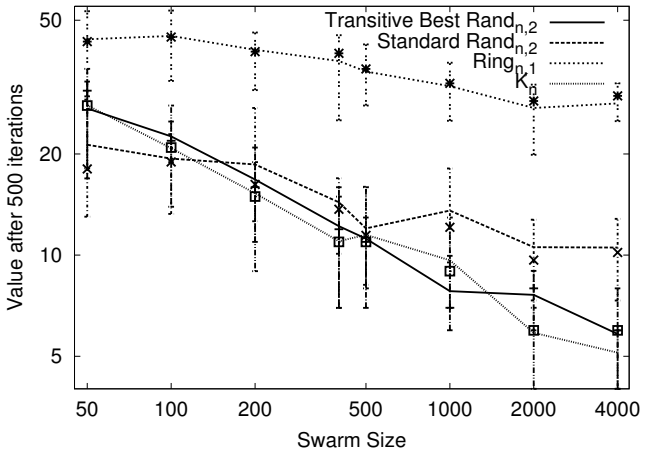

Fig. 15. PSO performance on Rastrigin for transitive best $Rand_{n,2}$, $Rand_{n,2}$, and $Ring_{n,2}$. The primary plots show the average performance over 20 independent runs, and the error bars show the median and the 10th and 90th percentiles.

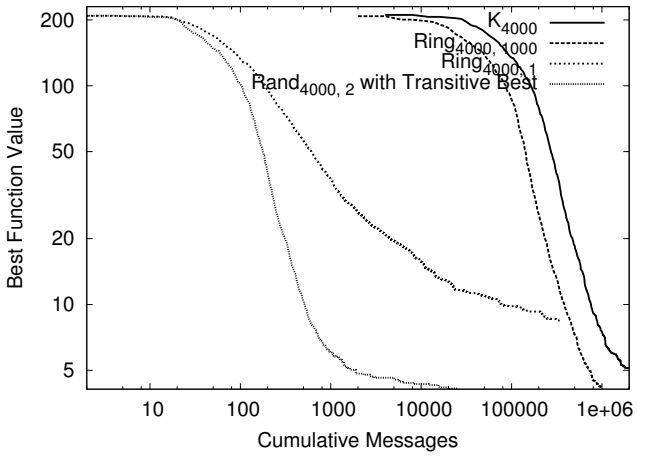

Fig. 16. PSO performance on Rastrigin with respect to communication.

Hearsay PSO with $Rand_{n,2}$ outperforms standard PSO with most other topologies.

D. Subswarms

Another way to parallelize PSO is to perform independent runs, or subswarms, on different processors and take the best result after the runs have completed. Formally, $Subswarms_{n,k}$ is a static topology consisting of n independent components of k fully-connected particles. This requires no modification of the PSO implementation and works surprisingly well. If local communication is inexpensive, then the subswarms topology is extremely efficient because it involves no external communication.

"Communicating subswarms" occasionally share neighborhood bests between subswarms. If external communication is expensive, this can be effective because messages are not sent every iteration. The communicating subswarms topology $CommunicatingSwarms$ is dynamic: it is $Subswarms_{n,k}$ during most iterations but is K_{nk} every ith iteration. More sophisticated ways to connect multiple subswarms have been proposed [9], [13], but communicating subswarms are particularly simple to parallelize.

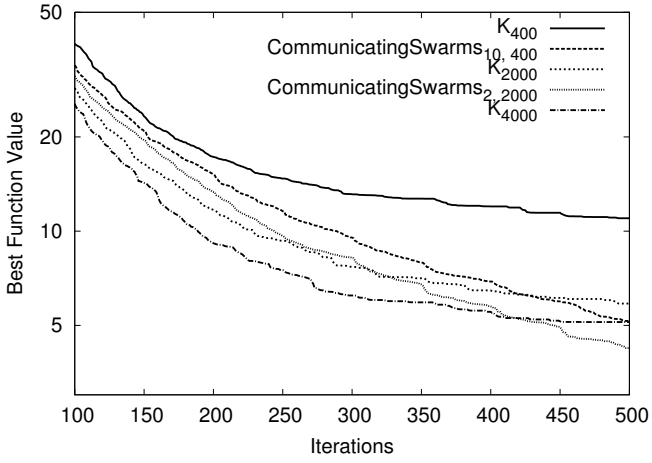

Fig. 17. PSO performance on Rastrigin with communicating subswarms.

Figure 17 shows two $CommunicatingSwarms$ topologies with 4000 total particles in comparison with related K_n swarms. These subswarms communicate every 50 iterations, and as a result, there is an interesting dip after every 50th iteration. Communicating subswarms outperform a single swarm of the smaller size, and the results even seem promising in comparison with a single large swarm.

VI. Conclusions and Future Work

We have shown that large swarms can improve the performance of Particle Swarm Optimization on several standard benchmark functions. Large swarms inherently delay convergence, which can be helpful in optimizing deceptive and moderately deceptive functions such as Griewank and Rastrigin. These results apply to both sequential and parallel environments.

However, performance gains are particularly compelling in the context of parallel computation. Where faster convergence is desirable but where communication is expensive relative to function evaluations, we have proposed adaptations of PSO. We have given a more formal approach to topologies, making it easier to describe complex and dynamic topologies. Random topologies with a stable neighborhood best increase the flow of information without destabilizing motion. Hearsay PSO allows particles to communicate more than just their personal best, which helps information spread even more quickly. This allows topologies to be more sparsely connected, which reduces the communication overhead of PSO in large scale parallel environments.

The subswarms topology deals with the situation where local communication is inexpensive (such as between cores in a multicore processor) and external communication is more expensive (like between multicore processors in a cluster) or impossible (with serial-only code).

In the future, we expect to extend our work with large swarms. Large scale parallel computing environments open the door to very large swarms. A cluster of a thousand 8-core processors could allow optimization with hundreds of thousands or perhaps millions of particles. Based on our experiences so far it seems likely that we will discover new issues that will drive further algorithmic advances. For example, we expect that large swarms like larger societies will demand richer communication for the swarm to work effectively. Sharing information such as history may allow particles to move more intelligently without increasing communication overhead. We suspect that knowledge of subsets of particles ("partially fully-informed PSO") or may work well in cases where "fully-informed PSO" has failed for large swarms [14].

Space has limited this work to a subset of the interesting and applicable benchmark functions. In future work we need to consider more benchmarks. The use of directed graphs has also allowed us to describe and consider interesting topologies, but we have only begun to develop topologies that might be applicable in parallel computing environments.

REFERENCES

[1] James Kennedy and Russell C. Eberhart. Particle swarm optimization. In *International Conference on Neural Networks IV*, pages 1942–1948, Piscataway, NJ, 1995.

[2] Daniel Bratton and James Kennedy. Defining a standard for particle swarm optimization. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 120–127, 2007.

[3] J. F. Schutte, J. A. Reinbolt, B. J. Fregly, R. T. Haftka, and A. D. George. Parallel global optimization with the particle swarm algorithm. *International Journal for Numerical Methods in Engineering*, 61(13):2296–2315, December 2004.

[4] J.R. Perez and J. Basterrechea. Particle swarm optimization for antenna far-field radiation pattern reconstruction. In *36th European Microwave Conference*, pages 687–690, 2006.

[5] Rui Mendes. *Population Topologies and Their Influence in Particle Swarm Performance*. PhD thesis, Escola de Engenharia, Universidade do Minho, 2004.

[6] Maurice Clerc and James Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, 2002.

[7] Russell Merris. *Graph Theory*. John Wiley & Sons, New York, 2001.

[8] Arvind S. Mohais, Christopher Ward, and Christian Posthoff. Randomized directed neighborhoods with edge migration in particle swarm optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 1, pages 548–555, 2004.

[9] J. J. Liang and P. N. Suganthan. Dynamic multi-swarm particle swarm optimizer. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 124–129, 2005.

[10] Maurice Clerc. TRIBES - un exemple d'optimisation par essaim particulaire sans paramètres de contrôle. In *Optimisation par Essaim Particulaire*, Paris, France, 2003.

[11] Andrew W. McNabb, Christopher K. Monson, and Kevin D. Seppi. Parallel PSO using MapReduce. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 7–14, September 2007.

[12] Vladimiro Miranda, Hrvoje Keko, and Álvaro Jaramillo Duque. Stochastic star communication topology in evolutionary particle swarms. *International Journal of Computational Intelligence Research*, 4(2), 2008.

[13] Johannes Jordan, Sabine Helwig, and Rolf Wanka. Social interaction in particle swarm optimization, the ranked FIPS, and adaptive multi-swarms. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, pages 49–56. ACM, 2008.

[14] Marco A. Montes de Oca and Thomas Stützle. Convergence behavior of the fully informed particle swarm optimization algorithm. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, pages 71–78. ACM, 2008.