The Apiary Topology: Emergent Behavior in Communities of Particle Swarms

Andrew McNabb and Kevin Seppi

Computer Science Department, Brigham Young University
{a,k}@cs.byu.edu

Abstract. In the natural world there are many swarms in any geographical region. In contrast, Particle Swarm Optimization (PSO) is usually used with a single swarm of particles. We define a simple new topology called Apiary and show that parallel communities of swarms give rise to emergent behavior that is fundamentally different from the behavior of a single swarm of identical total size. Furthermore, we show that subswarms are essential for scaling parallel PSO to more processors with computationally inexpensive objective functions. Surprisingly, subswarms are also beneficial for scaling PSO to high dimensional problems, even in single processor environments.

Keywords: Particle Swarm Optimization, parallel PSO, swarm topology, subswarms, multiple swarms, parallel computation.

1 Introduction

Particle Swarm Optimization (PSO) is a continuous function optimization algorithm inspired by the flocking behaviors of birds and insects. It is typically used with small swarms of 20 to 50 particles organized in simple topologies that do not fully reflect the complex social interactions of insects. In agriculture, for example, bees are managed in sets of hives called apiaries. The number of hives in an apiary usually ranges from 10 to 150.

Using conventional topologies, a single swarm of particles often fails to scale both to large numbers of processors and to high-dimensional problems. First, with a large number of processors and an inexpensive objective function, communication costs make parallel PSO with a single swarm impractical. Parallel PSO naturally works well for problems with computationally expensive function evaluations, but for inexpensive objective functions, the time to communicate a single position can exceed the time to perform a function evaluation. Second, for high-dimensional problems, particles are prone to premature convergence. Even for Sphere, the simplest of benchmark functions, standard PSO struggles to find the global optimum when the number of dimensions is 400 or greater.

Multiple swarms have been used to scale parallel PSO for inexpensive objective functions but have not been considered for scaling to high-dimensional problems. Semi-independent swarms of particles provide a natural way to parallelize the computation of PSO across a set of processors without requiring

C.A. Coello Coello et al. (Eds.): PPSN 2012, Part II, LNCS 7492, pp. 164–173, 2012.

instantaneous communication [1]. However, the behavior of subswarms has not been explored, particularly with respect to high-dimensional problems.

The Apiary topology, proposed in Section 3, spreads the population of particles among a set of small subswarms. In this topology, a subswarm is a social entity which lies between the individual particle and the full population and which serves as another source of emergent behavior. Each subswarm consists of a fixed set of particles and is mostly independent of other subswarms. Periodically, a single particle in each subswarm communicates with a few particles in other subswarms. This communication between subswarms is rare and limited, so computation is particularly well suited to parallel computation.

The Apiary topology helps PSO scale, both to large numbers of processors and to high-dimensional objective functions. Unlike some other proposed PSO techniques using subswarms, this topology is simple, clearly defined, and appropriate for parallel PSO. Experiments, described in Section 4, show significant improvements over standard PSO. Even in single processor environments, apiaries produce better results in the same time and are less prone to premature convergence for every benchmark function we tested. These results are presented and discussed in Section 4.1. The standard parameters are justified in Section 4.2, along with indications of when these parameters might be changed. Parallel PSO with Apiary is compared in Section 4.3. Despite inexpensive functions being particularly challenging for parallelization, the run time is reduced from 256 minutes with a single processor to 17 minutes with 40 processors.

2 Background Material: Particle Swarm Optimization

Particle Swarm Optimization, proposed by Kennedy and Eberhart [2], simulates the motion of particles in the domain of an objective function. These particles search for the global optimum by evaluating the function as they move. During each iteration, each particle is pulled toward the best position it has sampled, known as the *personal best*, and the best position of any particle in its neighborhood, known as the *neighborhood best*.

Constricted PSO is generally considered the standard variant [3]. Each particle's position $\boldsymbol{x}_0$ and velocity $\boldsymbol{v}_0$ are initialized to random values based on a function-specific feasible region. During iteration t, the following equations update the i^{th} component of a particle's position $\boldsymbol{x}_t$ and velocity $\boldsymbol{v}_t$ with respect to the personal best $\boldsymbol{p}_{t-1}$ and neighborhood best $\boldsymbol{n}_{t-1}$ from the preceding iteration:

$$v_{t,i} = \chi \left[v_{t-1,i} + \phi^P u^P_{t-1,i}(x^P_{t-1,i} - x_{t-1,i}) + \phi^N u^N_{t-1,i}(x^N_{t-1,i} - x_{t-1,i}) \right] \quad (1)$$

$$x_{t,i} = x_{t-1,i} + v_{t,i} \quad (2)$$

where x^P is the personal best, x^N is the neighborhood best, ϕ^P and ϕ^N are usually set to 2.05, $u^P_{t,i}$ and $u^N_{t,i}$ are samples drawn from a standard uniform distribution, and $\chi = 2/\left|2 - \phi - \sqrt{\phi^2 - 4\phi}\right|$ where $\phi = \phi^P + \phi^N$ [4].

The neighborhoods within a swarm are defined by the *topology* graph. The choice of topology can have a significant effect on performance [5]. Additionally,

the topology determines task dependencies and overhead in parallel PSO [6]. The $Ring_{50}$ topology, a swarm of 50 particles where each particle has a single neighbor on either side, is a standard starting point [3].

3 The Apiary Topology

The Apiary topology is a dynamic topology of independent subswarms which occasionally communicate with each other. Each subswarm has an *inner topology*, and the subswarms are connected in an *outer topology*. In most iterations, the neighbors of each particle are defined purely by the inner topology of its subswarm. After a fixed number of independent *subiterations*, each subswarm communicates with its neighboring subswarms, as defined by the outer topology. Each subswarm sends its neighbors the best value from any of its particles. It updates the neighborhood best of a fixed set of particles (the neighborhood of the first particle in the swarm) with the values from neighboring subswarms.

In the Apiary topology, subswarms share important characteristics with communities in nature. Just as each bee colony has its own social structure, each subswarm has its own particles and its own topology. Like bee colonies, the subswarms are independent and rarely interact. Curiously, bees occasionally allow foreign forage bees to enter a hive if they are fully loaded [7], and the native bees will be able to learn from those foreign bees if they are from another colony or even another species [8]. Likewise, a single particle in each subswarm occasionally engages in light communication with neighboring swarms. In this simple structure, subswarms are simple entities with a balance of independence and interaction that favors emergent behavior.

This approach contrasts with previous attempts to define subpopulations in PSO. Dynamic Multi-Swarm PSO [9] periodically shuffles by reassigning all particles to random subswarms. This global reshuffling increases the amount of communication required in parallel PSO and is incompatible with asynchronous parallel PSO [10]. In contrast, neighborhoods in the Apiary topology are deterministic and require very little communication. Section 4.1 compares the performance of Dynamic Multi-Swarm PSO with that of the Apiary topology. Most subswarm approaches have introduced strategies—some of them quite complex—to manage the migration of particles between subswarms [11,12,13,14]. Other works have used subswarm-style topologies within a limited context [6], including completely independent subswarms [1]. Romero and Cotta's island-structured swarms [11], is limited to small numbers of large subswarms and low-dimensional problems, and its conclusions do not seem to apply to high-dimensional problems. In contrast to other approaches, the Apiary topology is static and thus well suited to any implementation of parallel PSO, and it requires very little communication between subswarms.

The inner and outer topologies, as well as the number of subiterations, are changeable parameters. We recommend *Ring* for both the outer and inner topologies, with a starting point of 5 particles per subswarm, 40 total subswarms, and 100 subiterations. These recommendations are justified in Section 4.2.

4 Experimental Results

The Apiary topology provides significant improvements for both serial and parallel PSO with respect to a variety of benchmark functions. Benchmark functions are computationally inexpensive enough for large-scale experimentation but share interesting properties with challenging real-life problems. We use the Ackley, Rastrigin, Rosenbrock, Schwefel 1.2, and Sphere benchmark functions [15] with both 250 and 500 dimensions. Experiements were run on a Linux cluster consisting of 320 nodes (Dell PowerEdge M610). Each node is equipped with two quad-core Intel Nehalem processors (2.8 GHz) and 24 GB of memory.

Each experiment was repeated at least 40 times. We report the median instead of the mean because these distributions are skewed. The 10^{th} and 90^{th} percentiles illuminate both the variability and skewness. We determine statistical significance using a one-sided Monte Carlo permutation test [16]. A t-test would be inappropriate because it uses the mean statistic and assumes a normal distribution, which we can not assume in part because of skew. Each table cell is bolded if it is better than every other entry in its row with a p-value of 0.05.

Each table and plot presents either the median number of evaluations required to reach a threshold or the median best value at a fixed number of evaluations or iterations. The notation $Ring_n$ denotes a ring topology where each particle has one neighbor on each side, and $Ring_m$–$Ring_n$ denotes an Apiary topology with a $Ring_m$ outer topology and a $Ring_n$ inner topology. Each benchmark function is accompanied by its dimensionality, for example, "Sphere-500."

The balance of this section seeks to identify some of the most interesting observation and give greater clarity and meaning to these results. Section 4.1 compares the $Ring_{40}$–$Ring_5$ apiary with the standard recommendation of $Ring$. Section 4.2 justifies the particular choice of $Ring_{40}$–$Ring_5$ as a standard starting point. Finally, Section 4.3 demonstrates the suitability of the Apiary topology to parallel PSO by demonstrating its efficiency in a typical parallel environment.

4.1 Apiaries in Serial PSO

Limiting the interaction between subswarms to once every 100 iterations might be expected to compromise the performance of serial PSO in exchange for improved parallel efficiency, but this social organization in fact improves performance even in serial PSO. Figures 1 and 2 show the progress toward convergence for 500 dimensional Rastrigin and Sphere respectively. The $Ring_{40}$–$Ring_5$ apiaries require the same number of evaluations per iteration as the $Ring_{200}$ swarms, but they perform far better than the individual $Ring$ swarms. Note that the $Ring_{200}$ swarm in Figure 2 converges more slowly than the $Ring_{50}$ swarm because it requires more evaluations per iteration.

One might wonder whether the performance of the Apiary topology are dependent on the social interactions or whether they are merely due to the repetition of a high-variance experiment. After all, running 40 independent swarms of 5 particles would be expected to perform better than a single swarm of 5 particles. Figure 2 includes the abysmal results of such an $Independent_{40}$–$Ring_5$ topology,

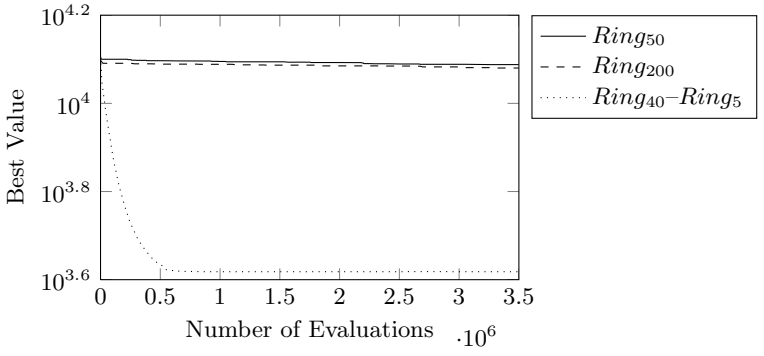


Fig. 1. Convergence plot for Rastrigin in serial PSO, comparing an apiary (using 100 subiterations) with a swarm of the same total number of total particles (200) and a swarm of 50 particles

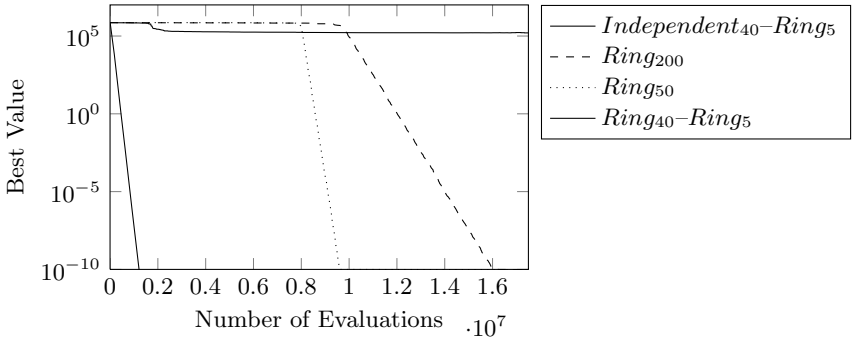


Fig. 2. Convergence plot for Sphere in serial PSO, comparing an apiary (using 100 subiterations) with a swarm of the same number of total particles (200) and a swarm of 50 particles

thus dispelling this possibility. The difference between the independent swarms and the apiary demonstrates emergent behavior.

We include results for the full range of benchmark functions in tabular form. In the case of Sphere, all runs of all PSO variants eventually converge to the global minimum. Table 1

reports the number of function evaluations to convergence. Table 2 reports the best value obtained at a fixed number of function evaluations for the other benchmark functions. The fixed number of evaluations for each function are equivalent to about 6 hours of computation, specifically: 6×10^6 for Ackley-250 and Ackley-500, 1×10^7 for Rastrigin-250, 3.5×10^6 for Rastrigin-500, 1×10^7 for Rosenbrock-250, 5×10^6 for Rosenbrock-500, 6×10^6 for Schwefel1.2-250, and 2×10^6 for Schwefel1.2-500. In all cases the apiary is best with statistical significance. Though the results for the Ackley function are statistically significant, the difference is small.

Table 1. Median number of function evaluations to reach a value of 10^{-10}. The best cell in each row is bolded if statistically significant

Function	$Ring_{50}$	$Ring_{200}$	$Ring_{40}$–$Ring_5$
Sphere-250	7.8×10^5	3×10^6	$\mathbf{5.9\times10^5}$
(10^{th}, 90^{th})	(7.6×10^5, 8.1×10^5)	(3×10^6, 3.1×10^6)	$\mathbf{(5.9\times10^5, 6\times10^5)}$
Sphere-500	9.5×10^6	1.6×10^7	$\mathbf{1.2\times10^6}$
(10^{th}, 90^{th})	(3.5×10^6, 2.6×10^7)	(1×10^7, 4×10^7)	$\mathbf{(1.2\times10^6, 1.2\times10^6)}$

Table 2. Median best value at a fixed number of function evaluations

Function	$Ring_{50}$	$Ring_{200}$	$Ring_{40}$–$Ring_5$
Ackley-250	20	20	**20**
(10^{th}, 90^{th})	(20, 20)	(20, 20)	**(20, 20)**
Ackley-500	20	20	**20**
(10^{th}, 90^{th})	(20, 21)	(20, 21)	**(20, 20)**
Rastrigin-250	2.6×10^3	2.3×10^3	$\mathbf{1.9\times10^3}$
(10^{th}, 90^{th})	(2.1×10^3, 2.9×10^3)	(2×10^3, 2.5×10^3)	$\mathbf{(1.7\times10^3, 2.1\times10^3)}$
Rastrigin-500	1.2×10^4	1.2×10^4	$\mathbf{4.1\times10^3}$
(10^{th}, 90^{th})	(1.1×10^4, 1.3×10^4)	(8.5×10^3, 1.2×10^4)	$\mathbf{(3.9\times10^3, 4.5\times10^3)}$
Rosenbrock-250	70	3.7×10^2	**0.0012**
(10^{th}, 90^{th})	(0.029, 3.4×10^2)	(2.8×10^2, 4.5×10^2)	$\mathbf{(6.1\times10^{-9}, 4)}$
Rosenbrock-500	4.3×10^{12}	4.1×10^{12}	$\mathbf{8.8\times10^2}$
(10^{th}, 90^{th})	(4×10^{12}, 4.6×10^{12})	(3.8×10^{12}, 4.3×10^{12})	$\mathbf{(7\times10^2, 1.1\times10^3)}$
Schwefel1.2-250	7.4×10^4	2.3×10^5	$\mathbf{1.6\times10^4}$
(10^{th}, 90^{th})	(4.6×10^4, 1.5×10^5)	(2.1×10^5, 2.7×10^5)	$\mathbf{(1.2\times10^4, 2.1\times10^4)}$
Schwefel1.2-500	1.6×10^6	2.2×10^6	$\mathbf{8.1\times10^5}$
(10^{th}, 90^{th})	(1.1×10^6, 2.3×10^6)	(1.8×10^6, 2.8×10^6)	$\mathbf{(7.1\times10^5, 9.4\times10^5)}$

For some functions, the Apiary topology outperforms Dynamic Multi-Swarm PSO [9] (DMS-PSO) in serial, while for other functions, DMS-PSO outperforms the Apiary topology. For Sphere-500, the Apiary topology finds the minimum faster with a small but statistically significant advantage, while for Sphere-250, the situation is reversed (the full table is omitted due to space constraints). Table 3 shows similarly mixed results for the other benchmark functions.

4.2 Apiary Parameters

We now justify the basic apiary parameters of a $Ring_{40}$ outer topology, a $Ring_5$ inner topology, and 100 subiterations. General recommendations set $Ring_{50}$ as a standard swarm topology [3] or even higher for difficult problems [6]. Previous subswarm topologies have suggested that 50–100 particles per subswarm [11] or 32 particles per subswarm [1] give ideal performance. In contrast, we recommend starting with small subswarms of about 5 particles.

Increasing the number of particles per subswarm or the total number of subswarms provide improvements only in some circumstances. Table 4 compares

Table 3. Median best value at a fixed number of function evaluations. The topology is $Ring_{40}$–$Ring_{25}$ for Rastrigin and $Ring_{40}$–$Ring_5$ for Rosenbrock and Schwefel 1.2.

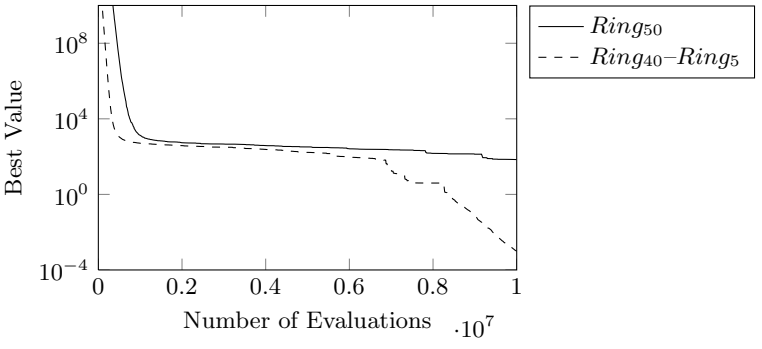
Function	Apiary	DMS-PSO
Rastrigin-250	1.5×10^3	$\mathbf{4.8\times10^2}$
(10^{th}, 90^{th})	(1.4×10^3, 1.6×10^3)	$\mathbf{(4\times10^2, 5.9\times10^2)}$
Rastrigin-500	3.3×10^3	$\mathbf{1.3\times10^3}$
(10^{th}, 90^{th})	(3×10^3, 3.6×10^3)	$\mathbf{(1.2\times10^3, 1.6\times10^3)}$
Rosenbrock-250	**0.0012**	1.3×10^2
(10^{th}, 90^{th})	$\mathbf{(6.1\times10^{-9}, 4)}$	(1.4, 2.3×10^2)
Rosenbrock-500	8.8×10^2	8.3×10^2
(10^{th}, 90^{th})	(7×10^2, 1.1×10^3)	(6.4×10^2, 9.8×10^2)
Schwefel1.2-250	$\mathbf{1.6\times10^4}$	7×10^4
(10^{th}, 90^{th})	$\mathbf{(1.2\times10^4, 2.1\times10^4)}$	(5.1×10^4, 8.9×10^4)
Schwefel1.2-500	$\mathbf{8.1\times10^5}$	1.4×10^6
(10^{th}, 90^{th})	$\mathbf{(7.1\times10^5, 9.4\times10^5)}$	(1.2×10^6, 1.7×10^6)

Table 4. Median best value at n function evaluations

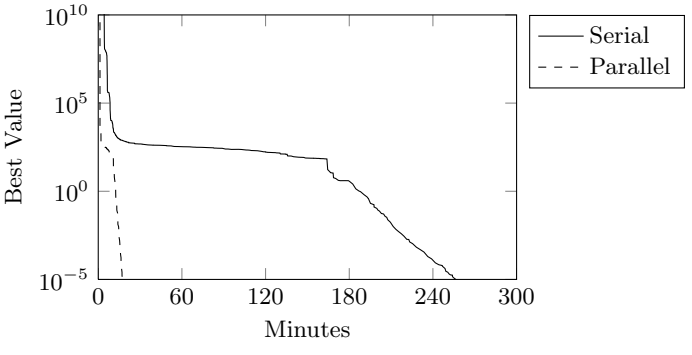
Function	$Ring_{40}$–$Ring_5$	$Ring_{200}$–$Ring_5$	$Ring_{40}$–$Ring_{25}$
Rastrigin-250	1.9×10^3	1.7×10^3	$\mathbf{1.5\times10^3}$
(10^{th}, 90^{th})	(1.7×10^3, 2.1×10^3)	(1.6×10^3, 1.9×10^3)	$\mathbf{(1.4\times10^3, 1.6\times10^3)}$
Rastrigin-500	4.1×10^3	3.8×10^3	$\mathbf{3.3\times10^3}$
(10^{th}, 90^{th})	(3.9×10^3, 4.5×10^3)	(3.6×10^3, 4×10^3)	$\mathbf{(3\times10^3, 3.6\times10^3)}$
Rosenbrock-250	**0.0012**	2.6×10^2	0.27
(10^{th}, 90^{th})	$\mathbf{(6.1\times10^{-9}, 4)}$	(2×10^2, 3.2×10^2)	(0.0016, 76)
Rosenbrock-500	$\mathbf{8.8\times10^2}$	5×10^3	9.4×10^2
(10^{th}, 90^{th})	$\mathbf{(7\times10^2, 1.1\times10^3)}$	(3.4×10^3, 1.4×10^4)	(8.2×10^2, 1.2×10^3)
Schwefel1.2-250	$\mathbf{1.6\times10^4}$	1.5×10^5	3.7×10^4
(10^{th}, 90^{th})	$\mathbf{(1.2\times10^4, 2.1\times10^4)}$	(1.3×10^5, 1.6×10^5)	(2.8×10^4, 4.8×10^4)
Schwefel1.2-500	$\mathbf{8.1\times10^5}$	1.6×10^6	1×10^6
(10^{th}, 90^{th})	$\mathbf{(7.1\times10^5, 9.4\times10^5)}$	(1.5×10^6, 1.8×10^6)	(8.8×10^5, 1.2×10^6)

$Ring_{40}$–$Ring_5$ to $Ring_{200}$–$Ring_5$, an apiary with 5 times as many subswarms, and to $Ring_{40}$–$Ring_{25}$, an apiary with 5 times as many particles per subswarm. For most of the benchmark functions, the $Ring_{40}$–$Ring_5$ apiary performs significantly better than either of the larger topologies. Likewise, the $Ring_{40}$–$Ring_5$ topology significantly outperforms the others for Sphere-250 and Sphere-500 (the table is omitted due to space). For such functions, the increased number of evaluations per iteration offsets any increased exploration provided by the larger swarms. On the other hand, both of the larger topologies are better for Rastrigin-250, Rastrigin-500, and Rosenbrock-500. As the number of local minima in Rastrigin increases exponentially with the number of dimensions, we conclude that larger swarms are preferable for highly multimodal objective functions.

Changing the communication between swarms can affect performance dramatically. Using a more connected outer topology, such as *Complete*, gives poor



(a) Convergence plot (in serial)



(b) Apiary in serial and parallel

Fig. 3. Convergence plots of the Apiary topology for the Rosenbrock function with respect to function evaluations and time

performance in serial PSO in addition to requiring more communication in parallel PSO. Sharing the best value of an arbitrary member of each subswarm instead of the best particle of each subswarm also reduces performance. Setting the number of subiterations to 100 is high enough to provide reasonable task granularity even for the least expensive benchmark functions in parallel PSO.

4.3 Parallel Performance of Apiaries

Benchmark functions are extremely inexpensive, yet despite the high relative cost of communication, the Apiary topology performs extremely well in parallel. Figure 3 shows the results for the Rosenbrock function with both serial and parallel computation. Performing 100 iterations on 5 particles requires only 0.2 seconds, and parallel PSO took about 0.5 seconds per iteration. With any realistically expensive function, the overhead of 0.3 seconds would be negligible.

In a parallel context with a large number of spare processors, there may be limited additional overhead in increasing the number of subswarms. In this light, we revisit the conclusions from Section 4.2. In this context, the number of

Table 5. Median best value at n iterations

Function	$Ring_{40}$–$Ring_5$	$Ring_{200}$–$Ring_5$	$Ring_{40}$–$Ring_{25}$
Rastrigin-250	1.9×10^3	1.7×10^3	$\mathbf{1.5\times10^3}$
(10^{th}, 90^{th})	(1.7×10^3, 2.1×10^3)	(1.6×10^3, 1.9×10^3)	$\mathbf{(1.4\times10^3, 1.6\times10^3)}$
Rastrigin-500	4.2×10^3	3.8×10^3	$\mathbf{3.4\times10^3}$
(10^{th}, 90^{th})	(3.9×10^3, 4.5×10^3)	(3.6×10^3, 4×10^3)	$\mathbf{(3\times10^3, 3.6\times10^3)}$
Rosenbrock-250	3.6×10^2	$\mathbf{2.5\times10^2}$	2.9×10^2
(10^{th}, 90^{th})	(2.6×10^2, 4.4×10^2)	$\mathbf{(1.9\times10^2, 3.2\times10^2)}$	(2.1×10^2, 3.5×10^2)
Rosenbrock-500	2×10^4	$\mathbf{4.2\times10^3}$	1.5×10^4
(10^{th}, 90^{th})	(4.5×10^3, 3.1×10^7)	$\mathbf{(3\times10^3, 9.4\times10^3)}$	(3.5×10^3, 7.5×10^6)
Schwefel1.2-250	1.7×10^5	$\mathbf{1.4\times10^5}$	1.7×10^5
(10^{th}, 90^{th})	(1.4×10^5, 2×10^5)	$\mathbf{(1.3\times10^5, 1.6\times10^5)}$	(1.3×10^5, 2×10^5)
Schwefel1.2-500	1.8×10^6	$\mathbf{1.6\times10^6}$	1.8×10^6
(10^{th}, 90^{th})	(1.6×10^6, 2.2×10^6)	$\mathbf{(1.5\times10^6, 1.8\times10^6)}$	(1.5×10^6, 2.1×10^6)

iterations of PSO is a more appropriate measure than the number of function evaluations [6]. With respect to iterations, Table 4 compares $Ring_{40}$–$Ring_5$ with $Ring_{200}$–$Ring_5$ and $Ring_{40}$–$Ring_{25}$, which loosely represent the situations where additional processors or time are available, respectively. If extra resources are available, they clearly provide improvements in the pursuit of better answers.

5 Conclusions and Future Work

Organizing particle swarms into communities of subswarms significantly improves the performance of PSO. We attribute the improvement to emergent behavior from the social interaction of particles. We speculate that small groups of particles might make progress on implicit subproblems. Likewise, subswarms might help other subswarms get unstuck if they have prematurely converged in individual dimensions. In any case, the behavior of particle swarm apiaries is not explained by amount of communication, but rather the structure of the swarms.

Furthermore, we have shown that apiaries are particularly well-suited to parallel computation. With low communication and adjustable task granularity, the topology is easily adapted to varying computational architectures. With an inexpensive benchmark function, parallel PSO was able to perform about 2 outer iterations per second and provide a speedup of 15 on 40 processors. For any non-trivial function, the performance would be even more pronounced. Unlike other multi-swarm topologies like DMS-PSO [9], which requires frequent global communication, the Apiary topology requires very little communication.

We believe there are several interesting areas that are open to future work. In particular, organizing subswarms into hierarchies is a promising possibility. Apiaries are effective with extremely small subswarms, so a hierarchical structure can be built with a low branching factor. For example, a three-layer apiary would only have $5^3 = 125$ particles, and a four-layer apiary would have $5^4 = 625$ particles, well within the range that can be computed on a medium-size cluster.

Acknowledgments. The Fulton Supercomputing Lab at Brigham Young University generously provided 10 processor-years of resources.

References

1. Schutte, J.F., Reinbolt, J.A., Fregly, B.J., Haftka, R.T., George, A.D.: Parallel global optimization with the particle swarm algorithm. International Journal for Numerical Methods in Engineering 61(13), 2296–2315 (2004)
2. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proc. International Conference on Neural Networks IV (1995)
3. Bratton, D., Kennedy, J.: Defining a standard for particle swarm optimization. In: Proc. IEEE Swarm Intelligence Symposium, pp. 120–127 (2007)
4. Clerc, M., Kennedy, J.: The particle swarm—explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6(1), 58–73 (2002)
5. Mendes, R.: Population Topologies and Their Influence in Particle Swarm Performance. PhD thesis, Universidade do Minho, Guimaraes, Portugal (2004)
6. McNabb, A., Gardner, M., Seppi, K.: An exploration of topologies and communication in large particle swarms. In: Proc. IEEE Congress on Evolutionary Computation, pp. 712–719 (2009)
7. Sammataro, D., Avitabile, A.: Beekeeper's Handbook, 3rd edn. Cornell University Press (1998)
8. Sample, I.: Bees translate dances of foreign species. The Guardian (June 3, 2008)
9. Liang, J., Suganthan, P.: Dynamic multi-swarm particle swarm optimizer. In: Proc. IEEE Swarm Intelligence Symposium, pp. 124–129 (2005)
10. Venter, G., Sobieszczanski-Sobieski, J.: A parallel particle swarm optimization algorithm accelerated by asynchronous evaluations. In: Proc. 6th World Congresses of Structural and Multidisciplinary Optimization (2005)
11. Romero, J., Cotta, C.: Optimization by Island-Structured Decentralized Particle Swarms. In: Reusch, B. (ed.) Computational Intelligence, Theory and Applications. AISC, vol. 33, pp. 25–33. Springer, Heidelberg (2005)
12. Chu, S.C., Pan, J.S.: Intelligent parallel particle swarm optimization algorithms. Parallel Evolutionary Computations, 159–175 (2006)
13. Lorion, Y., Bogon, T., Timm, I., Drobnik, O.: An agent based parallel particle swarm optimization—APPSO. In: Swarm Intelligence Symposium (2009)
14. Wang, H., Qian, F.: An improved particle swarm optimizer with shuffled subswarms and its application in soft-sensor of gasoline endpoint. In: Proc. International Conference on Intelligent Systems and Knowledge Engineering (2007)
15. Tang, K., Li, X., Suganthan, P., Yang, Z., Weise, T.: Benchmark functions for the CEC 2010 special session and competition on large scale global optimization. Technical report, IEEE Congress on Evolutionary Computation (November 2009)
16. Dwass, M.: Modified randomization tests for nonparametric hypotheses. The Annals of Mathematical Statistics 28(1), 181–187 (1957)