Matthew Gardner
Brigham Young University

Many problems in science can be solved analytically, with simple (or sometimes rather complicated) calculations. Computing Plank's constant, or the speed of light, for example, involves merely taking measurements and doing a few calculations with the data gathered. Other problems, however, are more complicated, and cannot be solved analytically. One way to solve such problems is to propose a model and compare the model's output to gathered data. If the output matches the data, one can suppose that the model is correct. A physical model could have many parameters that need to be specified, though, and finding correct values for those parameters is often a non-trivial task. One needs to search through all possible values of the parameters and find the values that most closely match the model output to the data. Such a task is an optimization problem.

There are many methods of optimizing the parameters to a function. Some methods look at the landscape of the function as they are searching and follow hills to the best values, as in the gradient descent method. Other methods take ideas from biology or metallurgy, like genetic algorithms and simulated annealing. Particle swarm optimization is a fairly recently developed optimization technique that draws on ideas from sociology [1]. This work seeks to improve on particle swarm optimization, investigating issues that arise when large parallel clusters are used to run the algorithm.

Particle swarm optimization tries to intelligently search a multi-dimensional space by mimicking the swarming and flocking behavior of birds and other animals. It is a sociological algorithm that depends on interaction between particles to quickly and consistently find the optimal solution to a problem. The algorithm keeps track of a number of potential solutions, called particles, which move somewhat randomly through the search space. Particles remember the best place, or solution, they have been to and are attracted back to that place, as well as to the best solution other particles have seen. Specifically, the formulas for updating the position and velocity of a particle are as follows:

$$v_{t+1} = \chi\,[v_t + \varphi_1\, U() \otimes (p - x_t) + \varphi_2\, U() \otimes (g - x_t)]$$

$$x_{t+1} = x_t + v_{t+1}$$

where v_t is the velocity of a particle, x_t is its position, $\varphi_1 = \varphi_2 = 2.05$, χ is a constriction constant set to about .73 to ensure convergence, U() is a vector of random numbers drawn from a uniform distribution, p is the best position the current particle has seen, and g is the best position other particles have seen. The $\otimes$ operator is an element-wise vector multiplication.

The sociology in this algorithm defines which other particles form the “neighborhood” of a given particle – the other particles whose best solution it sees. There are many ways to define a particle's neighborhood, varying from the entire rest of the swarm to just one other particle.

The way a neighborhood is defined can have drastic effects on the performance of the algorithm – the more neighbors each particle has, the faster information spreads throughout the particles, and the quicker the algorithm converges. For some problems it is possible that the algorithm converges too quickly and gets caught in a local optimum – it stops on a hill when there is a mountain next to it. For those kinds of problems, less communication, or smaller neighborhoods, is often preferable.

Much work has been done on the sociological aspect of particle swarm optimization, trying to determine the best neighborhood structure to use for various classes of problems. This work focused on swarms of tens of particles to several hundred particles run on a single machine. Conclusions were reached about which neighborhood structures worked best, but only in this limited context [2]. With the advent of parallel processing and the potential to have swarms of thousands or hundreds of thousands of particles, some of the earlier work needs to be rethought. We have conducted and are conducting research that focuses on neighborhood structures in very large swarms, especially when the algorithm is run in parallel on multiple machines.

Many different neighborhood structures, or topologies, have been proposed and tested [3]. Two

Matthew Gardner
Brigham Young University

of them are most common – the star or gbest topology, where all particles communicate with all other particles, and the ring or lbest topology, where each particle only communicates with its two immediate neighbors [2]. The star topology is often preferable for many problems, because it allows information to be shared quickly throughout the swarm and help the algorithm converge more quickly. A common complaint about the star topology is that it will converge too quickly on a local optimum, missing the global best solution. For a swarm of 50 particles those sentiments are accurate, and most researchers were satisfied to stop with a swarm that size. We have shown, however, that increasing the number of particles in a swarm will help the algorithm to avoid premature convergence, making star a viable topology even for somewhat complicated functions [4].

In a parallel world, another problem with the star topology is that communication between processors is expensive relative to the amount of time it takes to evaluate most benchmark functions. Communicating all of the information from every particle at every iteration can take orders of magnitude more time than doing a simple evaluation, and render parallelization of the algorithm worthless. We developed a modification to the algorithm along with a new topology that approximates the star topology with much less communication. If each particle at each iteration randomly chooses a small set of particles to send its information to, eventually information will be communicated throughout the whole swarm. In the standard particle swarm optimization algorithm, the best neighbor that a particle is attracted to (the g in the formula above) is taken from its current neighborhood. With the random topology we proposed, particles can get confused being attracted to a different neighbor at every iteration, and performance significantly decreases. We modified the algorithm slightly such that each particle remembers the best neighbor it has ever had, and is attracted to that position instead of its current best neighbor. Using this modification and a random topology, we saw performance that was indistinguishable from star with only 5% of the communication, and the speed up is significant even in a serial implementation [4].

We have run these experiments with swarms of up to 4000 particles. So far, all of the experiments were run in a serial implementation of particle swarm optimization on a single machine. With more than 4000 particles the algorithm becomes very computationally expensive on a single machine. We have implemented a parallel version of particle swarm optimization using Google's MapReduce framework. Soon we hope to have it working well enough to run swarms of hundreds of thousands of particles on hundreds of machines at a time, and further explore the issues involved with parallel particle swarm optimization. Many new possibilities for topologies arise when the algorithm is run in parallel with swarms that large. We have proposed, but not yet fully tested, a topology of subswarms, where each of a hundred machines would evaluate 500 or 1000 particles and send only its best value to all other machines, drastically reducing the amount of inter-processor communication. Our preliminary results show that this approach is very promising [4].

References:

1. James Kennedy and Russell C. Eberhart. Particle swarm optimization. In *International Conference on Neural Networks IV*, pages 1942–1948, Piscataway, NJ, 1995. IEEE Service Center.
2. D. Bratton and J. Kennedy. Defining a standard for particle swarm optimization. In *Swarm Intelligence Symposium*, 2007. SIS 2007. IEEE, pages 120–127, 2007.
3. James Kennedy and Rui Mendes. Population Structure and Particle Swarm Performance. In *Evolutionary Computation, 2002 CEC2002. Congress on*, volume 2, pages 1671–1676 Vol.2, 2002.
4. Andrew W. McNabb, Matthew J. Gardner, and Kevin D. Seppi. Topology in Large Particle Swarms. Submitted to *Evolutionary Computation, 2009 CEC2009. Congress on.*