Honors Thesis Proposal

Matthew Gardner

March 18, 2009

1 Background and Significance

Optimization problems are ubiquitous in our modern world. Businesses such as Google need to decide where to place ads, scientists need to fit models to data, and airlines need to schedule their flights. All of those problems are optimization problems, and optimization techniques can be used to find solutions. Optimization algorithms have been derived from all sorts of natural phenomena. Some methods look at the landscape of the function as they are searching and follow hills to the best values, as in the gradient ascent method. Other methods take ideas from biology or metallurgy, like genetic algorithms and simulated annealing. Particle swarm optimization is a recently developed optimization technique that draws on ideas from the sociology of flocking birds. All of these methods, however, are fundamentally sequential in nature—they need to be run in a specific order, and because of that, they are typically only run on one machine. The problems people want to solve are getting bigger, and larger problems need to run on multiple machines in parallel. Parallel implementations of these algorithms have been attempted, but they do not often perform as well as sequential versions for the amount of computation performed. The work I propose focuses on one of those algorithms, particle swarm optimization, and improving its performance in a large-scale, parallel world.

2 Statement of Intent

I will study the feasibility and potential performance gain of a technique which we call speculative execution in particle swarm optimization. By speculative execution I mean the use of additional computational resources, presumably extra processors running in parallel, to perform two or more iterations of particle swarm optimization at a time, speeding up the algorithm significantly.

3 Procedures

The bulk of the work for this project will be writing code and running experiments. I have already read many conference papers on particle swarm optimization to know the current state of the field, so I will know what the results need to be compared to. In implementing speculative execution, I will take care that the algorithm behaves exactly like the original, standard particle swarm optimization (PSO) algorithm in every aspect except the speed-up. My work can then be accurately compared to other work in the field.

In running experiments, many trials will be run and averaged, as there is a random component in this algorithm. I will use statistical tests for all comparisons to the original PSO algorithm. The success criterion will be a better performance of the algorithm as measured by the lowest function evaluation seen over a number of function evaluations for several benchmark and real-world problems. Speculative execution requires more function evaluations per particle than a naïve parallel PSO, PSO with the same number of particles, so the two algorithms will be compared at points where they use equal amounts of function evaluations per iteration.

4 Preliminary Prospectus of Finished Thesis

The finished thesis will be presented in the format of a peer-reviewed conference paper, including the following sections:

1. Introduction
2. Related Work
3. Particle Swarm Optimization
4. Speculative Execution
5. Speculative Execution in PSO
6. Experiments
7. Results and Conclusions
8. Future Work
9. References

5 Preliminary Research

5.1 Particle Swarm Optimization

Particle swarm optimization was proposed in 1995 by James Kennedy and Russell Eberhart. It tries to intelligently search a multi-dimensional space by mimicking the swarming and flocking behavior of birds and other animals. [2] 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 they have been, or solution they have evaluated, 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 $\vec{x}_t$ and velocity $\vec{v}_t$ of a particle at iteration t are as follows:

$$\vec{v}_{t+1} = \chi\,[\vec{v}_t + \phi_1\,\mathrm{U}() \otimes (\vec{p} - \vec{x}_t) + \phi_2\,\mathrm{U}() \otimes (\vec{g} - \vec{x}_t)] \tag{1}$$

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

where U() is a vector of random numbers drawn from a uniform distribution, the $\otimes$ operator is an element-wise vector multiplication, $\vec{p}$ (called pbest) is the best position the current particle has seen, and $\vec{g}$ (called gbest) is the best position any of the other particles have seen. ϕ_1, ϕ_2, and χ are parameters with prescribed values required to ensure convergence (2.05, 2.05, and .73, respectively). [1] [3]

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.

5.2 Speculative Execution

Speculative execution in a program is the execution of code that may or may not end up actually being needed. Modern processors routinely do this when a conditional branch is encountered; they try to predict which branch will actually be needed and continue their execution. If it turns out that the prediction was incorrect, the work is discarded. But if the prediction was right, the program can execute much faster than if it had waited on the branch.

In many optimization algorithms speculative execution is impossible, because the next iteration of the algorithm depends on the evaluation of the function during the previous iteration. In simulated annealing, for instance, the probability of accepting a new position depends on the difference in function value between the new position and the current one. In gradient descent, the gradient of the function must be determined (which includes actually evaluating it) to know how far to move before sampling the function again. But particle swarm optimization is rather unique in that it does not require the evaluation of the function to know what the next position of each particle will be. It simply depends on *which* particle ended up having the best position and whether or not the current particle found a better spot than it had seen before. That means you can just compute all possible next positions—all combinations of which particle is the new gbest and whether or not pbest was updated ($\vec{g}$ and $\vec{p}$ from (1))—and evaluate each position in parallel. That computes two iterations at the same time. It uses a lot of extra work to get the second iteration, but for some functions performing more iterations makes more progress than adding extra particles.

6 Qualifications of Investigator

I am a Computer Science major. My main focuses in my undergraduate career have been optimization, natural language processing, and machine learning. The optimization emphasis has come mainly in my work with Dr. Kevin Seppi, for whom I am a research assistant. I have spent the last year working with him, working mainly on particle swarm optimization topics. We worked together with Dr. Branton Campbell, a physics professor, to solve a crystallography problem for him using a parallel implementation of PSO. Those results will be submitted to Nature Materials. Recently we, along with Andrew McNabb, a graduate student who works in the lab, submitted a paper to the Congress on Evolutionary Computation, a major venue for optimization techniques. The paper was on topologies and communication in large particle swarms, and formed the beginnings of the work I propose.

7 Qualifications of Faculty Advisor

Dr. Kevin Seppi is the head of the Applied Machine Learning Laboratory at BYU. Three of his last five graduate students have done their graduate work in particle swarm optimization, and he has authored or co-authored over 10 papers on the topic. I took CS 236 from him in the fall of 2007 and decided I wanted to do research in the area he was researching, so I joined his lab. I too became interested in PSO and have worked in that area with him for the past year.

8 Schedule

Run preliminary tests to ensure feasibility	February 28, 2009
Finish writing and debugging code	April 30, 2009
Finish gathering data and statistics	June 30, 2009
Complete analysis of data and begin writing thesis	September 1, 2009
Submit completed copy of thesis to advisor	November 15, 2009
Submit paper detailing results to CEC 2010	December 20, 2009
Final thesis and portfolio submitted	February 1, 2010
Thesis defense completed	March 1, 2010
Final four thesis copies submitted	March 8, 2010

9 Expenses/Budget

This project will not require any outside funding. The only expenditure will be my own time researching, conducting experiments, and writing. I have received a grant from the Office of Research and Creative Activities to perform this research, for which I am grateful.

10 Closure

Thank you for reviewing my proposal. I look forward to exploring speculative execution in particle swarm optimization.

References

[1] 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.

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

[3] Riccardo Poli. Dynamics and stability of the sampling distribution of particle swarm optimisers via moment analysis. *Journal of Artificial Evolution and Applications*, 8(2):1–10, 2008.