15-418: Project Proposal

TITLE: Parallelizing the Artificial Bee Colony (ABC) Algorithm

SUMMARY:
The Artificial Bee Colony (ABC) algorithm is an optimization algorithm which mimics the foraging behavior of bees. We will parallelize this algorithm on multi-core CPU platforms using Open-MP and Open-MPI and analyze their performances.

BACKGROUND:
An explanation of the algorithm:
The Artificial Bee Colony (ABC) algorithm is a well known and commonly used algorithm when it comes to optimization problems. It was created based on how honey bee swarms have intelligent foraging behaviour. In the basic version of the model, the bees are split into 3 different groups: employed bees, onlookers bees, and scout bees. A main assumption is that there is one food source for each employed bee. The employed bees carry information regarding the food source and then share this information to other bees through their dancing. If an employed bees' food source is abandoned, they will become a scout and search for a new food source. The onlooker bees gain information regarding the food sources and choose them through the dances.

Basic algorithm looks like:

1. Initialize food sources for each employed bee
2. Repeat/loop the following
    1. Every employed bee goes to its associated food source and determines how much honey is at the food source then goes to dance in the hive
    2. Each onlooker bee watches the dances and selects a food source through a greedy strategy and goes to that source. After it picks a neighbour around it, it determines how much honey is in the neighbour.
    3. Figure out which are the abandoned food sources and replace them with new food sources found by the scout bees.
    4. Store information about best food source
3. Loop until requirements for best food source are met or until a certain number of iterations

The three main places in which this algorithm can and will be parallelized is 1. Initialization, 2.1. Employed bee tasks and 2.2 onlooker bee tasks.

THE CHALLENGE:
The original or sequential ABC algorithm is known to have a slow speed of convergence, a long search time, and can lead to issues of falling into local optimums at the end of a search. It will be difficult to adjust the algorithm such that the solution isn't stuck at a local optimum at the end of a search. One way we aim to try to fix this problem is through simulated annealing.

Furthermore, what may make it difficult to parallelize is the information that needs to be shared across processors within the stages through shared memory or message passing. We need to make sure that there aren't multiple processors writing to the same data. We also need to make sure that we divide/parallelize across the best axis possible, and will try different strategies when designing the size of the bee swarm and dividing the bee types such that the workload is as efficient and evenly distributed as possible across the processors.

Additionally, since we are trying both OpenMP and OpenMPI, another challenging aspect is that we will have to make sure that our algorithm can handle information sharing through either shared memory or message passing. This will also allow us to be able to see which strategy allows the greatest speedup.

RESOURCES:
We will be using the multi-core nodes on the GHC and Bridges PSC machines. We will use the Open-MP and Open-MPI libraries but we will build and write the Artificial Bee Colony algorithm from scratch in C++.

GOALS AND DELIVERABLES:
Plan to Achieve (100% completion):

- Optimization of the ABC algorithm for better accuracy and performance in general
- Implementation of a parallel ABC algorithm using Open-MP
- Implementation of a parallel ABC algorithm using Open-MPI
- Detailed analysis and comparison of both implementations

Hope to Achieve (125% completion):

- Implementation of parallel ABC algorithm using CUDA
- Analysis of CUDA implementation vs Open-MP and Open-MPI
- Step-by-step visualization of the ABC algorithm

Sub-MVP (75% completion):

- Optimization of the ABC algorithm for better accuracy and performance in general
- Implementation and analysis of ABC algorithm using Open-MP

Our final analysis will include speedup graphs and graphs comparing our various implementations of the parallel algorithm and sequential algorithm. We plan on answering which implementation results in the largest speedup/which is most efficient and how many threads for the corresponding implementation leads to these results.

PLATFORM CHOICE:
Computer: The ABC algorithm that we are implementing involves many different kinds of bees processes) performing independent work but also communicating with each other. Therefore, we chose the GHC and Bridges PSC machines because they have multi-core CPU functionality which works well for our use case.

Language: We are going to use C++ as it allows us to leverage the Open-MP and Open-MPI libraries to create a multi-threaded algorithm to run on the cores.

SCHEDULE:

March 23rd	Project Proposal Due
April 1st	Optimized Sequential ABC Algorithm Complete
April 11th	Milestone Report Due/Open-MP Implementation complete
April 20th	Open-MPI Implementation Complete
April 26th	Analysis and comparison of both implementations
April 29th	Final Report Due (Work on demo/poster session information
May 5th	Poster Session