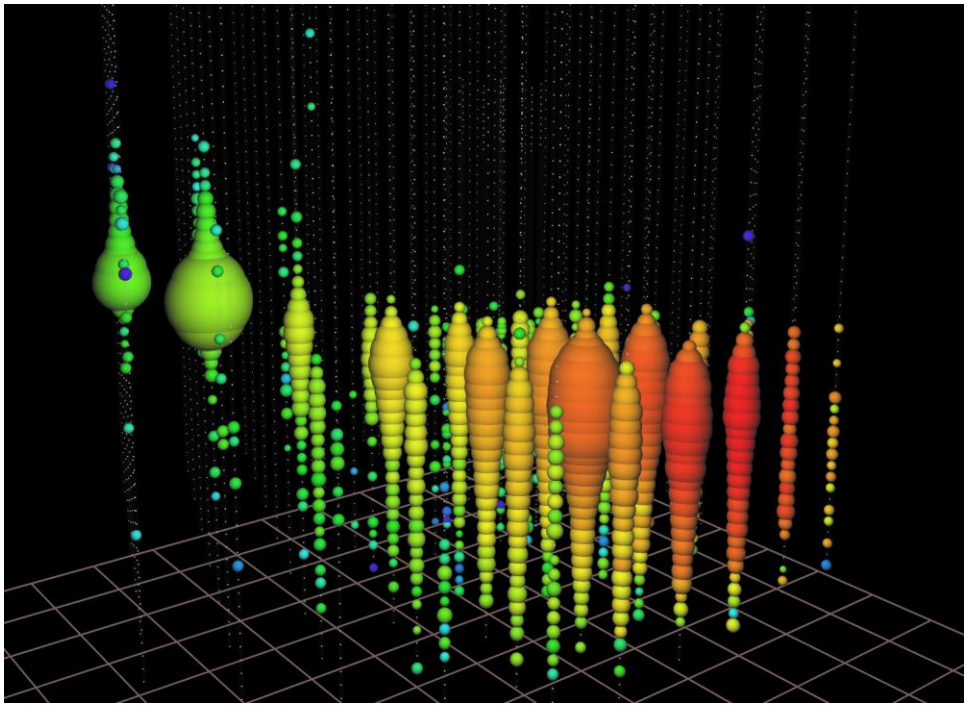
Parallelization of Markov Chain Monte Carlo Neutrino Event Reconstruction

Le Nguyen



(Simulated aftermath of neutrino interaction with the IceCube detector)

I am an undergraduate research assistance working with the IceCube Neutrino Detection Observatory group at MSU. My work is to take the resulting data from the aftermath of a simulated neutrino interaction with the IceCube detector and work backwards to figure out what characteristics the neutrino had before the interaction. To do this, I use key features from the event (the after math of the neutrino interaction) and used them to pull up relevant likelihood table of the type of neutrino that may have produce this event. For instance, if I have 13 GeV of deposited energy in the aftermath of my event and I wanted to know that mass of my neutrino I could pull up a likelihood distribution of the masses of neutrinos that deposited 13 GeV of energy to find the most likely mass.

I do this for 10 parameters of the event to get a precise description of the neutrino that caused it. This means I am trying to maximize a 10-dimensional likelihood space (technically it is minimizing because I am using an inverse log likelihood) which is a computational nightmare. It's a high dimensional irregular likelihood space that most optimization algorithms cannot handle, so the choice algorithm Markov Chain Monte Carlo (MCMC). MCMC uses a set of random but selective walkers to step through a likelihood space and converge at the minimum of maximum. In my research I do this by wrapping in-house IceCube software in python. The IceCube software spits out likelihood values and python takes them and decides where to take the next step in the likelihood space.

Unfortunately, I cannot mess with the IceCube software I am using. It's a black box, it gives me likelihood values, and I deal with everything else. My plan is to optimize the writing of the python wrapper as well as intelligently scheduling my jobs of the HPCC to cut down on run time. MCMC is a pleasantly parallel problem, multiple independent random walkers can be exploring the likelihood space at the same time and compile their results at the end. A good first step would be to take a serial walker and split it into multiple smaller walkers that run in parallel on different threads. Instead of a single walker taking 1 million steps, 100 walkers could take 100 thousand. This takes the same amount of CPU hours, but saves on real time. I think a good first step is to see how little steps across how many walks can I get away with while still having accurate results – not having too little steps to explore the space or to converge on a value.

After I do this, I want to minimize the run time of each individual walker. The only part of script I have control over is the python wrapper, so I want to optimize the execution of the python wrapper with a library like Numba (https://numba.pydata.org/). Numba is a library that converts python into a complied language. The goal is to have the ease of use of python programming with the speed of c/fortran. By optimally distributing the number of steps I need to take among many walkers and making the walkers run as fast as possible I will minimize the run time of the entire algorithm.

The benchmarking on this project will be based upon time and accuracy. Since I am working with simulated data I know the true values I should end up with. I know the accuracy a serial MCMC in my likelihood space and I want to achieve similar or greater accuracy to it. Also, I know how long it takes for the serial code to run as well. It won't be hard to beat the serial run time, so I am going for minimization of run time while maintaining accuracy.

Schedule

- Thursday February 11 - Project Proposal Milestone Due
- Week of February 15 – Review MCMC code (it's been a minute since I've had time to get into it)
- Week of February 22 - Create a submission script to run MCMC as an array of jobs
- Week of March 1 - Analysis of steps per walker and accuracy
- Week of March 8 - Analysis of steps per walker and accuracy (hopeful find a good value)
- Week of March 15 - Analysis of steps per walker and accuracy? (definitely have a good value)
- Week of March 21 - Learn Numba
- Week of March 22 - Implement Numba, Part one write up
- Thursday March 25 - Project Part 1 Due
- Week of March 28 - Implement Numba? Or find a different package if it doesn't work.
- Week of March 29 - Things will go wrong and I will need this time for something
- Week of April 5 - Last minute changes and begin write up
- Week of April 12 - Write up of project
- Thursday April 15 - Final Project due