Project Checkpoint: Parallel Bond Order
Assignment

Kevin Shi (kshi), Patrick Nguyen (phn)

1 Summary:

We have (we think) finished the literature review and understand the problem space. We still need to schedule an appointment with the professor that proposed the idea to confirm this.

We also have finished a basic, naive sequential implementation of a perfect matching algorithm detailed in https://github.com/ghutchis/parallel-matching/blob/master/papers/Mulmuley-1987.pdf

The algorithm boils down to taking lots of determinants and creating a Tutte matrix then using some magic to get a perfect matching.

We are a couple of days late behind our schedule as the background reading was a lot dense than expected and we were a lot busier than expected.

A large problem we realized within initial parallelization of the algorithm is that the graph ends up as very sparse matrices of large numbers that we need to run determinants on. We are currently unable to test it reasonably on large datasets due to the time it takes to run these determinants. We expect to spend a significant amount of time on optimizing the matrix operations on these sparse, large number matrices.

2 Goals and Deliverables:

We are on track to deliver a parallel OpenMP implementation for the presentation. We’re currently unsure of the level of performance we can get on this as parallel sparse matrix determinants are not well explored.

At this point we’re pretty sure putting it in CUDA is a ”nice to have” goal as thinking through the memory and cache problems with sparse matrix determinants would be a whole different story within CUDA. We expect it won’t be as daunting as it looks after we actually get the OpenMP implementation down, but it still remains a stretch.

Our deliverable at the parallelism competition will be graphs of the speedup we get on the parallel bond matching in OpenMP versus a sequential version. We could also possibly run a small demo showing our program assigning a bond order to a molecule on a small molecule.
3 Concerns

Memory is hard. Large numbers are even harder.

The algorithm involves making a matrix consisting of powers of two, where the exponents are weights ranging from 0 to the number of edges in the graph. Needless to say, this can be very intractable. Our naive sequential algorithm works for small datasets, but we will need to optimize the representation of the matrix to make this problem tractable. This will affect how we take the determinant of the matrix, which is the bulk of the algorithm, and so we will need to do a lot of investigation on the math behind the algorithm to make scaling up feasible.

Furthermore, we have to deal with very large, sparse matrices, and so representing such matrices with the above constraint will be even more of a challenge.

4 Schedule

**Done:** Week 1 (4/3 - 4/9): Read literature about parallel perfect matching, talk to professor about dataset details, plan out higher level details, begin sequential implementation.

**Done:** Week 2 (4/10 - 4/16): Complete sequential implementation, do obvious optimizations in OpenMp, complete checkpoint report.

Week 3 (4/17 - 4/23): Create benchmark framework, do optimizations relating to memory access patterns and caching, number representation (i.e. MATRICIESSSSSSS)

Week 4 (4/24 - 4/30): Optimize to chemical bonds data set with further assumptions (i.e. MATRICIES PART 2)

Week 5 (5/1 - 5/8): Complete final touches, complete final writeup