Name Goutam Murlidhar

Major-Master of Information Management and Systems

Research Interests-

Before beginning my masters, I worked as a lead engineer at a startup. There I worked on implementing data mining systems that analyzed social networks to identify influence. Since then I have been very interested in large graph analysis. I am interested in a couple of areas using graph theory. I am interested in how graphs can be used to model real world entities like social networks, power grids maps and so on. I am especially interested in building infrastructure to store and analyze this information.

Goal from the class-

Like I outlined in my research interest. I am interested in using parallel computing methods and algorithms to design and develop infrastructure to analyze these large graphs. I believe that parts of the class around parallelizing graph algorithms would be especially to useful to me. By the end of the course I hope to gain the fundamentals required to implement these graph based algorithms in a parallel nature and use this knowledge to design and implement software infrastructure specialized for large scale graph analysis especially in the domain of social network analysis.
The problem of community detection is around identifying groups of people or nodes in the network that interact frequently with each other. Popular algorithms to do this include hierarchical clustering, random walk finding a clique and so on. Some of these methods like finding a clique are known to be NP hard. Parallel approaches to the problem are very effective. Research has been done in using a multi k-core graphs to identify multiple communities subgraphs within a social network. Large scale big data systems like Apache Hbase can be used to effectively parallelize and build distributed algorithms that easily parallelize.[1]

Hierarchal clustering based algorithms also lend themselves to parallelization. Prams and butterfly architectures can be used to effectively to solve this problem. Clustering using a single link metric can be parallelized by assigning a single cluster to each core and as the clusters are combined the combined cluster is processed by one of the cores and the other core becomes idle. If you have n items to cluster, then you can use n/logn processor. The entire algorithm can thus be done in O (nlogn) time.[2]

Challenges-

One of the major challenges in community identification is that a large number of parallel algorithms are very susceptible to noisy data. Additionally because communities in social networks are dynamic it makes it more challenging to identify them.