**DENSEST SUBGRAPH DISCOVERY ON THE GPU**

by

Hunter G. Gareau

A Thesis

Submitted to the

(insert department name)

College of Science and Mathematics

In partial fulfillment of the requirement

For the degree of

Master of Science in Computer Science

at

Rowan University

(insert defense date)

Thesis Chair: (insert Professor Guo info)

Committee Members:

(insert Yu info)

(insert Rabbitz info)

© 2025 Hunter G. Gareau

**Acknowledgements**

I would like to deeply thank Professor Guimu Guo for his guidance throughout our research. We have worked together for a long time on this project, and I wouldn’t have been able to do any of this without his skills, direction, and help.

(insert thanks for other Committee members)

(insert thanks for other student?)

**Abstract**

Hunter Gareau

DENSEST SUBGRAPH DISCOVERY ON THE GPU

2024 – 2025

Dr. Guimu Guo

Master of Science in Computer Science

(Abstract starts here)

**Table of Contents**

Abstract iv

List of Figures vi

List of Tables vii

Chapter 1: Introduction 1

Chapter 2: (insert here)

**List of Figures**

Figure Page

Figure 1. (Title here) (Page #)

List of Tables

Table Page

Table 1. (Title here) (Page #)

**Chapter 1**

**Introduction**

When it comes to analyzing large or complex groups of data, it is often useful to examine the connections and relationships shared between its subjects. Graphs can be used to model such relations. Graphs are composed of two components: vertices (which represent individual members of a data set) and edges (which represent the connections between these members). A graph is usually represented as *G = (V, E)*, where *V* is the set of vertices in the graph, and *E* is the set of edges in the graph. A couple of the major types of data that graphs can be used to model are social networks (e.g. Facebook, Twitter, etc.) and biological data (DNA, neural networks, etc.) [2]. It’s also worth noting that edges can be directed, as in a connection between two vertices specifically goes from one to another. A directed graph can be used to model other types of data sets, or specific types of relations, such as people following others in an online social network.

The analyzation of these graphs using various tools or techniques to find additional data and patterns is known as Graph Mining. While there are many studies and problems in graph mining, a fundamental one is known as the *densest subgraph discovery problem* (the DSD). The aim of the DSD is that given an undirected graph *G*, you must find a subgraph *S* such that it has the highest density of all subgraphs of *G*. The density of a graph is represented as *e/v* (where *e* is the number of edges in the graph and *v* is the number of vertices in the graph). The denser a graph is, the more connected the members of that graph are. And thus, in simple terms, the DSD aims to find the most connected group of vertices within a graph. Additionally, density can also be applied to network motifs, which are small structures of vertices and edges such as shapes or cliques. In this case, it would be the number of a given motif over the number of vertices, finding the most connected group of those motifs [3]. The densest subgraph (and thus solutions to the DSD) is a notable piece of information to have for a dataset and has plenty of notable applications in real data sets including finding and filtering out fake users or identifying echo chambers in social networks, or identification of regulatory motifs in DNA or gene annotation graphs in biological data [2].

Being such a notable problem, there are of course many solutions to the DSD. However most, if not all, are serialized programs that run on the CPU. By the nature of graph mining, it is almost always required to process every vertex in these large graph datasets, which can certainly take time in a serialized program processing these one by one. By programming in parallel, a great number of vertices can be processed in concurrence, and thus save time and be more efficient. And this is where the GPU comes in. The CPU and GPU differ in their processing cores. The CPU runs on a handful of powerful processing cores that can take complex orders, while the GPU runs on many weaker processing cores that take simpler orders. So, while a powerful CPU can certainly run a taxing program efficiently, utilizing the full power by running commands in parallel across the GPU’s many cores is much more computationally efficient. Although this comes with the limitation that the GPU’s commands are much more restrictive than that of the CPU [1]. But the computational power of parallel programming on the GPU is certainly well suited for graph mining, and thus the DSD.

With the complexities of graph mining (and in this case the DSD), however, one would have a very tough time writing a program to analyze a graph only using the simple commands available to the GPU. But thanks to NVIDIA, there’s a tool to work around that. CUDA, which stands for Compute Unified Device Architecture, is a parallel computing platform and application programming interface (API) model. CUDA allows for a serialized C++ program run on the CPU to execute threads in parallel on the GPU, being able to leverage the massive computational power of the GPU as needed [1].

And that brings us to the goal of the project. Programming a solution to the DSD which utilizes the GPU through CUDA. This goal is two-fold, being both a more efficient solution to the DSD, as well as providing further research on the computational power of parallel programming on the GPU.

**Chapter 2**

**Existing Solutions**

(need to gather info on existing solutions, but begin this section here)