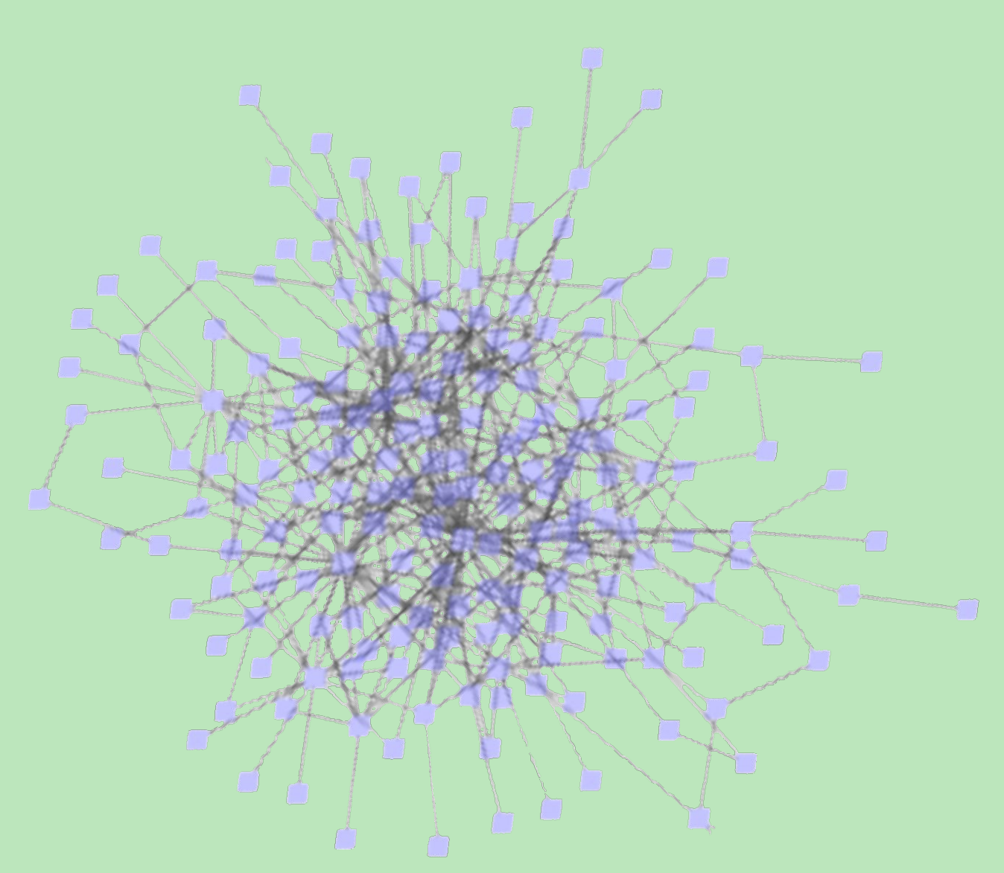
Predicting Optimal Thread Configuration Based on Input Graph Properties with Machine Learning

Arch Henderson III[1], Amanda Hooge[2], and Apan Qasem[3]

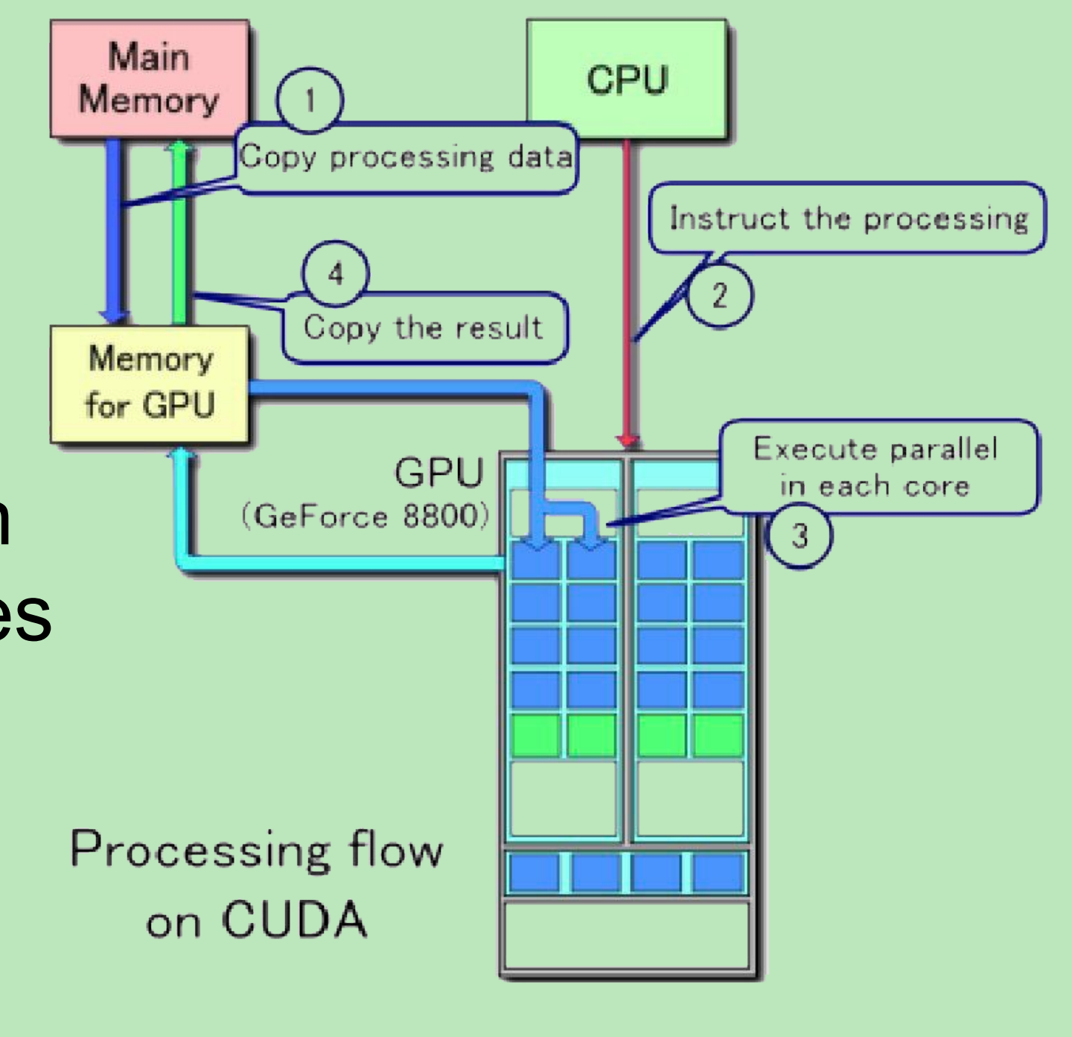
[1]Louisiana State University in Shreveport [2]Texas State University [3]Texas State University

Motivation


Graph of noun/adjective adjacencies in "David Copperfield"

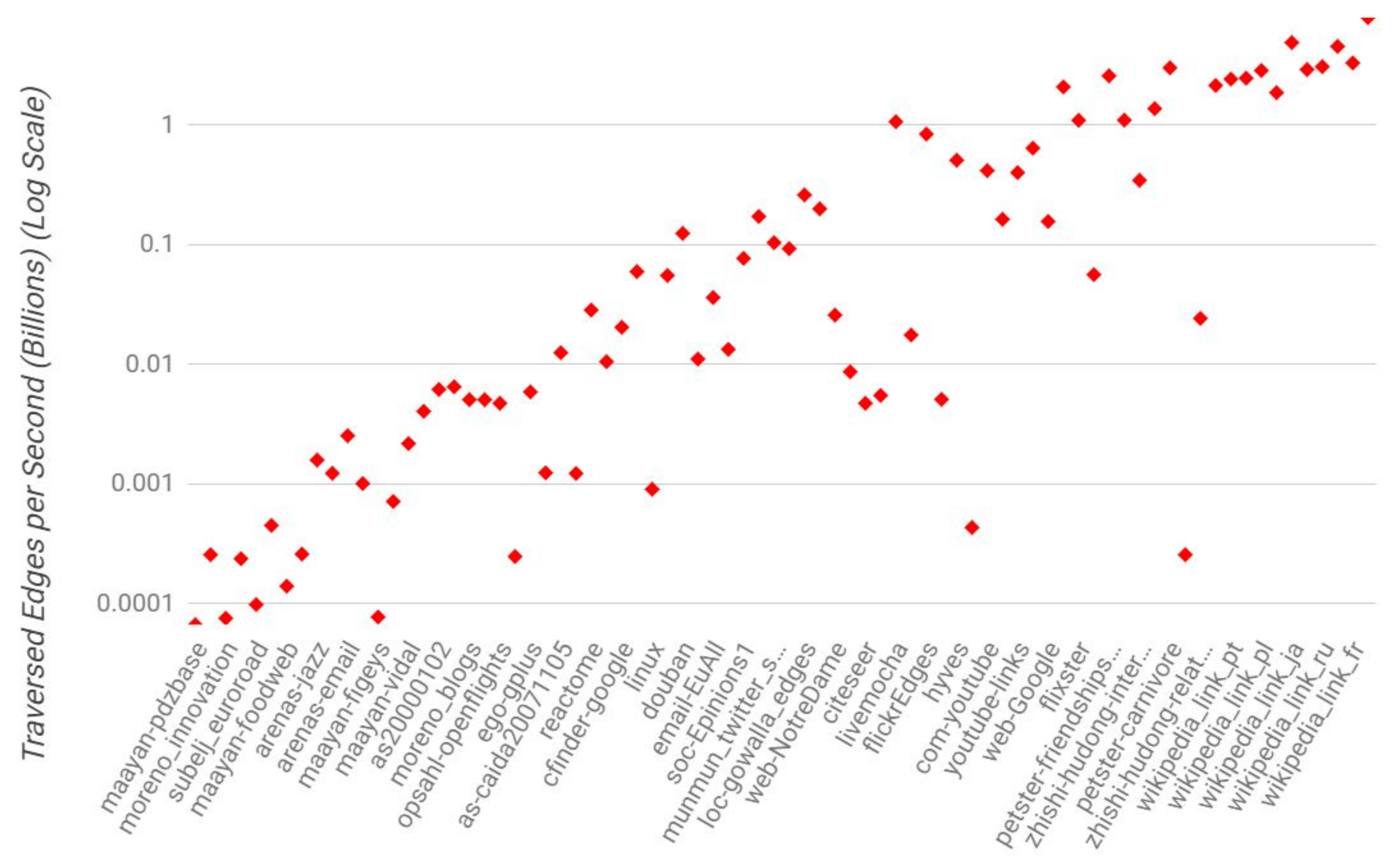
The **graph** data structure has many applications, and graphs can become massive with large databases, so efficient traversal is a critical concern. There is extensive research in graph algorithm optimization, and parallel traversal using GPUs is a popular method.

Graphics Processing Units (GPUs) work by using many simple cores to process commands in parallel. Parallel computing works well for graph traversal due to the many nodes that must be processed. However, irregularity in real-world graphs makes performance unpredictable across graphs with varying properties.



GPU Architecture

In this project, we sought to understand why GPU programs perform the way they do on different graph inputs. Using machine learning to analyze the relationship between graph properties and algorithmic configurations, we developed a supervised classifier that can predict the optimal thread/block size with good accuracy.



Each graph's performance at their fastest configuration, ordered by edge count. There is observable irregularity across the graphs.

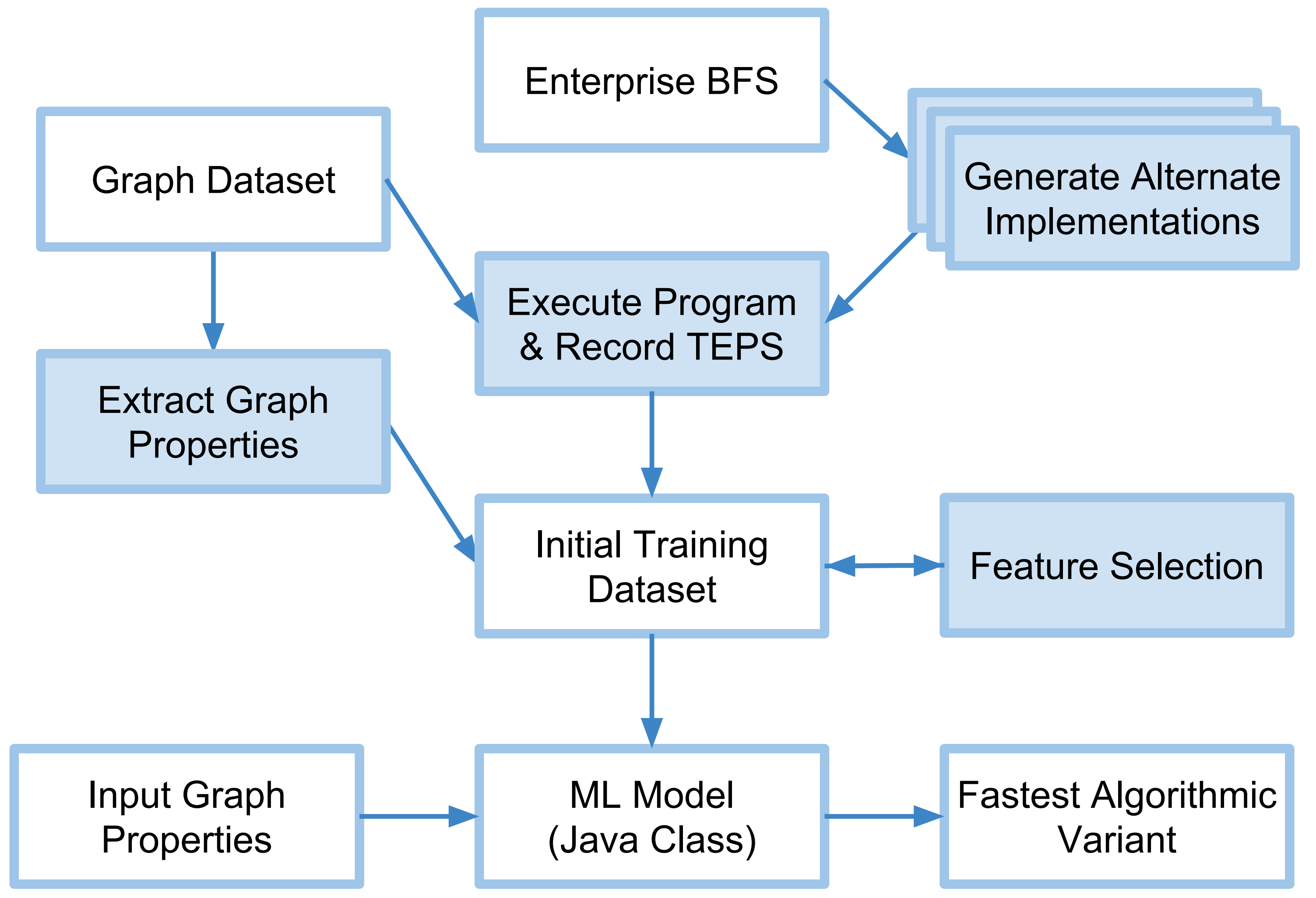
Methodology

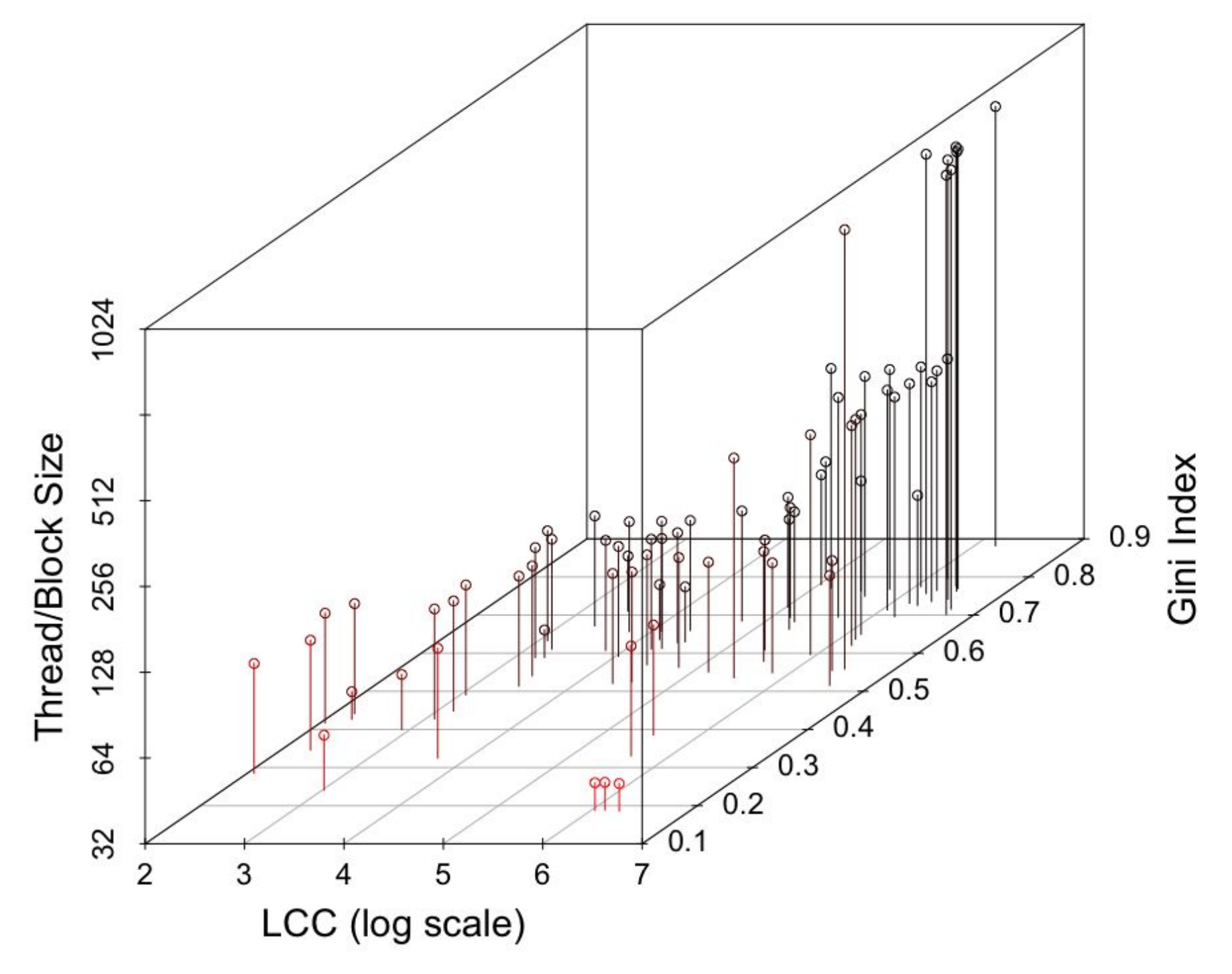
We used 78 unweighted graphs, 40 directed and 38 undirected, obtained from the University of Koblenz-Landau's KONECT dataset. We used Hang Liu et al's highly optimized **Breadth-First Search** algorithm, *Enterprise*.

- We altered compiler optimization level, register pressure, and threads per block/blocks per grid (referred to as thread/block size, these were assigned the same value). Our baseline was O3 optimization, 16 bit register, and a thread/block size of 256.

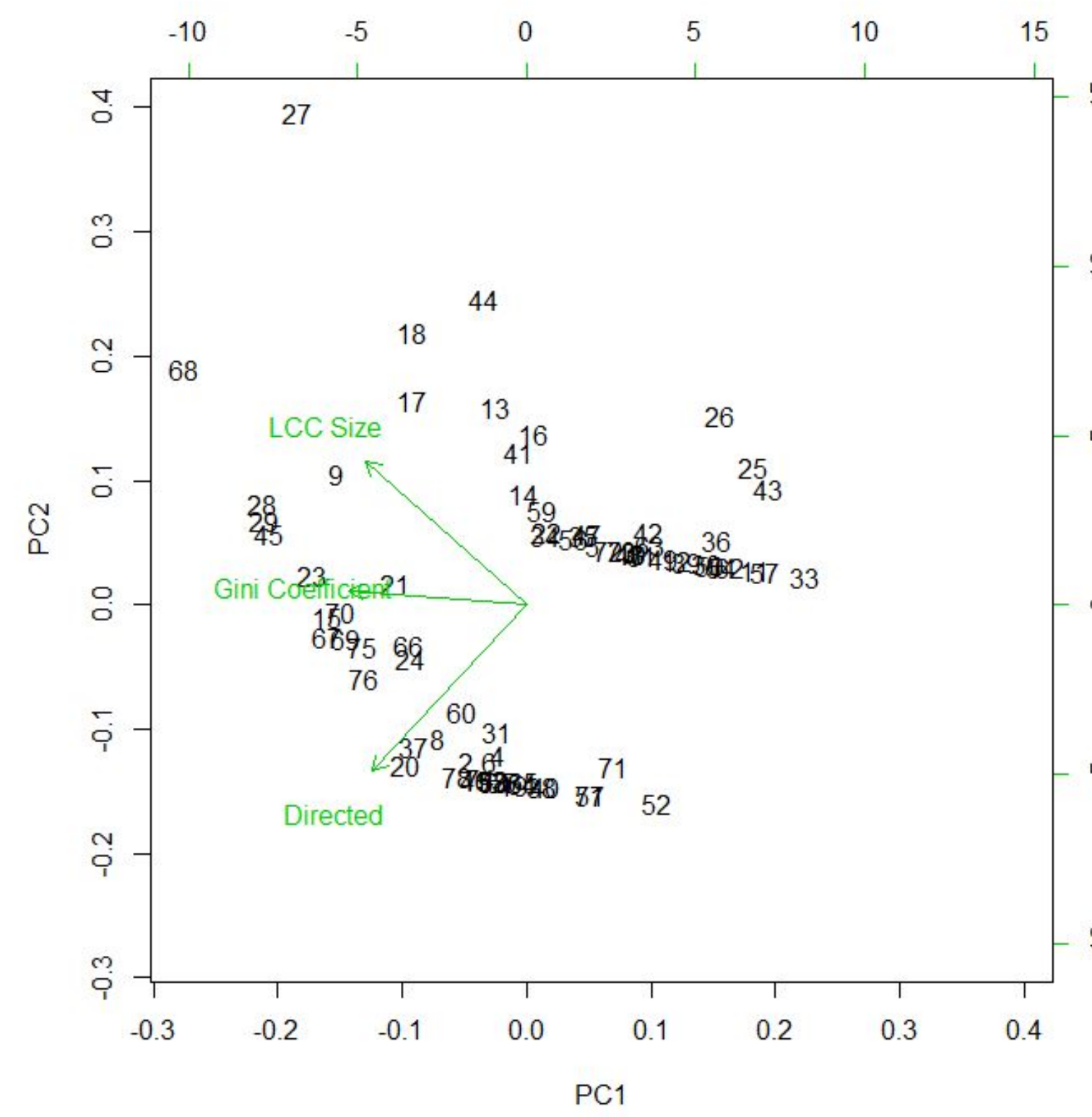
Optimization Levels	Register Pressures	Thread/Block Size	
• O1	• 16	• 32	• 256
• O2	• 32	• 64	• 512
• O3	• 64	• 128	• 1024
	• 512		

- Each graph was run through Enterprise with each configuration multiple times.
- We used 25 graph properties to train our ML model to predict the fastest configuration for each graph.

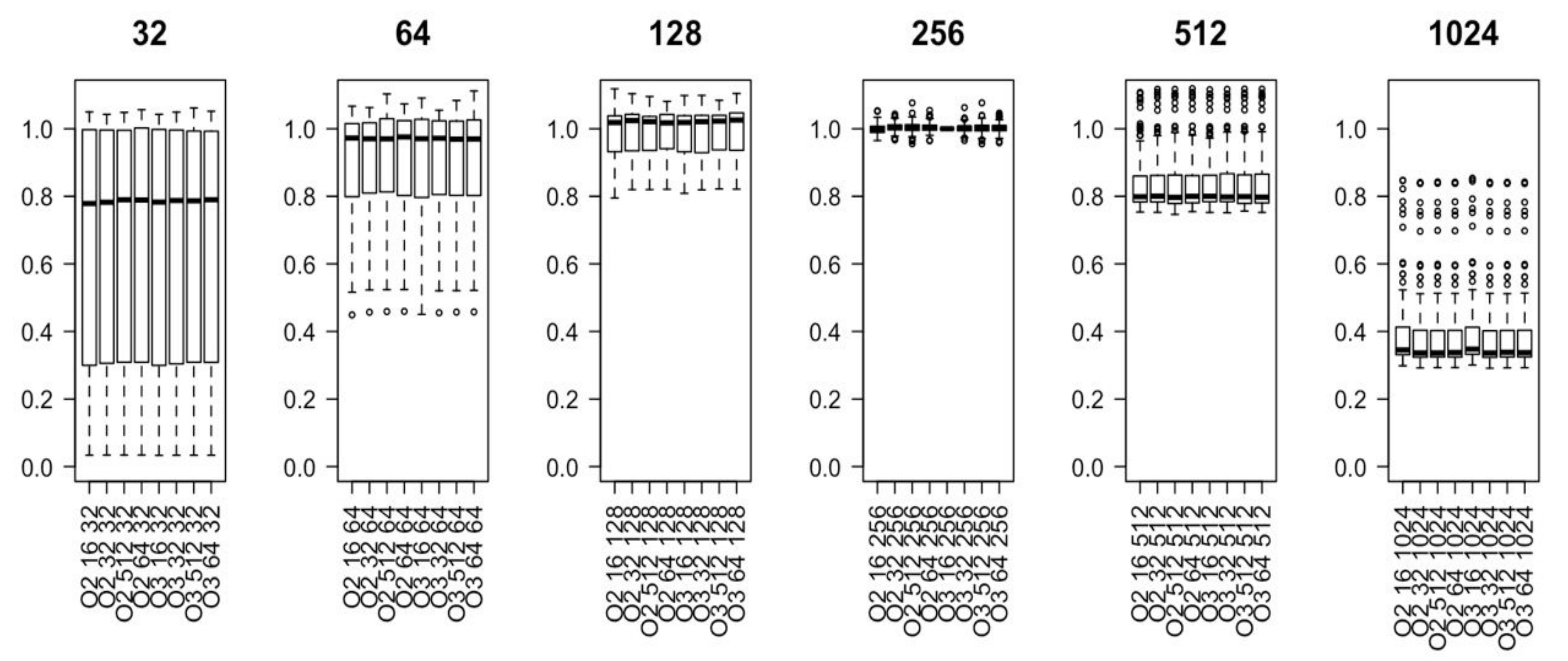




Correlation between significant features



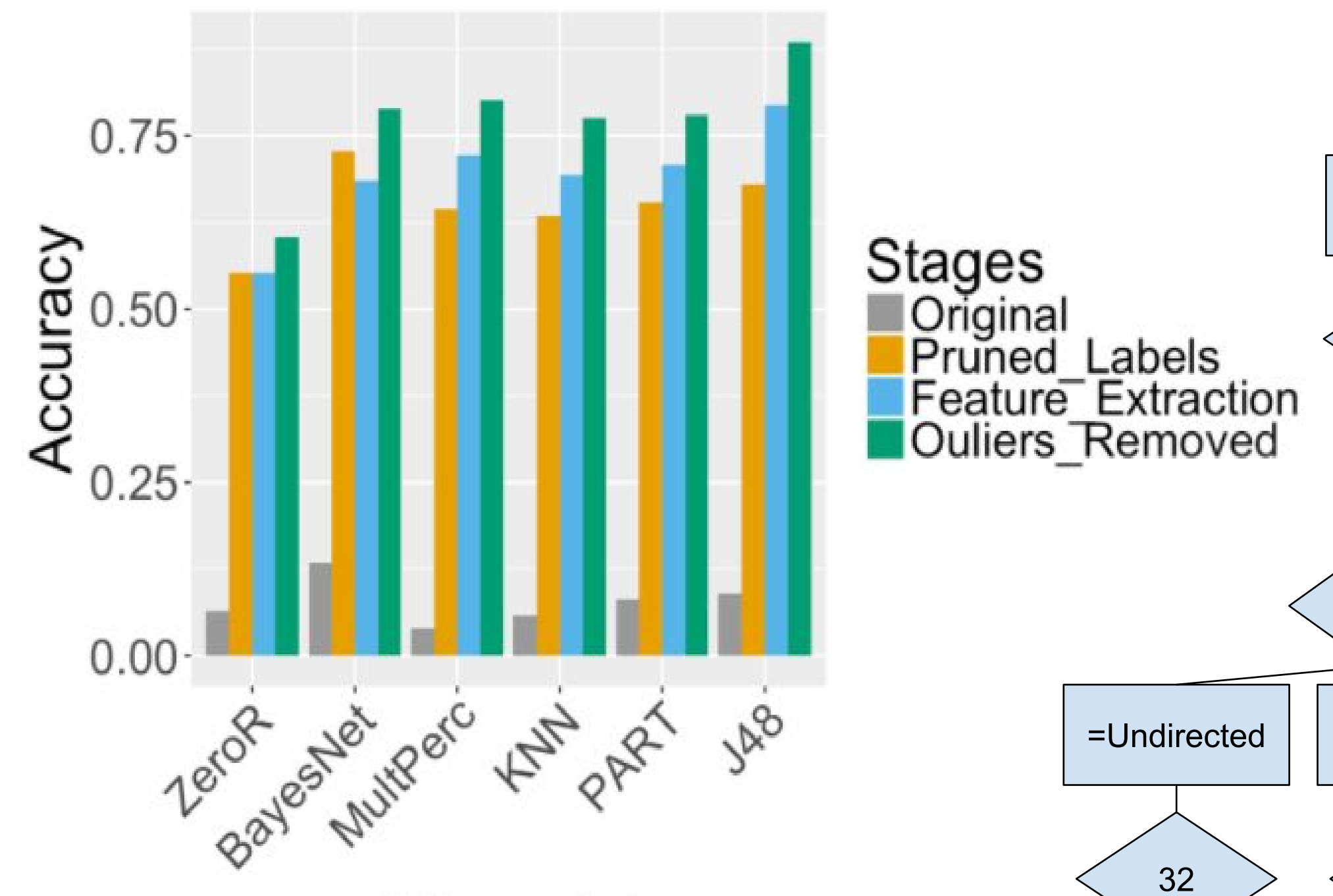
Significant feature PCA



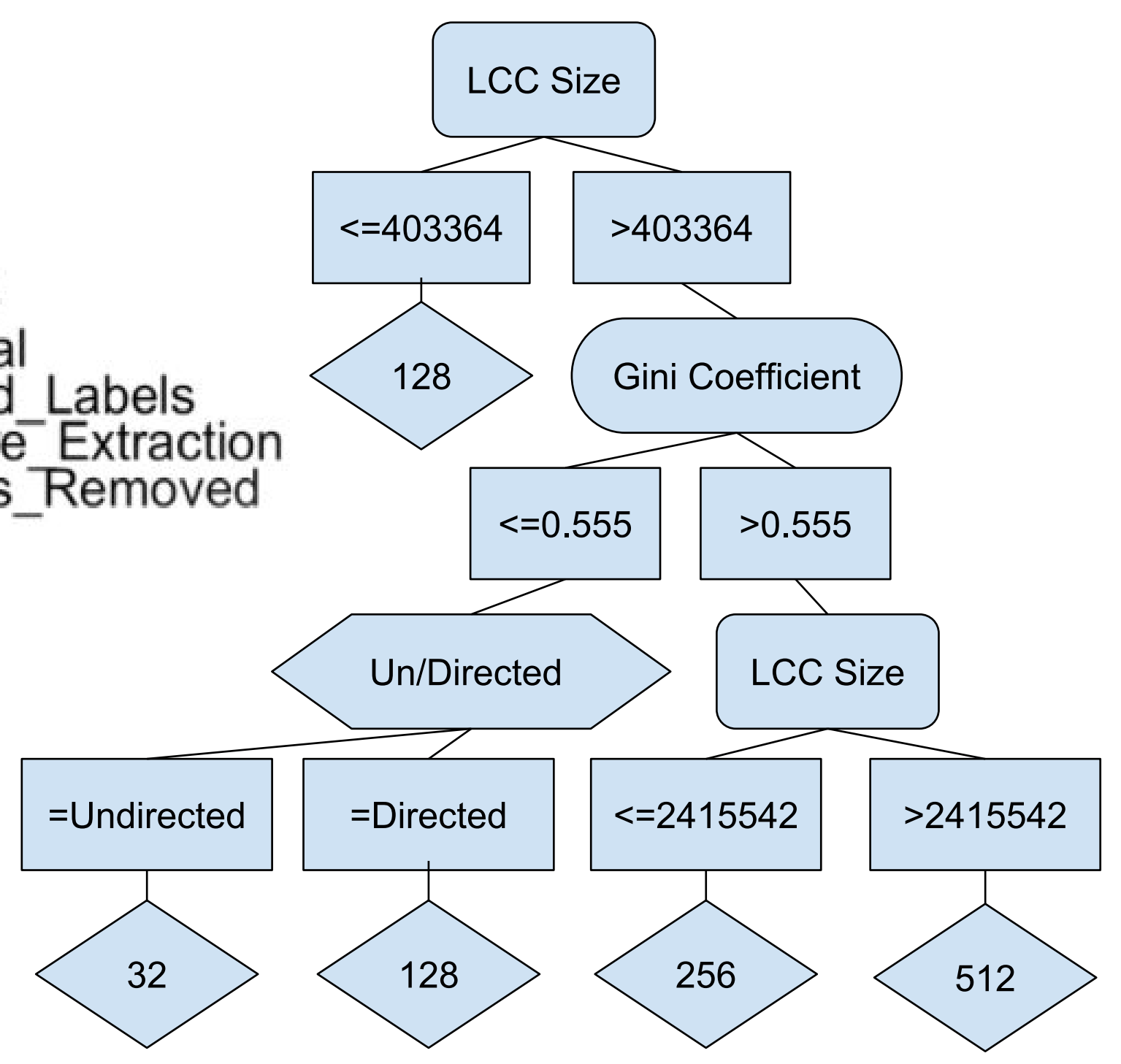
Each configuration compared to the baseline, grouped by thread/block size

Results and Analysis

- While register pressure and optimization level have little impact, choosing a proper thread/block size increases performance by as much as 12% over the baseline on some graphs.
- Because 78 input graphs yielded 33 fastest configuration labels, the models suffered from poor accuracy. To overcome this, we focused solely on predicting thread/block size, as it impacted performance the most.
- After incrementally selecting the most correlated properties, we were left with three that could predict thread/block size with reasonable accuracy: Un/Directed, Gini Coefficient (degree distribution), and Largest Connected Component (LCC) Size.
- Using a J48 decision tree classifier, our model was able to predict thread/block size with ~85% accuracy.



Different ML models' accuracies during each stage of data processing



J48 Decision Tree that yielded ~85% accuracy

Acknowledgments

We thank the National Science Foundation for funding the research under the Research Experiences for Undergraduates site programs (CNS-1358939, CNS-1659807) at Texas State University to perform this piece of work and the infrastructure provided by a NSF-CRI 1305302 award.