Advanced Algorithms and Parallel Programming Spring 2018

Advanced Algorithms Project

Semsi Yigit Ozgumus

Politecnico di Milano

July 17, 2018

Table of Contents

Current Section

What is a Strongly Connected Component ?

What is a Strongly Connected Component ?

- if a Vertex is reachable by any vertex in the graph, then that vertex is **strongly connected**.

What is a Strongly Connected Component ?

- if a Vertex is reachable by any vertex in the graph, then that vertex is **strongly connected**.
- if a partition of vertices that are a subgraph of the graph are all strongly connected, then that subgraph is a **strongly connected component** of that graph.

What is a Strongly Connected Component ?

- if a Vertex is reachable by any vertex in the graph, then that vertex is **strongly connected**.
- if a partition of vertices that are a subgraph of the graph are all strongly connected, then that subgraph is a **strongly connected component** of that graph.

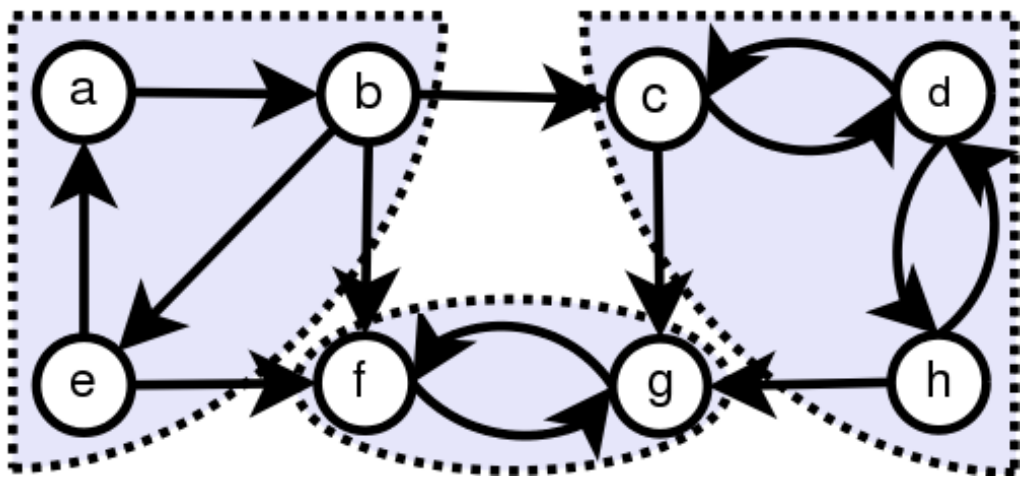


Figure: Graph with strongly connected components

Methods to solve

Methods to solve

- Tarjan's Algorithm

Methods to solve

- Tarjan's Algorithm
  - Time Complexity $\rightarrow O(|V| + |E|)$

Methods to solve

- Tarjan's Algorithm
  - Time Complexity $\rightarrow O(|V| + |E|)$
  - Space Complexity $\rightarrow O(|V| \cdot (2 + 5w))$

Methods to solve

- Tarjan's Algorithm
  - Time Complexity $\rightarrow O(|V| + |E|)$
  - Space Complexity $\rightarrow O(|V| \cdot (2 + 5w))$
- Nuutila's Version

Methods to solve

- Tarjan's Algorithm
  - Time Complexity $\rightarrow O(|V| + |E|)$
  - Space Complexity $\rightarrow O(|V| \cdot (2 + 5w))$
- Nuutila's Version
  - Space Complexity $\rightarrow O(|V| \cdot (1 + 4w))$

Methods to solve

- Tarjan's Algorithm
  - Time Complexity $\rightarrow O(|V| + |E|)$
  - Space Complexity $\rightarrow O(|V| \cdot (2 + 5w))$
- Nuutila's Version
  - Space Complexity $\rightarrow O(|V| \cdot (1 + 4w))$
- Pearce's Version

Methods to solve

- Tarjan's Algorithm
  - Time Complexity $\rightarrow O(|V| + |E|)$
  - Space Complexity $\rightarrow O(|V| \cdot (2 + 5w))$
- Nuutila's Version
  - Space Complexity $\rightarrow O(|V| \cdot (1 + 4w))$
- Pearce's Version
  - Space Complexity $\rightarrow O(|V| \cdot (1 + 3w))$

Current Section

Project Structure

- In terms of functionality the project has 4 different functions.

Project Structure

- In terms of functionality the project has 4 different functions.
  1. Generate a Random Directed Graph with options (Python)

Project Structure

- In terms of functionality the project has 4 different functions.
  1. Generate a Random Directed Graph with options (Python)
  2. Run Experiement with algoritmhs while keeping track of completion time and storage consumption (tarjan vs. Nuutila vs. Pearce)(C++)

Project Structure

- In terms of functionality the project has 4 different functions.
  1. Generate a Random Directed Graph with options (Python)
  2. Run Experiement with algoritmhs while keeping track of completion time and storage consumption (tarjan vs. Nuutila vs. Pearce)(C++)
  3. Debug mode for testing a singular graph with algorithms provided(C++).

Project Structure

- In terms of functionality the project has 4 different functions.
  1. Generate a Random Directed Graph with options (Python)
  2. Run Experiement with algoritmhs while keeping track of completion time and storage consumption (tarjan vs. Nuutila vs. Pearce)(C++)
  3. Debug mode for testing a singular graph with algorithms provided(C++).
  4. Visualization of the experiments, can only be done with cvs files (Python).

Project Structure

- All the steps are connected to each other and glued together to create a pipeline to emulate terminal application.

Project Structure

- All the steps are connected to each other and glued together to create a pipeline to emulate terminal application.
- All documentation done in Doxygen

Project Structure

- All the steps are connected to each other and glued together to create a pipeline to emulate terminal application.
- All documentation done in Doxygen
- Visualization of the experiment is done with python, csv files can be further explored using the provided jupyter notebook.

Python Part

- There are 2 scripts that are embedded into the project.
  1. generate_graph_directories.py

Python Part

- There are 2 scripts that are embedded into the project.
  1. generate_graph_directories.py
  2. analysis.py

Python Part

- There are 2 scripts that are embedded into the project.
  1. generate_graph_directories.py
  2. analysis.py
- They can be used outside of application

Python Part

```
yigitozgumus@Yigits-MacBook-Pro|~/dev/AAPP_Project on master!?
± python src/python/analysis.py -h
usage: analysis.py [-h] [--index I] C

Visualize the experiment results

positional arguments:
  C           The Target Location of the experiment csv file

optional arguments:
  -h, --help  show this help message and exit
  --index I   Optional file selection variable
```

Figure: Analysis script help command result

Python Part

```
yigitozgumus@Yigits-MacBook-Pro|~/dev/AAPP_Project on master!?
± python src/python/generate_graph_directories.py -h
usage: generate_graph_directories.py [-h] [--single S [S ...]] [--node N] G F

Generate directed graphs with different edge creation probabilities.

positional arguments:
  G                     Number of graphs for each edge creation interval
  F                     The Target Location of the Graph files

optional arguments:
  -h, --help            show this help message and exit
  --single S [S ...]    Single node class Graph creation for fast testing
                        between 5_50, 50_100 , 100_500 , 500_1000
  --node N              Specific Node selection for Graph Creation. This option
                        takes precedence over the Single node class selection.
```

Figure: generate graph directories script help command result

C++ Part

- Application Class uses:

C++ Part

- Application Class uses:
  - Analyzer, Visualize and Session Classes

C++ Part

- Application Class uses:
  - Analyzer, Visualize and Session Classes
- Analyzer Class uses:

C++ Part

- Application Class uses:
  - Analyzer, Visualize and Session Classes
- Analyzer Class uses:
  - Nuutila, Pearce, Tarjan, Timer and GraphComponent Classes

C++ Part

- Application Class uses:
  - Analyzer, Visualize and Session Classes
- Analyzer Class uses:
  - Nuutila, Pearce, Tarjan, Timer and GraphComponent Classes
- StorageItems struct is used by all algorithm classes to transfer the execution data

C++ Part

- Application Class uses:
  - Analyzer, Visualize and Session Classes
- Analyzer Class uses:
  - Nuutila, Pearce, Tarjan, Timer and GraphComponent Classes
- StorageItems struct is used by all algorithm classes to transfer the execution data

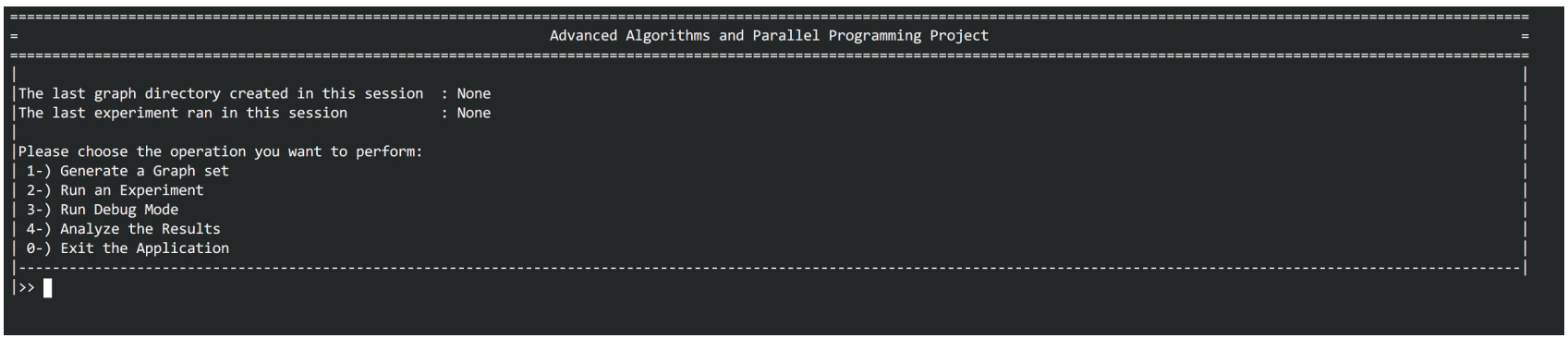
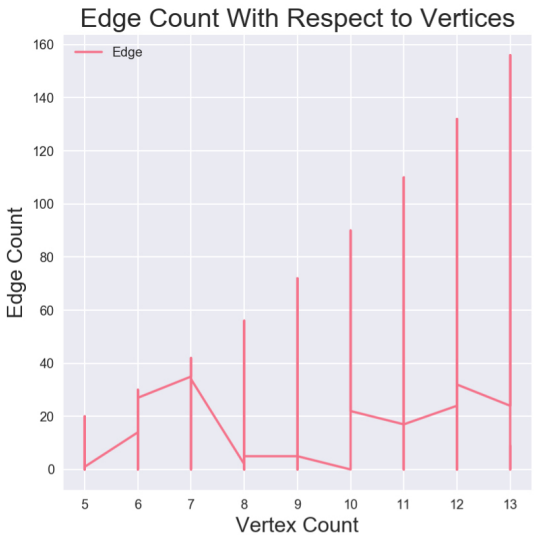
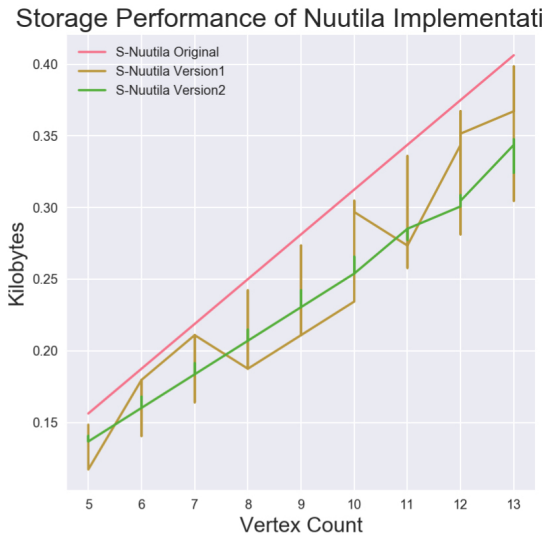


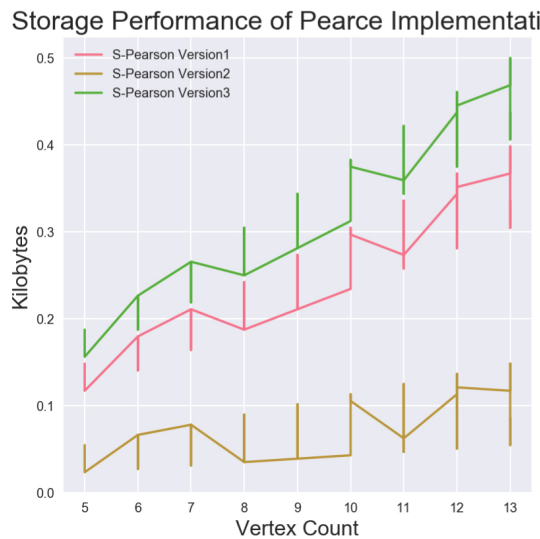
Figure: The main screen of the Application

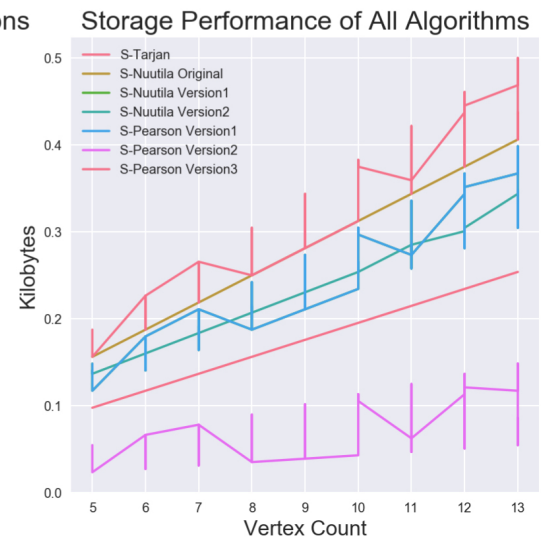
Current Section

Analysis result of the first 5000 Graphs

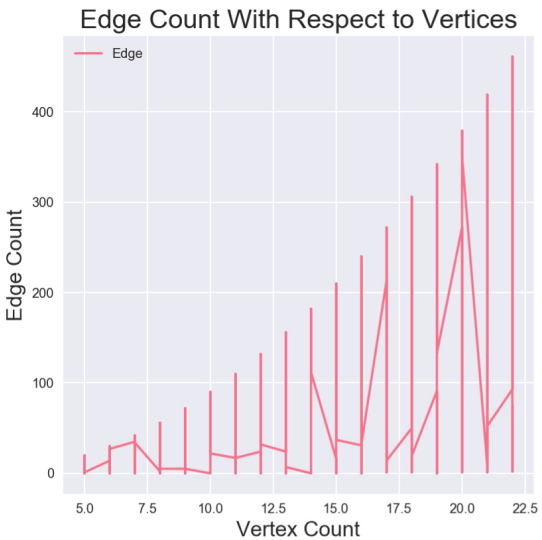


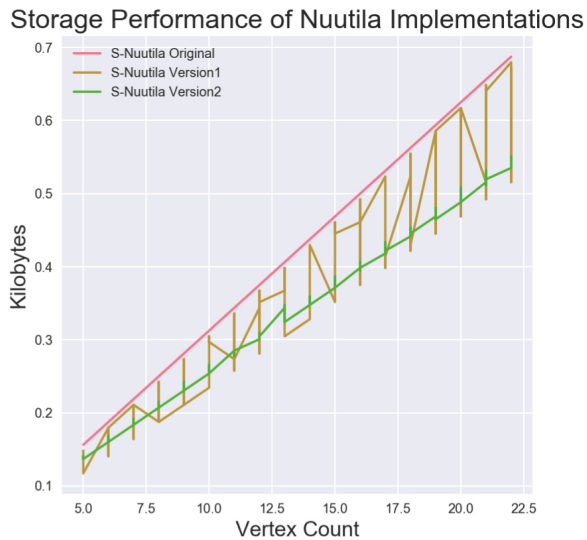


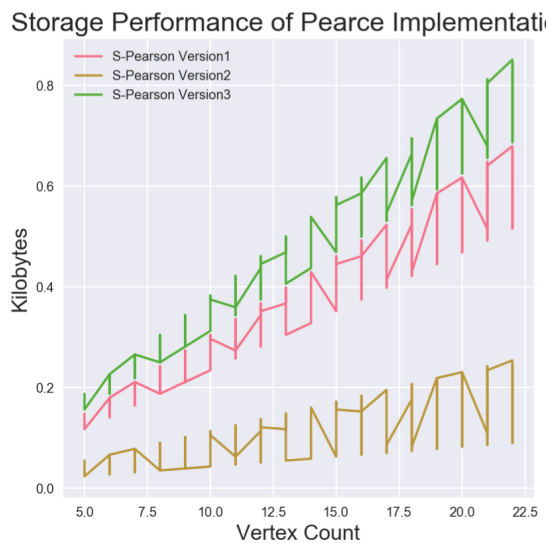


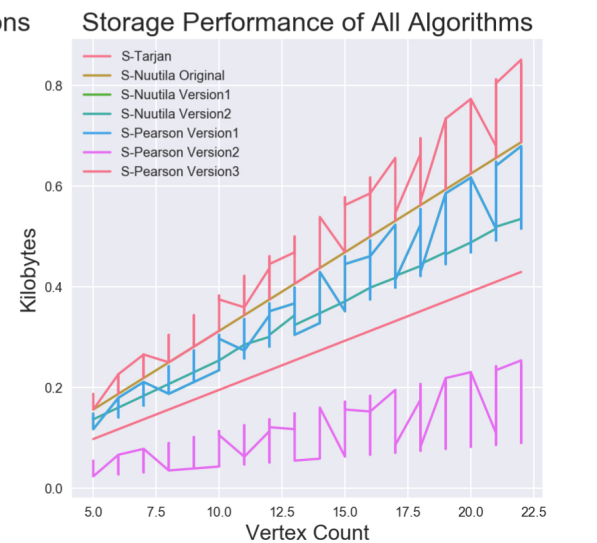


Analysis result of the first 10000 Graphs

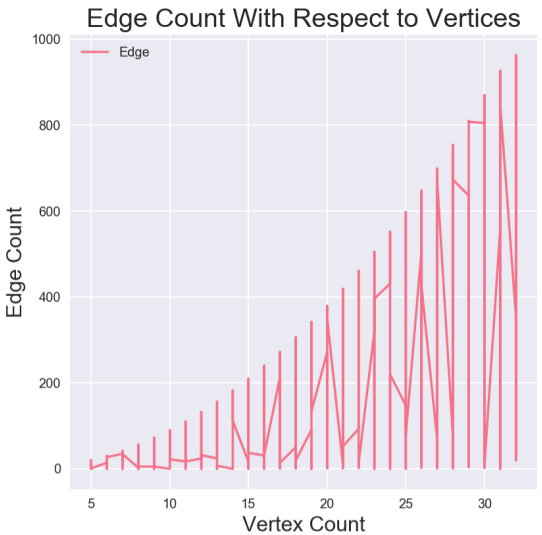


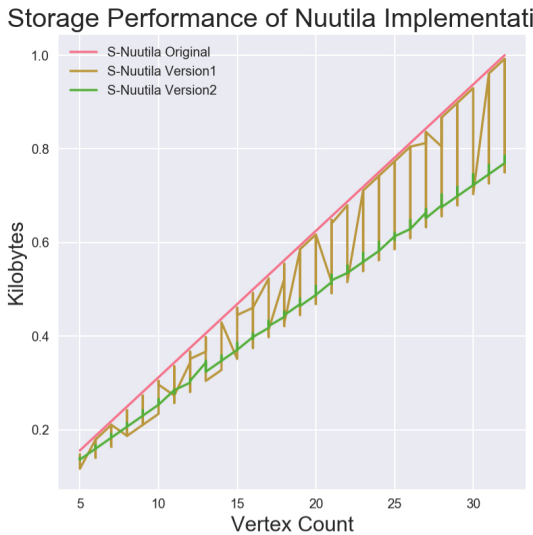


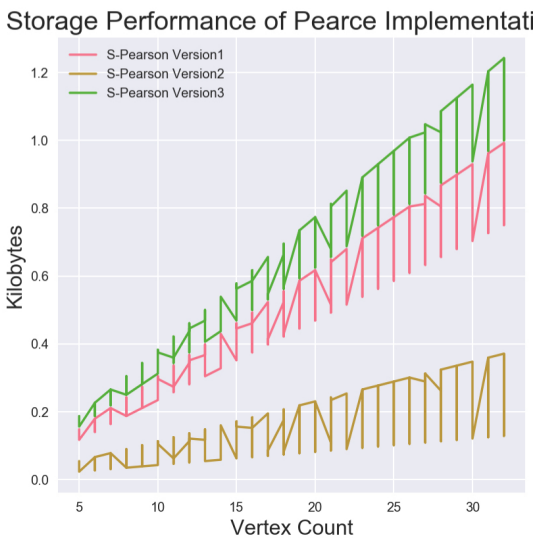


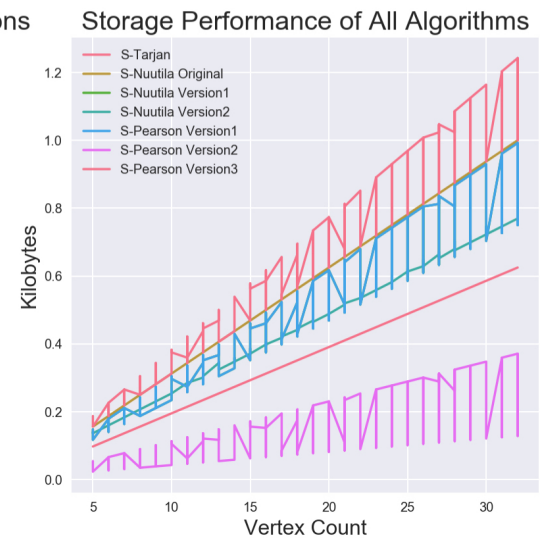


Analysis result of the first 15000 Graphs

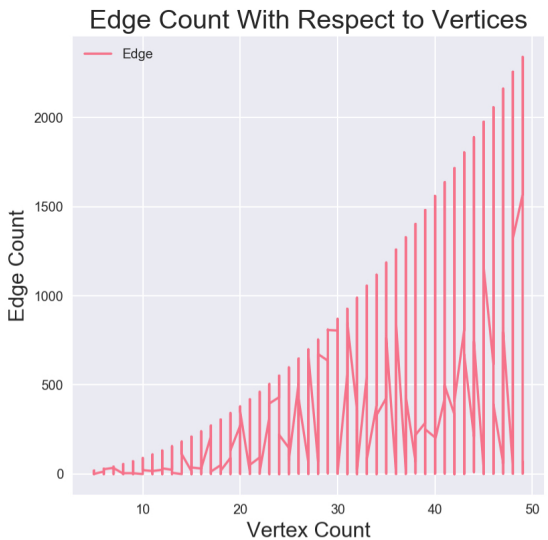


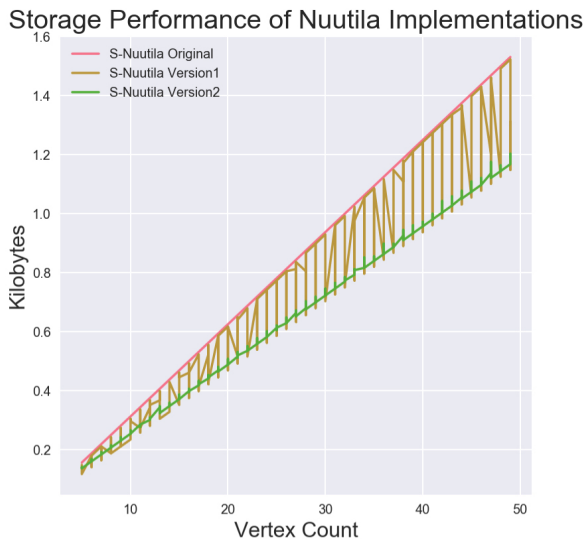


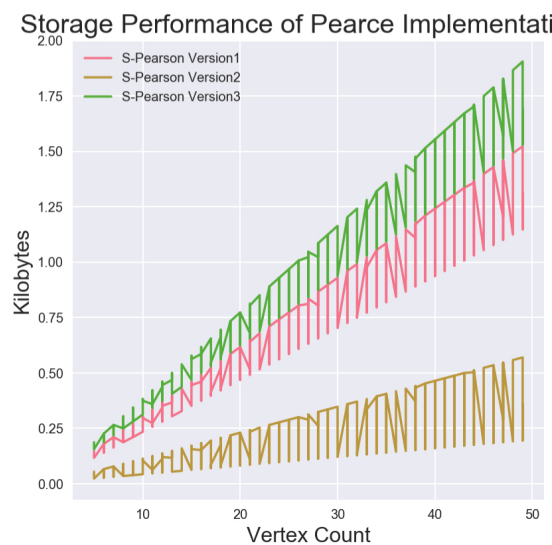


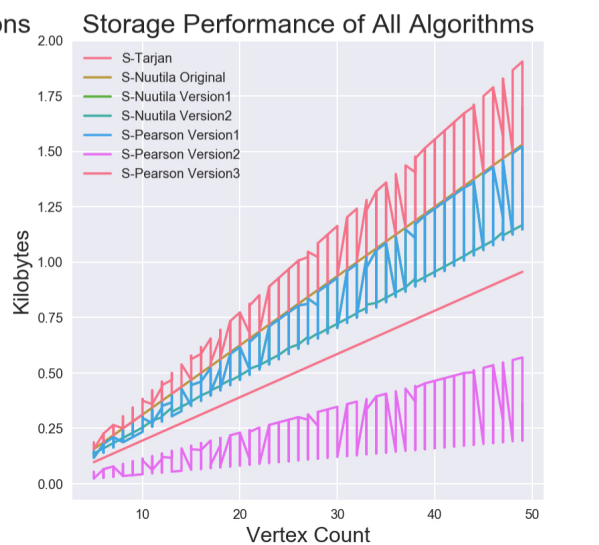


Analysis result of the first 25000 Graphs

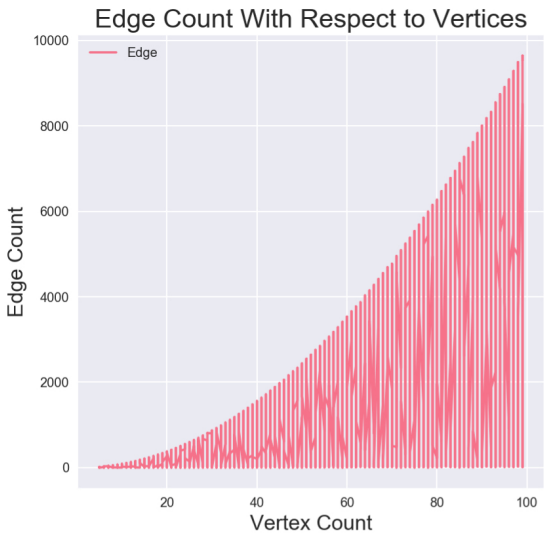


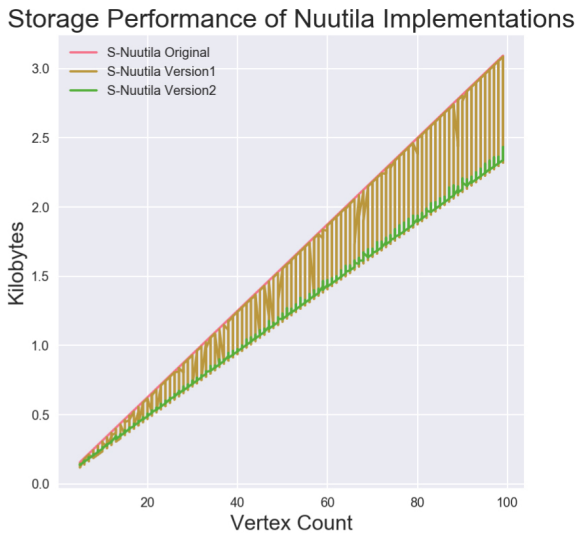


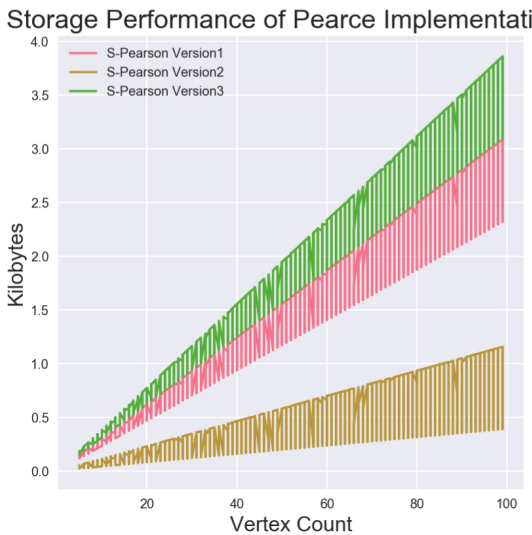


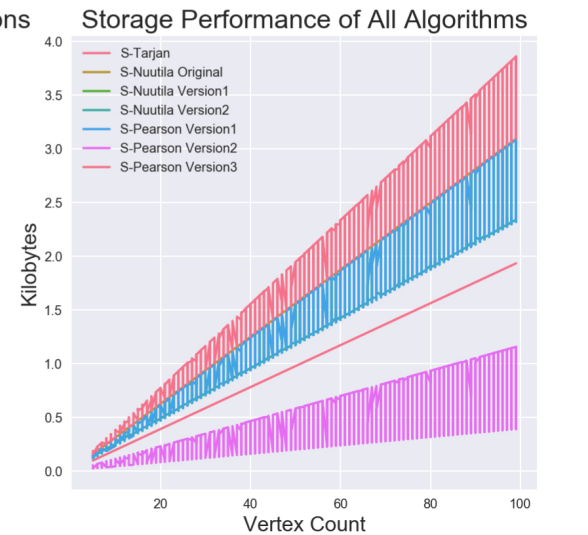


Analysis result of the first 50000 Graphs

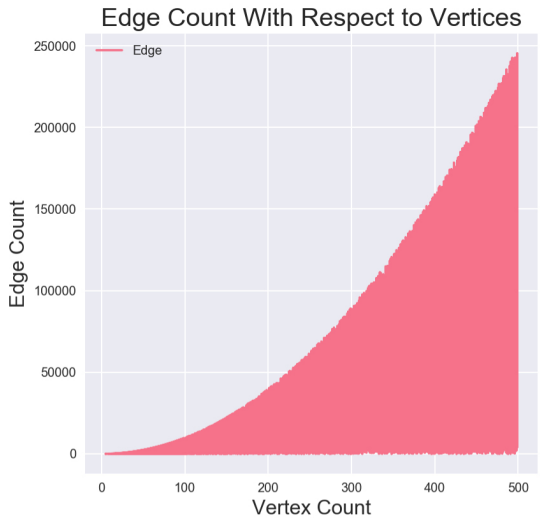


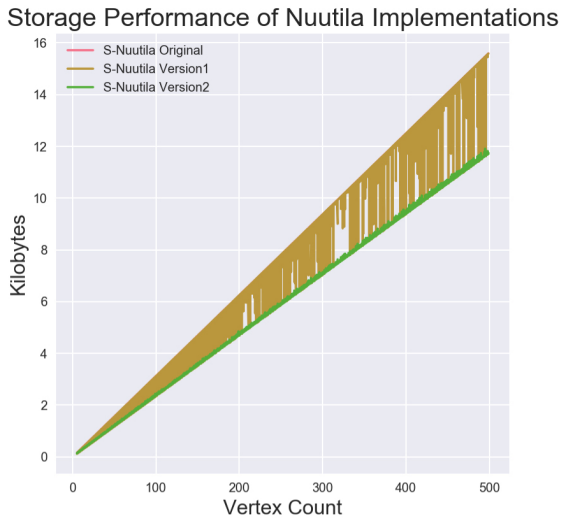


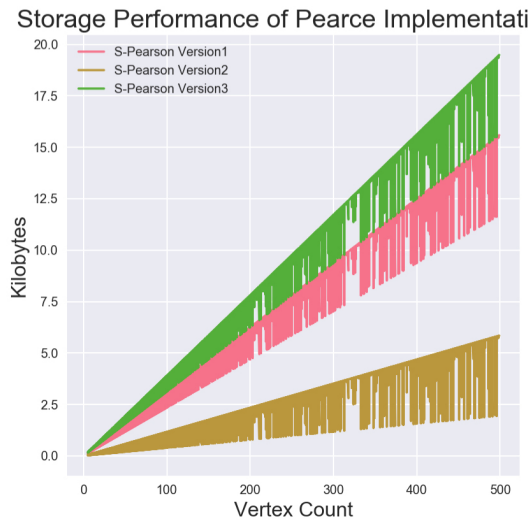


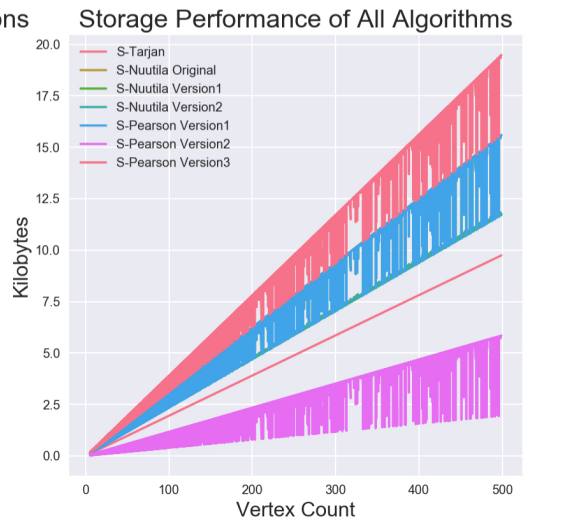


Analysis result of the first 75000 Graphs

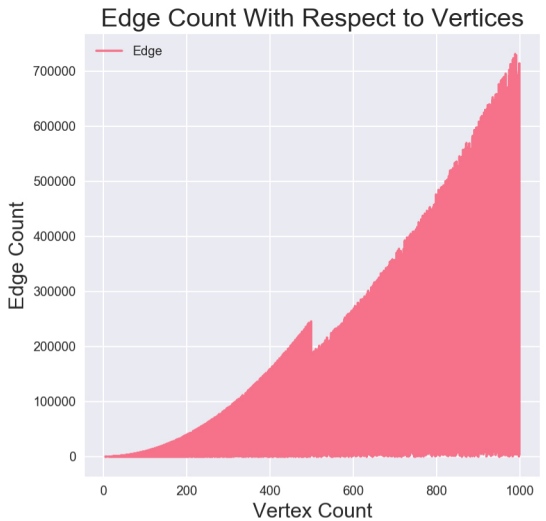


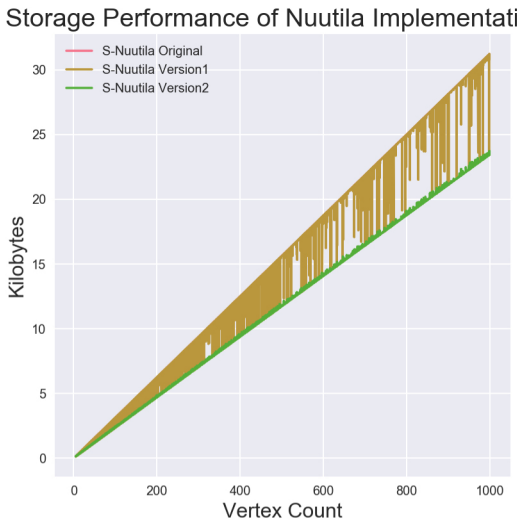


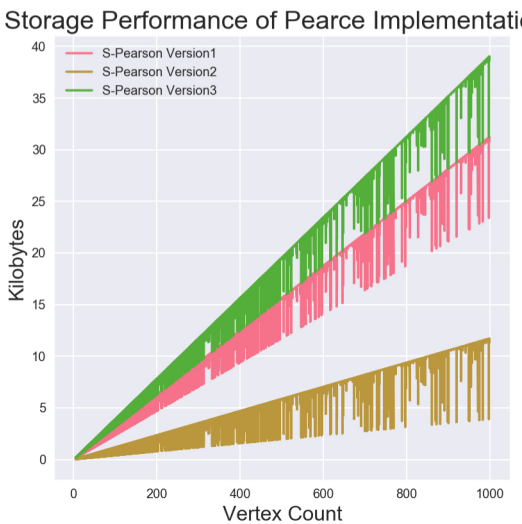


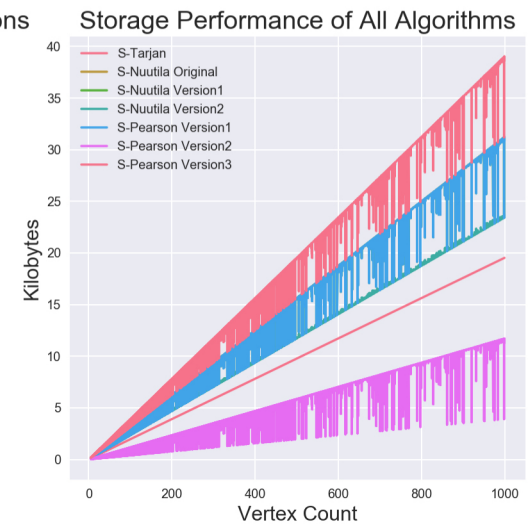


Analysis result of all of the Graphs









Analysis Results

- For Nuutila algorithms, Version 2 uses the least amount of storage.

Analysis Results

- For Nuutila algorithms, Version 2 uses the least amount of storage.
- For Pearce algorithms, Version 2 Uses the least amount of storage.

Analysis Results

- For Nuutila algorithms, Version 2 uses the least amount of storage.
- For Pearce algorithms, Version 2 Uses the least amount of storage.
- Among all the algorithms, Pearce Version 2 uses the least amount of storage .

Demo ?