UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE ED ELETTRICA E MATEMATICA APPLICATA

Corso di Laurea Magistrale in Ingegneria Informatica

Parallelization and performance evaluation of Tarjan's strongly connected components algorithm

Lecturer:

Ch.mo Prof.

Francesco MOSCATO

fmoscato@unisa.it

Authors:

Antonio LANGELLA

Mat. 0622702011

a.langella31@studenti.unisa.it

Michele MARSICO

Mat. 0622702012

m.marsico10@studenti.unisa.it

Salvatore PAOLINO

Mat. 0622702016

s.paolino6@studenti.unisa.it

Davide RISI

Mat. 0622702013

d.risi2@studenti.unisa.it

Free as in Freedom

ANNO ACCADEMICO 2022/2023

CONTENTS

1 Introduction **9**

1.1 Problem description . . . 9

1.2 Theoretical recalls on used technologies . . . 10

1.2.1 GPU . . . 10

1.2.2 CUDA . . . 10

1.2.3 MPI . . . 12

1.3 Metrics and methodology . . . 12

1.3.1 Speedup . . . 12

1.3.2 Efficiency . . . 12

1.3.3 Amdahl's law . . . 13

1.3.4 An analytical approach for calculating the speedup upper bound . . . 13

1.3.5 Our methodology for calculating the speedup upper bound . . . 15

1.3.6 Input graphs used for the measurements . . . 16

1.4 Sequential implementation of Tarjan's algorithm . . . 17

1.5 Experimental setup . . . 17

1.5.1 Hardware . . . 17

1.5.1.1 Machine A . . . 18

CPU . . . 18

Memory . . . 19

GPU . . . 20

1.5.1.2 Machine B . . . 21

CPU . . . 21

Memory . . . 22

1.5.1.3 Machine C . . . 23

CPU . . . 23

Memory . . . 24
1.5.1.4 Machine P4 . . . 25
CPU . . . 25
Memory . . . 25
1.5.1.5 Machine P3 . . . 26
CPU . . . 26
Memory . . . 26
1.5.2 Software . . . 27

2 MPI implementation **29**
2.1 Algorithm idea . . . 29
2.1.1 Description of the algorithm . . . 29
2.1.1.1 Communication . . . 30
2.2 Experimental results . . . 30
2.2.1 Fully connected graph . . . 32
2.2.1.1 Optimization 0 . . . 32
2.2.1.2 Optimization 1 . . . 32
2.2.1.3 Optimization 2 . . . 33
2.2.1.4 Optimization 3 . . . 33
2.2.2 Fully disconnected graph . . . 34
2.2.2.1 Optimization 0 . . . 34
2.2.2.2 Optimization 1 . . . 34
2.2.2.3 Optimization 2 . . . 35
2.2.2.4 Optimization 3 . . . 35
2.2.3 Random-150000 graph . . . 36
2.2.3.1 Optimization 0 . . . 36
2.2.3.2 Optimization 1 . . . 36
2.2.3.3 Optimization 2 . . . 37
2.2.3.4 Optimization 3 . . . 37
2.2.4 Random-250000 graph . . . 38
2.2.4.1 Optimization 0 . . . 38
2.2.4.2 Optimization 1 . . . 38
2.2.4.3 Optimization 2 . . . 39
2.2.4.4 Optimization 3 . . . 39
2.2.5 Random-500000 graph . . . 40
2.2.5.1 Optimization 0 . . . 40
2.2.5.2 Optimization 1 . . . 40
2.2.5.3 Optimization 2 . . . 41

2.2.5.4 Optimization 3 . . . 41
2.2.6 Tile-52000 graph . . . 42
2.2.6.1 Optimization 0 . . . 42
2.2.6.2 Optimization 1 . . . 42
2.2.6.3 Optimization 2 . . . 43
2.2.6.4 Optimization 3 . . . 43
2.2.7 Tile-205000 graph . . . 44
2.2.7.1 Optimization 0 . . . 44
2.2.7.2 Optimization 1 . . . 44
2.2.7.3 Optimization 2 . . . 45
2.2.7.4 Optimization 3 . . . 45
2.2.8 Tile-410000 graph . . . 46
2.2.8.1 Optimization 0 . . . 46
2.2.8.2 Optimization 1 . . . 46
2.2.8.3 Optimization 2 . . . 47
2.2.8.4 Optimization 3 . . . 47
2.2.9 Tile-820000 graph . . . 48
2.2.9.1 Optimization 0 . . . 48
2.2.9.2 Optimization 1 . . . 48
2.2.9.3 Optimization 2 . . . 49
2.2.9.4 Optimization 3 . . . 49
2.3 Considerations . . . 50

3 CUDA implementation with global memory 51

3.1 Algorithm idea . . . 51
3.1.1 Remarkable results in research literature . . . 51
3.1.2 Graph preprocessing: trivial SCCs deletion . . . 52
3.1.3 A CUDA-friendly graph representation . . . 52
3.1.4 Steps of the algorithm . . . 53
3.2 Choosing the block size . . . 53
3.3 Experimental results . . . 55
3.3.1 Reference sequential algorithm for evaluating the speedup . . . 55
3.3.2 Measured metrics . . . 55
3.3.3 Fully connected graph . . . 57
3.3.3.1 Optimization 0 . . . 57
3.3.3.2 Optimization 1 . . . 57
3.3.3.3 Optimization 2 . . . 58
3.3.3.4 Optimization 3 . . . 58

3.3.4 Fully disconnected graph . . . 59
3.3.4.1 Optimization 0 . . . 59
3.3.4.2 Optimization 1 . . . 59
3.3.4.3 Optimization 2 . . . 60
3.3.4.4 Optimization 3 . . . 60
3.3.5 Random-150000 graph . . . 61
3.3.5.1 Optimization 0 . . . 61
3.3.5.2 Optimization 1 . . . 61
3.3.5.3 Optimization 2 . . . 62
3.3.5.4 Optimization 3 . . . 62
3.3.6 Random-250000 graph . . . 63
3.3.6.1 Optimization 0 . . . 63
3.3.6.2 Optimization 1 . . . 63
3.3.6.3 Optimization 2 . . . 64
3.3.6.4 Optimization 3 . . . 64
3.3.7 Random-500000 graph . . . 65
3.3.7.1 Optimization 0 . . . 65
3.3.7.2 Optimization 1 . . . 65
3.3.7.3 Optimization 2 . . . 66
3.3.7.4 Optimization 3 . . . 66
3.3.8 Tile-52000 graph . . . 67
3.3.8.1 Optimization 0 . . . 67
3.3.8.2 Optimization 1 . . . 67
3.3.8.3 Optimization 2 . . . 68
3.3.8.4 Optimization 3 . . . 68
3.3.9 Tile-205000 graph . . . 69
3.3.9.1 Optimization 0 . . . 69
3.3.9.2 Optimization 1 . . . 69
3.3.9.3 Optimization 2 . . . 70
3.3.9.4 Optimization 3 . . . 70
3.3.10 Tile-410000 graph . . . 71
3.3.10.1 Optimization 0 . . . 71
3.3.10.2 Optimization 1 . . . 71
3.3.10.3 Optimization 2 . . . 72
3.3.10.4 Optimization 3 . . . 72
3.3.11 Tile-820000 graph . . . 73
3.3.11.1 Optimization 0 . . . 73

3.3.11.2 Optimization 1 . . . 73
3.3.11.3 Optimization 2 . . . 74
3.3.11.4 Optimization 3 . . . 74
3.4 Considerations . . . 75

4 Hybrid CUDA and MPI implementation 76
4.1 Algorithm idea . . . 76
4.2 Experimental results . . . 76
4.2.1 Fully connected graph . . . 78
4.2.1.1 Optimization 0 . . . 78
4.2.1.2 Optimization 1 . . . 78
4.2.1.3 Optimization 2 . . . 79
4.2.1.4 Optimization 3 . . . 79
4.2.2 Fully disconnected graph . . . 80
4.2.2.1 Optimization 0 . . . 80
4.2.2.2 Optimization 1 . . . 80
4.2.2.3 Optimization 2 . . . 81
4.2.2.4 Optimization 3 . . . 81
4.2.3 Random-150000 graph . . . 82
4.2.3.1 Optimization 0 . . . 82
4.2.3.2 Optimization 1 . . . 82
4.2.3.3 Optimization 2 . . . 83
4.2.3.4 Optimization 3 . . . 83
4.2.4 Random-250000 graph . . . 84
4.2.4.1 Optimization 0 . . . 84
4.2.4.2 Optimization 1 . . . 84
4.2.4.3 Optimization 2 . . . 85
4.2.4.4 Optimization 3 . . . 85
4.2.5 Random-500000 graph . . . 86
4.2.5.1 Optimization 0 . . . 86
4.2.5.2 Optimization 1 . . . 86
4.2.5.3 Optimization 2 . . . 87
4.2.5.4 Optimization 3 . . . 87
4.2.6 Tile-52000 graph . . . 88
4.2.6.1 Optimization 0 . . . 88
4.2.6.2 Optimization 1 . . . 88
4.2.6.3 Optimization 2 . . . 89
4.2.6.4 Optimization 3 . . . 89

4.2.7 Tile-205000 graph . . . 90
4.2.7.1 Optimization 0 . . . 90
4.2.7.2 Optimization 1 . . . 90
4.2.7.3 Optimization 2 . . . 91
4.2.7.4 Optimization 3 . . . 91
4.2.8 Tile-410000 graph . . . 92
4.2.8.1 Optimization 0 . . . 92
4.2.8.2 Optimization 1 . . . 92
4.2.8.3 Optimization 2 . . . 93
4.2.8.4 Optimization 3 . . . 93
4.2.9 Tile-820000 graph . . . 94
4.2.9.1 Optimization 0 . . . 94
4.2.9.2 Optimization 1 . . . 94
4.2.9.3 Optimization 2 . . . 95
4.2.9.4 Optimization 3 . . . 95
4.3 Considerations . . . 96

5 CUDA implementation with texture memory 97
5.1 Algorithm idea . . . 97
5.2 Experimental results . . . 98
5.2.1 Fully connected graph . . . 100
5.2.1.1 Optimization 0 . . . 100
5.2.1.2 Optimization 1 . . . 100
5.2.1.3 Optimization 2 . . . 101
5.2.1.4 Optimization 3 . . . 101
5.2.2 Fully disconnected graph . . . 102
5.2.2.1 Optimization 0 . . . 102
5.2.2.2 Optimization 1 . . . 102
5.2.2.3 Optimization 2 . . . 103
5.2.2.4 Optimization 3 . . . 103
5.2.3 Random-150000 graph . . . 104
5.2.3.1 Optimization 0 . . . 104
5.2.3.2 Optimization 1 . . . 104
5.2.3.3 Optimization 2 . . . 105
5.2.3.4 Optimization 3 . . . 105
5.2.4 Random-250000 graph . . . 106
5.2.4.1 Optimization 0 . . . 106
5.2.4.2 Optimization 1 . . . 106

5.2.4.3 Optimization 2 . . . 107
5.2.4.4 Optimization 3 . . . 107
5.2.5 Random-500000 graph . . . 108
5.2.5.1 Optimization 0 . . . 108
5.2.5.2 Optimization 1 . . . 108
5.2.5.3 Optimization 2 . . . 109
5.2.5.4 Optimization 3 . . . 109
5.2.6 Tile-52000 graph . . . 110
5.2.6.1 Optimization 0 . . . 110
5.2.6.2 Optimization 1 . . . 110
5.2.6.3 Optimization 2 . . . 111
5.2.6.4 Optimization 3 . . . 111
5.2.7 Tile-205000 graph . . . 112
5.2.7.1 Optimization 0 . . . 112
5.2.7.2 Optimization 1 . . . 112
5.2.7.3 Optimization 2 . . . 113
5.2.7.4 Optimization 3 . . . 113
5.2.8 Tile-410000 graph . . . 114
5.2.8.1 Optimization 0 . . . 114
5.2.8.2 Optimization 1 . . . 114
5.2.8.3 Optimization 2 . . . 115
5.2.8.4 Optimization 3 . . . 115
5.2.9 Tile-820000 graph . . . 116
5.2.9.1 Optimization 0 . . . 116
5.2.9.2 Optimization 1 . . . 116
5.2.9.3 Optimization 2 . . . 117
5.2.9.4 Optimization 3 . . . 117
5.3 Considerations . . . 118

A MPI measurements on a Raspberry cluster 119
A.1 Experimental results . . . 119
A.1.1 Fully disconnected graph . . . 121
A.1.1.1 Optimization 0 . . . 121
A.1.1.2 Optimization 1 . . . 121
A.1.1.3 Optimization 2 . . . 122
A.1.1.4 Optimization 3 . . . 122
A.1.2 Random-150000 graph . . . 123
A.1.2.1 Optimization 0 . . . 123

A.1.2.2 Optimization 1 . . . 123
A.1.2.3 Optimization 2 . . . 124
A.1.2.4 Optimization 3 . . . 124
A.1.3 Random-250000 graph . . . 125
A.1.3.1 Optimization 0 . . . 125
A.1.3.2 Optimization 1 . . . 125
A.1.3.3 Optimization 2 . . . 126
A.1.3.4 Optimization 3 . . . 126
A.1.4 Random-500000 graph . . . 127
A.1.4.1 Optimization 0 . . . 127
A.1.4.2 Optimization 1 . . . 127
A.1.4.3 Optimization 2 . . . 128
A.1.4.4 Optimization 3 . . . 128
A.1.5 Tile-52000 graph . . . 129
A.1.5.1 Optimization 0 . . . 129
A.1.5.2 Optimization 1 . . . 129
A.1.5.3 Optimization 2 . . . 130
A.1.5.4 Optimization 3 . . . 130
A.1.6 Tile-205000 graph . . . 131
A.1.6.1 Optimization 0 . . . 131
A.1.6.2 Optimization 1 . . . 131
A.1.6.3 Optimization 2 . . . 132
A.1.6.4 Optimization 3 . . . 132
A.1.7 Tile-410000 graph . . . 133
A.1.7.1 Optimization 0 . . . 133
A.1.7.2 Optimization 1 . . . 133
A.1.7.3 Optimization 2 . . . 134
A.1.7.4 Optimization 3 . . . 134
A.1.8 Tile-820000 graph . . . 135
A.1.8.1 Optimization 0 . . . 135
A.1.8.2 Optimization 1 . . . 135
A.1.8.3 Optimization 2 . . . 136
A.1.8.4 Optimization 3 . . . 136

B API Documentation **137**

References **138**

List of Figures **140**

CHAPTER 1

INTRODUCTION

The assigned problem is to provide a parallel version of the Tarjan's algorithm to find Strongly Connected Components (SCC) in a graph with an implementation that had to use an hybrid message passing/CUDA paradigm, implemented by using MPI and CUDA. In particular, in this report different implementations of the Tarjan's algorithm will be presented, and their performances will be evaluated:

- a sequential version described in 1.4;
- a sequential version with graph preprocessing described in 3.3.1;
- a parallel version with MPI described in 2.1;
- a parallel version with CUDA described in 3.1.4;
- a parallel version based on a hybrid CUDA/MPI approach described in 4.1.
- a parallel version with CUDA which exploits texture memory described in 5.1;

In the following pages the problem faced is going to be described in a detailed way, along with the most important theoretical concepts about GPUs (Graphics Processing Unit), CUDA and MPI. Great attention is going to be reserved to the description of the different case studies considered.

1.1 Problem description

The goal of this project is to parallelize and evaluate performances of Tarjan's Algorithm, mainly by using MPI and CUDA. **Tarjan's strongly connected components algorithm** is an algorithm in graph theory for finding the strongly connected components (SCCs) of a directed graph. It runs in **O(n+m)**, where n is the number of vertices and m is the number of edges, thanks to a depth-first traversal procedure to output the list of all SCCs for a given directed graph [1].

In the mathematical theory of directed graphs, a graph is said to be **strongly connected** if every vertex is reachable from every other vertex. The **strongly connected components** of an arbitrary directed graph form a partition into subgraphs that are themselves strongly connected [2].

The algorithm takes a directed graph as input, and produces a partition of the graph's vertices into the graph's strongly connected components. Each vertex of the graph appears in exactly one of the strongly connected components. Any vertex that is not on a directed cycle forms a strongly connected component all by itself: for example, a vertex whose in-degree or out-degree is 0, or any vertex of an acyclic graph.

The basic idea of the algorithm is this: a depth-first search (DFS) begins from an arbitrary start node (and subsequent depth-first searches are conducted on any nodes that have not yet been found). As usual with DFS, the search visits every node of the graph exactly once, declining to revisit any node that has already been visited. Thus, the collection of search trees is a spanning forest of the graph. The strongly connected components will be recovered as certain subtrees of this forest. The roots of these subtrees are called the "roots" of the strongly connected components. Any node of a strongly connected component might serve as a root, if it happens to be the first node of a component that is discovered by search [3].

The nodes are thus placed in a stack in the order in which they are visited. When the search moves up from a visited subtree, the corresponding nodes are taken from the stack, and it is determined whether each node is the root of a strongly connected component. If it is, then it forms a strongly connected component together with all those taken before.

1.2 Theoretical recalls on used technologies

1.2.1 GPU

A graphics processing unit (GPU) is a specialized electronic circuit designed to manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device [4]. GPUs are used in embedded systems, smartphones, PC, workstations, and game consoles. Since the beginning of the century GPUs have been used not only for graphic computations but also for general-purpose computing (typically done by CPUs). The main differences between GPUs and CPUs are:

- GPUs have much more parallel execution units than CPUs.
- GPUs have deeper pipes than CPUs.
- CPUs can handle complex control logic, while GPUs are optimized for simple control logic.

GPUs are perfect to run in parallel the same set of simple instructions on a lot of different data.

1.2.2 CUDA

CUDA (Compute Unified Device Architecture) is a parallel computing platform and application programming interface (API), created by Nvidia, that allows software to use certain types of graphics processing units (GPUs) for general purpose processing, an approach called general-purpose computing on

GPUs (GPGPU). CUDA is a software layer that gives direct access to the GPU's virtual instruction set and parallel computational elements, for the execution of compute kernels [5]. To fully understand the results of this report, it is fundamental to give some definitions:

- Host is the computer on which GPU is mounted.
- Device is another word through which GPU is indicated.
- Compute Capability (c.c.) identifies the features supported by the GPU hardware (its version).
- Kernel is a CUDA function written to run on device.
- Execution cores are the computing elements mounted on a GPU. They are a sort of ALUs (Arithmetic Logic Unit).
- Streaming Multiprocessor (SM) is a GPU unit that consists of registers, several caches, warp schedulers and execution cores.
- Warp is a grouping of 32 threads.
- Thread is a flow of execution.
- Block is a group of threads organized in 1D, 2D or 3D logical arrays.
- Grid is a group of blocks organized in 1D, 2D or 3D logical arrays.

GPUs are characterized by a particular memory hierarchy. All CUDA threads in a block have access to the resources of the Streaming Multiprocessor assigned to the block to which the threads belong, that are registers and shared memory. Threads belonging to different blocks cannot share registers and shared memory. The **Shared Memory** is a memory with a configurable size (32 to 48 KByte), it has a very low latency (2 clocks per cycle), and a throughput of 32 bit/2 cycles. This type of memory is not persistent, that means its status is not maintained between different kernel calls. Because it is shared between all threads in a block, synchronization is needed.

All CUDA threads, also, can access to all the memory types available on the GPU:

- **Global Memory** is the largest memory available on a device. It is comparable to a RAM for CPU and its status is maintained among different kernel launches. This memory can be accessed both read/write from all threads of the kernel grid, and it has a very high bandwidth (throughput up to 144-177 GB/s) and a very high latency (about 400-800 clock cycles).
- **Constant Memory** is a read-only memory, and it's very efficient when all threads in a warp request the same memory address. The throughput is the same as the shared memory.
- **Texture Memory** is a read-only memory with a dedicated cache, which means that subsequent uses of the same texture memory are extremely fast. It can be used to obtain a greater efficient bandwidth,

reducing calls to global memory on DRAM. Each SM has various texture fetch units that are dedicated units to access the texture memory.

CPU can access and initialize both constant and texture memories. Global, constant and texture memories have persistent storage duration.

1.2.3 MPI

Message Passing Interface (MPI) is a standardized and portable message-passing standard designed to function on parallel computing architectures. The MPI standard defines the syntax and semantics of library routines that are useful to a wide range of users writing portable message-passing programs in C, C++, and Fortran [6].

MPI is based on a Distributed Memory Model, in which several nodes send and receive messages, so that a sort of memory abstraction is created. MPI is not a programming language, but a framework in which a fundamental role is played by the middleware MPI, which manages synchronization among parallel processes and messages mechanism.

One of the main MPI concepts is the meaning of communicator, that is a sort of container of processes, in which each of them is identified by a rank. The size of a communicator is given by the number of processes it manages and cannot be changed after its creation.

1.3 Metrics and methodology

1.3.1 Speedup

Speedup is a number that measures the relative performance of two systems processing the same problem. More technically, it is the improvement in speed of execution of a task executed on two similar architectures with different resources [7]. Speedup can also be used as a measure of the relative speed of a parallel algorithm compared to its sequential counterpart. The concept of speedup is connected to the Amdahl's law. The curve of the speedup depends on the number of processors P, the size N of the problem, and the time complexity of an algorithm. So in most cases, speedup functions, plotted considering the same algorithm launched on a different number of processors, are not linear but present a particular trend that can be explained in the light of machine characteristics, algorithm complexity, parallelizable portions, framework used, etc.

1.3.2 Efficiency

Furthermore, it exists another metric that measure the quality of a parallel algorithm, the **efficiency**. It measures how many times a parallel version of an algorithm executed on P processors is faster than the same parallel version executed on only one processor. In particular, the relative efficiency is:

$$E_r = \frac{T_1}{P * T_P} \tag{1.1}$$

where T_1 is the time required to execute the algorithm on one processor, P is the number of processors and T_P is the time required to execute the same algorithm on P processors.

1.3.3 Amdahl's law

Amdahl's law is a formula which gives the theoretical speedup of an algorithm when only a percentage p of the algorithm can be parallelized. Amdahl's law is often used in parallel computing to predict the theoretical speedup when using multiple processors. For example, if a program needs 20 hours to complete using a single thread, but a one-hour portion of the program cannot be parallelized, therefore only the remaining 19 hours' (p = 0.95) execution time can be parallelized, then regardless of how many threads are devoted to a parallelized execution of this program, the minimum execution time cannot be less than one hour. Hence, the theoretical speedup is limited to at most 20 times the single thread performance, ($\frac{1}{1-p} = 20$).

Amdahl's law can be formulated in the following way:

$$S_{latency}(s) = \frac{1}{(1-p) + \frac{p}{s}} \tag{1.2}$$

where:

- $S_{latency}$ is the theoretical speedup of the execution of the whole task;
- s is the speedup of the part of the task that benefits from improved system resources;
- p is the proportion of execution time that the part benefiting from improved resources originally occupied.

Furthermore, these equations

$$\begin{cases} S_{latency}(s) \leq \dfrac{1}{(1-p)} \\ \lim\limits_{s \to \infty} S_{latency}(s) = \dfrac{1}{(1-p) + \frac{p}{s}} \end{cases} \tag{1.3}$$

show that the theoretical speedup of the execution of the whole task increases with the improvement of the resources of the system and that regardless of the magnitude of the improvement, the theoretical speedup is always limited by the part of the task that cannot benefit from the improvement.

Amdahl's law can be used to compute the upper bound of the speedup of an algorithm, assuming that the speedup of the parallel part is equal to the number of processes.

1.3.4 An analytical approach for calculating the speedup upper bound

As said before, in order to use Amdahl's law to calculate an upper bound for the speedup of a parallel program, we need to estimate the percentage p of execution time of the part that could benefit from parallelization in the sequential program.

A possible approach for estimating p is to manually calculate the costs of the parallelizable and sequential parts of the algorithm by counting the number of elemental operations performed. The implicit hypotesis is

that each elemental operation approximately takes the same time, thus the execution time of an algorithm would be proportional to it's cost. Notice how using the asymptotic complexity would lead to a wrong estimation, because it ignores lower order terms and multiplication factors that could be significant for determining the program execution time.

In this paragraph we will see an example application of this approach for estimating p for our version of Tarjan's algorithm parallelized with CUDA. For further information on this algorithm, see 3.1.4.

The functions called by the sequential algorithm are `cuda_graph_load_from_file()` `scc_set_init()` `preprocess()` `cuda_graph_to_graph()` `graph_tarjan_foreach()` `graph_num_vertex()` `scc_set_save_to_file()` `graph_free()` `cuda_graph_free()` `scc_set_free()` and `free()`. From the taken measurements, we determined that most of the functions' execution time is negligible. The ones that influence the execution time the most are `cuda_graph_load_from_file()` `preprocess()` `cuda_graph_to_graph()` and `graph_tarjan_foreach()`. For each of these functions we calculate the cost function $T(\cdot)$, with:

- v: number of vertices
- e: number of edges
- v_1: vertices not deleted after preprocess
- v_2: vertices deleted after preprocess
- e_1: edges not deleted during preprocess.

The function `cuda_graph_load_from_file()` is used to read the graph from a binary file in a format that is compatible with CUDA elaborations. The cost of this function depends on the number of vertices and edges in the graph: $T_1(v, e) = 9v + 4e + 11$.

The function `preprocess()` is used to delete trivial SCCs from the graph. This is the only function that will be parallelized by CUDA. The cost of this function depends on the number of vertices and edges in the graph: $T_2(v, e) = 3ve + 13v + 6$

The function `cuda_graph_to_graph()` is used to convert the CUDA-friendly representation of the graph into an adjacency map representation used by Tarjan's algorithm. The cost of this function depends on the number of vertices and edges in the graph after the preprocessing, as well as the number of deleted vertices: $T_3(v_1, v_2, e_1) = 10v_1 + 24e_1 + 3v_2 + 6$

The function `graph_tarjan_foreach()` actually executes Tarjan's algorithm on the preprocessed graph. Calculating the cost of this function is not trivial, for it strongly depends on the topology of the input graph. Factors that influence the cost of this function are: number of connected components, number of SCCs, degree of each vertex in the graph. In order to calculate the cost, here we assume to execute the function on a fully connected graph: $T_4(v, e) = 12v^2 + 58v + 38$.

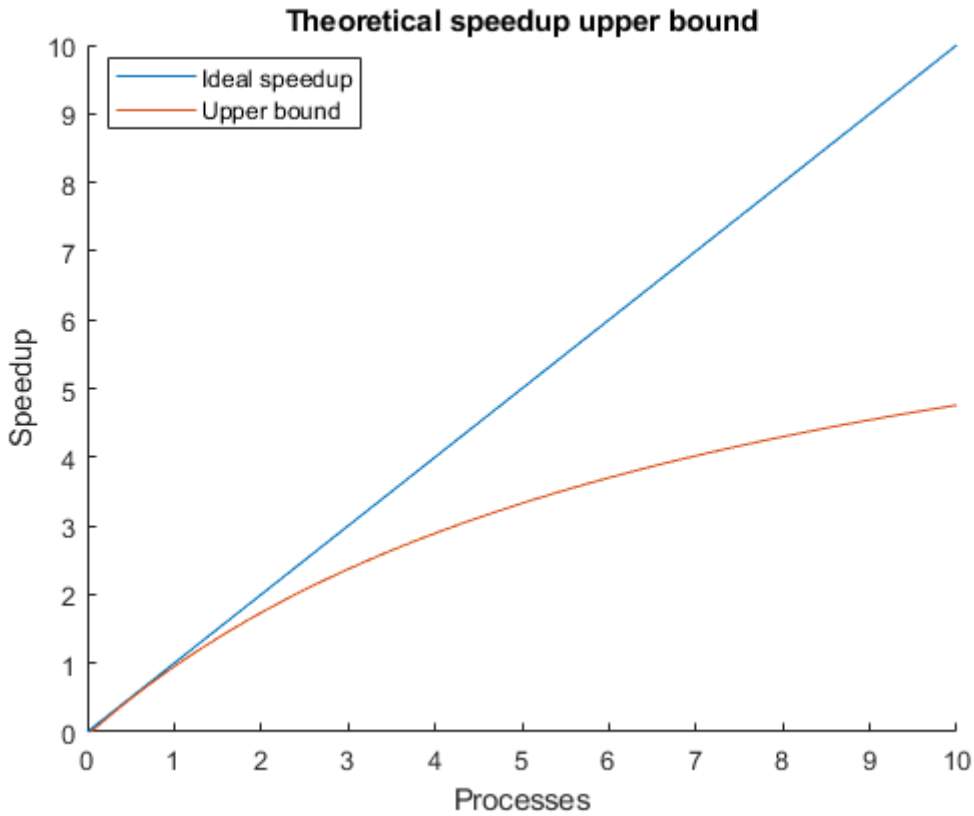
Given the assumption made for estimating the cost of `graph_tarjan_foreach()`, we have that $e = v(v-1)$,

$v_1 = v$, $e_1 = e$ and $v_2 = 0$. Thus

$$\begin{aligned} p(v) &= \frac{T_2(v,e)}{T_1(v,e) + T_2(v,e) + T_3(v_1, v_2, e_1) + T_4(v,e)} \\ &= \frac{3ve + 13v + 6}{12v^2 + 80v + 4e + 3ve + 10v_1 + 24e_1 + 3v_2 + 61} \\ &= \frac{3v^3 - 3v^2 + 13v + 6}{3v^3 + 37v^2 + 62v + 61} \end{aligned}$$

For instance, with a fully connected graph with 100 vertices, the fraction of parallelizable time is $p(100) = 0.88$. By applying Amdahl's law the maximum theoretical speedup is:

$$S_{max}(P) = \frac{1}{0.12 + \frac{0.88}{P}} = \frac{P}{0.12P + 0.88}$$



1.3.5 Our methodology for calculating the speedup upper bound

The analytical approach seen before has the following problems:

- Manually calculating the cost of the algorithm is time-consuming
- The correctness of the result relies on the strong hypothesis that execution time will be proportional to the cost of the algorithm
- The cost of the algorithm (and thus p) strongly depends on the input size of the algorithm
- Different algorithm will have different parallelized sections (for example with CUDA we parallelize the preprocessing of the graph, while with MPI we parallelized Tarjan's algorithm)

For these reasons in the following measurements we applied Amdahl's law by calculating p directly from the measurements of the corresponding sequential algorithm executed with the right input. For instance, in order to plot the speedup upper bound for the CUDA algorithm described in 3.1.4, we measure $t_{preprocess}$ (the time needed by the sequential program to pre-process the graph) and $t_{elapsed}$ (total running time of the sequential algorithm), such that we calculate

$$p = \frac{t_{preprocess}}{t_{elapsed}}$$

for each input graph. After that we calculate the maximum theoretical speedup by applying Amdahl's law as seen before. In the following chapters all of the upper bounds have been determined with this approach.

1.3.6 Input graphs used for the measurements

The implemented algorithms have been tested with different graphs, to test them under difference circumstances and to have a reliable estimate of the performances. Specifically, the following it is tested:

- Fully connected graph;

  A fully connected graph with 12500 nodes (each node is connected via edges to all other nodes in the graph).

  Such a graph was created using the sgg tool, in the root directory *./tarjan/* is as follows:

  ./bin/sgg.out ./data/fully-connected-12500.bin 12500 0

- Fully disconnected graph;

  A fully disconnected graph with 1000000 nodes and each node has no edge.

  Such a graph was created using the sgg tool, in the root directory *./tarjan/* is as follows:

  ./bin/sgg.out ./data/fully-disconnected-1000000.bin 1000000 1

- Random graph;

  Randomly generated graphs with variable number of nodes. The number of outgoing edges of each node is distributed according to the Bernoulli distribution with a given mean and variance. The Bernoulli distribution was used since it can be seen as a Gaussian distribution that operates only with positive values.

  Such a graph was created using the rsg tool, in the root directory *./tarjan/* is as follows:

  ./bin/rsg.out ./data/random-150000.bin 150000 1 0.5

  ./bin/rsg.out ./data/random-250000.bin 250000 1 0.5

  ./bin/rsg.out ./data/random-500000.bin 500000 1 0.5

  In generating the graphs, a low mean was chosen to avoid creating graphs in which there is only one SSC that includes all nodes in the graph.

  - random with 150000 nodes;
  - random with 250000 nodes;

  - random with 500000 nodes;

- Tile graph;

  Graphs generated from a seed graph. Using an integer n, a graph with 2^n replications of the seed graph is generated while retaining all the edges already present in the seed. In addition, edges are added between the different seeds of the final graph according to a Bernoulli distribution.

  Such a graph was created using the rgg tool, in the root directory *./tarjan/* is as follows:

  ./bin/rgg.out ./data/seed.bin ./data/tile-52000.bin 9 0.25

  ./bin/rgg.out ./data/seed.bin ./data/tile-205000.bin 11 0.25

  ./bin/rgg.out ./data/seed.bin ./data/tile-410000.bin 12 0.25

  ./bin/rgg.out ./data/seed.bin ./data/tile-820000.bin 13 0.25

  The seed graph used was accurately chosen to avoid the formation of one single large SCC in the final graph after its replication.

  - tile with 52000 nodes;
  - tile with 205000 nodes;
  - tile with 410000 nodes;
  - tile with 820000 nodes;

1.4 Sequential implementation of Tarjan's algorithm

We decided to represent graphs with the adjacency map representation for performance reasons. This representation is based on the adjacency list representation [8], but it uses hash tables [9] instead of lists. Our sequential implementation of Tarjan's algorithm is based on the algorithm described in [10]. The main difference is that we implemented `disc`, `low` and `stackMember` data structures as hash tables instead of arrays. This is necessary because we represented graphs with the adjacency map representation. This fact gives us an important advantage: we don't have to put constraints on the IDs of the node of the graph. In other words, the implementation from geeksforgeeks requires that vertices are numbered from 0 to $N - 1$ while we can run our implementation on a graph with arbitrary node IDs. This is a great advantage because in our MPI implementation (described in 2.1), slave processes will execute this algorithm on arbitrary subgraphs.

1.5 Experimental setup

Information provided below refers to hardware and software used during the evaluation of performances.

1.5.1 Hardware

Several machines were used to make the measurements. In the following, the following coding will be used:

- **Machine A**: Dell XPS17 laptop PC
- **Machine B**: HP Elitebook laptop
- **Machine C** A desktop PC
- **Machine P4**: A raspberry Pi4
- **Machine P3**: A raspberry Pi3

In particular:

- for the measurements of the `sequential`, `sequential_pre`, `cuda` and `cuda_texture` programs the machine A was used;
- for the measurements of the `mpi` and `cuda_mpi` programs, a cluster consisting of the machine A (master), the machines B and C (slaves) was used.

On the next pages we report the CPU, memory specifications of each machine as well as the GPU specifications of machine A.

1.5.1.1 Machine A

CPU

This CPU has 20 processors, but in order not to be redundant we only present the specifications relative to a single core.

```
processor : 0
vendor_id : GenuineIntel
cpu family : 6
model : 154
model name : 12th Gen Intel(R) Core(TM) i9-12900HK
stepping : 3
microcode : 0x423
cpu MHz : 3823.573
cache size : 24576 KB
physical id : 0
siblings : 20
core id : 0
cpu cores : 14
apicid : 0
initial apicid : 0
fpu : yes
fpu_exception : yes
cpuid level : 32
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology
nonstop_tsc cpuid aperfmperf tsc_known_freq pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16
xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm
3dnowprefetch cpuid_fault epb invpcid_single ssbd ibrs ibpb stibp ibrs_enhanced tpr_shadow vnmi flexpriority ept vpid
```

```
ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid rdseed adx smap clflushopt clwb intel_pt sha_ni xsaveopt
xsavec xgetbv1 xsaves split_lock_detect avx_vnni dtherm ida arat pln pts hwp hwp_notify hwp_act_window hwp_epp
hwp_pkg_req umip pku ospke waitpkg gfni vaes vpclmulqdq rdpid movdiri movdir64b fsrm md_clear serialize arch_lbr
flush_l1d arch_capabilities
vmx flags : vnmi preemption_timer posted_intr invvpid ept_x_only ept_ad ept_1gb flexpriority apicv tsc_offset vtpr
mtf vapic ept vpid unrestricted_guest vapic_reg vid ple shadow_vmcs ept_mode_based_exec tsc_scaling usr_wait_pause
bugs : spectre_v1 spectre_v2 spec_store_bypass swapgs eibrs_pbrsb
bogomips : 5836.80
clflush size : 64
cache_alignment : 64
address sizes : 39 bits physical, 48 bits virtual
power management:

[...]
```

Memory

```
MemTotal:       32551280 kB
MemFree:        24968688 kB
MemAvailable:   28890932 kB
Buffers:          117464 kB
Cached:          4253812 kB
SwapCached:            0 kB
Active:          2244180 kB
Inactive:        4590456 kB
Active(anon):       9964 kB
Inactive(anon):  2610540 kB
Active(file):    2234216 kB
Inactive(file):  1979916 kB
Unevictable:       58336 kB
Mlocked:              16 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:                12 kB
Writeback:             0 kB
AnonPages:       2521784 kB
Mapped:           703340 kB
Shmem:            169540 kB
KReclaimable:     154884 kB
Slab:             346852 kB
SReclaimable:     154884 kB
SUnreclaim:       191968 kB
KernelStack:       15456 kB
PageTables:        28420 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:    16275640 kB
Committed_AS:    8098092 kB
VmallocTotal:   34359738367 kB
VmallocUsed:      118664 kB
```

```
VmallocChunk:          0 kB
Percpu:            27360 kB
HardwareCorrupted:     0 kB
AnonHugePages:         0 kB
ShmemHugePages:        0 kB
ShmemPmdMapped:        0 kB
FileHugePages:         0 kB
FilePmdMapped:         0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
Hugetlb:               0 kB
DirectMap4k:     1111116 kB
DirectMap2M:     9052160 kB
DirectMap1G:    24117248 kB
```

GPU

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.60.13    Driver Version: 525.60.13    CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ...  On   | 00000000:01:00.0 Off |                  N/A |
| N/A   60C    P0    60W /  N/A |   1699MiB /  6144MiB |    100%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

Device 0: "NVIDIA GeForce RTX 3060 Laptop GPU"
  CUDA Driver Version / Runtime Version          12.0 / 12.0
  CUDA Capability Major/Minor version number:    8.6
  Total amount of global memory:                 5938 MBytes (6226378752 bytes)
  (30) Multiprocessors, (128) CUDA Cores/MP:     3840 CUDA Cores
  GPU Max Clock rate:                            1282 MHz (1.28 GHz)
  Memory Clock rate:                             6001 Mhz
  Memory Bus Width:                              192-bit
  L2 Cache Size:                                 3145728 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  1536
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
```

```
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 2 copy engine(s)
  Run time limit on kernels:                     Yes
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Disabled
  Device supports Unified Addressing (UVA):      Yes
  Device supports Compute Preemption:            Yes
  Supports Cooperative Kernel Launch:            Yes
  Supports MultiDevice Co-op Kernel Launch:      Yes
  Device PCI Domain ID / Bus ID / location ID:   0 / 1 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 12.0, CUDA Runtime Version = 12.0,
NumDevs = 1, Device0 = NVIDIA GeForce RTX 3060 Laptop GPU
Result = PASS
```

1.5.1.2 Machine B

CPU

This CPU has 16 processors, but in order not to be redundant we only present the specifications relative to a single core.

```
processor : 0
vendor_id : AuthenticAMD
cpu family : 23
model : 96
model name : AMD Ryzen 7 PRO 4750U with Radeon Graphics
stepping : 1
microcode : 0x8600106
cpu MHz : 1400.000
cache size : 512 KB
physical id : 0
siblings : 16
core id : 0
cpu cores : 8
apicid : 0
initial apicid : 0
fpu : yes
fpu_exception : yes
cpuid level : 16
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ht
syscall nx mmxext fxsr_opt pdpe1gb rdtscp lm constant_tsc rep_good nopl nonstop_tsc cpuid extd_apicid aperfmperf rapl
pni pclmulqdq monitor ssse3 fma cx16 sse4_1 sse4_2 movbe popcnt aes xsave avx f16c rdrand lahf_lm cmp_legacy svm
extapic cr8_legacy abm sse4a misalignsse 3dnowprefetch osvw ibs skinit wdt tce topoext perfctr_core perfctr_nb bpext
perfctr_llc mwaitx cpb cat_l3 cdp_l3 hw_pstate ssbd mba ibrs ibpb stibp vmmcall fsgsbase bmi1 avx2 smep bmi2 cqm
```

```
rdt_a rdseed adx smap clflushopt clwb sha_ni xsaveopt xsavec xgetbv1 xsaves cqm_llc cqm_occup_llc cqm_mbm_total
cqm_mbm_local clzero irperf xsaveerptr rdpru wbnoinvd cppc arat npt lbrv svm_lock nrip_save tsc_scale vmcb_clean
flushbyasid decodeassists pausefilter pfthreshold avic v_vmsave_vmload vgif v_spec_ctrl umip rdpid overflow_recov
succor smca
bugs : sysret_ss_attrs spectre_v1 spectre_v2 spec_store_bypass retbleed
bogomips : 3393.86
TLB size : 3072 4K pages
clflush size : 64
cache_alignment : 64
address sizes : 48 bits physical, 48 bits virtual
power management: ts ttp tm hwpstate cpb eff_freq_ro [13] [14]

[...]
```

Memory

```
MemTotal:       15689856 kB
MemFree:        11708020 kB
MemAvailable:   13510496 kB
Buffers:          174636 kB
Cached:          1808632 kB
SwapCached:            0 kB
Active:           804460 kB
Inactive:        2491180 kB
Active(anon):       4032 kB
Inactive(anon):  1401556 kB
Active(file):     800428 kB
Inactive(file):  1089624 kB
Unevictable:          32 kB
Mlocked:              32 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:               296 kB
Writeback:             0 kB
AnonPages:       1312624 kB
Mapped:           564092 kB
Shmem:            104404 kB
KReclaimable:     246816 kB
Slab:             387140 kB
SReclaimable:     246816 kB
SUnreclaim:       140324 kB
KernelStack:       16208 kB
PageTables:        28588 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:     7844928 kB
Committed_AS:    7658604 kB
VmallocTotal:   34359738367 kB
VmallocUsed:       53240 kB
VmallocChunk:          0 kB
```

```
Percpu:            16704 kB
HardwareCorrupted:     0 kB
AnonHugePages:         0 kB
ShmemHugePages:        0 kB
ShmemPmdMapped:        0 kB
FileHugePages:         0 kB
FilePmdMapped:         0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
Hugetlb:               0 kB
DirectMap4k:      403148 kB
DirectMap2M:     8331264 kB
DirectMap1G:     8388608 kB
```

1.5.1.3 Machine C

CPU

This CPU has 8 processors, but in order not to be redundant we only present the specifications relative to a single core.

```
processor : 0
vendor_id : GenuineIntel
cpu family : 6
model : 60
model name : Intel(R) Core(TM) i7-4770K CPU @ 3.50GHz
stepping : 3
microcode : 0x28
cpu MHz : 3741.591
cache size : 8192 KB
physical id : 0
siblings : 8
core id : 0
cpu cores : 4
apicid : 0
initial apicid : 0
fpu : yes
fpu_exception : yes
cpuid level : 13
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc
cpuid aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2
movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm cpuid_fault invpcid_single pti ssbd ibrs ibpb
stibp tpr_shadow vnmi flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid xsaveopt
dtherm ida arat pln pts md_clear flush_l1d
vmx flags : vnmi preemption_timer invvpid ept_x_only ept_ad ept_1gb flexpriority tsc_offset vtpr mtf vapic ept
vpid unrestricted_guest ple
bugs : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds swapgs itlb_multihit srbds mmio_unknown
```

```
bogomips : 7000.09
clflush size : 64
cache_alignment : 64
address sizes : 39 bits physical, 48 bits virtual
power management:

[...]
```

Memory

```
MemTotal:       16327224 kB
MemFree:         5239300 kB
MemAvailable:   10413200 kB
Buffers:          161024 kB
Cached:          5267036 kB
SwapCached:            0 kB
Active:          2004012 kB
Inactive:        8325384 kB
Active(anon):       3704 kB
Inactive(anon):  5048344 kB
Active(file):    2000308 kB
Inactive(file):  3277040 kB
Unevictable:          64 kB
Mlocked:              64 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:               336 kB
Writeback:             0 kB
AnonPages:       4901248 kB
Mapped:          1209004 kB
Shmem:            180484 kB
KReclaimable:     231936 kB
Slab:             411900 kB
SReclaimable:     231936 kB
SUnreclaim:       179964 kB
KernelStack:       18784 kB
PageTables:        64396 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:     8163612 kB
Committed_AS:   16559804 kB
VmallocTotal:   34359738367 kB
VmallocUsed:      107468 kB
VmallocChunk:          0 kB
Percpu:             8064 kB
HardwareCorrupted:     0 kB
AnonHugePages:      2048 kB
ShmemHugePages:        0 kB
ShmemPmdMapped:        0 kB
FileHugePages:         0 kB
```

```
FilePmdMapped:         0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
Hugetlb:               0 kB
DirectMap4k:      694616 kB
DirectMap2M:    12881920 kB
DirectMap1G:     4194304 kB
```

1.5.1.4 Machine P4

CPU

This CPU has 4 processors, but in order not to be redundant we only present the specifications relative to a single core.

```
processor : 0
model name : ARMv7 Processor rev 3 (v7l)
BogoMIPS : 108.00
Features : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt vfpd32 lpae evtstrm crc32
CPU implementer : 0x41
CPU architecture: 7
CPU variant : 0x0
CPU part : 0xd08
CPU revision : 3

[...]
```

Memory

```
MemTotal:         8118400 kB
MemFree:         47392052 kB
MemAvailable:    58732104 kB
Buffers:           784 kB
Cached:           120920 kB
SwapCached:            0 kB
Active:           234940 kB
Inactive:         123088 kB
Active(anon):     216512 kB
Inactive(anon):    524 kB
Active(file):      8428 kB
Inactive(file):   122564 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:                 0 kB
Writeback:             0 kB
AnonPages:        216260 kB
Mapped:            18480 kB
Shmem:              6248 kB
```

```
Slab:              16336 kB
SReclaimable:       8076 kB
SUnreclaim:         8260 kB
KernelStack:        1224 kB
PageTables:         1496 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:     4058720 kB
Committed_AS:    3255508 kB
VmallocTotal:   1114112 kB
VmallocUsed:       17304 kB
VmallocChunk:    1096508 kB
HardwareCorrupted:     0 kB
AnonHugePages:         0 kB
ShmemHugePages:        0 kB
ShmemPmdMapped:        0 kB
CmaTotal:              0 kB
CmaFree:               0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
DirectMap4k:       16296 kB
DirectMap2M:      907264 kB
```

1.5.1.5 Machine P3

CPU

This CPU has 4 processors, but in order not to be redundant we only present the specifications relative to a single core.

```
processor : 0
model name : ARMv7 Processor rev 3 (v7l)
BogoMIPS : 108.00
Features : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt vfpd32 lpae evtstrm crc32
CPU implementer : 0x41
CPU architecture: 7
CPU variant : 0x0
CPU part : 0xd08
CPU revision : 3

[...]
```

Memory

```
MemTotal:         948336 kB
MemFree:          709824 kB
MemAvailable:     793468 kB
Buffers:            8156 kB
Cached:           120540 kB
```

```
SwapCached:            0 kB
Active:           103924 kB
Inactive:         103544 kB
Active(anon):      76588 kB
Inactive(anon):    52472 kB
Active(file):      27336 kB
Inactive(file):    51072 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:                 4 kB
Writeback:             0 kB
AnonPages:         76444 kB
Mapped:            19248 kB
Shmem:              6248 kB
Slab:              16360 kB
SReclaimable:       8092 kB
SUnreclaim:         8268 kB
KernelStack:        1224 kB
PageTables:         1496 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:      472432 kB
Committed_AS:     297516 kB
VmallocTotal:   1114112 kB
VmallocUsed:       17296 kB
VmallocChunk:    1096508 kB
HardwareCorrupted:     0 kB
AnonHugePages:         0 kB
ShmemHugePages:        0 kB
ShmemPmdMapped:        0 kB
CmaTotal:              0 kB
CmaFree:               0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
DirectMap4k:       16296 kB
DirectMap2M:      907264 kB
```

1.5.2 Software

Every machine used runs **Ubuntu 22.04 LTS** as its operating system.

The used compilers were:

- **C compiler**: gcc 11.3.0;

- **MPI framework**: MPICH 4.0;
- **CUDA compiler**: Cuda compilation tools, release 12.0, V12.0.76, Build cuda_12.0.r12.0/compiler.31968024_0.

In addition, Python 3.10.6 was used for creating the tables and speedup graphs based on the measurements.

CHAPTER 2

MPI IMPLEMENTATION

In this chapter we will discuss about our distributed implementation of Tarjan's algorithm made with MPI (`mpi`) and present the experimental results that this algorithm achieved.

2.1 Algorithm idea

2.1.1 Description of the algorithm

Tarjan's parallel algorithm on MPI uses a Master/Slave communication model; i.e. a model in which one process, the Master, controls the communication of several processes, the Slaves. The Master node is identified by rank 0 while all Slave nodes are identified by a rank other than 0.
The realised algorithm works on the abstract data type graph to represent the mathematical concept of an oriented graph. The implementation of the graph is realised via an adjacency map. For reasons of effeciency in search operations, the adjacency map was implemented with a double HashMap. The first HashMap has as key the node ID and as value a set of node IDs reachable via outgoing edges; the second HashMap always has as key the node ID and as value a set of node IDs reachable via incoming edges.

The first step of the MPI algorythm involves reading from a binary file the information about the graph whose SCCs we wish to find. The contents of the file must then be processed in order to derive the internal data structure on which to perform operations on the graph. In addition, a further data structure must be generated and initialised to keep track of all SCCs that are found during the execution of the programme.

The MPI algorithm requires that the communicator processes have different functionalities depending on whether the node is a Master or a Slave.

- If the node is the Master: the process takes care of splitting the graph into smaller chunks of a fixed size, called CHUNK SIZE, and sending the chunk of the graph to a Slave node, which will take care of searching for SCCs within the subgraph. Once the processing of the graph has been completed and

the SCCs of each chunk have been found, further processing must be carried out in order to find new SCCs that were not found in the previous processing of the graph. So, we proceed by carrying out new iterations of the previous operations doubling the size of the CHUNK SIZE at each iteration, all this until the CHUNK SIZE is at least equal to the number of nodes in the graph.

The Master node, once it has sent a chunk of the graph to the Slave node, waits until a Slave node has finished locating the SCCs in its assigned chunk. The Slave node sends each SCC it finds to the Master node, the latter proceeds to merge the found SCC into a supernode and stores the found SCC in a special data structure. After that, if the Slave node has finished execution on the chunk of the graph, the Master node sends a new chunk of the graph to the Slave node that has just finished. This process lasts as long as there are chunks available for processing.

- If the node is a Slave: the process receives a chunk of the graph from the Master node after which it applies the Tarjan algorithm on the received subgraph. As soon as the process finds an SCC it proceeds to send it to the Master node. When the process has processed the entire subgraph, it communicates this information to the master node while waiting for a new chunk of the graph to be processed.

  Slave processes terminate only when the Master process sends it a special termination message.

2.1.1.1 Communication

The algorithm expects that the exchanged messages are dynamic arrays. Whenever a process wants to send a message, it must first communicate the size of the array, after which it can proceed to send the message. The MPI algorithm requires two special messages:

1. The Slave node communicates to the Master node that it has finished processing the subgraph by sending an array (which normally represents the found SCC) of size zero. In this way, the Master node interprets this information as the end of the processing of the chunk assigned to the Slave process and thus the latter is ready to do more work.

2. The Master node, when it has to finish because all the operations provided by the MPI algorithm have been performed, sends a message to all Slave nodes of a dynamic array (which normally represents a graph chunk) of size zero. The Slave node interprets this information as a termination message for all tasks and finishes its execution.

2.2 Experimental results

In this subsection are presented the results of the parallel version of Tarjan's algorithm with MPI. Each graph is tested with all the compiler optimization (O0, O1, O2, O3) and with 2, 4 and 8 processors MPI. In the next pages for each graph, set the optimization, there are a table and a graph of the speedup. In the table there are:

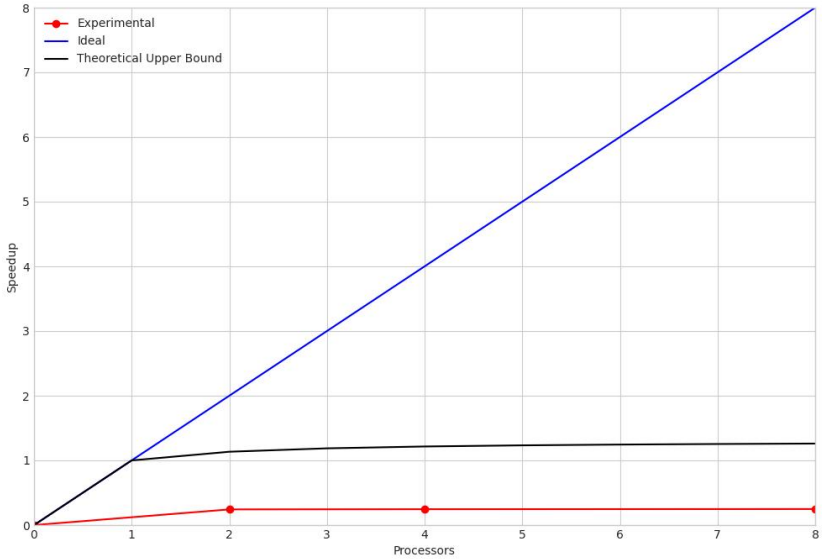
- **Version**: is the version of the program. It can be "Serial" if it's the sequential version of the program, or it can be "Parallel" if it's the parallel version with MPI.
- **Threads**: is the number of processors. For the "Serial" version, the value of this field is 1, because it's the sequential version of the program executed on only one processor of the CPU.
- **vertices**: is the number of vertices of the graph.
- **init**: is the time spent to initialize the program, in particular it's the time spent to create and initialize the data structures, and to read the graph from a file.
- **tarjan**: is the time spent to execute Tarjan's algorithm. In the "Parallel" version is the time it takes to run Tarjan's algorithm on each MPI slave node, along with the time it takes to partition the graph into chunks, exchange messages between processes, store found SCCs, and create supernodes.
- **split**: is the time spent to split the graph in the "Parallel" version. This phase is not provided by the sequential algorithm, in fact this field for this version is reported with the value "no data".
- **merge**: is the time spent to merge the SCCs found. This phase involves merging the nodes of an SCC found by the Tarjan's algorithm into a supernode, and storing the SCC found in a dedicated data structure. This phase is not provided by the sequential algorithm, in fact this field for this version is reported with the value "no data".
- **user**: is the total number of CPU-seconds that the process used directly (in user mode), in seconds.
- **system**: is the total number of CPU-seconds used by the system on behalf of the process (in kernel mode), in seconds.
- **pCPU**: is the percentage of the CPU that this process got. This is just user + system times divided by the total running time.
- **elapsed**: is the elapsed real (wall clock) time used by the process, in seconds.
- **Speedup**: is the speedup of the program.
- **Efficiency**: is the efficiency of the program.

All values except those in the `Version`, `Threads`, `Speedup`, and `Efficiency` columns are means. These means were calculated from the files of the measurements taken. `Speedup`, and `Efficiency` are calculated with the formulas in Cap.1 using the results of these means.

2.2.1 Fully connected graph

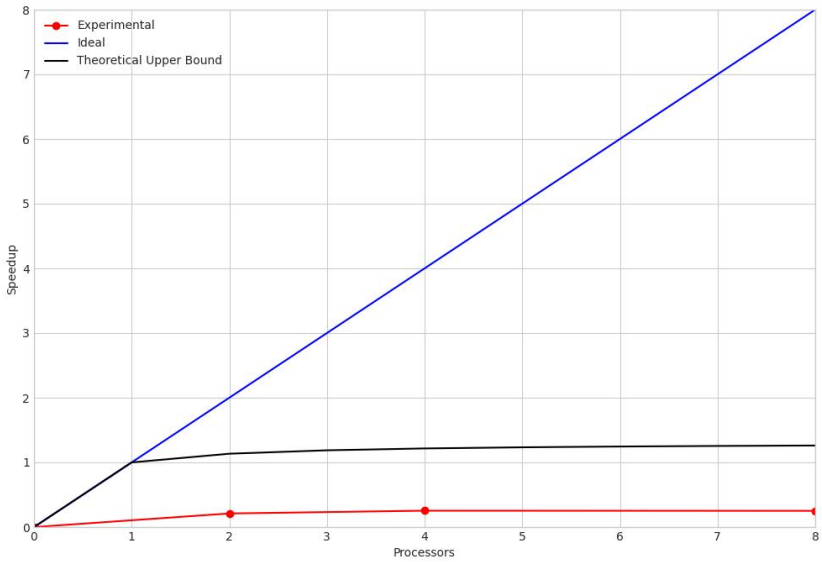
2.2.1.1 Optimization 0

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	12500.000	9.346	2.901	no data	no data	11.489	0.794	99.050	12.291	1.000	1.000
Parallel	2	12500.000	14.190	34.241	0.765	21.401	95.211	1.845	189.920	50.749	0.242	0.121
Parallel	4	12500.000	14.532	33.037	1.037	20.906	73.690	21.679	189.240	50.094	0.245	0.061
Parallel	8	12500.000	14.273	32.407	1.064	19.931	103.310	37.523	282.360	49.542	0.248	0.031



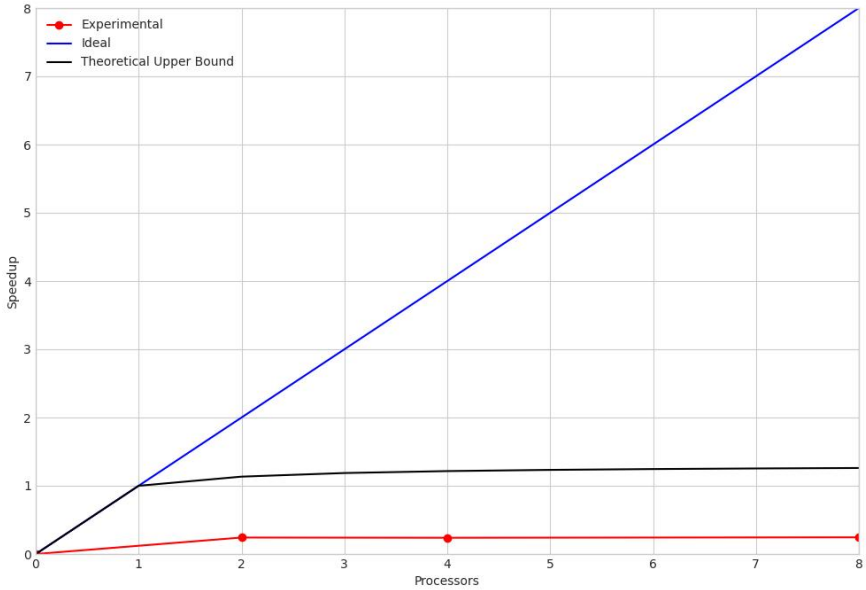
2.2.1.2 Optimization 1

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	12500.000	9.340	2.908	no data	no data	11.494	0.790	99.010	12.293	1.000	1.000
Parallel	2	12500.000	17.123	38.979	0.828	26.820	110.540	1.856	191.360	58.378	0.211	0.105
Parallel	4	12500.000	13.798	32.175	0.993	19.384	70.820	21.350	188.800	48.505	0.253	0.063
Parallel	8	12500.000	13.922	32.231	1.033	20.337	101.666	37.604	282.760	48.938	0.251	0.031



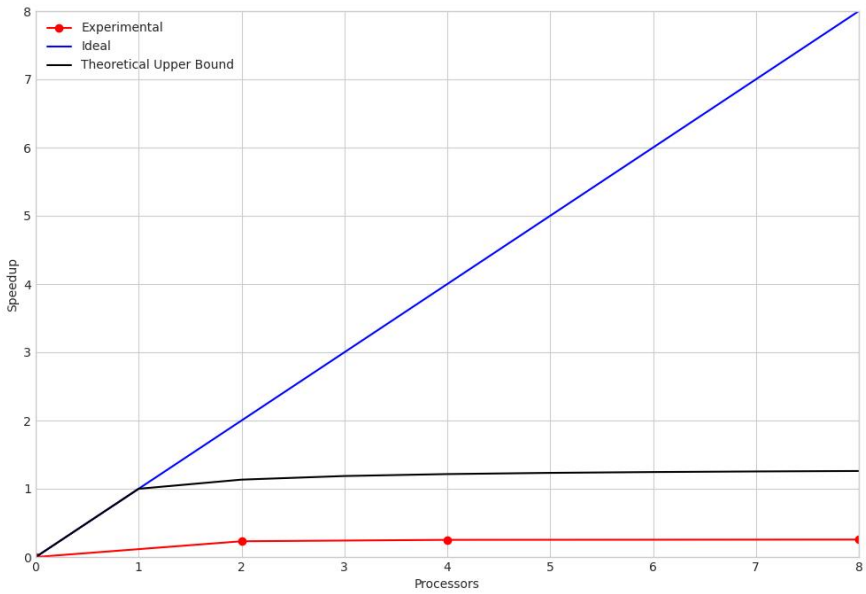
2.2.1.3 Optimization 2

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	12500.000	9.380	2.912	no data	no data	11.539	0.790	99.060	12.335	1.000	1.000
Parallel	2	12500.000	13.779	35.013	0.740	20.397	95.915	1.864	190.000	51.124	0.241	0.121
Parallel	4	12500.000	14.963	34.118	1.021	21.231	75.825	22.695	188.920	51.808	0.238	0.060
Parallel	8	12500.000	14.121	32.418	1.049	20.585	102.422	37.886	279.520	50.475	0.244	0.031



2.2.1.4 Optimization 3

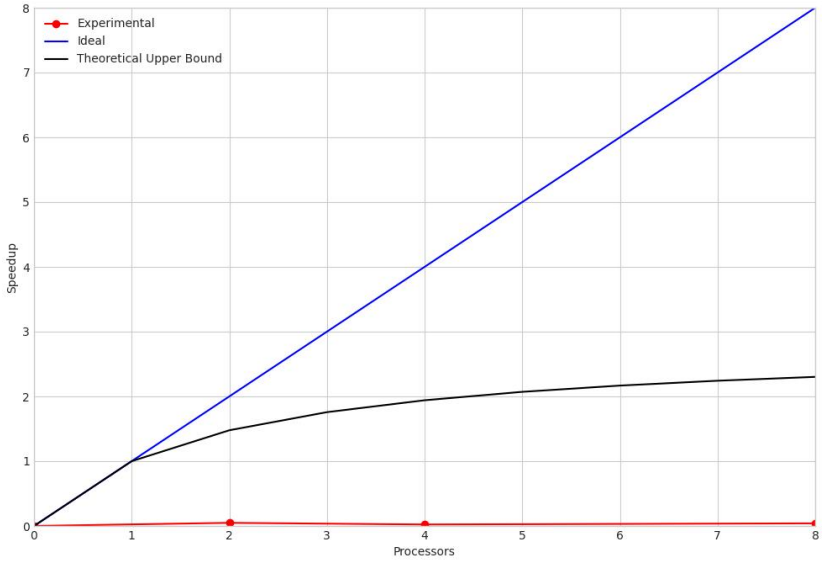
Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	12500.000	9.354	2.906	no data	no data	11.509	0.788	99.060	12.304	1.000	1.000
Parallel	2	12500.000	15.042	36.175	0.782	23.025	100.742	1.889	190.960	53.363	0.231	0.115
Parallel	4	12500.000	13.714	32.562	1.004	20.141	71.932	21.424	189.840	48.914	0.252	0.063
Parallel	8	12500.000	13.613	31.749	1.033	19.748	99.907	36.993	282.400	48.134	0.256	0.032



2.2.2 Fully disconnected graph

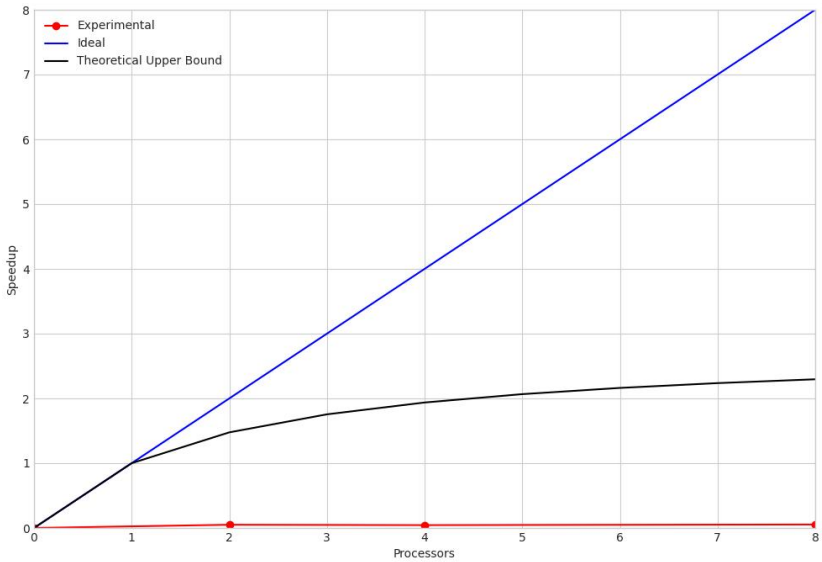
2.2.2.1 Optimization 0

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	1000000.000	0.075	0.134	no data	no data	0.171	0.031	99.430	0.207	1.000	1.000
Parallel	2	1000000.000	0.098	3.402	0.146	0.000	6.746	0.338	169.120	4.274	0.048	0.024
Parallel	4	1000000.000	0.104	3.819	0.148	0.000	10.143	4.806	148.760	9.098	0.023	0.006
Parallel	8	1000000.000	0.092	2.393	0.135	0.000	5.534	2.119	196.600	5.102	0.041	0.005



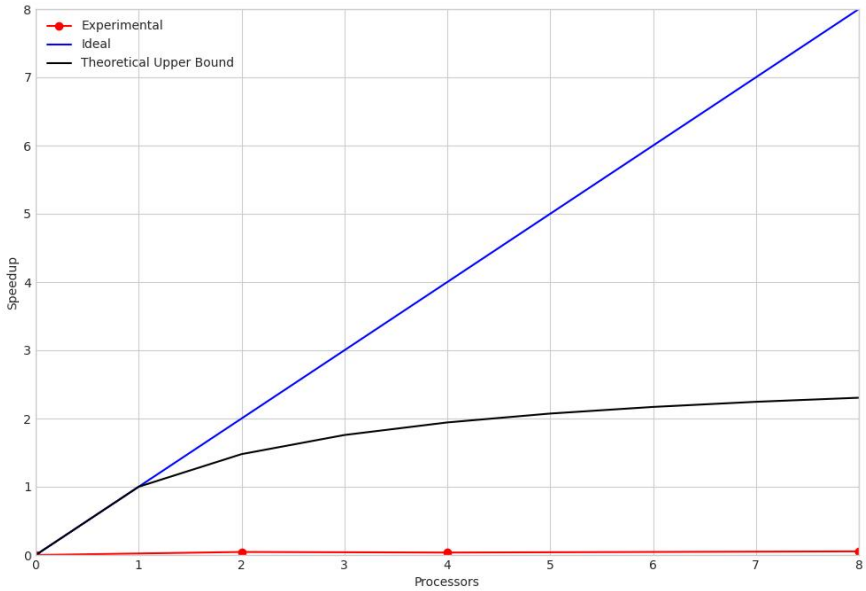
2.2.2.2 Optimization 1

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	1000000.000	0.075	0.134	no data	no data	0.173	0.028	99.370	0.207	1.000	1.000
Parallel	2	1000000.000	0.103	3.195	0.149	0.000	6.351	0.330	166.320	4.164	0.050	0.025
Parallel	4	1000000.000	0.101	2.993	0.146	0.000	4.522	1.778	139.720	4.700	0.044	0.011
Parallel	8	1000000.000	0.100	2.387	0.142	0.000	5.509	2.136	202.520	3.886	0.053	0.007



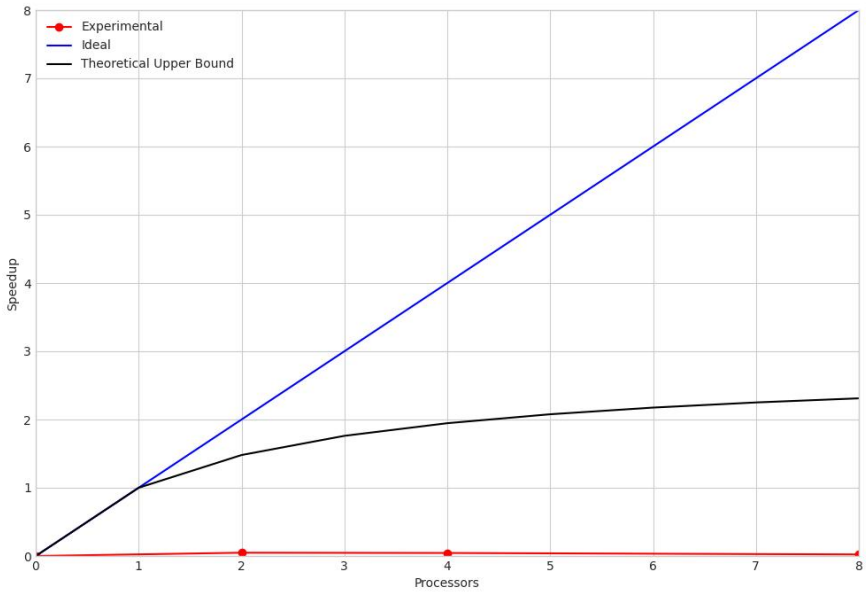
2.2.2.3 Optimization 2

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	1000000.000	0.075	0.133	no data	no data	0.172	0.028	99.270	0.206	1.000	1.000
Parallel	2	1000000.000	0.095	3.797	0.143	0.000	7.518	0.356	174.800	4.609	0.045	0.022
Parallel	4	1000000.000	0.099	3.713	0.145	0.000	5.870	2.523	148.360	5.696	0.036	0.009
Parallel	8	1000000.000	0.093	2.346	0.136	0.000	5.478	2.045	201.040	3.848	0.053	0.007



2.2.2.4 Optimization 3

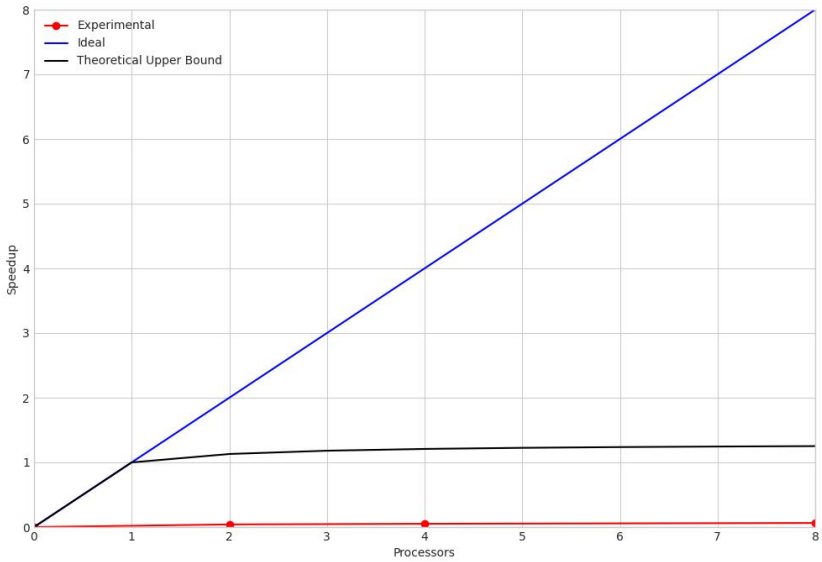
Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	1000000.000	0.075	0.133	no data	no data	0.172	0.029	99.250	0.205	1.000	1.000
Parallel	2	1000000.000	0.101	3.443	0.149	0.000	6.832	0.345	172.440	4.256	0.048	0.024
Parallel	4	1000000.000	0.096	2.930	0.142	0.000	4.441	1.727	138.400	4.614	0.044	0.011
Parallel	8	1000000.000	0.103	2.314	0.147	0.000	5.420	2.050	197.240	8.919	0.023	0.003



2.2.3 Random-150000 graph

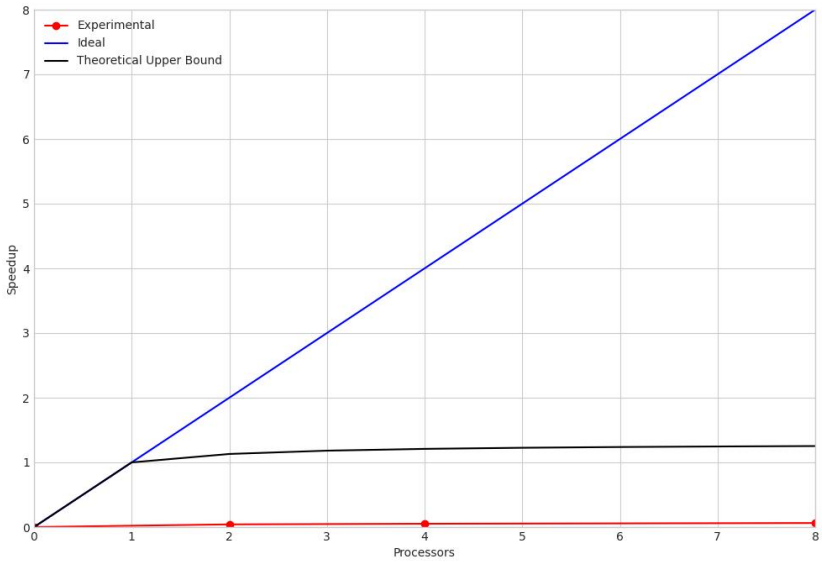
2.2.3.1 Optimization 0

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	150000.000	0.122	0.035	no data	no data	0.144	0.006	99.400	0.154	1.000	1.000
Parallel	2	150000.000	0.142	2.753	0.065	0.000	5.736	0.122	158.840	3.773	0.041	0.020
Parallel	4	150000.000	0.156	1.835	0.063	0.000	3.156	0.935	141.280	3.006	0.051	0.013
Parallel	8	150000.000	0.158	1.161	0.060	0.000	3.246	0.997	183.000	2.437	0.063	0.008



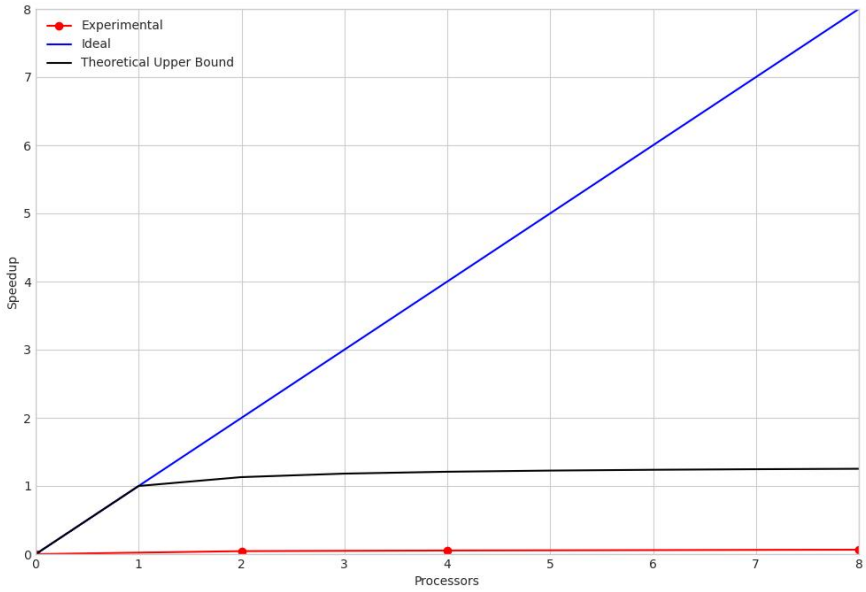
2.2.3.2 Optimization 1

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	150000.000	0.121	0.035	no data	no data	0.142	0.005	99.420	0.152	1.000	1.000
Parallel	2	150000.000	0.156	2.794	0.067	0.000	5.846	0.123	169.440	3.634	0.042	0.021
Parallel	4	150000.000	0.159	1.807	0.064	0.000	3.094	0.947	142.560	2.944	0.052	0.013
Parallel	8	150000.000	0.148	1.205	0.059	0.000	3.333	1.014	188.400	2.426	0.063	0.008



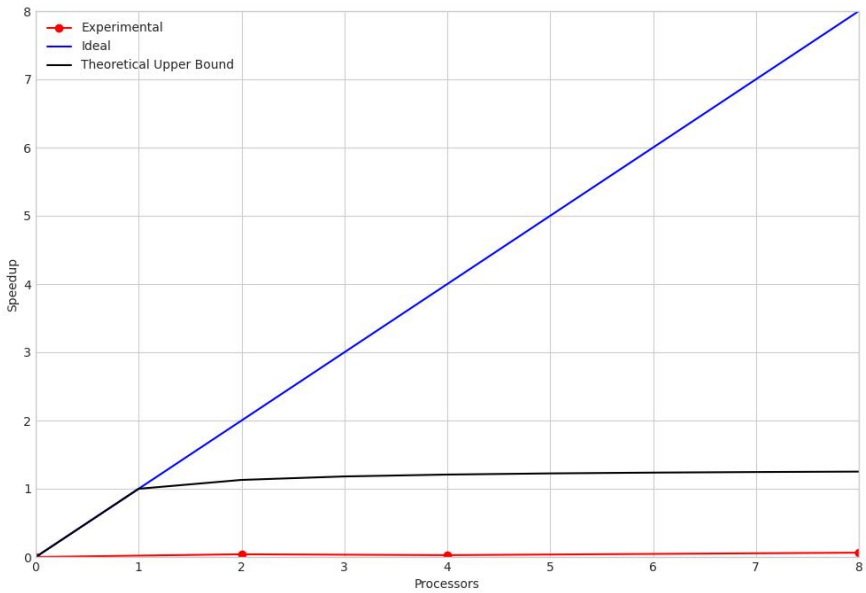
2.2.3.3 Optimization 2

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	150000.000	0.120	0.035	no data	no data	0.142	0.004	99.400	0.150	1.000	1.000
Parallel	2	150000.000	0.165	2.613	0.069	0.000	5.503	0.122	169.840	3.426	0.044	0.022
Parallel	4	150000.000	0.144	1.825	0.061	0.000	3.101	0.940	145.320	2.833	0.053	0.013
Parallel	8	150000.000	0.145	1.175	0.058	0.000	3.273	0.957	188.920	2.312	0.065	0.008



2.2.3.4 Optimization 3

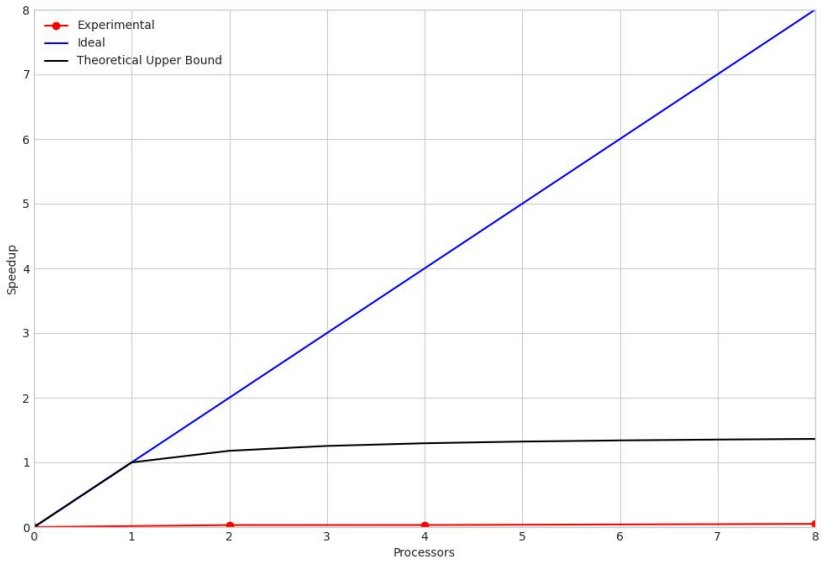
Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	150000.000	0.120	0.035	no data	no data	0.140	0.006	99.330	0.150	1.000	1.000
Parallel	2	150000.000	0.150	2.790	0.066	0.000	5.829	0.121	163.440	3.734	0.040	0.020
Parallel	4	150000.000	0.149	3.829	0.063	0.000	5.937	2.722	143.760	5.508	0.027	0.007
Parallel	8	150000.000	0.150	1.199	0.060	0.000	3.340	1.029	192.160	2.359	0.064	0.008



2.2.4 Random-250000 graph

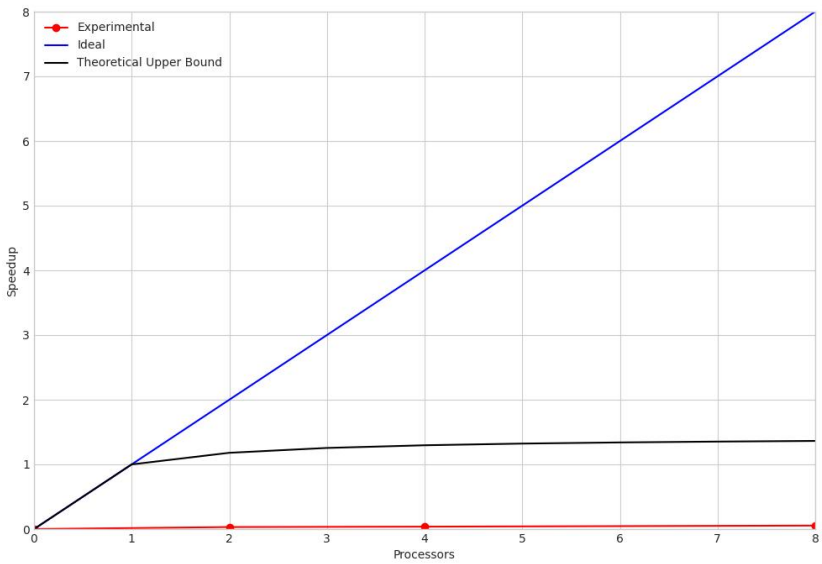
2.2.4.1 Optimization 0

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	250000.000	0.150	0.064	no data	no data	0.195	0.010	99.320	0.210	1.000	1.000
Parallel	2	250000.000	0.199	5.585	0.139	0.000	11.434	0.217	173.720	6.791	0.031	0.015
Parallel	4	250000.000	0.182	3.842	0.127	0.000	7.906	2.869	156.680	6.638	0.032	0.008
Parallel	8	250000.000	0.181	2.543	0.122	0.000	6.558	2.084	213.840	4.216	0.050	0.006



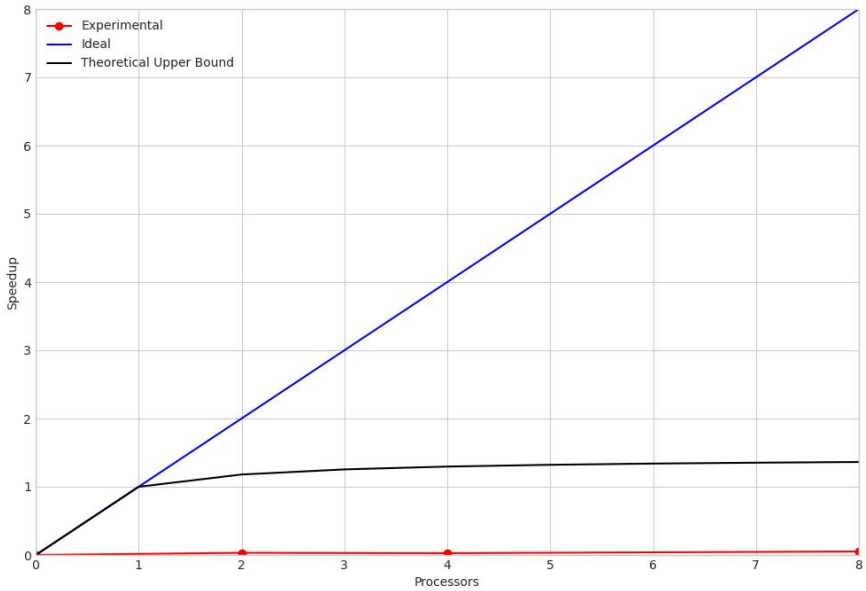
2.2.4.2 Optimization 1

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	250000.000	0.150	0.064	no data	no data	0.195	0.011	99.270	0.210	1.000	1.000
Parallel	2	250000.000	0.198	5.482	0.141	0.000	11.224	0.213	173.640	6.723	0.031	0.016
Parallel	4	250000.000	0.200	3.773	0.136	0.000	6.207	1.939	155.200	5.624	0.037	0.009
Parallel	8	250000.000	0.193	2.552	0.124	0.000	6.552	2.150	229.040	3.876	0.054	0.007



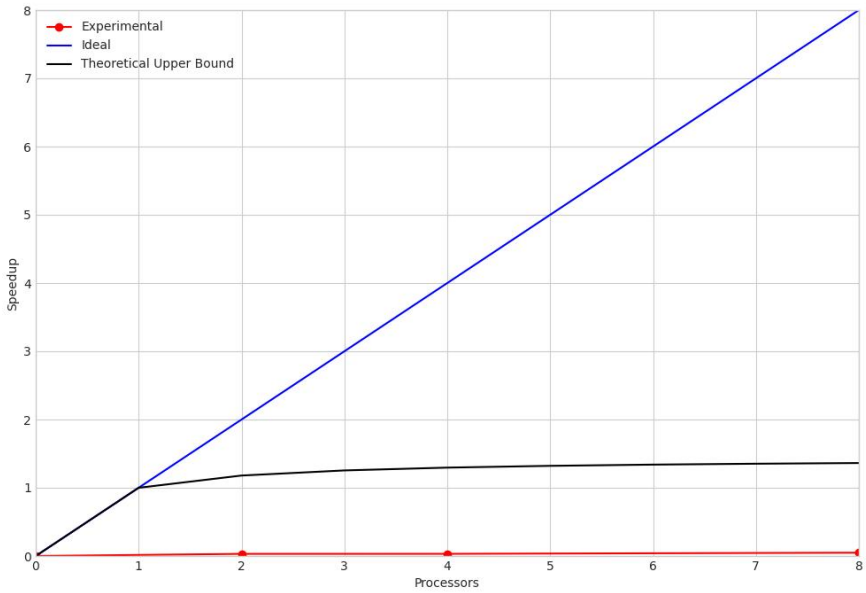
2.2.4.3 Optimization 2

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	250000.000	0.150	0.064	no data	no data	0.195	0.011	99.400	0.210	1.000	1.000
Parallel	2	250000.000	0.193	5.515	0.135	0.000	11.285	0.207	174.160	6.638	0.032	0.016
Parallel	4	250000.000	0.192	5.780	0.131	0.000	8.950	3.704	154.960	7.928	0.027	0.007
Parallel	8	250000.000	0.184	2.552	0.122	0.000	6.512	2.131	219.320	4.095	0.051	0.006



2.2.4.4 Optimization 3

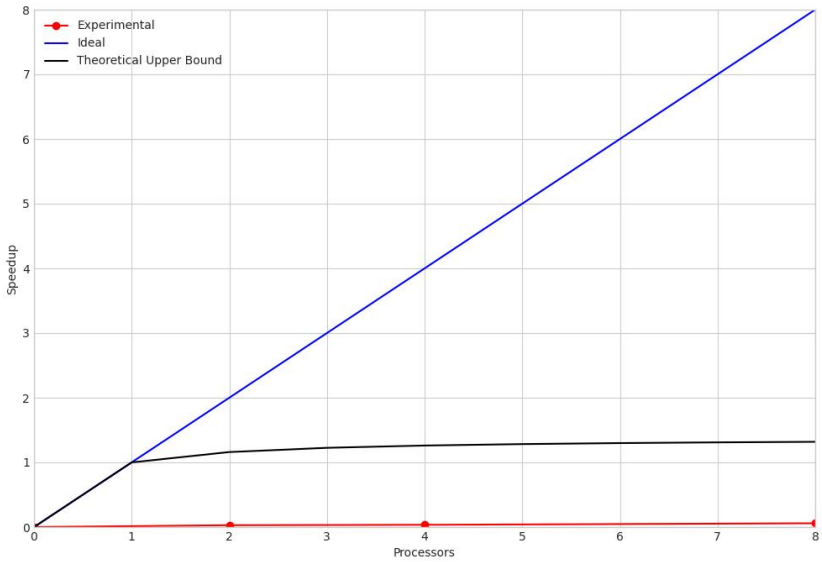
Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	250000.000	0.150	0.064	no data	no data	0.193	0.012	99.330	0.211	1.000	1.000
Parallel	2	250000.000	0.213	5.600	0.148	0.000	11.488	0.220	176.360	6.710	0.031	0.016
Parallel	4	250000.000	0.198	3.716	0.134	0.000	6.044	1.896	146.200	6.608	0.032	0.008
Parallel	8	250000.000	0.179	2.533	0.123	0.000	6.523	2.042	208.520	4.343	0.048	0.006



2.2.5 Random-500000 graph

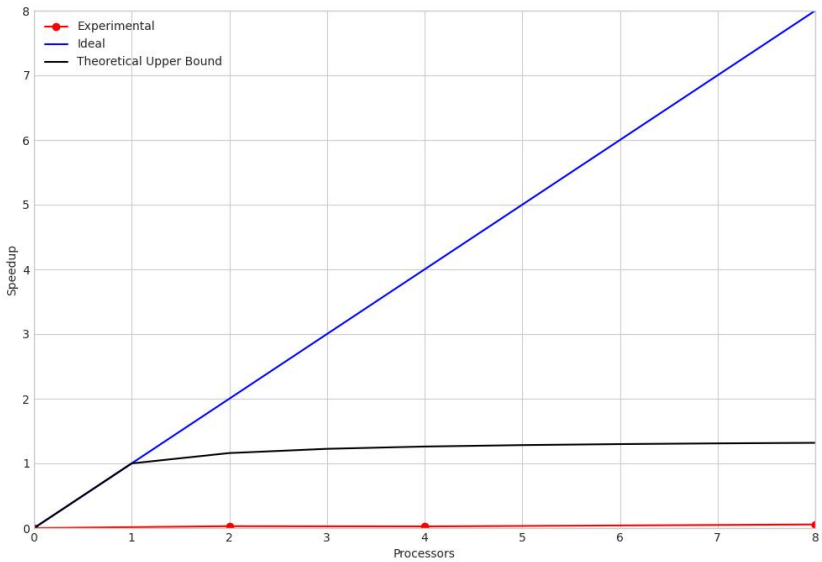
2.2.5.1 Optimization 0

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	500000.000	0.377	0.143	no data	no data	0.486	0.025	99.330	0.518	1.000	1.000
Parallel	2	500000.000	0.461	16.013	0.319	0.000	32.583	0.458	182.680	18.054	0.029	0.014
Parallel	4	500000.000	0.495	10.160	0.324	0.000	16.430	5.004	168.800	14.880	0.035	0.009
Parallel	8	500000.000	0.432	6.616	0.292	0.000	16.806	5.049	250.760	8.773	0.059	0.007



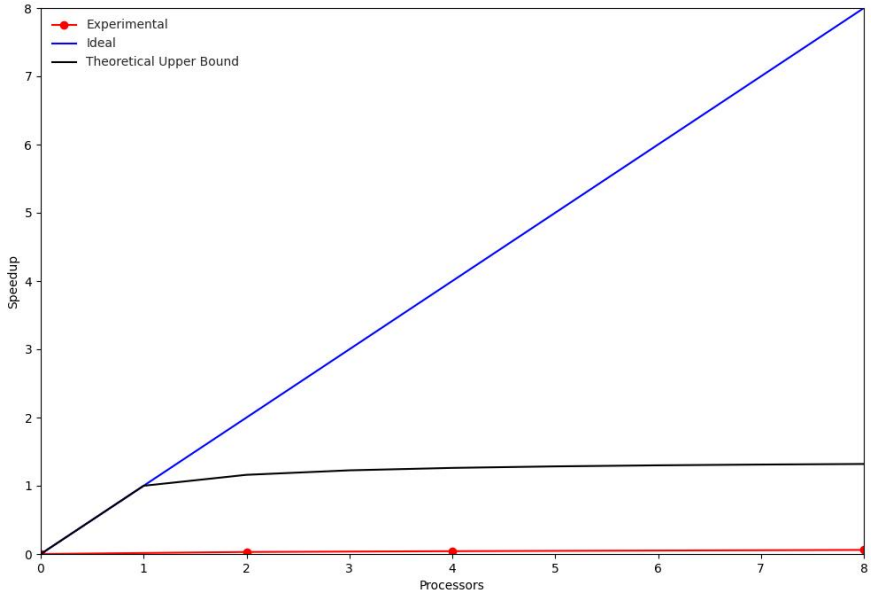
2.2.5.2 Optimization 1

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	500000.000	0.376	0.143	no data	no data	0.486	0.025	99.370	0.517	1.000	1.000
Parallel	2	500000.000	0.488	15.936	0.335	0.000	32.499	0.443	183.160	17.910	0.029	0.014
Parallel	4	500000.000	0.479	10.137	0.320	0.000	16.354	5.001	163.000	20.530	0.025	0.006
Parallel	8	500000.000	0.457	6.564	0.307	0.000	16.629	5.085	236.040	9.290	0.056	0.007



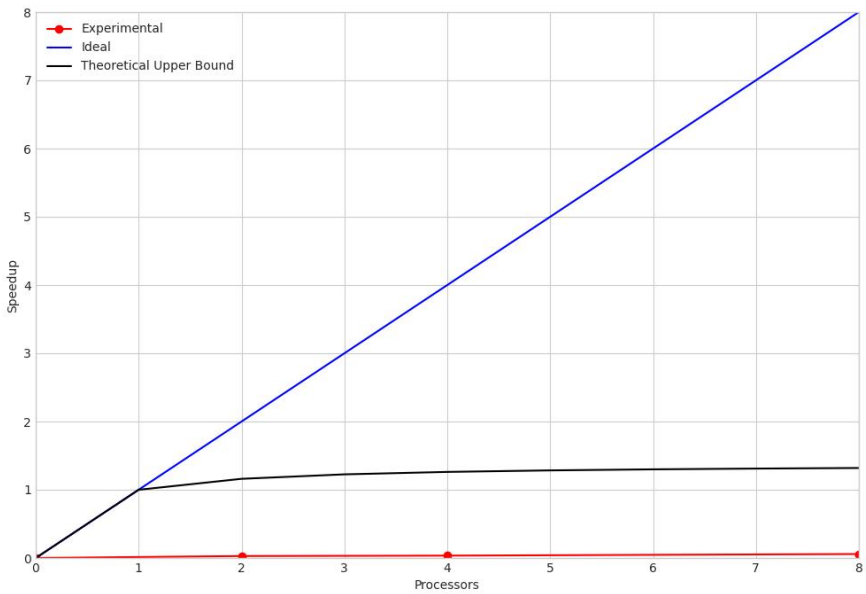
2.2.5.3 Optimization 2

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	500000.000	0.375	0.143	no data	no data	0.487	0.024	99.350	0.517	1.000	1.000
Parallel	2	500000.000	0.461	15.507	0.321	0.000	31.584	0.446	183.520	17.417	0.030	0.015
Parallel	4	500000.000	0.472	10.427	0.317	0.000	16.805	5.116	176.440	12.427	0.042	0.010
Parallel	8	500000.000	0.449	6.611	0.294	0.000	16.864	5.022	255.160	8.612	0.060	0.008



2.2.5.4 Optimization 3

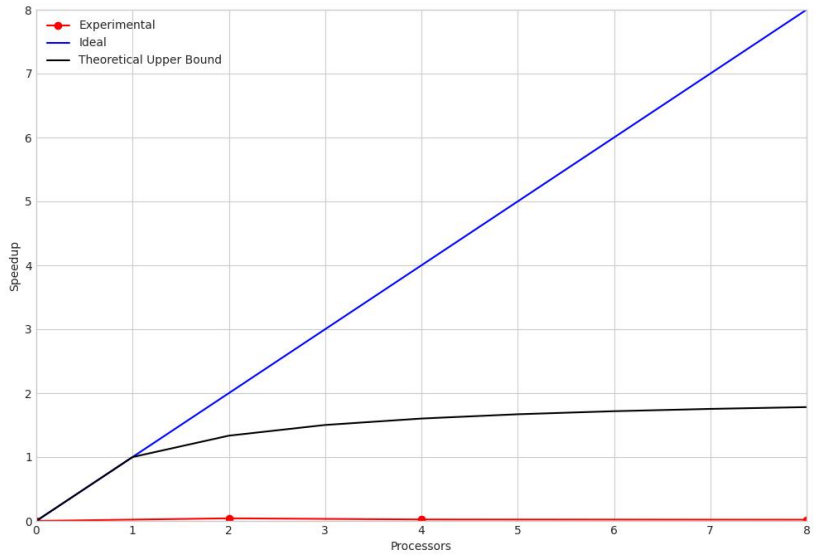
Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	500000.000	0.377	0.144	no data	no data	0.488	0.025	99.410	0.519	1.000	1.000
Parallel	2	500000.000	0.477	16.105	0.329	0.000	32.785	0.467	183.920	18.018	0.029	0.014
Parallel	4	500000.000	0.442	10.068	0.310	0.000	16.188	4.950	164.400	15.000	0.035	0.009
Parallel	8	500000.000	0.483	6.626	0.308	0.000	16.853	5.164	249.600	8.910	0.058	0.007



2.2.6 Tile-52000 graph

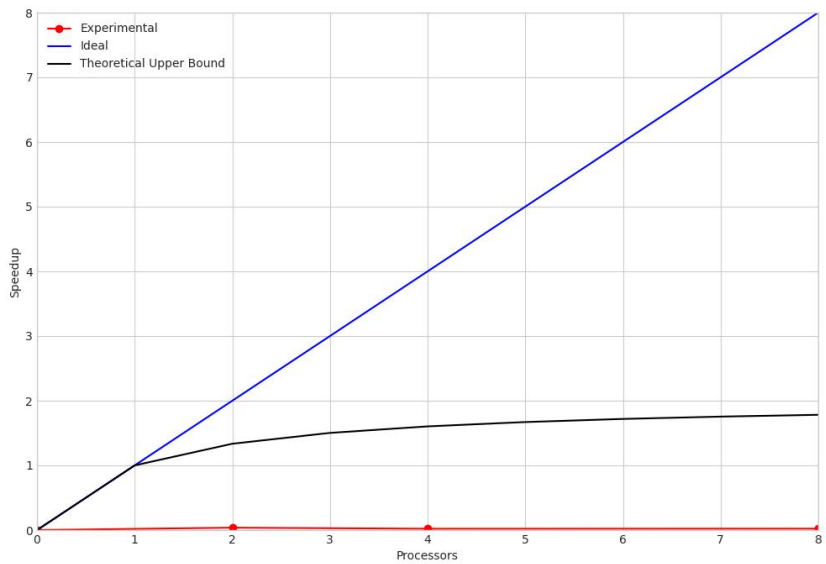
2.2.6.1 Optimization 0

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	51200.000	0.011	0.010	no data	no data	0.014	0.001	98.450	0.020	1.000	1.000
Parallel	2	51200.000	0.015	0.112	0.009	0.006	0.270	0.045	75.120	0.483	0.041	0.021
Parallel	4	51200.000	0.015	0.151	0.009	0.006	0.302	0.138	56.320	0.864	0.023	0.006
Parallel	8	51200.000	0.015	0.160	0.009	0.006	0.474	0.224	78.240	1.029	0.019	0.002



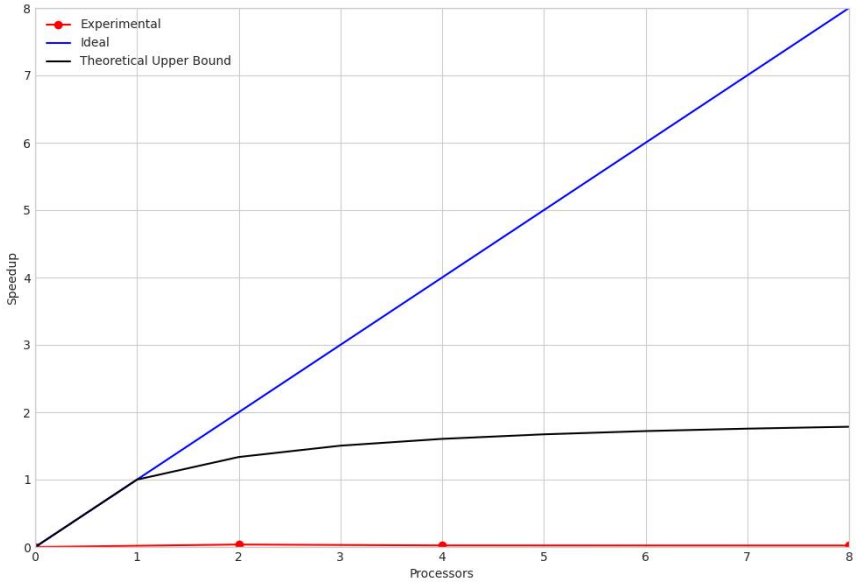
2.2.6.2 Optimization 1

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	51200.000	0.011	0.010	no data	no data	0.014	0.000	98.300	0.020	1.000	1.000
Parallel	2	51200.000	0.014	0.115	0.009	0.006	0.282	0.039	76.120	0.539	0.037	0.019
Parallel	4	51200.000	0.016	0.149	0.009	0.007	0.293	0.136	52.320	0.975	0.021	0.005
Parallel	8	51200.000	0.015	0.163	0.008	0.006	0.477	0.223	82.160	0.921	0.022	0.003



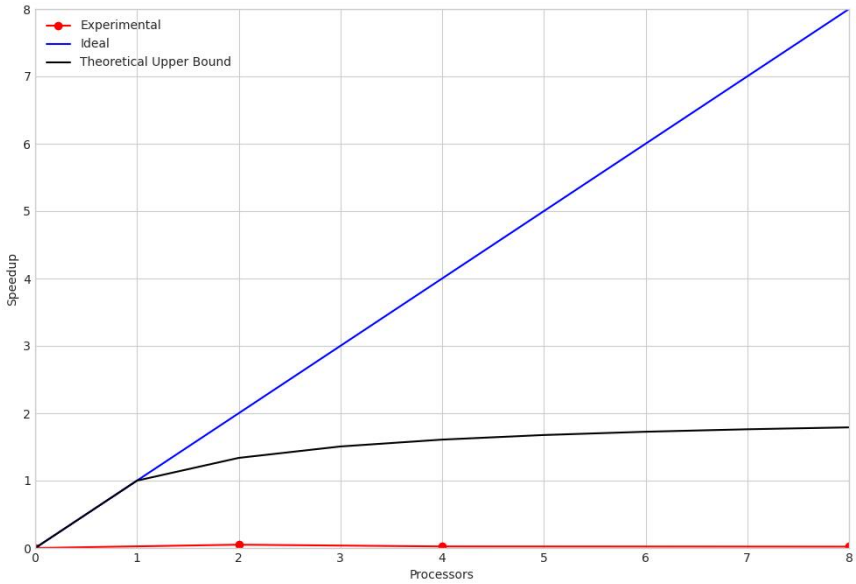
2.2.6.3 Optimization 2

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	51200.000	0.011	0.010	no data	no data	0.014	0.000	98.500	0.020	1.000	1.000
Parallel	2	51200.000	0.015	0.111	0.009	0.006	0.270	0.043	73.280	0.546	0.037	0.018
Parallel	4	51200.000	0.015	0.151	0.008	0.006	0.292	0.140	55.440	0.856	0.023	0.006
Parallel	8	51200.000	0.014	0.157	0.008	0.006	0.458	0.216	80.840	0.912	0.022	0.003



2.2.6.4 Optimization 3

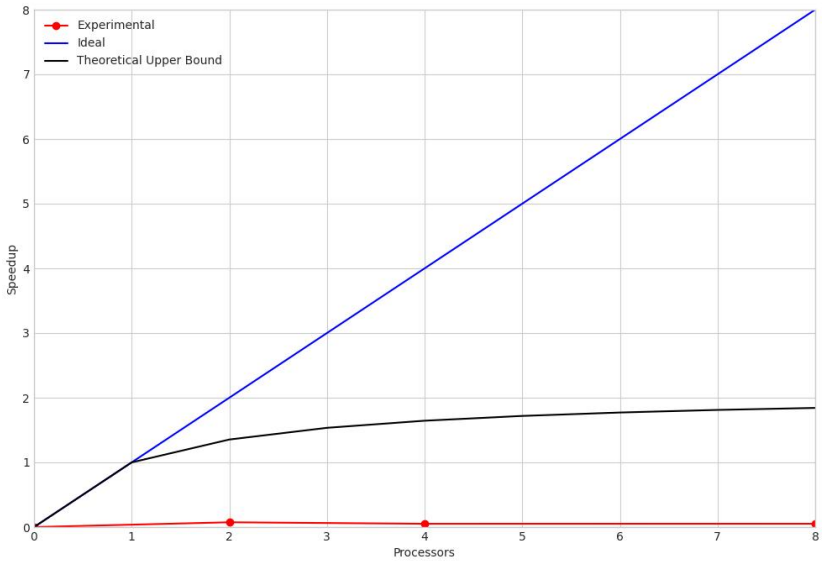
Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	51200.000	0.011	0.010	no data	no data	0.013	0.000	97.950	0.020	1.000	1.000
Parallel	2	51200.000	0.015	0.110	0.009	0.006	0.268	0.043	77.560	0.406	0.049	0.025
Parallel	4	51200.000	0.015	0.155	0.008	0.006	0.293	0.148	56.160	0.864	0.023	0.006
Parallel	8	51200.000	0.014	0.160	0.008	0.007	0.470	0.220	80.200	0.995	0.020	0.003



2.2.7 Tile-205000 graph

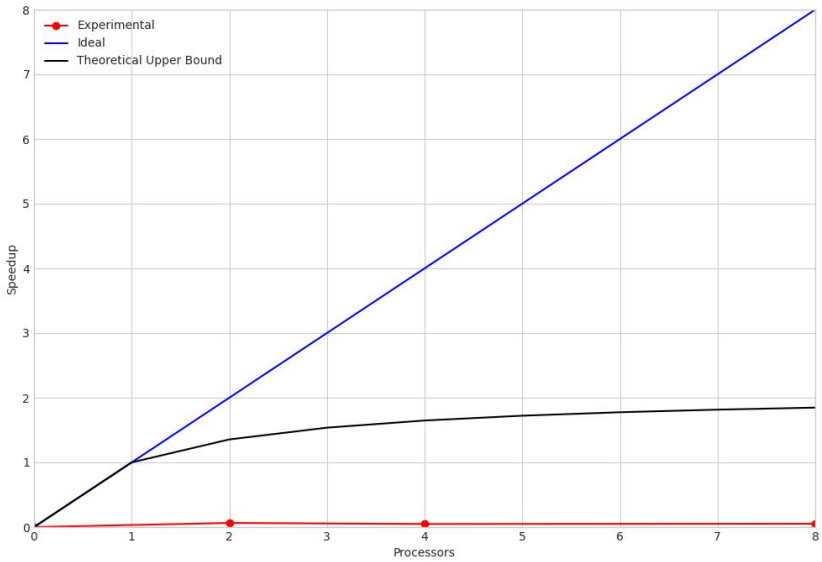
2.2.7.1 Optimization 0

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	204800.000	0.043	0.042	no data	no data	0.067	0.010	98.460	0.080	1.000	1.000
Parallel	2	204800.000	0.061	0.628	0.054	0.023	1.332	0.116	142.920	1.075	0.074	0.037
Parallel	4	204800.000	0.059	0.664	0.051	0.024	1.115	0.438	104.240	1.563	0.051	0.013
Parallel	8	204800.000	0.057	0.601	0.051	0.025	1.525	0.618	147.120	1.552	0.052	0.006



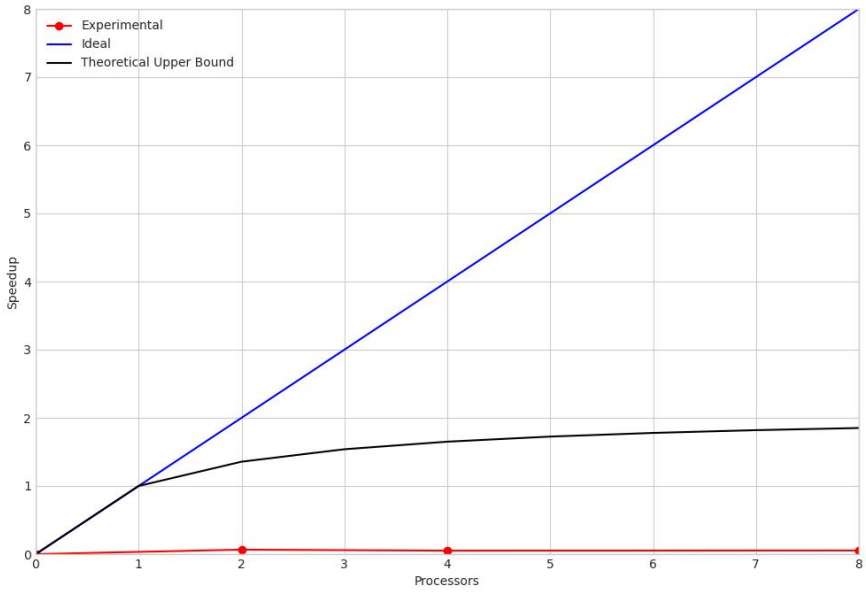
2.2.7.2 Optimization 1

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	204800.000	0.043	0.042	no data	no data	0.068	0.009	98.210	0.080	1.000	1.000
Parallel	2	204800.000	0.052	0.706	0.050	0.021	1.472	0.114	141.200	1.252	0.064	0.032
Parallel	4	204800.000	0.054	0.678	0.048	0.023	1.131	0.440	101.800	1.669	0.048	0.012
Parallel	8	204800.000	0.056	0.612	0.050	0.025	1.554	0.627	148.960	1.536	0.052	0.007



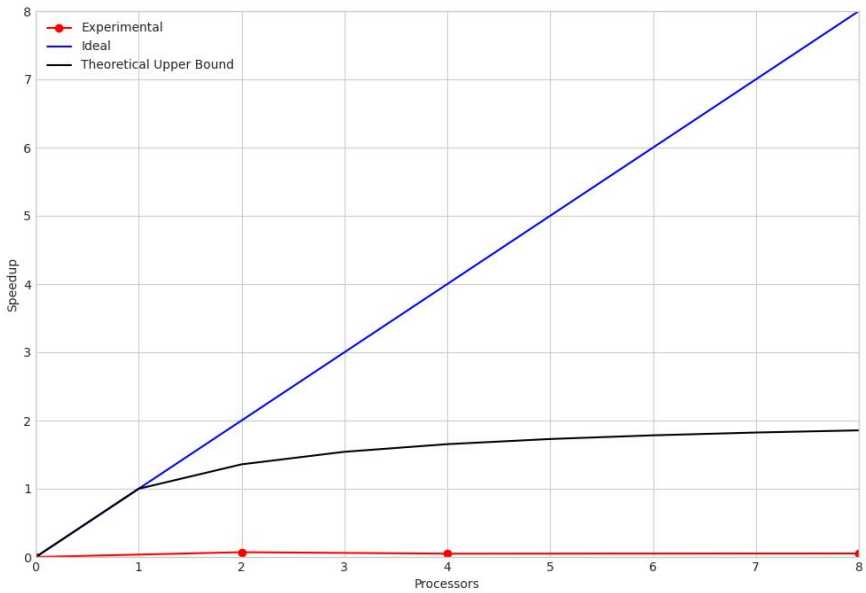
2.2.7.3 Optimization 2

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	204800.000	0.043	0.042	no data	no data	0.068	0.009	98.420	0.080	1.000	1.000
Parallel	2	204800.000	0.057	0.679	0.052	0.022	1.430	0.112	140.440	1.218	0.066	0.033
Parallel	4	204800.000	0.060	0.677	0.051	0.025	1.138	0.446	103.920	1.585	0.051	0.013
Parallel	8	204800.000	0.055	0.603	0.051	0.025	1.542	0.611	148.760	1.502	0.053	0.007



2.2.7.4 Optimization 3

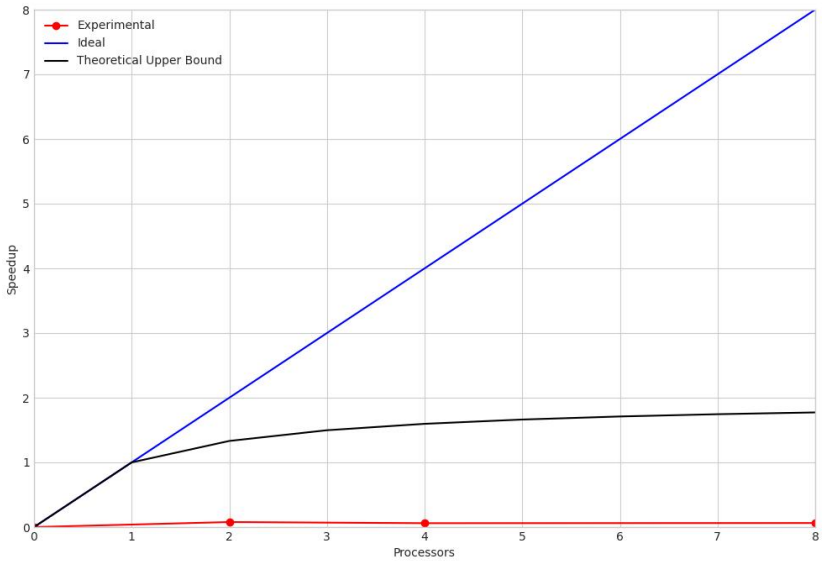
Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	204800.000	0.043	0.042	no data	no data	0.067	0.011	98.690	0.080	1.000	1.000
Parallel	2	204800.000	0.055	0.697	0.052	0.022	1.465	0.112	145.800	1.135	0.071	0.035
Parallel	4	204800.000	0.060	0.675	0.051	0.025	1.139	0.438	104.120	1.625	0.049	0.012
Parallel	8	204800.000	0.057	0.612	0.052	0.026	1.534	0.648	149.120	1.536	0.052	0.007



2.2.8 Tile-410000 graph

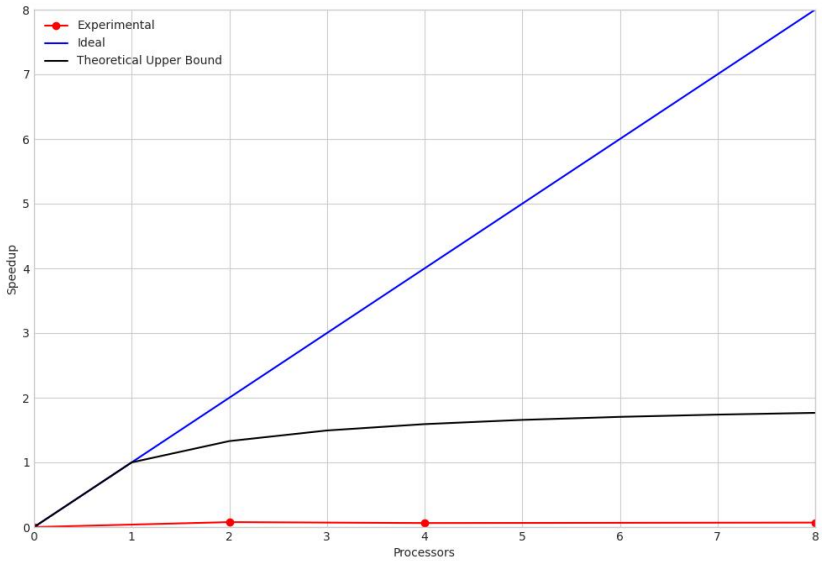
2.2.8.1 Optimization 0

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	409600.000	0.087	0.085	no data	no data	0.145	0.022	99.230	0.170	1.000	1.000
Parallel	2	409600.000	0.117	1.541	0.120	0.046	3.186	0.214	161.840	2.186	0.078	0.039
Parallel	4	409600.000	0.113	1.590	0.114	0.047	2.543	0.981	132.640	2.825	0.060	0.015
Parallel	8	409600.000	0.117	1.324	0.116	0.051	3.308	1.309	181.680	2.720	0.063	0.008



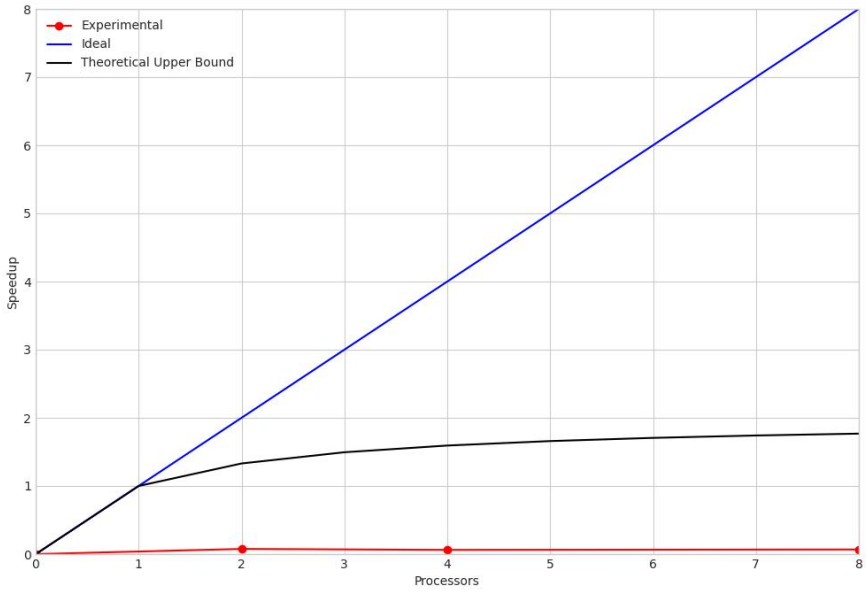
2.2.8.2 Optimization 1

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	409600.000	0.087	0.084	no data	no data	0.140	0.025	99.270	0.170	1.000	1.000
Parallel	2	409600.000	0.114	1.568	0.117	0.044	3.222	0.226	162.160	2.214	0.077	0.038
Parallel	4	409600.000	0.118	1.540	0.116	0.049	2.459	0.974	132.880	2.725	0.062	0.016
Parallel	8	409600.000	0.111	1.263	0.113	0.050	3.152	1.201	184.320	2.440	0.070	0.009



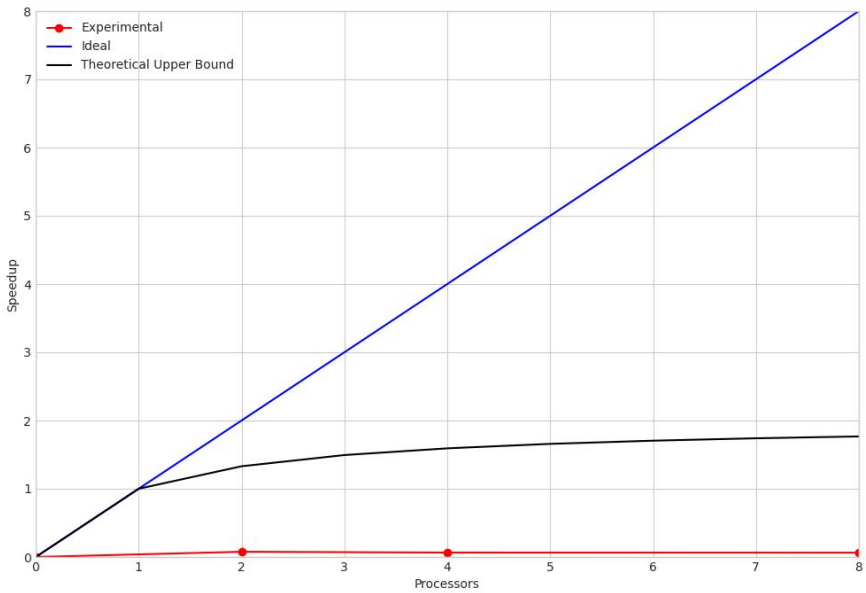
2.2.8.3 Optimization 2

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	409600.000	0.087	0.084	no data	no data	0.140	0.025	99.350	0.170	1.000	1.000
Parallel	2	409600.000	0.113	1.593	0.118	0.045	3.263	0.231	162.720	2.231	0.076	0.038
Parallel	4	409600.000	0.112	1.620	0.111	0.046	2.575	1.007	137.640	2.742	0.062	0.015
Parallel	8	409600.000	0.109	1.262	0.112	0.049	3.148	1.202	182.280	2.498	0.068	0.009



2.2.8.4 Optimization 3

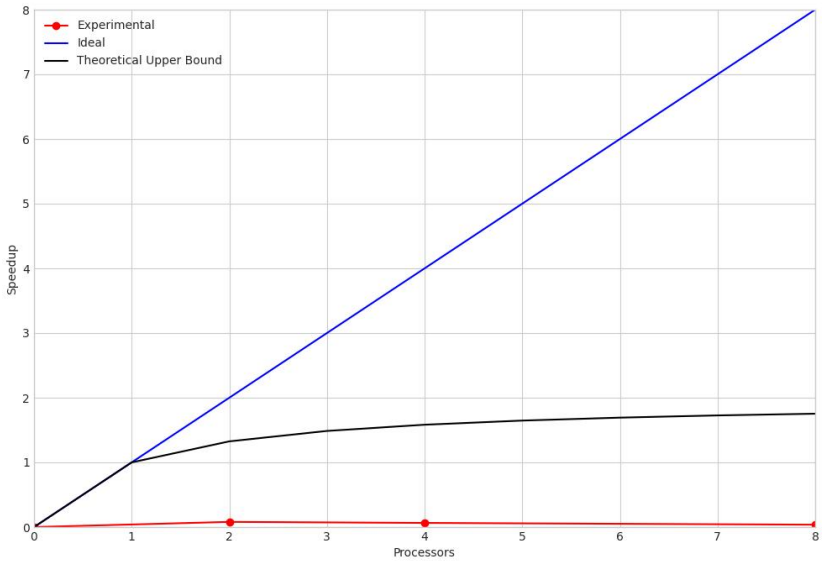
Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	409600.000	0.087	0.084	no data	no data	0.141	0.024	99.170	0.170	1.000	1.000
Parallel	2	409600.000	0.118	1.526	0.121	0.046	3.154	0.217	160.640	2.214	0.077	0.038
Parallel	4	409600.000	0.117	1.559	0.116	0.048	2.536	0.934	138.120	2.616	0.065	0.016
Parallel	8	409600.000	0.118	1.346	0.117	0.052	3.377	1.299	181.880	2.670	0.064	0.008



2.2.9 Tile-820000 graph

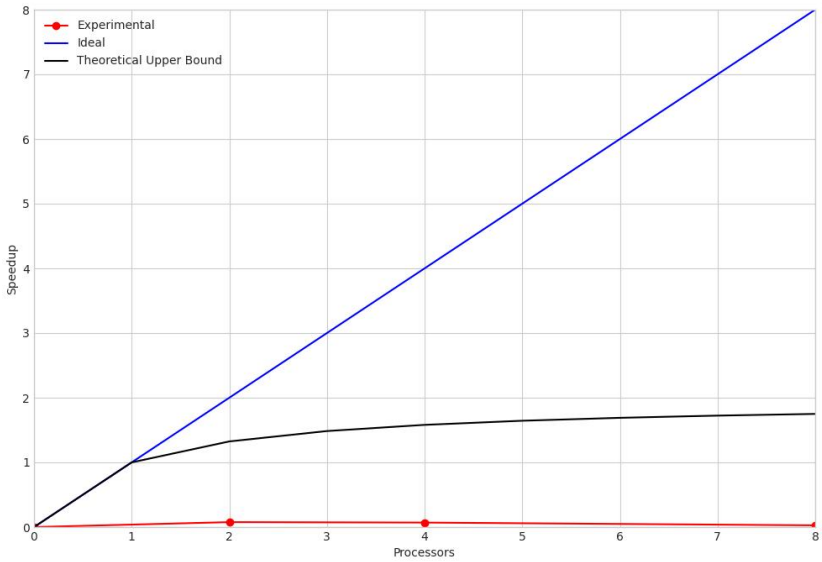
2.2.9.1 Optimization 0

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	819200.000	0.174	0.168	no data	no data	0.291	0.047	99.360	0.342	1.000	1.000
Parallel	2	819200.000	0.242	3.328	0.269	0.094	6.810	0.436	172.960	4.266	0.080	0.040
Parallel	4	819200.000	0.228	3.507	0.255	0.094	5.724	2.184	154.440	5.308	0.064	0.016
Parallel	8	819200.000	0.217	2.818	0.257	0.098	9.338	3.968	208.100	9.268	0.037	0.005



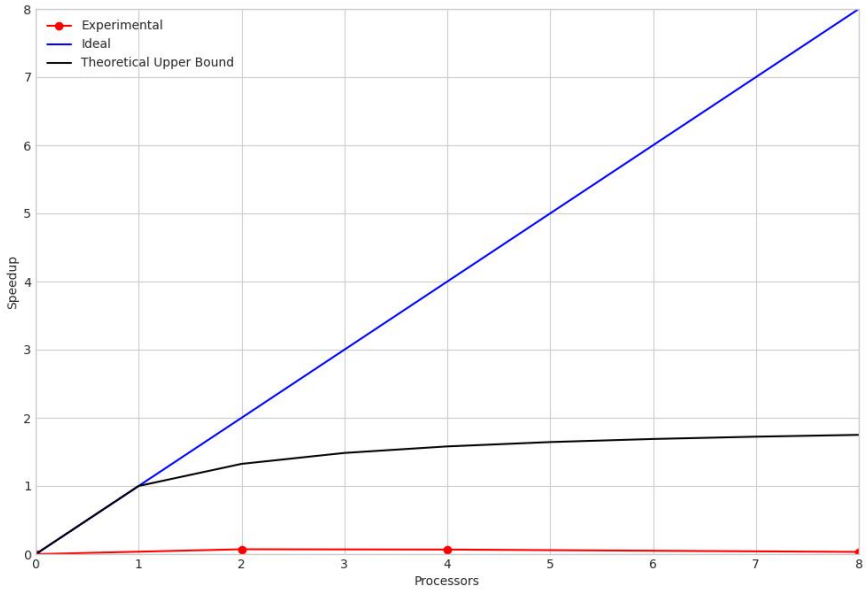
2.2.9.2 Optimization 1

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	819200.000	0.174	0.167	no data	no data	0.289	0.049	99.460	0.342	1.000	1.000
Parallel	2	819200.000	0.245	3.480	0.274	0.096	7.127	0.434	172.720	4.463	0.077	0.038
Parallel	4	819200.000	0.238	3.476	0.260	0.096	5.475	2.090	158.200	4.890	0.070	0.017
Parallel	8	819200.000	0.228	2.699	0.260	0.099	7.397	2.972	203.400	12.659	0.027	0.003



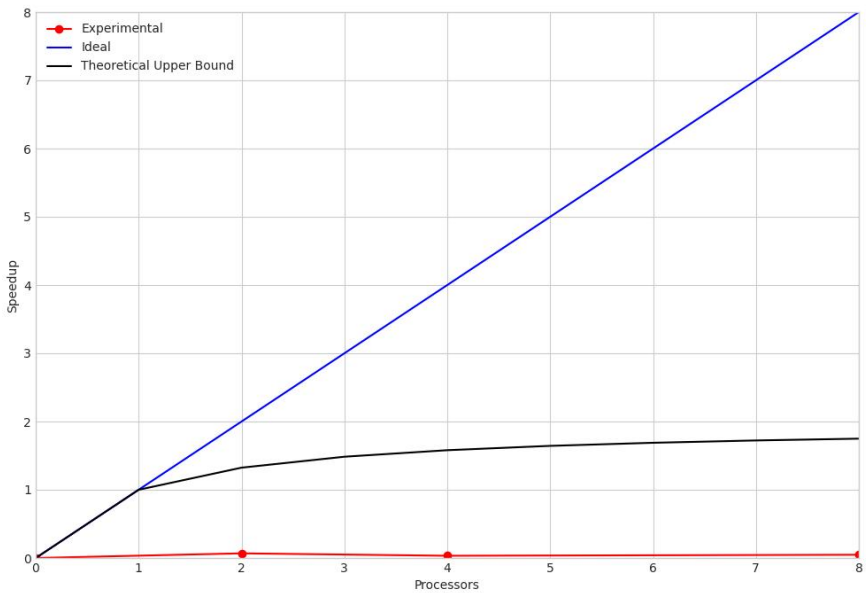
2.2.9.3 Optimization 2

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	819200.000	0.173	0.167	no data	no data	0.286	0.051	99.410	0.341	1.000	1.000
Parallel	2	819200.000	0.228	3.724	0.264	0.090	7.558	0.456	171.880	4.784	0.071	0.036
Parallel	4	819200.000	0.238	3.495	0.263	0.097	5.523	2.082	155.120	5.094	0.067	0.017
Parallel	8	819200.000	0.232	2.723	0.259	0.100	6.890	2.637	186.520	10.542	0.032	0.004



2.2.9.4 Optimization 3

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	819200.000	0.172	0.166	no data	no data	0.284	0.051	99.360	0.340	1.000	1.000
Parallel	2	819200.000	0.231	3.687	0.267	0.091	7.495	0.452	167.520	4.969	0.068	0.034
Parallel	4	819200.000	0.223	3.553	0.256	0.093	5.558	2.133	146.720	10.340	0.033	0.008
Parallel	8	819200.000	0.223	2.797	0.254	0.097	6.987	2.698	209.280	7.146	0.048	0.006



2.3 Considerations

As expected, the split and merge phases of the MPI implementation of Tarjan's algorithm severely worsen the performances of the sequential algorithm. As a result, we are always observing a speedup which is smaller than 1. Communication time between the nodes of the cluster is another factor that worsened the performances.

CHAPTER 3

CUDA IMPLEMENTATION WITH GLOBAL MEMORY

In this chapter we will discuss about our implementation of Tarjan's algorithm accelerated by CUDA (`cuda_only`) and present the experimental results that this algorithm.

3.1 Algorithm idea

3.1.1 Remarkable results in research literature

Tarjan's algorithm is difficult to parallelize because it is based on DFS graph traversal. For a long time, many researchers believed that no parallel DFS algorithms existed [11]. In the literature, some algorithm for parallelizing DFS were developed, but those algorithms suffer from strong constraints. For example, Reif proposed a parallel algorithm for lexicographical DFS [12]. Others such as Aggarwal and Anderson, have described algorithms for performing DFS on both undirected and directed graphs [13], but the algorithm is randomized and it requires n^{2376} processors. Other attempts require further restrictions to the graphs under question; for example, Chaudhuri and Hagerup proposed a solution that only works for acyclic and planar graphs.

The most popular algorithms for parallelizing the SCC search in CUDA are the Forward-Backward algorithm (illustrated in Fig. 3.1), the Coloring algorithm and the OBF algorithm [14] [15]. All of these algorithm are based on BFS traversal because it is much easier to parallelize compared to DFS. Implementing those algorithms in CUDA wouldn't meet the goal of parallelizeing Tarjan's algorithm which is based on DFS. For these reasons, we decided to take a completely different approach: we preprocess the graph to delete trivial SCCs and execute Tarjan's algorithm on the reduced graph. The idea is that, when the number trivial SCCs is substantial, Tarjan's algorithm would benefit from not having to process these nodes. We parallelized the graph preprocesing with CUDA.

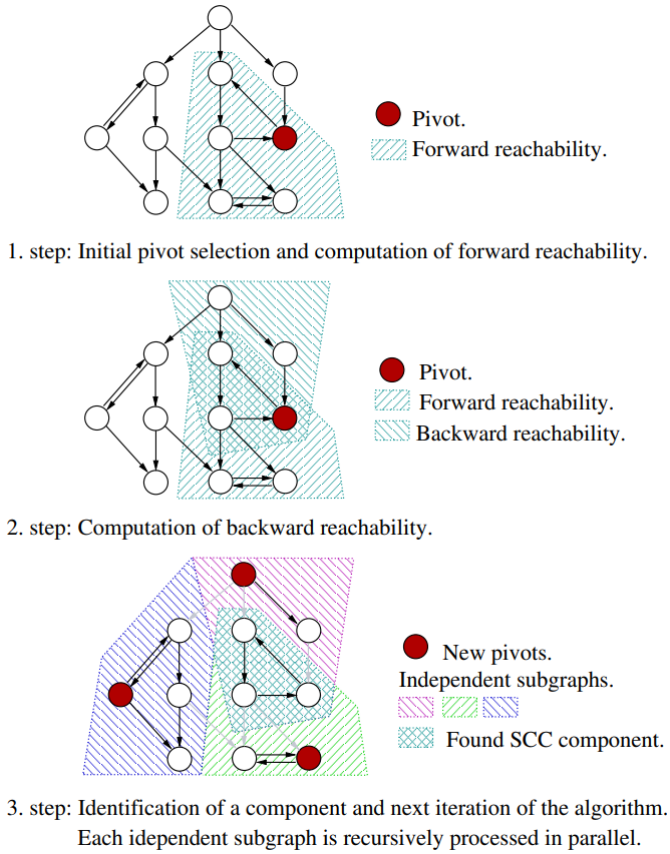


Figure 3.1: Key steps of the Forward-Backward Algorithm

3.1.2 Graph preprocessing: trivial SCCs deletion

The goal of graph preprocessing is to find and mark nodes belonging to trivial SCCs. A trivial SCC is an SCC composed of a single vertex. In other words, we are looking for vertices of the graph that have no immediate predecessors (in the case of leading components) or immediate successors (in the case of terminal components) in the graph. Such vertices may be iteratively removed from the subgraph as trivial SCCs. Futhermore, we also delete SCCs made of a single vertex with a self-loop. The result of the procedure is a bitmask of $|V|$ bits indicating which of the vertices have been eliminated.

3.1.3 A CUDA-friendly graph representation

In our sequential implementation of Tarjan's algorithm, discussed in 1.4, we used the adjacency maps to represent the graphs. This representation is not CUDA-friendly because it is a linked data structure. In order to process the graph with CUDA, we defined the `cuda_graph` struct: we encode the graph as the adjacency list that is represented as two one-dimensional arrays. One array stores the target vertices of edges sorted according to source vertex (`adj_lists`). The second array keeps an index to the first array for each vertex: the index points to the first edge emanating from the vertex (`adj_list_indexes`). A graphical representation of this data structure is shown in Fig. 3.2.

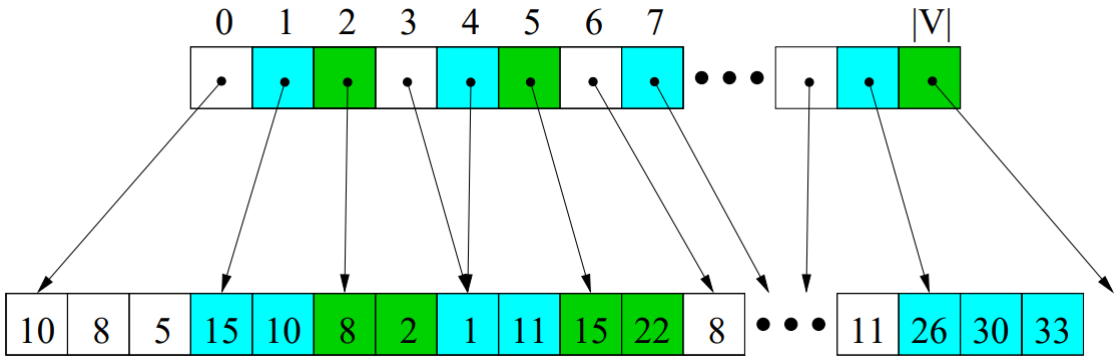


Figure 3.2: CUDA-friendly graph representation

3.1.4 Steps of the algorithm

The steps of our CUDA-accelerated algorithm are the following:

1. Read the graph from the input file as a `cuda_graph` (represented as seen above)
2. Load the graph on the global memory of the device
3. Execute a CUDA kernel to mark the nodes to be deleted on a bitmask (as described in 3.1.2)
4. Move the bitmask from the device to the host
5. Convert the `cuda_graph` in a `graph` (represented as seen in 1.4), ignoring the nodes flagged in the bitmask
6. Execute the sequential version of Tarjan's algorithm on the reduced `graph`

3.2 Choosing the block size

Before starting with the measurements, we need to choose the optimal number of threads per thread block. The goal is to maximize SM thread occupancy. The GPU used for CUDA acceleration is an RTX 3060Ti, whose compute capability is 8.6. From the computing capability, the following constraints are determined:

- Maximum number of threads per SM: 1536
- Maximum number of blocks per SM: 16
- Maximum number of threads per block: 1024
- Maximum number of registers per block: 65536
- Maximum number of registers per thread: 255
- Maximum number of registers per SM: 65536

Furthermore, by compiling the kernel with `--ptxas-options=-v` argument, we determined that it needs 10 registers to be executed. We consider different number of block sizes:

- $Blocksize = 32threads$

  $\Rightarrow Number\ of\ blocks\ per\ SM\ =\ 1536\ /\ 32\ =\ 48$

  $\Rightarrow 48\ blocks\ >\ 16\ blocks/SM \Leftrightarrow SM\ can't\ be\ fully\ occupied$

  The actual number of threads per SM is $16\ blocks\ *\ 32\ thread/block\ =\ 512\ threads$

  The occupancy level is 33%

- $Blocksize = 64threads$

  $\Rightarrow Number\ of\ blocks\ per\ SM\ =\ 1536\ /\ 64\ =\ 24$

  $\Rightarrow 24\ blocks\ >\ 16\ blocks/SM \Leftrightarrow SM\ can't\ be\ fully\ occupied$

  The actual number of threads per SM is $16\ blocks\ *\ 64\ thread/block\ =\ 1024\ threads$

  The occupancy level is 66%

- $Blocksize = 128threads$

  $\Rightarrow Number\ of\ blocks\ per\ SM\ =\ 1536\ /\ 128\ =\ 12$

  $\Rightarrow 12\ blocks\ <=\ 16\ blocks/SM \Leftrightarrow SM\ can\ be\ fully\ occupied$

  The actual number of threads per SM is $12\ blocks\ *\ 128\ thread/block\ =\ 1536\ threads$

  The occupancy level is 100%

- $Blocksize = 256threads$

  $\Rightarrow Number\ of\ blocks\ per\ SM\ =\ 1536\ /\ 256\ =\ 6$

  $\Rightarrow 6\ blocks\ <=\ 16\ blocks/SM \Leftrightarrow SM\ can\ be\ fully\ occupied$

  The actual number of threads per SM is $6\ blocks\ *\ 256\ thread/block\ =\ 1536\ threads$

  The occupancy level is 100%

- $Blocksize = 512threads$

  $\Rightarrow Number\ of\ blocks\ per\ SM\ =\ 1536\ /\ 512\ =\ 3$

  $\Rightarrow 3\ blocks\ <=\ 16\ blocks/SM \Leftrightarrow SM\ can\ be\ fully\ occupied$

  The actual number of threads per SM is $3\ blocks\ *\ 512\ thread/block\ =\ 1536\ threads$

  The occupancy level is 100%

- $Blocksize = 1024threads$

  $\Rightarrow Number\ of\ blocks\ per\ SM\ =\ \lfloor 1536\ /\ 1024 \rfloor\ =\ 1$

  $\Rightarrow 1\ blocks\ <=\ 16\ blocks/SM \Leftrightarrow SM\ can\ be\ fully\ occupied$

  The actual number of threads per SM is $1\ blocks\ *\ 1024\ thread/block\ =\ 1024\ threads$

  The occupancy level is 66%

In order to get maximum SM occupancy we choose a block size of 512 threads, which guarantees the execution of 1536 threads per SM. This needs a total of $1536 * 10 = 15360$ registers to be executed which is compatible with the constraints set by the computing capability of our GPU.

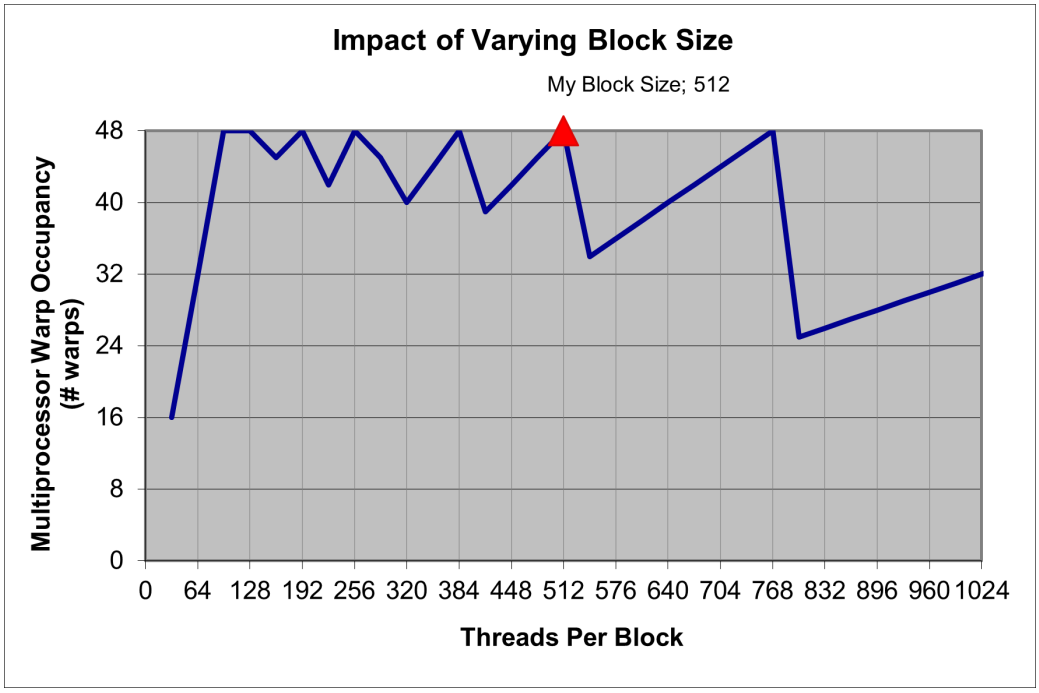


Figure 3.3: Variation of warp occupancy with respect to block size

3.3 Experimental results

3.3.1 Reference sequential algorithm for evaluating the speedup

Comparing the CUDA implementation we described in 3.1.4 with the sequential version of Tarjan's algorithm we used in chapter 2 would be unfair: for instance, on a fully connected graph (which has no trivial SCCs) the CUDA preprocessing would be useless and would simply delay the execution of Tarjan's algorithm. In order to correctly evaluate the speedup of our CUDA implementation we decided to write the `sequential_pre` program which performs a sequential preprocessing of the graph as described in 3.1.2 before applying Tarjan's algorithm. This implementation performs the same operations as our CUDA implementation, thus it has been used as a reference for measuring the speedup of the latter.

Notice that, in order to graph the theoretical upper bound of the speedup, we applied Amdahl's law by taking into account that the preprocessing of the graph is the "parallel part" of our algorithm while Tarjan's algorithm is executed sequentially. In other words, we estimated the percentage of "parallelizable" execution time to be

$$p = \frac{t_{preprocess}}{t_{elapsed}}$$

where $t_{elapsed}$ is the total elapsed time.

3.3.2 Measured metrics

In this subsection are presented the results of the parallel version of Tarjan's algorithm with CUDA. Each graph is tested with all the compiler optimization (O0, O1, O2, O3) and with $\frac{n}{4}$, $\frac{n}{2}$, $\frac{3*n}{4}$, n CUDA threads, where n is the number of vertices of the graph tested. In the next pages for each graph, set the

optimization, there are a table and a graph of the speedup. In the table there are:

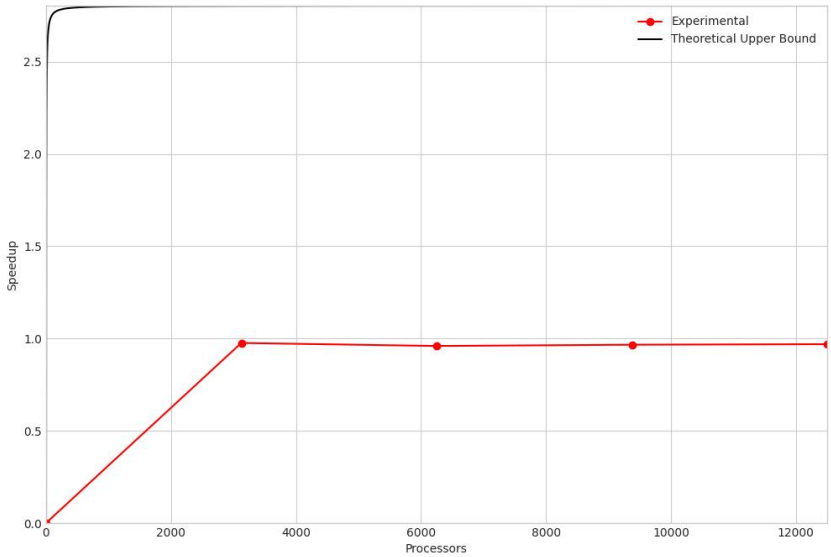
- **Version**: is the version of the program. It can be "Serial" if it's the sequential with preprocessing version of the program, or it can be "Parallel" if it's the parallel version with CUDA.
- **Threads**: is the number of thread CUDA. For the "Serial" version, the value of this field is 1, because it's the sequential with preprocessing version of the program executed on only one processor of the CPU.
- **vertices**: is the number of vertices of the graph after the preprocessing.
- **init**: is the time spent to initialize the program, in particular it's the time spent to create and initialize the data structures, and to read the graph from a file.
- **preprocess**: is the time spent to do the preprocessing of the graph.
- **conversion**: is the time spent to convert the graph into a graph adapted for CUDA.
- **finalize**: is the time spent to deallocate the necessary data structures.
- **tarjan**: the time spent to execute Tarjan's algorithm.
- **user**: is the total number of CPU-seconds that the process used directly (in user mode), in seconds.
- **system**: is the total number of CPU-seconds used by the system on behalf of the process (in kernel mode), in seconds.
- **pCPU**: is the percentage of the CPU that this process got. This is just user + system times divided by the total running time.
- **elapsed**: is the elapsed real (wall clock) time used by the process, in seconds.
- **Speedup**: the speedup of the program.
- **Efficiency**: the efficiency of the program.

All values except those in the `Version`, `Threads`, `Speedup`, and `Efficiency` columns are means. These means were calculated from the files of the measurements taken. `Speedup`, and `Efficiency` are calculated with the formulas in Cap.1 using the results of these means. For the sequential algorithm, the `finalize` and `conversion` fields were not measured.

3.3.3 Fully connected graph

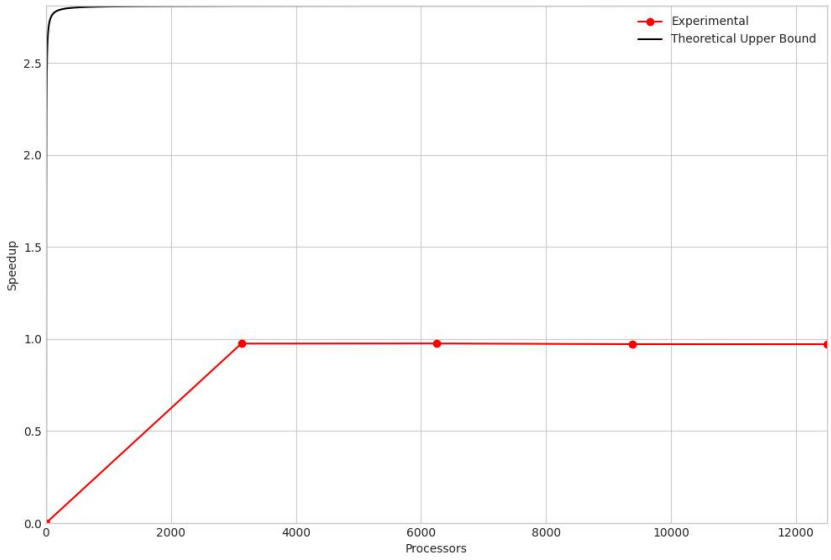
3.3.3.1 Optimization 0

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	12500.000	2.231	no data	9.478	no data	2.940	13.709	1.013	99.040	14.730	1.000	1.000
Parallel	3125	12500.000	2.346	0.017	0.004	9.663	2.977	14.016	1.055	99.000	15.092	0.976	0.000
Parallel	6250	12500.000	2.347	0.018	0.002	9.862	3.029	14.253	1.068	99.000	15.346	0.960	0.000
Parallel	9375	12500.000	2.345	0.018	0.002	9.790	2.994	14.152	1.065	99.000	15.239	0.967	0.000
Parallel	12500	12500.000	2.343	0.018	0.002	9.761	2.981	14.096	1.076	99.000	15.194	0.969	0.000



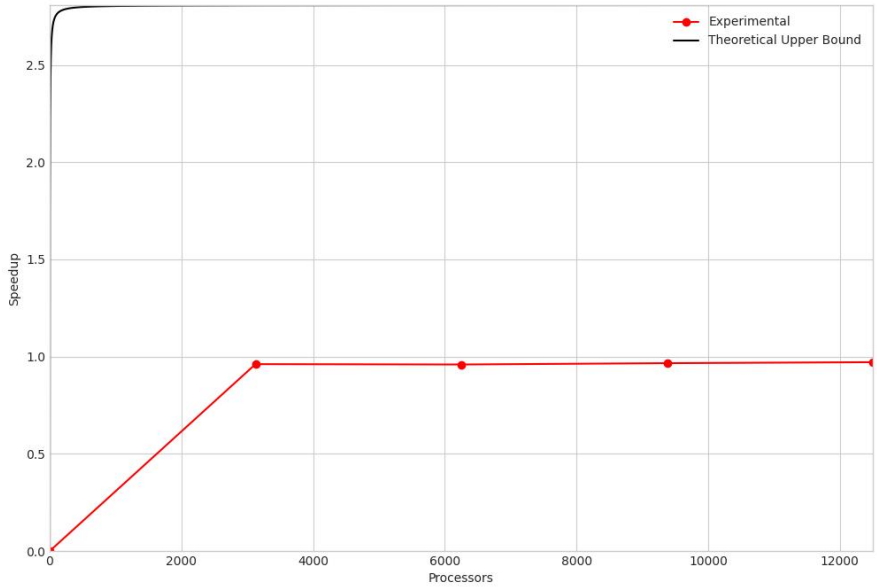
3.3.3.2 Optimization 1

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	12500.000	2.237	no data	9.523	no data	2.942	13.772	1.002	99.000	14.782	1.000	1.000
Parallel	3125	12500.000	2.372	0.017	0.004	9.689	2.990	14.096	1.043	99.000	15.163	0.975	0.000
Parallel	6250	12500.000	2.350	0.018	0.002	9.696	2.999	14.068	1.061	99.000	15.154	0.975	0.000
Parallel	9375	12500.000	2.335	0.018	0.002	9.747	3.019	14.128	1.060	99.000	15.211	0.972	0.000
Parallel	12500	12500.000	2.349	0.018	0.002	9.767	2.993	14.129	1.066	99.000	15.218	0.971	0.000



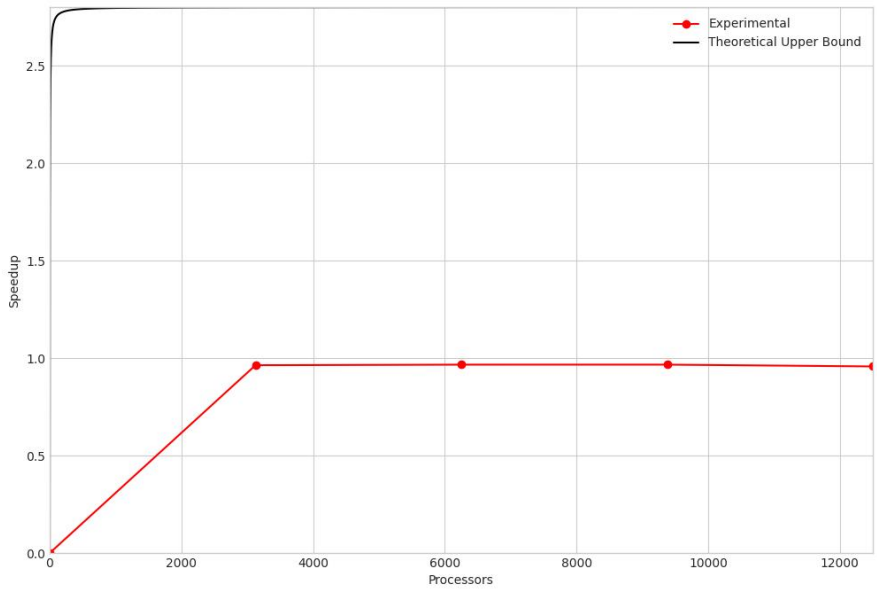
3.3.3.3 Optimization 2

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	12500.000	2.236	no data	9.454	no data	2.911	13.648	1.026	99.040	14.681	1.000	1.000
Parallel	3125	12500.000	2.353	0.017	0.004	9.790	3.023	14.190	1.061	99.000	15.274	0.961	0.000
Parallel	6250	12500.000	2.353	0.017	0.002	9.823	3.013	14.218	1.063	99.000	15.302	0.959	0.000
Parallel	9375	12500.000	2.343	0.019	0.002	9.747	2.996	14.106	1.068	99.000	15.196	0.966	0.000
Parallel	12500	12500.000	2.344	0.018	0.002	9.682	2.977	14.032	1.058	99.000	15.114	0.971	0.000



3.3.3.4 Optimization 3

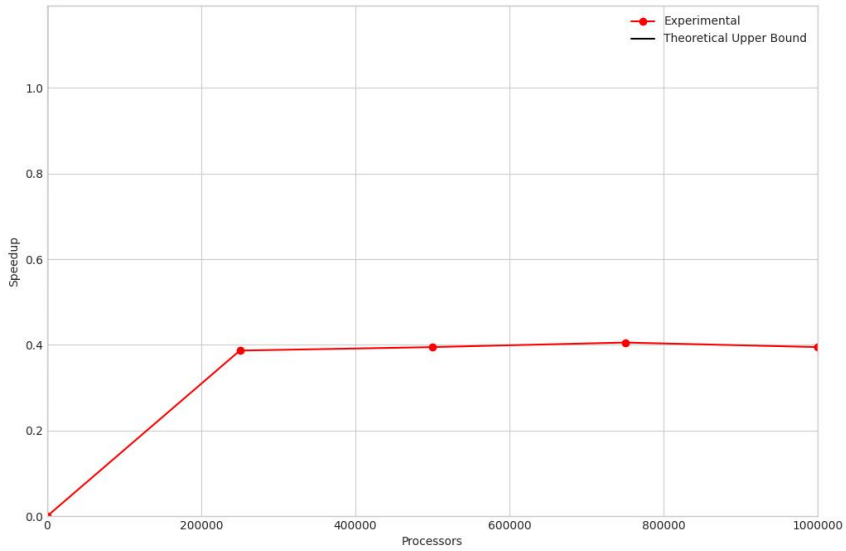
Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	12500.000	2.231	no data	9.460	no data	2.934	13.706	0.994	99.000	14.708	1.000	1.000
Parallel	3125	12500.000	2.355	0.018	0.004	9.778	3.011	14.173	1.061	99.000	15.257	0.964	0.000
Parallel	6250	12500.000	2.350	0.017	0.002	9.748	3.004	14.129	1.056	99.000	15.211	0.967	0.000
Parallel	9375	12500.000	2.342	0.018	0.002	9.762	2.994	14.124	1.065	99.000	15.210	0.967	0.000
Parallel	12500	12500.000	2.359	0.018	0.002	9.864	3.024	14.286	1.046	99.000	15.356	0.958	0.000



3.3.4 Fully disconnected graph

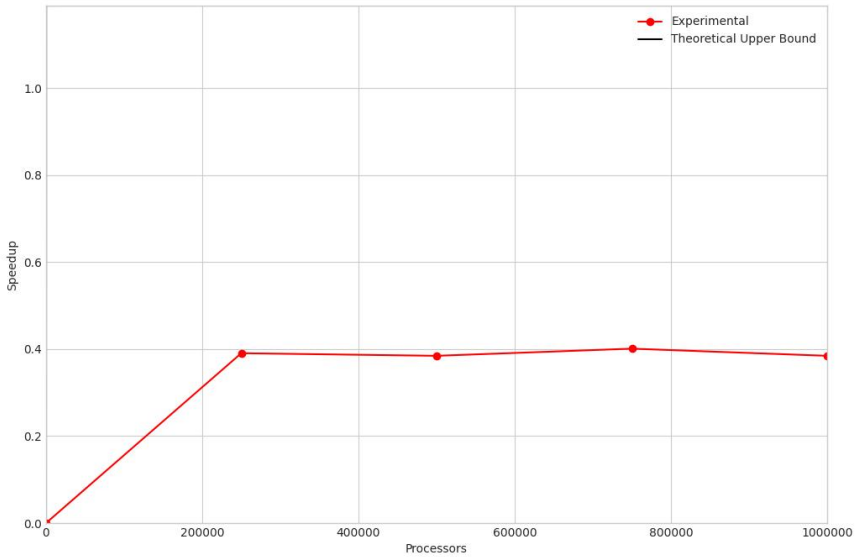
3.3.4.1 Optimization 0

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	0.000	0.026	no data	0.005	no data	0.000	0.024	0.000	97.480	0.030	1.000	1.000
Parallel	250000	0.000	0.059	0.000	0.000	0.001	0.000	0.024	0.036	85.520	0.078	0.387	0.000
Parallel	500000	0.000	0.058	0.000	0.001	0.001	0.000	0.025	0.036	89.000	0.076	0.395	0.000
Parallel	750000	0.000	0.058	0.000	0.001	0.001	0.000	0.026	0.034	90.280	0.074	0.405	0.000
Parallel	1000000	0.000	0.058	0.000	0.001	0.001	0.000	0.028	0.034	90.600	0.076	0.395	0.000



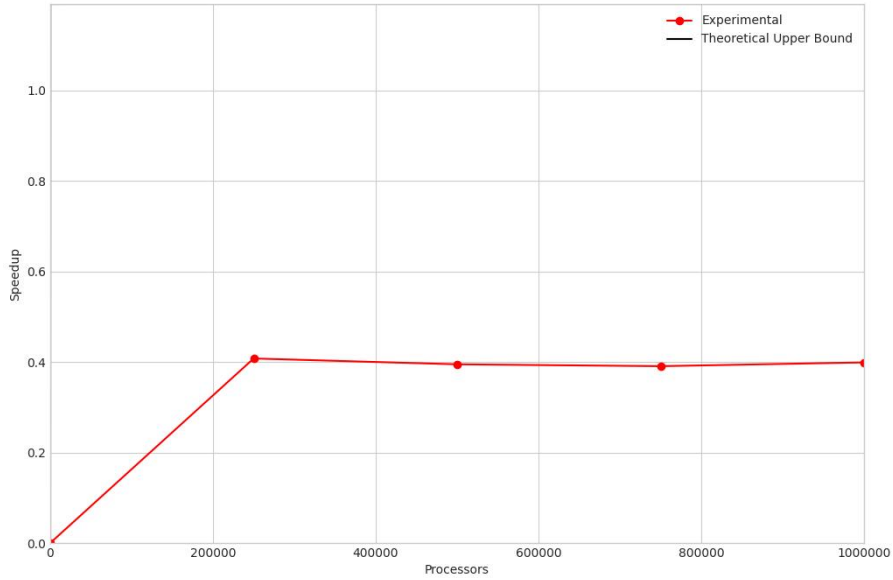
3.3.4.2 Optimization 1

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	0.000	0.026	no data	0.005	no data	0.000	0.023	0.000	97.320	0.030	1.000	1.000
Parallel	250000	0.000	0.058	0.000	0.000	0.001	0.000	0.024	0.036	86.400	0.077	0.391	0.000
Parallel	500000	0.000	0.058	0.000	0.001	0.001	0.000	0.027	0.034	87.600	0.078	0.385	0.000
Parallel	750000	0.000	0.059	0.000	0.001	0.001	0.000	0.027	0.036	90.160	0.075	0.401	0.000
Parallel	1000000	0.000	0.058	0.000	0.002	0.001	0.000	0.023	0.038	87.440	0.078	0.385	0.000



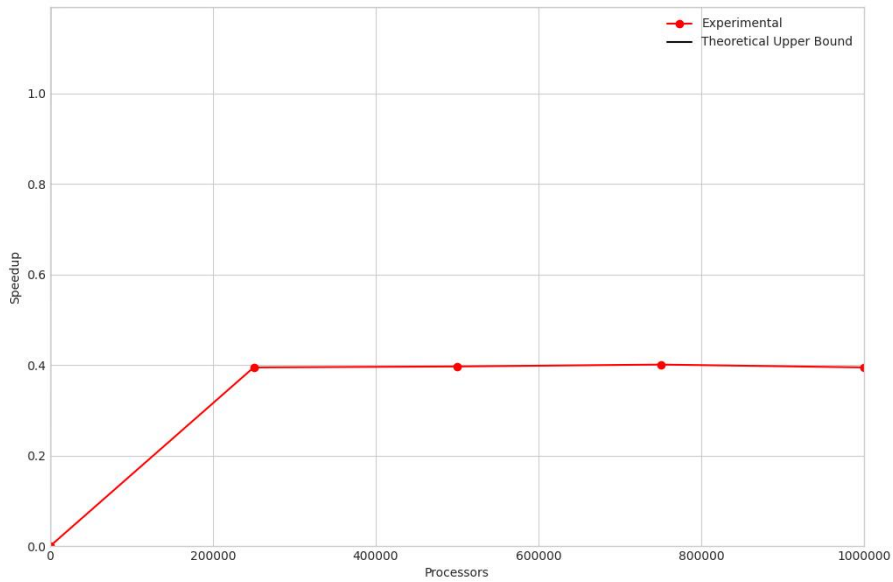
3.3.4.3 Optimization 2

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	0.000	0.026	no data	0.005	no data	0.000	0.026	0.000	98.240	0.030	1.000	1.000
Parallel	250000	0.000	0.058	0.000	0.000	0.001	0.000	0.025	0.037	90.320	0.074	0.408	0.000
Parallel	500000	0.000	0.058	0.000	0.001	0.001	0.000	0.024	0.035	88.600	0.076	0.395	0.000
Parallel	750000	0.000	0.058	0.000	0.001	0.001	0.000	0.026	0.035	87.880	0.077	0.391	0.000
Parallel	1000000	0.000	0.059	0.001	0.002	0.001	0.000	0.025	0.039	91.320	0.075	0.399	0.000



3.3.4.4 Optimization 3

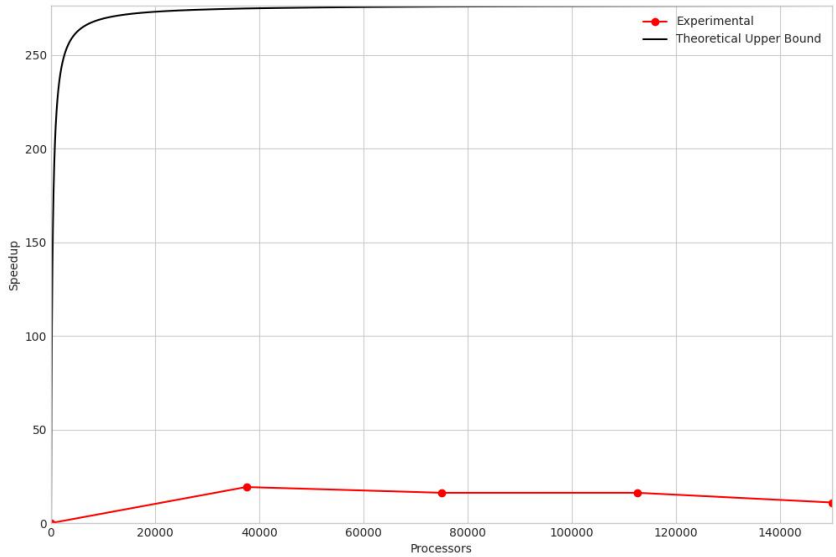
Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	0.000	0.026	no data	0.005	no data	0.000	0.023	0.000	97.440	0.030	1.000	1.000
Parallel	250000	0.000	0.059	0.000	0.000	0.001	0.000	0.027	0.035	89.360	0.076	0.395	0.000
Parallel	500000	0.000	0.058	0.000	0.001	0.001	0.000	0.024	0.036	89.280	0.076	0.397	0.000
Parallel	750000	0.000	0.058	0.000	0.001	0.001	0.000	0.025	0.036	91.000	0.075	0.401	0.000
Parallel	1000000	0.000	0.058	0.000	0.001	0.001	0.000	0.027	0.035	89.320	0.076	0.395	0.000



3.3.5 Random-150000 graph

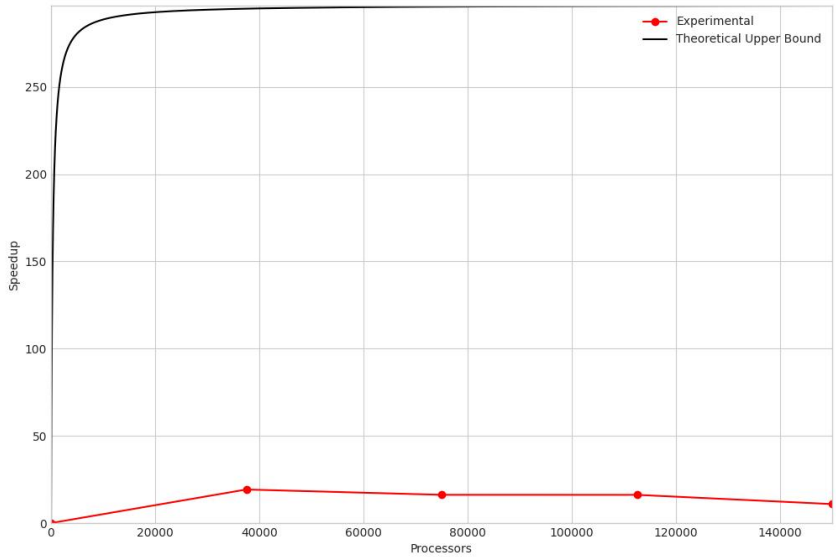
3.3.5.1 Optimization 0

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	71094.000	0.006	no data	11.645	no data	0.036	11.676	0.003	99.120	11.687	1.000	1.000
Parallel	37500	71094.000	0.038	0.000	0.477	0.027	0.038	0.541	0.043	96.280	0.607	19.260	0.001
Parallel	75000	71094.000	0.038	0.000	0.594	0.026	0.037	0.658	0.042	96.760	0.722	16.187	0.000
Parallel	112500	71094.000	0.039	0.000	0.593	0.027	0.038	0.661	0.040	97.240	0.722	16.196	0.000
Parallel	150000	71094.000	0.042	0.000	0.925	0.028	0.039	0.993	0.042	97.640	1.063	10.992	0.000



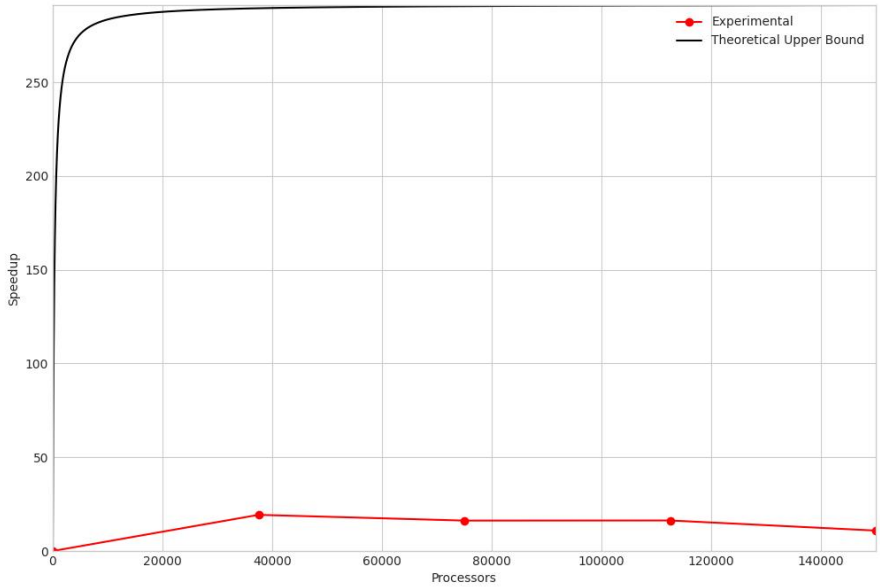
3.3.5.2 Optimization 1

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	71094.000	0.006	no data	11.675	no data	0.036	11.704	0.005	99.200	11.714	1.000	1.000
Parallel	37500	71094.000	0.039	0.000	0.477	0.027	0.038	0.544	0.039	96.280	0.609	19.242	0.001
Parallel	75000	71094.000	0.039	0.000	0.592	0.027	0.038	0.656	0.042	97.040	0.723	16.198	0.000
Parallel	112500	71094.000	0.039	0.000	0.592	0.027	0.038	0.656	0.044	96.800	0.724	16.180	0.000
Parallel	150000	71094.000	0.039	0.000	0.942	0.026	0.037	1.007	0.042	97.680	1.075	10.899	0.000



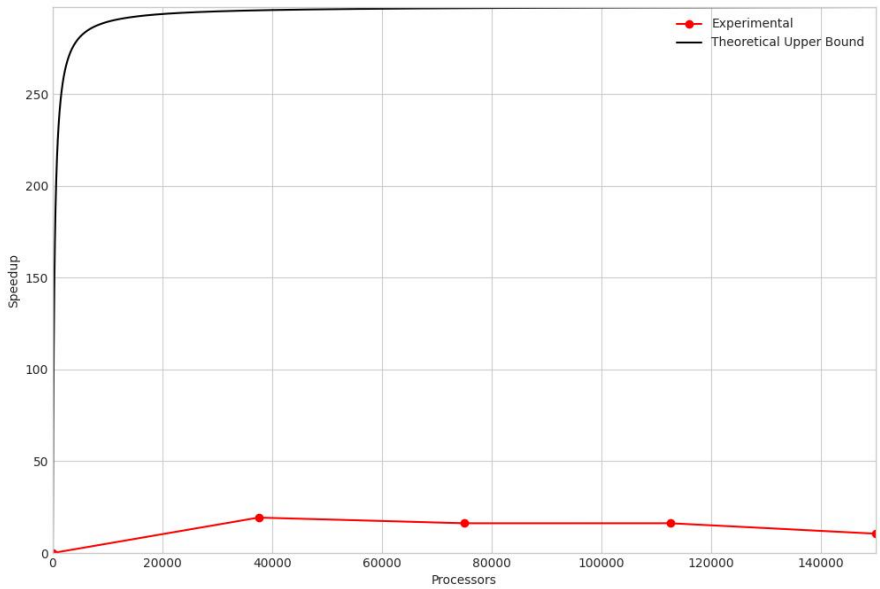
3.3.5.3 Optimization 2

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	71094.000	0.006	no data	11.658	no data	0.036	11.688	0.003	99.080	11.698	1.000	1.000
Parallel	37500	71094.000	0.039	0.000	0.478	0.027	0.038	0.542	0.043	96.600	0.606	19.291	0.001
Parallel	75000	71094.000	0.039	0.000	0.593	0.026	0.037	0.657	0.040	97.120	0.721	16.229	0.000
Parallel	112500	71094.000	0.040	0.000	0.593	0.026	0.037	0.657	0.044	97.480	0.718	16.292	0.000
Parallel	150000	71094.000	0.042	0.000	0.936	0.027	0.038	1.003	0.043	97.520	1.073	10.904	0.000



3.3.5.4 Optimization 3

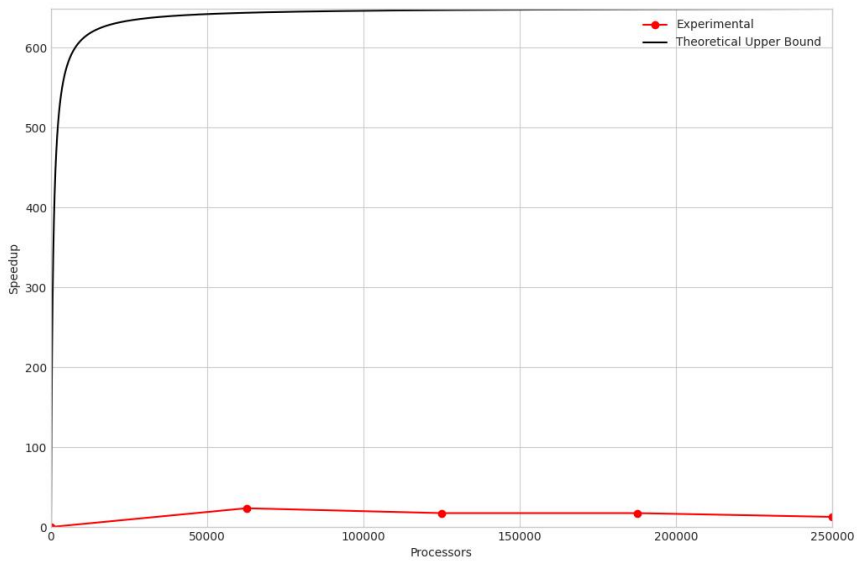
Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	71094.000	0.006	no data	11.672	no data	0.036	11.700	0.004	99.200	11.711	1.000	1.000
Parallel	37500	71094.000	0.039	0.000	0.478	0.027	0.038	0.544	0.041	96.600	0.607	19.299	0.001
Parallel	75000	71094.000	0.039	0.000	0.592	0.027	0.038	0.659	0.041	97.120	0.722	16.220	0.000
Parallel	112500	71094.000	0.039	0.000	0.592	0.027	0.038	0.657	0.040	96.960	0.722	16.229	0.000
Parallel	150000	71094.000	0.040	0.000	0.979	0.026	0.037	1.041	0.044	97.640	1.112	10.535	0.000



3.3.6 Random-250000 graph

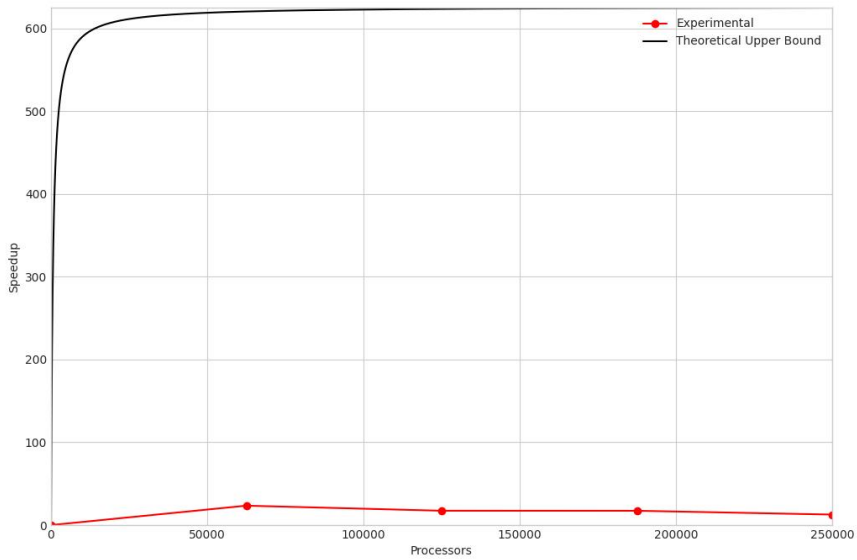
3.3.6.1 Optimization 0

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	118498.000	0.010	no data	32.428	no data	0.041	32.464	0.006	99.160	32.478	1.000	1.000
Parallel	62500	118498.000	0.044	0.000	1.222	0.046	0.043	1.310	0.048	97.960	1.386	23.433	0.000
Parallel	125000	118498.000	0.045	0.001	1.701	0.048	0.046	1.796	0.050	97.960	1.873	17.338	0.000
Parallel	187500	118498.000	0.045	0.000	1.701	0.048	0.046	1.795	0.050	98.120	1.873	17.338	0.000
Parallel	250000	118498.000	0.048	0.000	2.386	0.054	0.050	2.491	0.052	98.840	2.570	12.635	0.000



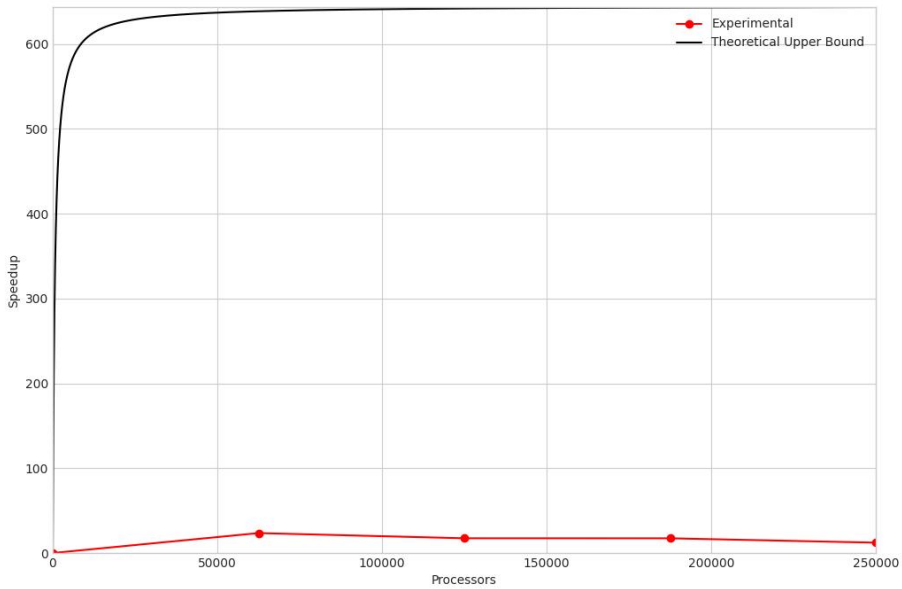
3.3.6.2 Optimization 1

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	118498.000	0.010	no data	32.409	no data	0.041	32.446	0.007	99.040	32.461	1.000	1.000
Parallel	62500	118498.000	0.044	0.000	1.222	0.045	0.043	1.311	0.048	98.000	1.385	23.434	0.000
Parallel	125000	118498.000	0.045	0.001	1.701	0.048	0.046	1.795	0.050	98.040	1.873	17.333	0.000
Parallel	187500	118498.000	0.045	0.000	1.703	0.048	0.046	1.799	0.048	98.040	1.876	17.300	0.000
Parallel	250000	118498.000	0.049	0.000	2.375	0.054	0.050	2.480	0.053	98.840	2.561	12.676	0.000



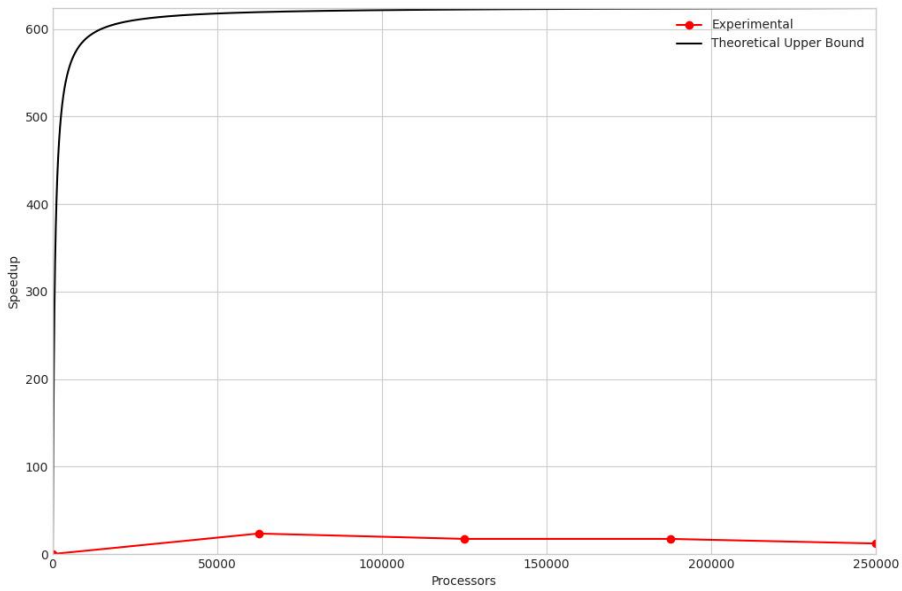
3.3.6.3 Optimization 2

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	118498.000	0.010	no data	32.485	no data	0.041	32.522	0.006	99.120	32.536	1.000	1.000
Parallel	62500	118498.000	0.043	0.000	1.222	0.045	0.043	1.312	0.046	98.000	1.386	23.474	0.000
Parallel	125000	118498.000	0.045	0.001	1.701	0.048	0.046	1.796	0.051	98.120	1.872	17.376	0.000
Parallel	187500	118498.000	0.046	0.001	1.702	0.049	0.046	1.799	0.049	98.000	1.877	17.336	0.000
Parallel	250000	118498.000	0.048	0.000	2.487	0.053	0.050	2.592	0.052	98.800	2.672	12.176	0.000



3.3.6.4 Optimization 3

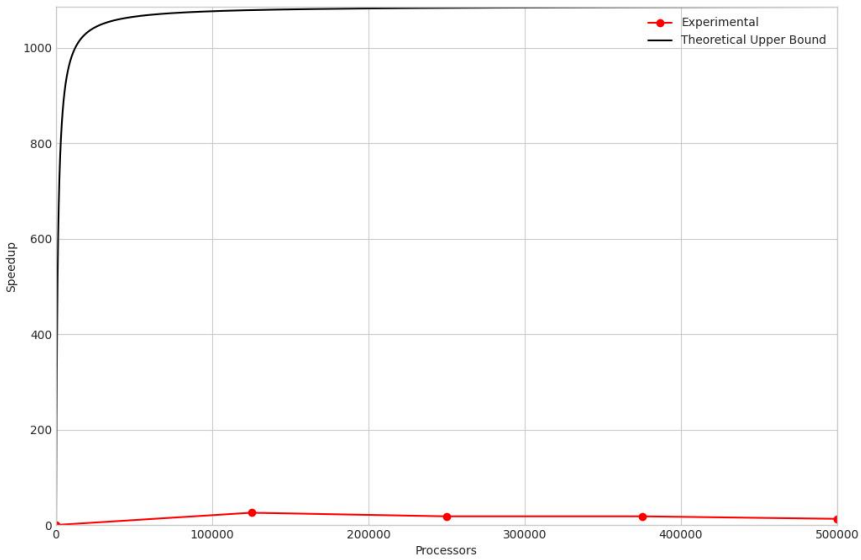
Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	118498.000	0.010	no data	32.394	no data	0.041	32.429	0.006	99.040	32.446	1.000	1.000
Parallel	62500	118498.000	0.045	0.000	1.222	0.045	0.043	1.310	0.050	97.960	1.387	23.396	0.000
Parallel	125000	118498.000	0.046	0.001	1.703	0.049	0.046	1.797	0.052	98.040	1.877	17.288	0.000
Parallel	187500	118498.000	0.045	0.000	1.704	0.049	0.046	1.804	0.046	98.000	1.878	17.280	0.000
Parallel	250000	118498.000	0.047	0.000	2.529	0.049	0.047	2.627	0.047	98.920	2.704	12.001	0.000



3.3.7 Random-500000 graph

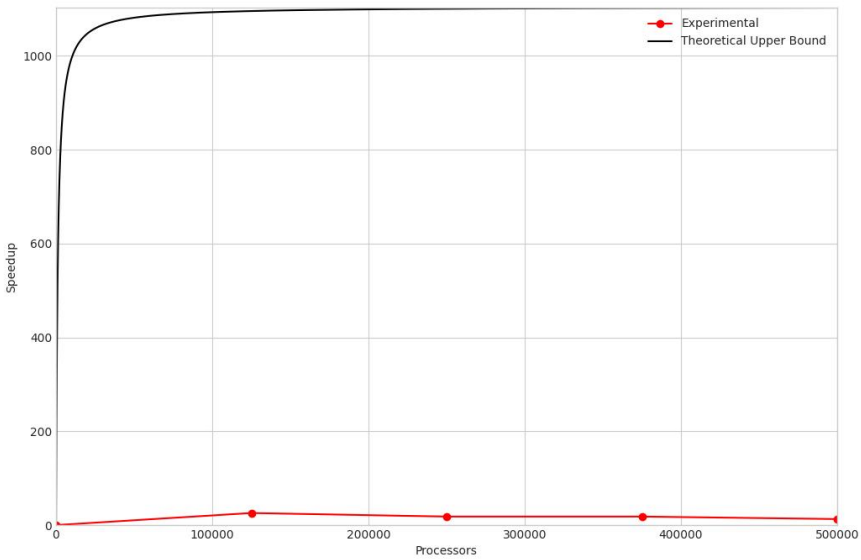
3.3.7.1 Optimization 0

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	236518.000	0.020	no data	128.240	no data	0.092	128.320	0.026	99.000	128.358	1.000	1.000
Parallel	125000	236518.000	0.064	0.000	4.633	0.148	0.123	4.904	0.074	99.000	5.003	25.655	0.000
Parallel	250000	236518.000	0.063	0.000	6.715	0.152	0.124	6.987	0.077	99.000	7.089	18.107	0.000
Parallel	375000	236518.000	0.062	0.000	6.709	0.151	0.124	6.985	0.069	99.000	7.081	18.128	0.000
Parallel	500000	236518.000	0.066	0.000	9.635	0.151	0.127	9.916	0.072	99.000	10.015	12.816	0.000



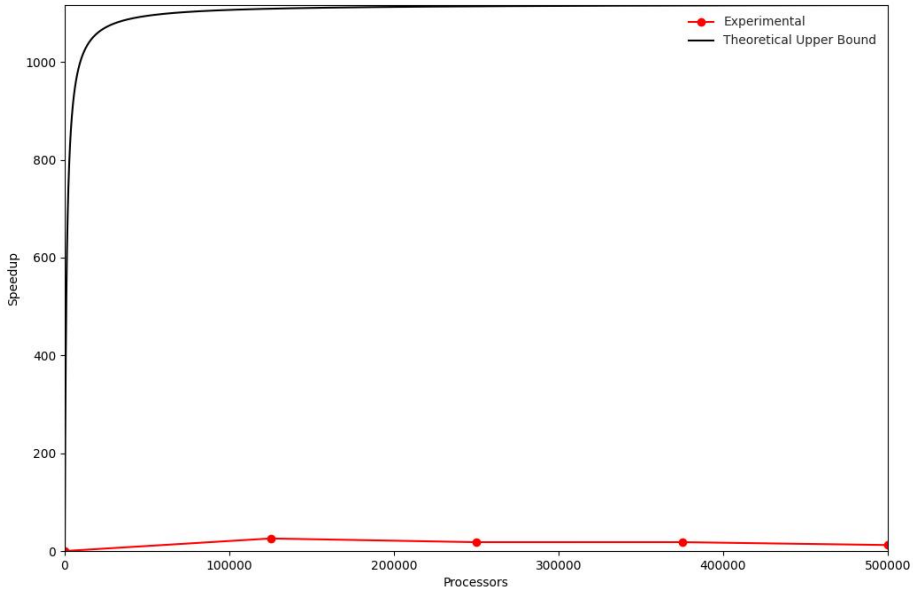
3.3.7.2 Optimization 1

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	236518.000	0.020	no data	128.384	no data	0.093	128.453	0.036	99.000	128.500	1.000	1.000
Parallel	125000	236518.000	0.064	0.000	4.623	0.151	0.125	4.898	0.074	99.000	4.999	25.706	0.000
Parallel	250000	236518.000	0.063	0.000	6.712	0.152	0.124	6.993	0.067	99.000	7.083	18.142	0.000
Parallel	375000	236518.000	0.063	0.000	6.712	0.152	0.124	6.986	0.075	99.000	7.086	18.135	0.000
Parallel	500000	236518.000	0.067	0.000	9.642	0.154	0.128	9.922	0.078	99.000	10.027	12.816	0.000



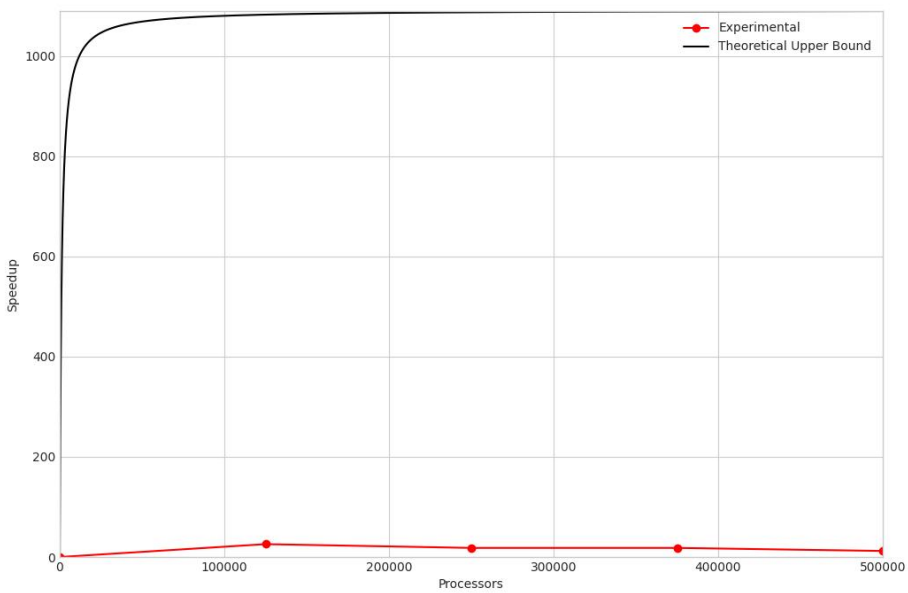
3.3.7.3 Optimization 2

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	236518.000	0.020	no data	127.986	no data	0.091	128.070	0.020	99.000	128.100	1.000	1.000
Parallel	125000	236518.000	0.064	0.000	4.622	0.151	0.125	4.900	0.070	99.000	4.998	25.630	0.000
Parallel	250000	236518.000	0.065	0.000	6.711	0.151	0.126	6.988	0.074	99.000	7.089	18.071	0.000
Parallel	375000	236518.000	0.064	0.000	6.708	0.152	0.125	6.983	0.074	99.000	7.082	18.087	0.000
Parallel	500000	236518.000	0.059	0.000	10.165	0.121	0.109	10.394	0.068	99.000	10.491	12.210	0.000



3.3.7.4 Optimization 3

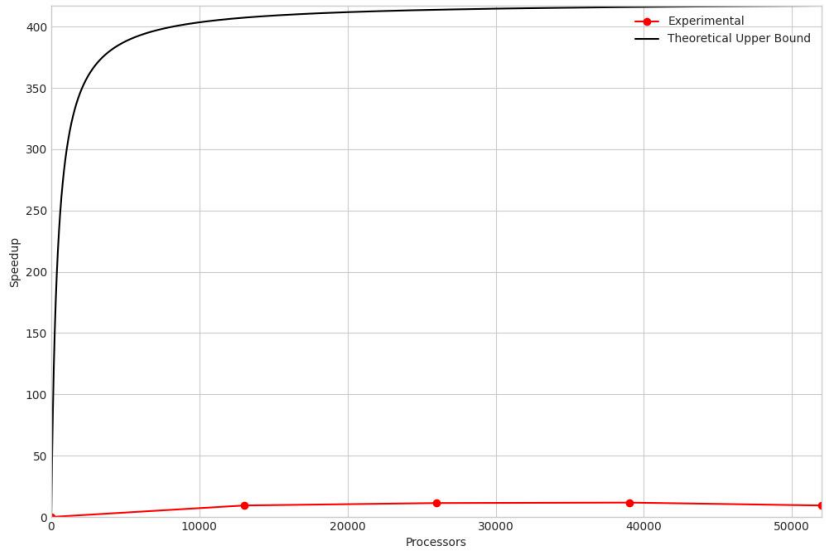
Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	236518.000	0.020	no data	128.266	no data	0.091	128.339	0.030	99.000	128.383	1.000	1.000
Parallel	125000	236518.000	0.064	0.000	4.626	0.150	0.124	4.900	0.074	99.000	5.001	25.670	0.000
Parallel	250000	236518.000	0.063	0.000	6.703	0.152	0.124	6.975	0.075	99.000	7.079	18.135	0.000
Parallel	375000	236518.000	0.064	0.000	6.705	0.151	0.124	6.981	0.072	99.000	7.078	18.138	0.000
Parallel	500000	236518.000	0.054	0.000	10.318	0.098	0.096	10.513	0.060	99.000	10.608	12.102	0.000



3.3.8 Tile-52000 graph

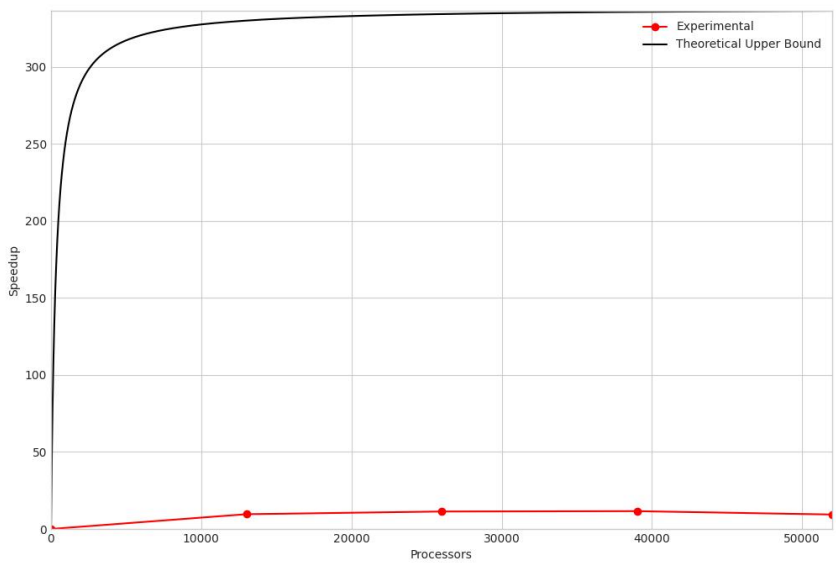
3.3.8.1 Optimization 0

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	25088.000	0.002	no data	1.602	no data	0.007	1.604	0.000	99.520	1.606	1.000	1.000
Parallel	13000	25088.000	0.038	0.000	0.100	0.007	0.007	0.117	0.040	92.840	0.172	9.359	0.001
Parallel	26000	25088.000	0.039	0.000	0.069	0.007	0.007	0.083	0.041	91.080	0.142	11.342	0.000
Parallel	39000	25088.000	0.038	0.000	0.069	0.007	0.007	0.087	0.038	93.600	0.137	11.740	0.000
Parallel	52000	25088.000	0.038	0.000	0.099	0.007	0.007	0.114	0.040	92.360	0.172	9.359	0.000



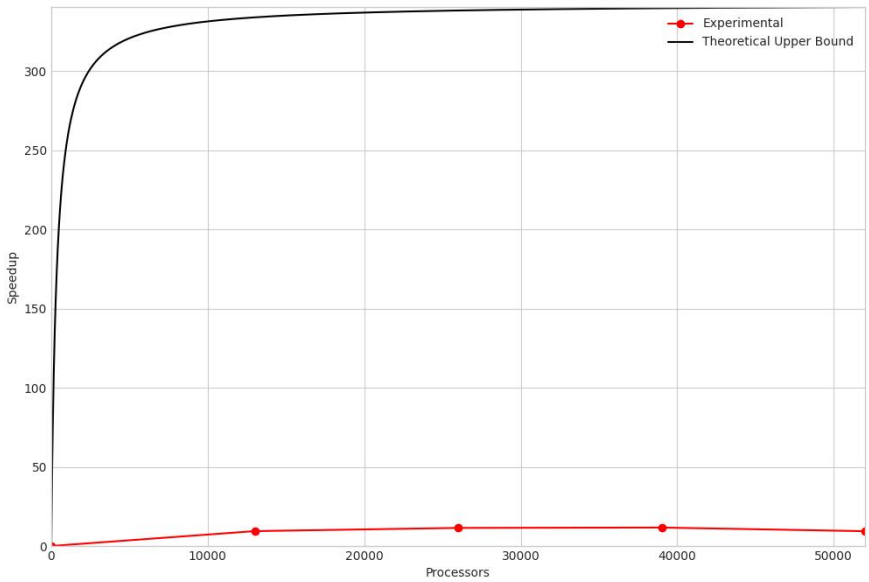
3.3.8.2 Optimization 1

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	25088.000	0.002	no data	1.605	no data	0.006	1.608	0.000	99.640	1.610	1.000	1.000
Parallel	13000	25088.000	0.037	0.000	0.100	0.007	0.007	0.116	0.037	93.800	0.168	9.583	0.001
Parallel	26000	25088.000	0.039	0.000	0.069	0.007	0.007	0.085	0.038	90.200	0.142	11.338	0.000
Parallel	39000	25088.000	0.039	0.001	0.069	0.007	0.007	0.086	0.038	91.920	0.139	11.566	0.000
Parallel	52000	25088.000	0.038	0.000	0.099	0.007	0.007	0.114	0.038	92.000	0.172	9.382	0.000



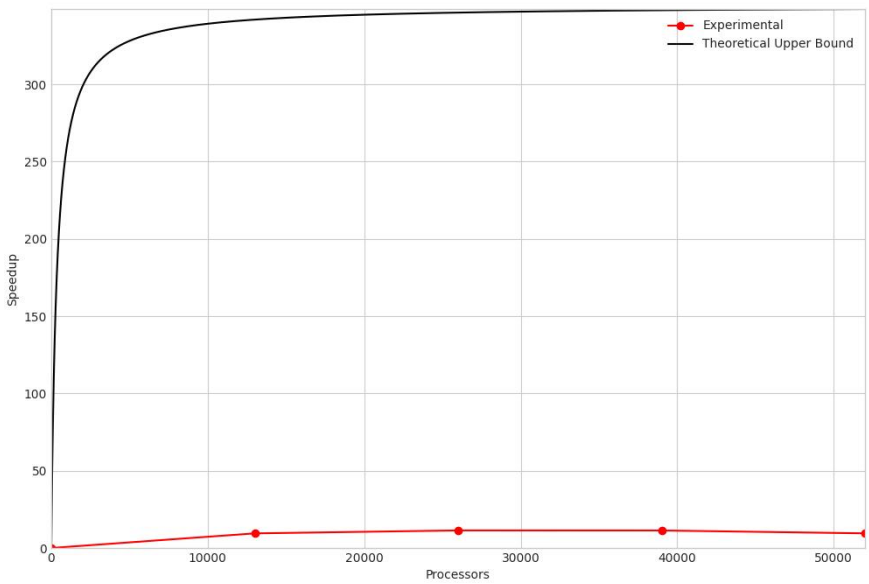
3.3.8.3 Optimization 2

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	25088.000	0.002	no data	1.605	no data	0.006	1.607	0.000	99.440	1.609	1.000	1.000
Parallel	13000	25088.000	0.037	0.000	0.100	0.007	0.007	0.116	0.036	92.600	0.171	9.400	0.001
Parallel	26000	25088.000	0.038	0.000	0.069	0.007	0.007	0.085	0.039	90.800	0.141	11.429	0.000
Parallel	39000	25088.000	0.039	0.000	0.069	0.007	0.007	0.084	0.040	93.160	0.138	11.661	0.000
Parallel	52000	25088.000	0.039	0.001	0.099	0.007	0.007	0.114	0.040	93.480	0.172	9.356	0.000



3.3.8.4 Optimization 3

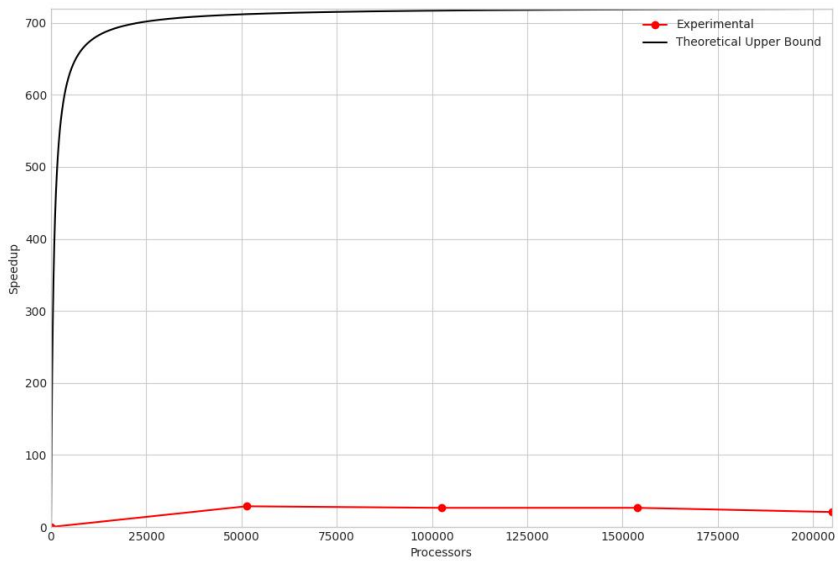
Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	25088.000	0.002	no data	1.592	no data	0.006	1.595	0.000	99.720	1.596	1.000	1.000
Parallel	13000	25088.000	0.038	0.000	0.100	0.007	0.007	0.118	0.037	93.840	0.169	9.457	0.001
Parallel	26000	25088.000	0.039	0.000	0.069	0.007	0.007	0.084	0.040	91.200	0.140	11.370	0.000
Parallel	39000	25088.000	0.039	0.001	0.069	0.007	0.007	0.085	0.040	91.560	0.141	11.338	0.000
Parallel	52000	25088.000	0.037	0.000	0.099	0.007	0.007	0.115	0.040	93.880	0.168	9.480	0.000



3.3.9 Tile-205000 graph

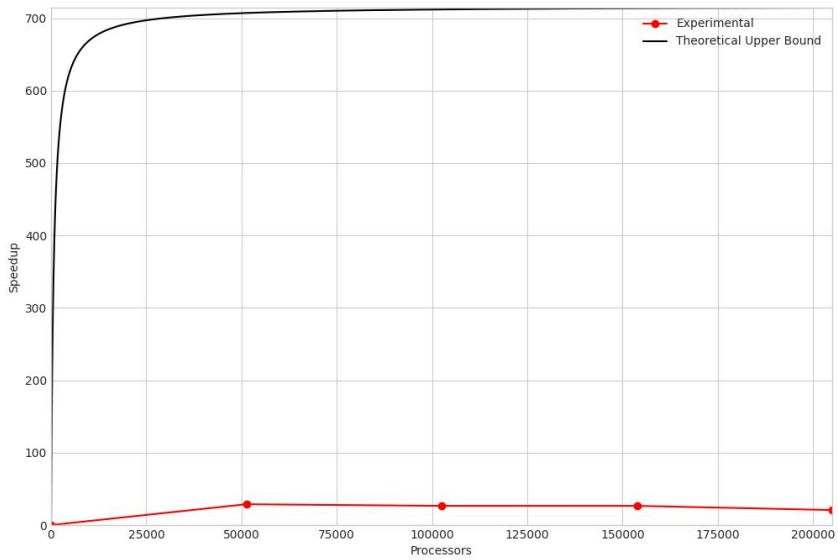
3.3.9.1 Optimization 0

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	100352.000	0.008	no data	25.436	no data	0.029	25.458	0.008	99.040	25.471	1.000	1.000
Parallel	51250	100352.000	0.042	0.001	0.754	0.029	0.030	0.813	0.047	97.000	0.886	28.748	0.001
Parallel	102500	100352.000	0.041	0.001	0.827	0.030	0.030	0.887	0.045	97.200	0.959	26.565	0.000
Parallel	153750	100352.000	0.042	0.001	0.828	0.029	0.030	0.888	0.045	97.400	0.957	26.610	0.000
Parallel	205000	100352.000	0.042	0.001	1.094	0.029	0.030	1.153	0.047	97.800	1.226	20.769	0.000



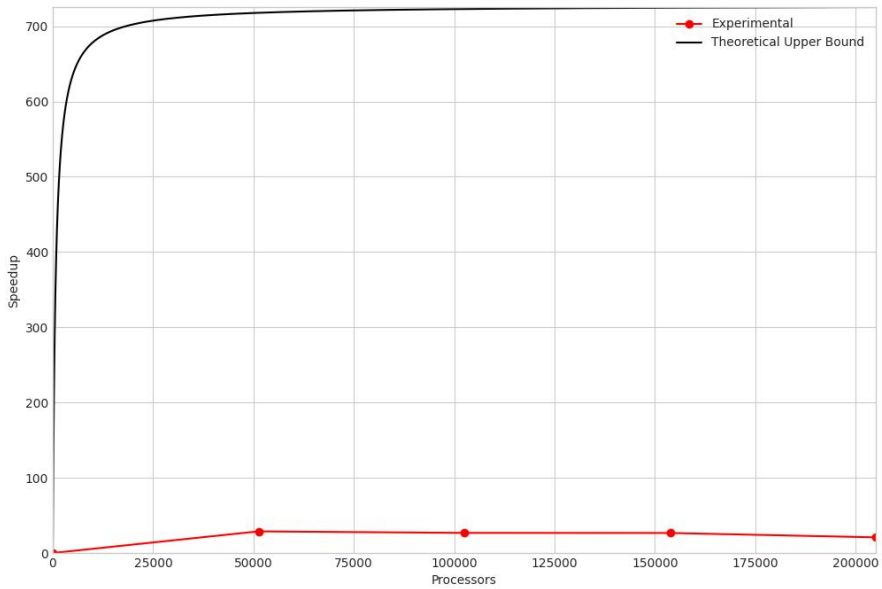
3.3.9.2 Optimization 1

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	100352.000	0.008	no data	25.458	no data	0.029	25.479	0.008	99.000	25.493	1.000	1.000
Parallel	51250	100352.000	0.042	0.001	0.755	0.029	0.030	0.818	0.042	97.000	0.886	28.786	0.001
Parallel	102500	100352.000	0.044	0.001	0.829	0.030	0.030	0.890	0.046	97.600	0.960	26.566	0.000
Parallel	153750	100352.000	0.041	0.001	0.828	0.030	0.030	0.889	0.044	97.200	0.960	26.555	0.000
Parallel	205000	100352.000	0.042	0.001	1.093	0.030	0.030	1.156	0.044	97.600	1.228	20.753	0.000



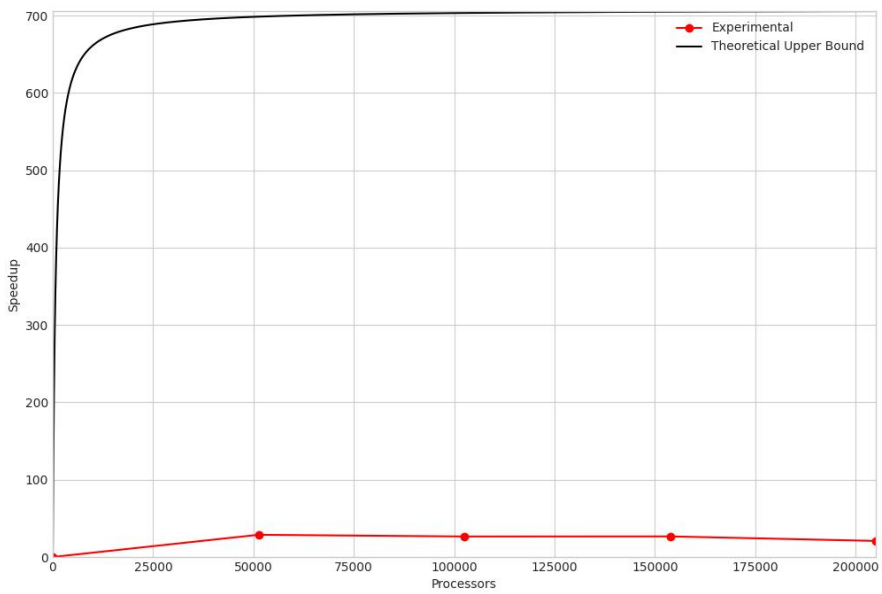
3.3.9.3 Optimization 2

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	100352.000	0.008	no data	25.469	no data	0.029	25.488	0.009	99.040	25.504	1.000	1.000
Parallel	51250	100352.000	0.042	0.001	0.755	0.029	0.030	0.815	0.046	97.080	0.886	28.773	0.001
Parallel	102500	100352.000	0.042	0.001	0.828	0.029	0.030	0.889	0.044	97.680	0.954	26.734	0.000
Parallel	153750	100352.000	0.042	0.001	0.829	0.029	0.030	0.889	0.046	97.480	0.958	26.611	0.000
Parallel	205000	100352.000	0.043	0.001	1.093	0.029	0.030	1.154	0.044	97.760	1.226	20.803	0.000



3.3.9.4 Optimization 3

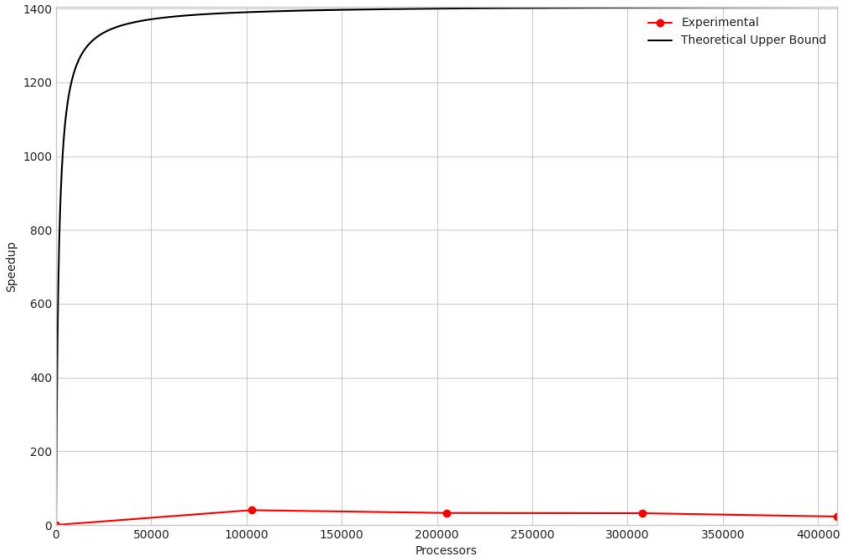
Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	100352.000	0.008	no data	25.455	no data	0.029	25.472	0.011	99.040	25.491	1.000	1.000
Parallel	51250	100352.000	0.043	0.001	0.754	0.030	0.030	0.816	0.046	96.960	0.888	28.719	0.001
Parallel	102500	100352.000	0.041	0.001	0.828	0.030	0.031	0.890	0.046	97.240	0.961	26.531	0.000
Parallel	153750	100352.000	0.043	0.001	0.828	0.030	0.031	0.889	0.048	97.520	0.957	26.631	0.000
Parallel	205000	100352.000	0.043	0.001	1.093	0.029	0.030	1.155	0.046	97.640	1.228	20.765	0.000



3.3.10 Tile-410000 graph

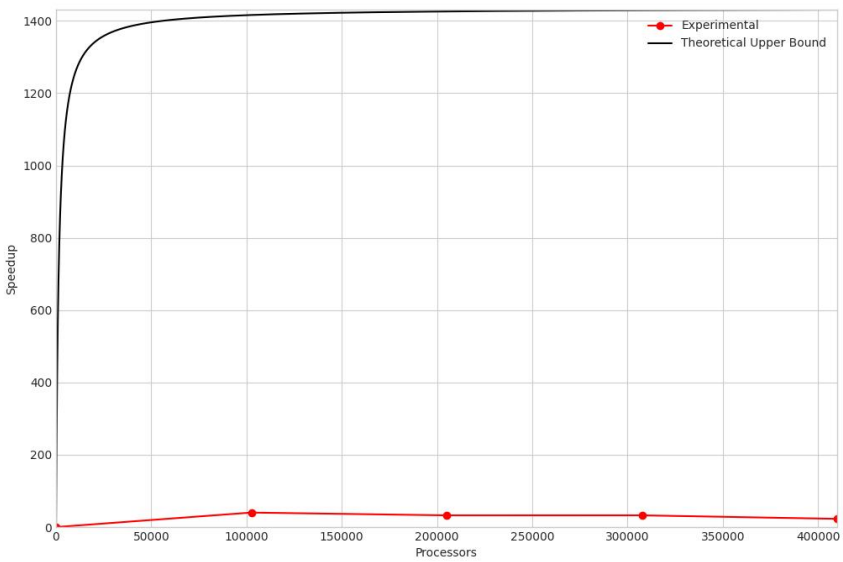
3.3.10.1 Optimization 0

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	200704.000	0.017	no data	101.611	no data	0.052	101.645	0.027	99.000	101.683	1.000	1.000
Parallel	102500	200704.000	0.051	0.001	2.329	0.056	0.053	2.443	0.053	98.440	2.531	40.178	0.000
Parallel	205000	200704.000	0.051	0.001	2.935	0.056	0.053	3.049	0.052	98.920	3.129	32.495	0.000
Parallel	307500	200704.000	0.052	0.001	2.992	0.057	0.054	3.106	0.057	98.880	3.190	31.871	0.000
Parallel	410000	200704.000	0.054	0.001	4.264	0.059	0.055	4.380	0.060	99.000	4.468	22.758	0.000



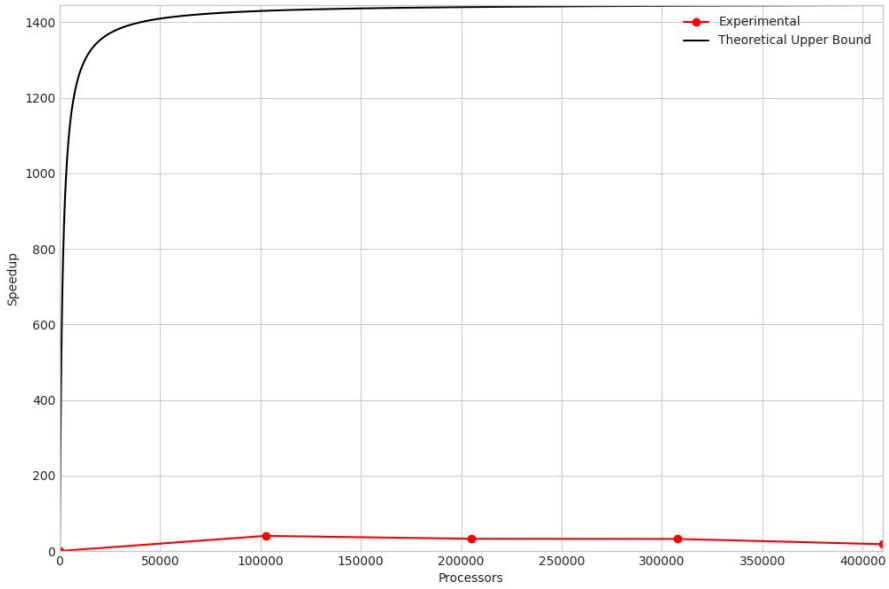
3.3.10.2 Optimization 1

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	200704.000	0.017	no data	101.562	no data	0.051	101.594	0.030	99.000	101.633	1.000	1.000
Parallel	102500	200704.000	0.051	0.001	2.330	0.056	0.053	2.444	0.054	98.520	2.531	40.152	0.000
Parallel	205000	200704.000	0.050	0.001	2.937	0.057	0.054	3.052	0.053	98.920	3.133	32.438	0.000
Parallel	307500	200704.000	0.050	0.001	2.937	0.056	0.053	3.050	0.053	98.920	3.130	32.467	0.000
Parallel	410000	200704.000	0.052	0.001	4.265	0.058	0.054	4.380	0.055	99.000	4.464	22.769	0.000



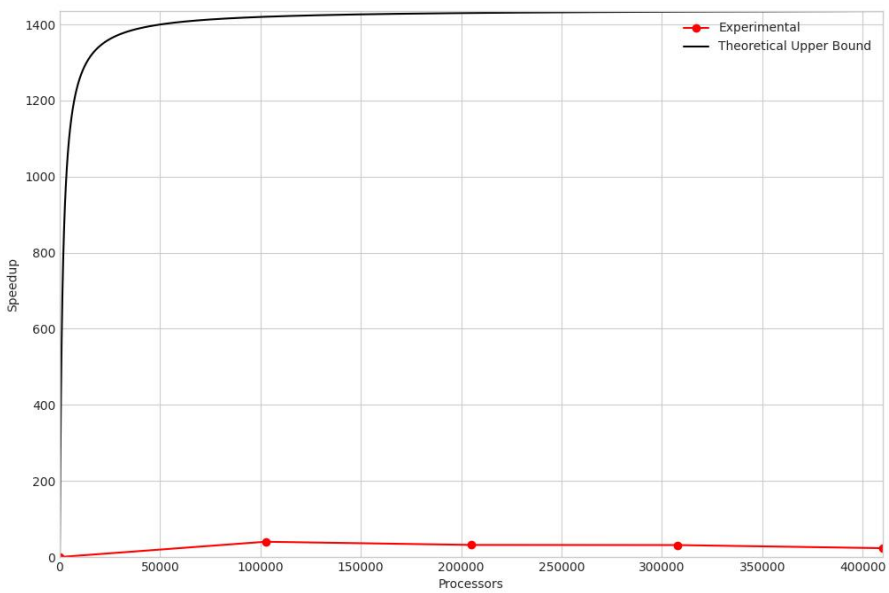
3.3.10.3 Optimization 2

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	200704.000	0.017	no data	101.449	no data	0.051	101.480	0.030	99.000	101.519	1.000	1.000
Parallel	102500	200704.000	0.051	0.001	2.327	0.057	0.054	2.437	0.057	98.440	2.531	40.113	0.000
Parallel	205000	200704.000	0.051	0.001	2.938	0.056	0.053	3.048	0.056	99.000	3.132	32.409	0.000
Parallel	307500	200704.000	0.052	0.001	2.978	0.057	0.054	3.092	0.055	98.880	3.176	31.969	0.000
Parallel	410000	200704.000	0.051	0.001	5.450	0.057	0.054	5.563	0.056	99.000	5.645	17.985	0.000



3.3.10.4 Optimization 3

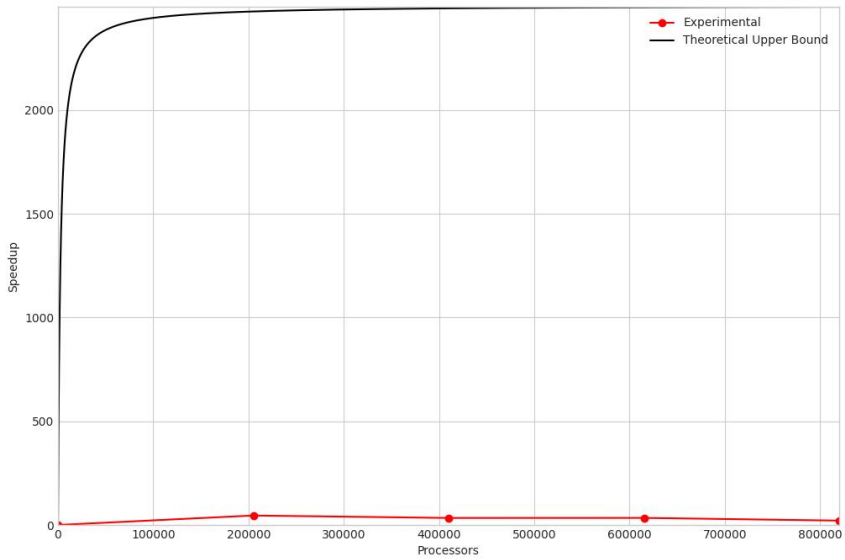
Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	200704.000	0.017	no data	101.432	no data	0.051	101.459	0.034	99.000	101.502	1.000	1.000
Parallel	102500	200704.000	0.051	0.002	2.328	0.057	0.054	2.442	0.055	98.440	2.530	40.113	0.000
Parallel	205000	200704.000	0.051	0.001	2.999	0.057	0.054	3.109	0.058	98.800	3.197	31.751	0.000
Parallel	307500	200704.000	0.052	0.001	3.035	0.057	0.054	3.151	0.053	98.880	3.230	31.425	0.000
Parallel	410000	200704.000	0.054	0.001	4.142	0.070	0.063	4.279	0.058	98.960	4.365	23.253	0.000



3.3.11 Tile-820000 graph

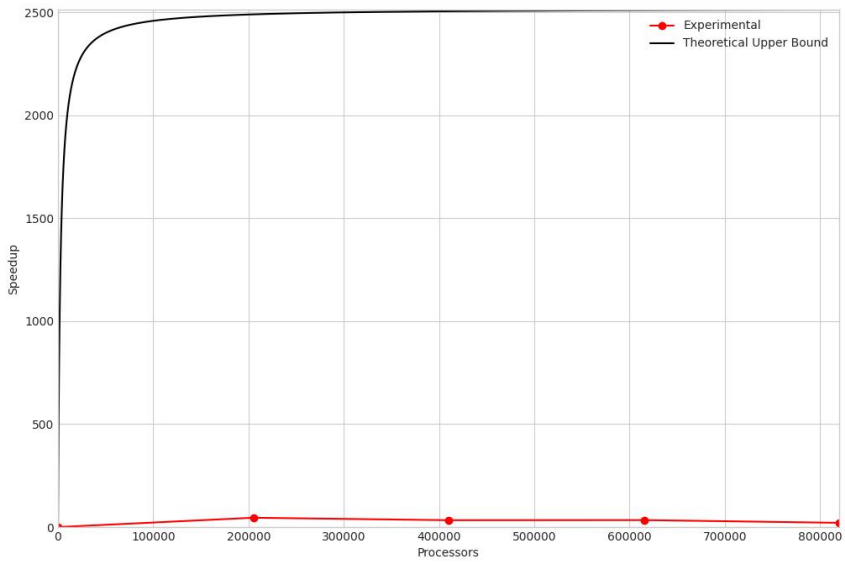
3.3.11.1 Optimization 0

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	401408.000	0.034	no data	403.937	no data	0.108	404.022	0.053	99.000	404.098	1.000	1.000
Parallel	205000	401408.000	0.081	0.003	8.418	0.188	0.150	8.758	0.093	99.000	8.874	45.539	0.000
Parallel	410000	401408.000	0.076	0.003	11.537	0.166	0.140	11.843	0.088	99.000	11.955	33.802	0.000
Parallel	615000	401408.000	0.084	0.003	11.411	0.189	0.157	11.756	0.097	99.000	11.881	34.013	0.000
Parallel	820000	401408.000	0.074	0.003	18.820	0.138	0.129	19.090	0.080	99.000	19.199	21.048	0.000



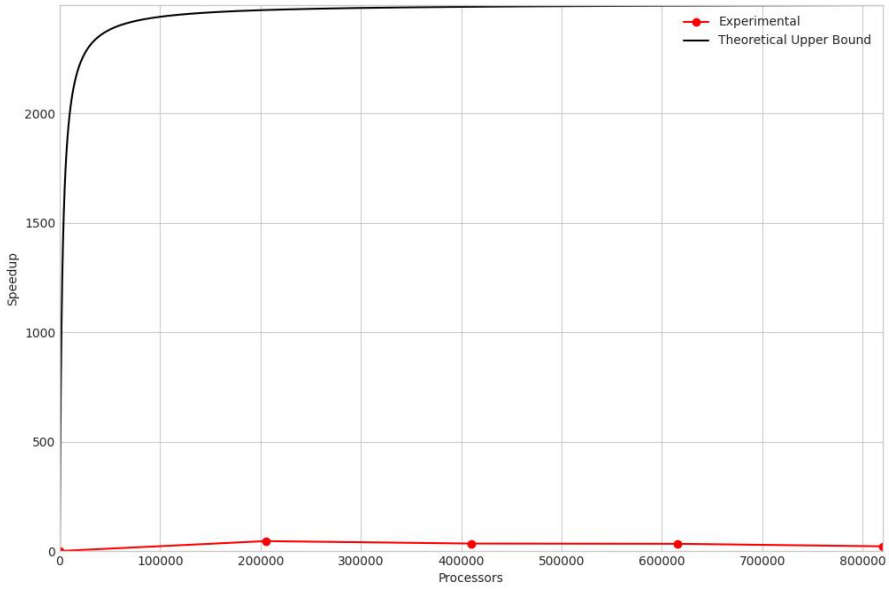
3.3.11.2 Optimization 1

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	401408.000	0.034	no data	404.900	no data	0.108	405.004	0.034	99.000	405.061	1.000	1.000
Parallel	205000	401408.000	0.078	0.003	8.514	0.160	0.141	8.818	0.088	99.000	8.934	45.341	0.000
Parallel	410000	401408.000	0.072	0.003	11.763	0.134	0.125	12.023	0.083	99.000	12.131	33.391	0.000
Parallel	615000	401408.000	0.077	0.003	11.516	0.169	0.141	11.827	0.088	99.000	11.936	33.935	0.000
Parallel	820000	401408.000	0.073	0.003	19.186	0.138	0.130	19.452	0.086	99.000	19.578	20.690	0.000



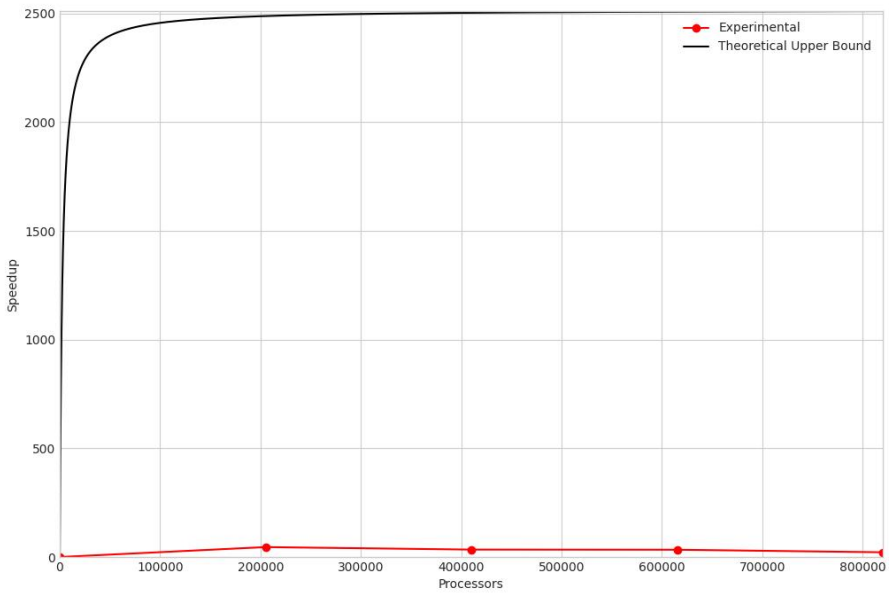
3.3.11.3 Optimization 2

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	401408.000	0.034	no data	404.260	no data	0.109	404.362	0.036	99.000	404.422	1.000	1.000
Parallel	205000	401408.000	0.080	0.003	8.490	0.180	0.150	8.823	0.091	99.000	8.939	45.241	0.000
Parallel	410000	401408.000	0.079	0.003	11.414	0.186	0.149	11.745	0.096	99.000	11.865	34.086	0.000
Parallel	615000	401408.000	0.073	0.003	11.874	0.148	0.132	12.154	0.083	99.000	12.261	32.984	0.000
Parallel	820000	401408.000	0.073	0.003	18.503	0.133	0.127	18.764	0.080	99.000	18.882	21.419	0.000



3.3.11.4 Optimization 3

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	401408.000	0.034	no data	404.698	no data	0.109	404.793	0.042	99.000	404.859	1.000	1.000
Parallel	205000	401408.000	0.080	0.003	8.421	0.188	0.153	8.760	0.096	99.000	8.879	45.598	0.000
Parallel	410000	401408.000	0.078	0.003	11.406	0.186	0.148	11.738	0.094	99.000	11.862	34.131	0.000
Parallel	615000	401408.000	0.078	0.003	11.652	0.176	0.144	11.974	0.091	99.000	12.087	33.496	0.000
Parallel	820000	401408.000	0.078	0.003	18.117	0.179	0.149	18.445	0.092	99.000	18.573	21.798	0.000



3.4 Considerations

Differently from the MPI implementation, with CUDA we managed to achieve a speedup which is greater than 1. That is because we are parallelizing the graph preprocessing, and, as a reference for calculating the speedup, we are using a version of the sequential algorithm which also performs the said preprocessing.

CHAPTER 4

HYBRID CUDA AND MPI IMPLEMENTATION

In this chapter we will discuss about our hybrid implementation of Tarjan's algorithm accelerated by CUDA (`cuda_mpi`) and present the experimental results that this algorithm achieved.

4.1 Algorithm idea

The algorithm is identical to the one described in 2.1, except that the master node executes the graph preprocesing with CUDA as described in 3.1.4. There are several reasons why we decided not to execute graph preprocessing on each node:

- The graph chunks assigned to the slave nodes are small. The overhead of using CUDA on such small graphs would hurt the algorithm more than it would benefit it.
- The `cuda_graph` representation described in 3.1.3 requires that the nodes are numbered from 0 to $N-1$. This is only true for the original graph on the master node, because later each slave processes subgraphs with arbitrary node IDs.

4.2 Experimental results

In this subsection are presented the results of the parallel version of Tarjan's algorithm with hybrid CUDA and MPI. Each graph is tested with all the compiler optimization (O0, O1, O2, O3) and with 2, 4 and 8 processors MPI, and the number of CUDA threads as number of nodes in the graph. In the next pages for each graph, set the optimization, there are a table and a graph of the speedup. In the table there are:

- **Version**: is the version of the program. It can be "Serial" if it's the sequential with preprocessing version of the program, or it can be "Parallel" if it's the parallel version with CUDA.

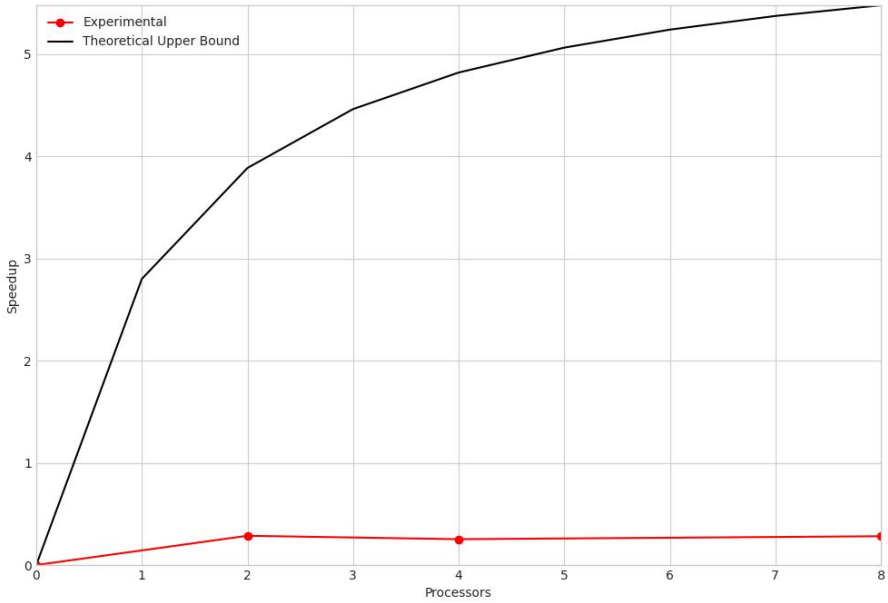
- **Threads**: is the number of processors. For the "Serial" version, the value of this field is 1, because it's the sequential with preprocessing version of the program executed on only one processor of the CPU.
- **vertices**: is the number of vertices of the graph.
- **vertices_after**: is the number of vertices of the graph after the preprocessing.
- **init**: is the time spent to initialize the program, in particular it's the time spent to create and initialize the data structures, and to read the graph from a file.
- **tarjan**: is the time spent to execute Tarjan's algorithm. In the "Parallel" version is the time it takes to run Tarjan's algorithm on each MPI slave node, along with the time it takes to partition the graph into chunks, exchange messages between processes, store found SCCs, and create supernodes.
- **split**: is the time spent to split the graph in the "Parallel" version. This phase is not provided by the sequential algorithm, in fact this field for this version is reported with the value "no data".
- **merge**: is the time spent to merge the SCCs found. This phase involves merging the nodes of an SCC found by the Tarjan's algorithm into a supernode, and storing the SCC found in a dedicated data structure. This phase is not provided by the sequential algorithm, in fact this field for this version is reported with the value "no data".
- **total_only_mpi**: is the time taken by the hybrid CUDA and MPI algorithm to execute the sequential Tarjan's algorithm on each MPI slave node.
- **preprocess**: is the time spent to do the preprocessing of the graph with CUDA.
- **conversion**: is the time spent to convert the graph into a graph adapted for CUDA.
- **user**: is the total number of CPU-seconds that the process used directly (in user mode), in seconds.
- **system**: is the total number of CPU-seconds used by the system on behalf of the process (in kernel mode), in seconds.
- **pCPU**: is the percentage of the CPU that this process got. This is just user + system times divided by the total running time.
- **elapsed**: is the elapsed real (wall clock) time used by the process, in seconds.
- **Speedup**: is the speedup of the program.
- **Efficiency**: is the efficiency of the program.

All values except those in the `Version`, `Threads`, `Speedup`, and `Efficiency` columns are means. These means were calculated from the files of the measurements taken. `Speedup`, and `Efficiency` are calculated with the formulas in Cap.1 using the results of these means. For the sequential algorithm, the `finalize` and `conversion` fields were not measured.

4.2.1 Fully connected graph

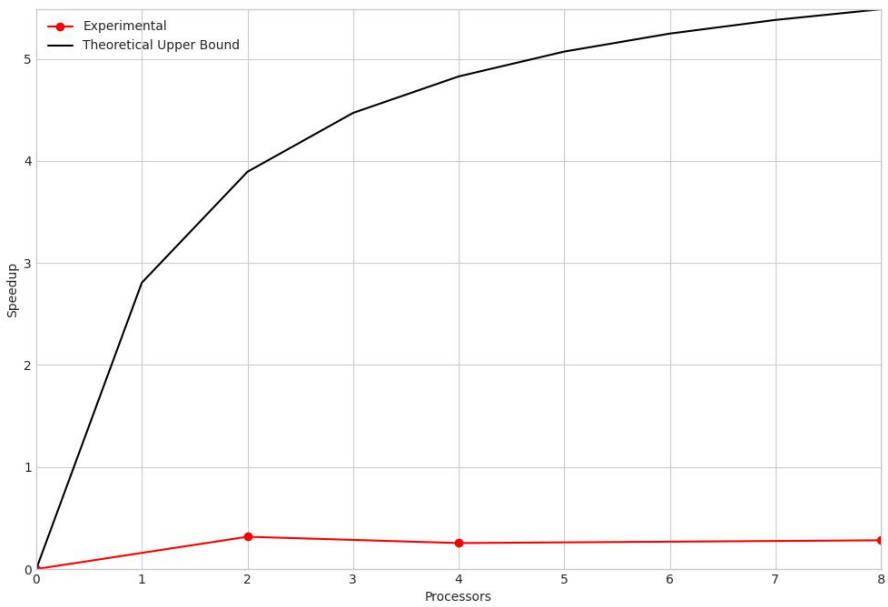
4.2.1.1 Optimization 0

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	12500.000	12500.000	2.231	2.940	no data	no data	no data	9.478	no data	no data	13.709	1.013	99.040	14.730	1.000	1.000
Parallel	2	12500.000	12500.000	2.626	32.709	0.751	20.487	32.709	0.002	13.766	0.023	96.439	2.066	190.560	51.417	0.286	0.143
Parallel	4	12500.000	12500.000	2.759	37.197	1.092	25.644	37.197	0.002	15.979	0.024	87.670	24.644	191.880	58.267	0.253	0.063
Parallel	8	12500.000	12500.000	2.638	32.626	1.062	20.456	32.626	0.002	14.211	0.023	108.475	40.306	283.040	52.240	0.282	0.035



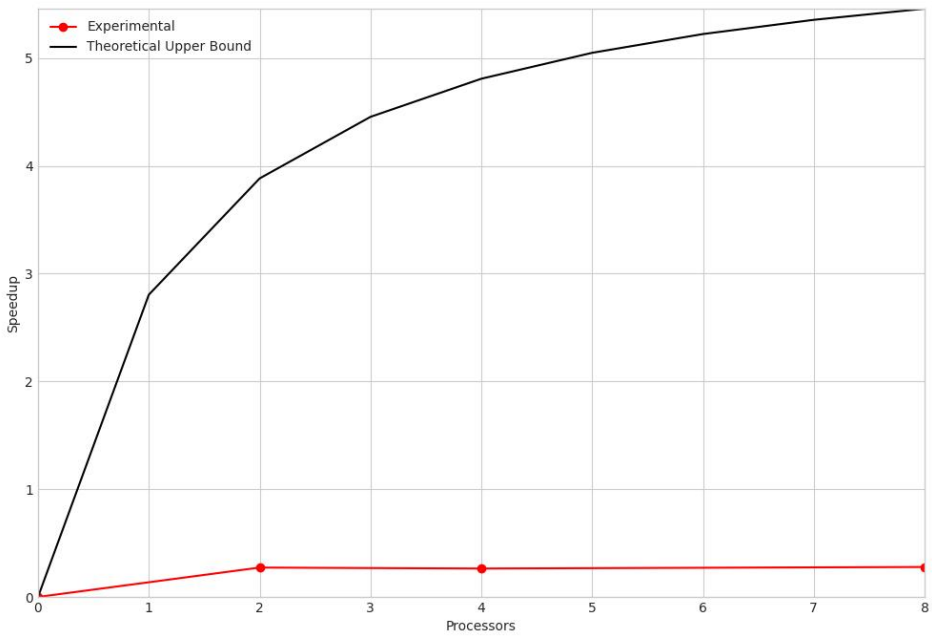
4.2.1.2 Optimization 1

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	12500.000	12500.000	2.237	2.942	no data	no data	no data	9.523	no data	no data	13.772	1.002	99.000	14.782	1.000	1.000
Parallel	2	12500.000	12500.000	2.540	29.787	0.685	16.110	29.787	0.001	12.093	0.022	87.141	2.002	189.920	46.725	0.316	0.158
Parallel	4	12500.000	12500.000	2.728	36.891	1.061	25.076	36.892	0.002	16.008	0.023	86.983	24.594	191.080	58.041	0.255	0.064
Parallel	8	12500.000	12500.000	2.654	32.594	1.052	20.480	32.594	0.002	14.508	0.023	109.675	40.664	284.920	52.487	0.282	0.035



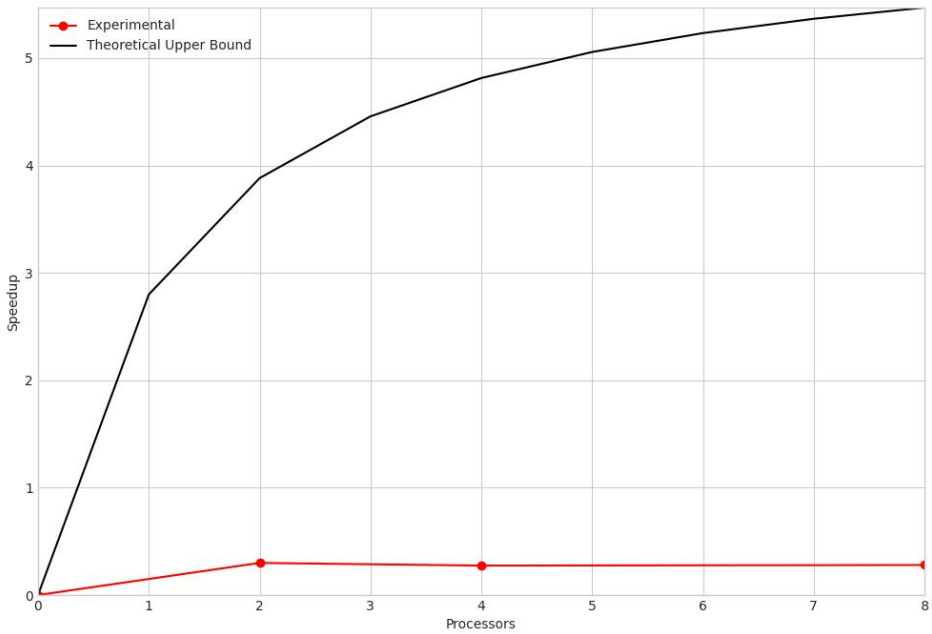
4.2.1.3 Optimization 2

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	12500.000	12500.000	2.236	2.911	no data	no data	no data	9.454	no data	no data	13.648	1.026	99.040	14.681	1.000	1.000
Parallel	2	12500.000	12500.000	2.653	34.390	0.768	22.348	34.390	0.002	14.496	0.023	101.318	2.068	191.080	53.834	0.273	0.136
Parallel	4	12500.000	12500.000	2.600	31.781	0.973	19.614	31.782	0.002	13.602	0.022	74.786	21.508	184.440	55.705	0.264	0.066
Parallel	8	12500.000	12500.000	2.673	33.683	1.032	19.590	33.683	0.002	14.105	0.023	110.227	41.531	286.840	52.766	0.278	0.035



4.2.1.4 Optimization 3

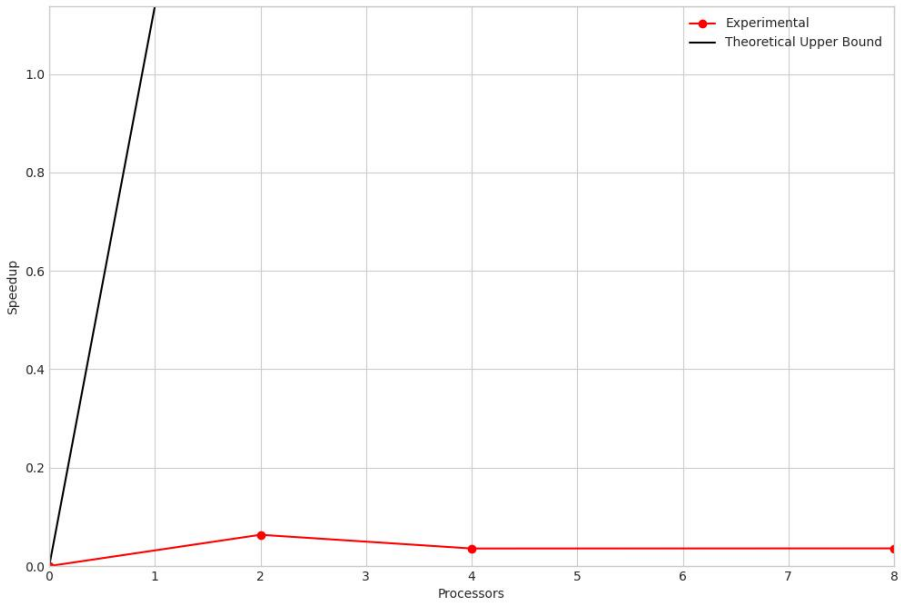
Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	12500.000	12500.000	2.231	2.934	no data	no data	no data	9.460	no data	no data	13.706	0.994	99.000	14.708	1.000	1.000
Parallel	2	12500.000	12500.000	2.527	31.823	0.707	17.738	31.823	0.002	12.411	0.022	91.772	2.059	190.080	49.122	0.299	0.150
Parallel	4	12500.000	12500.000	2.651	33.440	1.003	21.434	33.440	0.002	14.395	0.023	79.367	23.245	190.400	53.592	0.274	0.069
Parallel	8	12500.000	12500.000	2.667	32.555	1.049	20.455	32.555	0.001	14.513	0.022	109.590	40.632	284.200	52.545	0.280	0.035



4.2.2 Fully disconnected graph

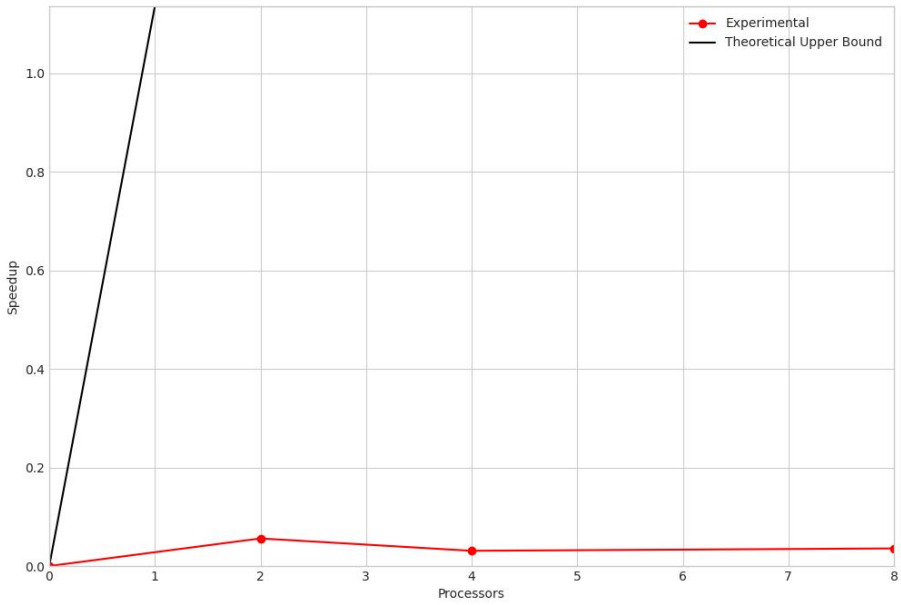
4.2.2.1 Optimization 0

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	1000000.000	0.000	0.026	0.000	no data	no data	no data	0.005	no data	no data	0.024	0.000	97.480	0.030	1.000	1.000
Parallel	2	1000000.000	0.000	0.089	0.000	0.000	0.000	0.000	0.002	0.001	0.001	0.179	0.092	65.720	0.472	0.064	0.032
Parallel	4	1000000.000	0.000	0.079	0.000	0.000	0.000	0.000	0.002	0.001	0.000	0.154	0.132	37.240	0.841	0.036	0.009
Parallel	8	1000000.000	0.000	0.074	0.000	0.000	0.000	0.000	0.002	0.001	0.000	0.268	0.168	56.240	0.836	0.036	0.004



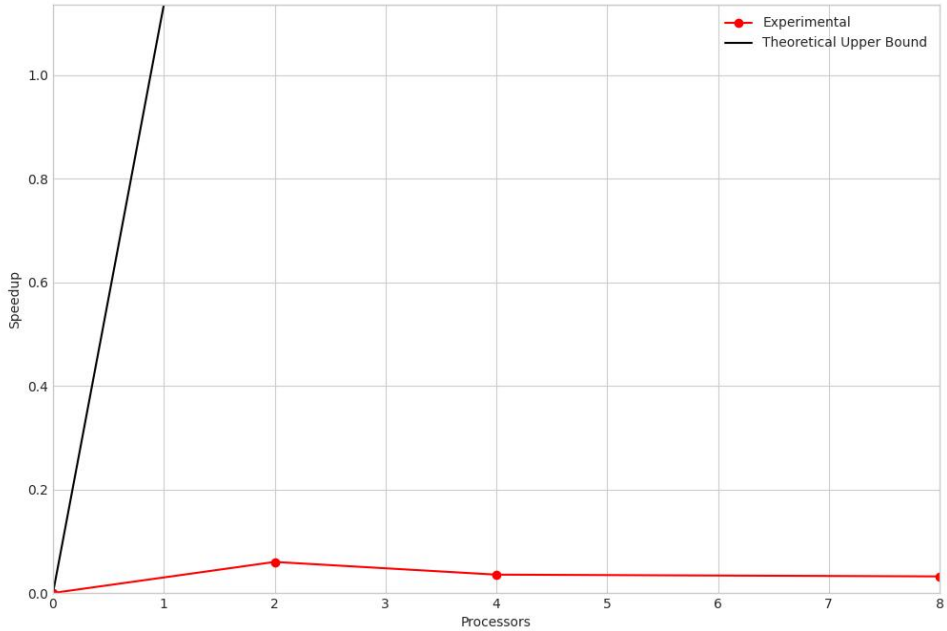
4.2.2.2 Optimization 1

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	1000000.000	0.000	0.026	0.000	no data	no data	no data	0.005	no data	no data	0.023	0.000	97.320	0.030	1.000	1.000
Parallel	2	1000000.000	0.000	0.084	0.000	0.000	0.000	0.000	0.002	0.001	0.000	0.169	0.088	62.600	0.536	0.056	0.028
Parallel	4	1000000.000	0.000	0.089	0.000	0.000	0.000	0.000	0.002	0.001	0.000	0.166	0.143	37.200	0.970	0.031	0.008
Parallel	8	1000000.000	0.000	0.074	0.000	0.000	0.000	0.000	0.002	0.001	0.000	0.282	0.159	58.000	0.840	0.036	0.004



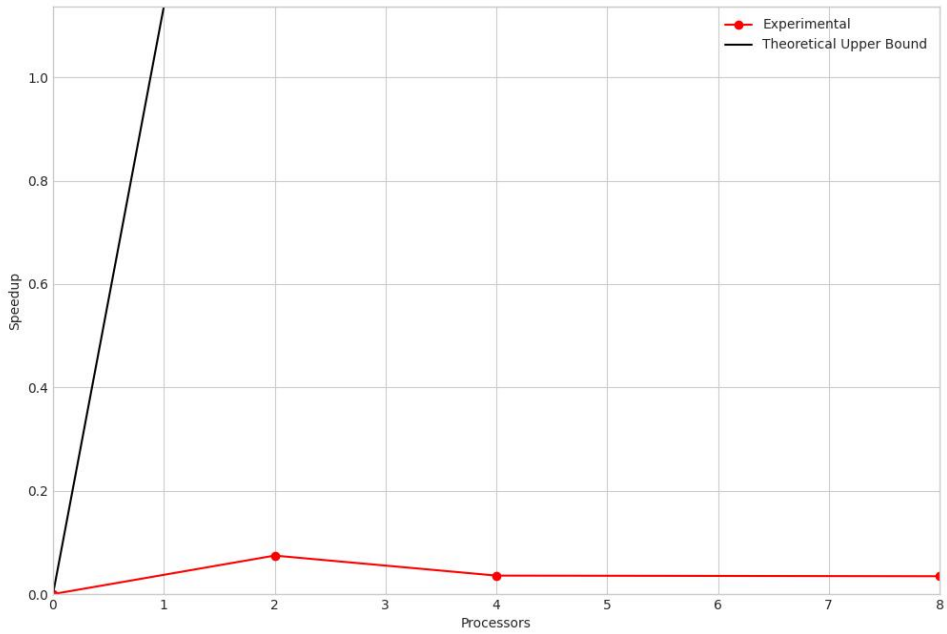
4.2.2.3 Optimization 2

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	1000000.000	0.000	0.026	0.000	no data	no data	no data	0.005	no data	no data	0.026	0.000	98.240	0.030	1.000	1.000
Parallel	2	1000000.000	0.000	0.078	0.000	0.000	0.000	0.000	0.002	0.001	0.000	0.162	0.081	61.440	0.500	0.060	0.030
Parallel	4	1000000.000	0.000	0.085	0.000	0.000	0.000	0.000	0.002	0.001	0.000	0.174	0.125	38.880	0.848	0.035	0.009
Parallel	8	1000000.000	0.000	0.074	0.000	0.000	0.000	0.000	0.002	0.001	0.000	0.310	0.166	58.200	0.938	0.032	0.004



4.2.2.4 Optimization 3

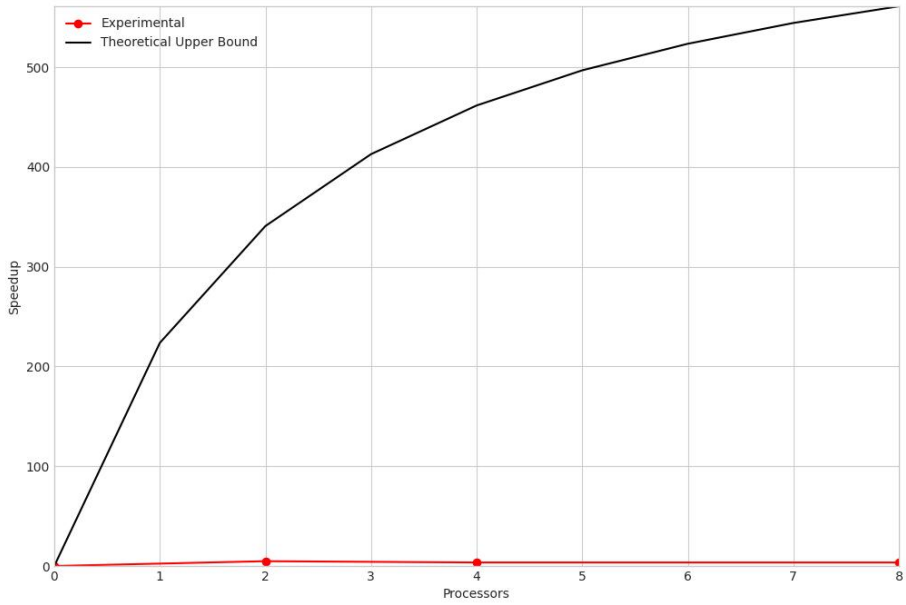
Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	1000000.000	0.000	0.026	0.000	no data	no data	no data	0.005	no data	no data	0.023	0.000	97.440	0.030	1.000	1.000
Parallel	2	1000000.000	0.000	0.071	0.000	0.000	0.000	0.000	0.002	0.001	0.000	0.150	0.080	59.520	0.403	0.074	0.037
Parallel	4	1000000.000	0.000	0.080	0.000	0.000	0.000	0.000	0.002	0.001	0.000	0.159	0.132	38.120	0.839	0.036	0.009
Parallel	8	1000000.000	0.000	0.068	0.000	0.000	0.000	0.000	0.002	0.001	0.000	0.262	0.156	55.200	0.869	0.035	0.004



4.2.3 Random-150000 graph

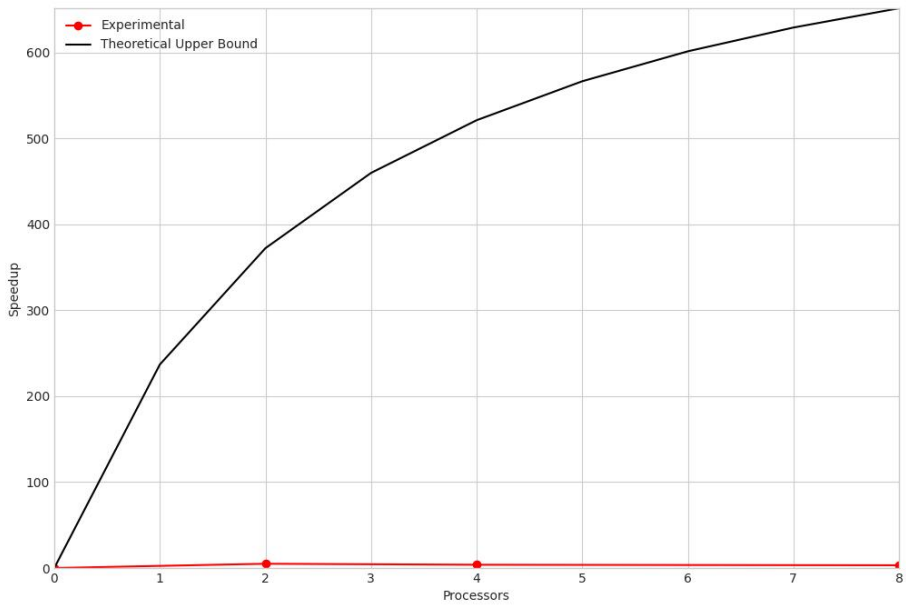
4.2.3.1 Optimization 0

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	150000.000	71094.000	0.006	0.036	no data	no data	no data	11.645	no data	no data	11.676	0.003	99.120	11.687	1.000	1.000
Parallel	2	150000.000	71094.000	0.057	0.538	0.026	0.000	0.539	1.101	0.031	0.000	3.424	0.116	156.680	2.405	4.860	2.430
Parallel	4	150000.000	71094.000	0.060	0.647	0.024	0.000	0.647	1.107	0.030	0.000	2.965	0.895	121.760	3.288	3.555	0.889
Parallel	8	150000.000	71094.000	0.058	0.607	0.024	0.000	0.607	1.113	0.031	0.000	4.459	1.638	188.920	3.313	3.528	0.441



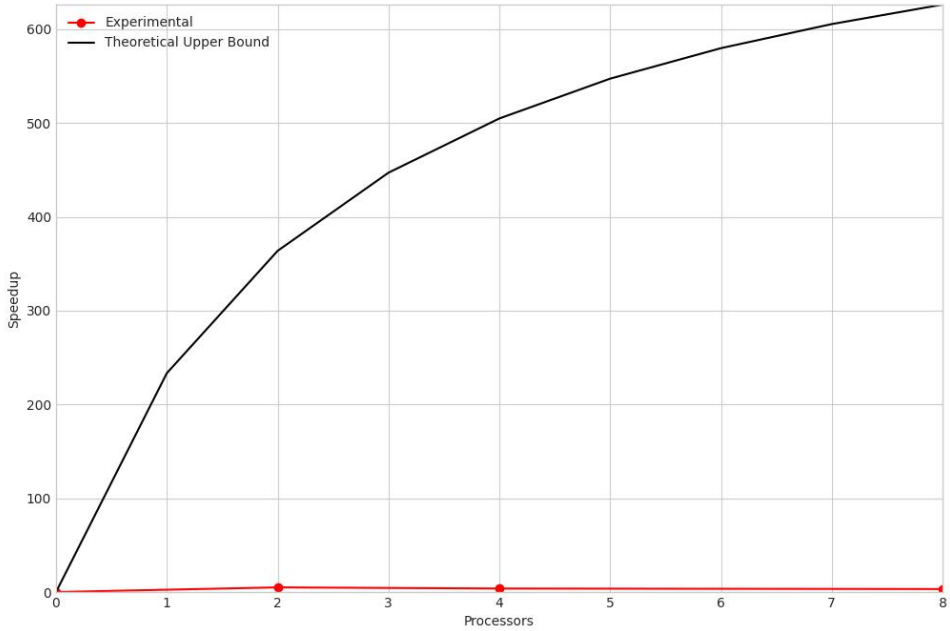
4.2.3.2 Optimization 1

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	150000.000	71094.000	0.006	0.036	no data	no data	no data	11.675	no data	no data	11.704	0.005	99.200	11.714	1.000	1.000
Parallel	2	150000.000	71094.000	0.059	0.552	0.026	0.000	0.552	1.099	0.031	0.000	3.444	0.128	159.760	2.356	4.973	2.487
Parallel	4	150000.000	71094.000	0.063	0.525	0.024	0.000	0.525	1.105	0.032	0.000	2.778	0.807	122.600	3.073	3.812	0.953
Parallel	8	150000.000	71094.000	0.056	0.736	0.024	0.000	0.736	1.112	0.029	0.000	4.637	1.775	180.000	3.644	3.215	0.402



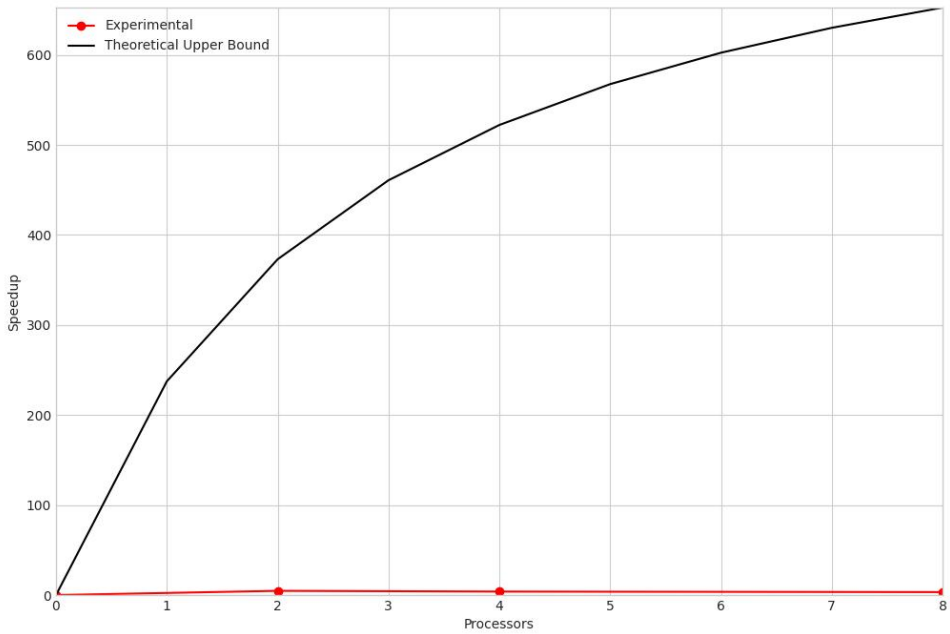
4.2.3.3 Optimization 2

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	150000.000	71094.000	0.006	0.036	no data	no data	no data	11.658	no data	no data	11.688	0.003	99.080	11.698	1.000	1.000
Parallel	2	150000.000	71094.000	0.054	0.543	0.026	0.000	0.543	1.106	0.030	0.000	3.436	0.116	159.840	2.342	4.994	2.497
Parallel	4	150000.000	71094.000	0.064	0.570	0.024	0.000	0.570	1.110	0.031	0.000	2.847	0.833	124.960	3.110	3.762	0.940
Parallel	8	150000.000	71094.000	0.069	0.702	0.025	0.000	0.702	1.105	0.033	0.000	4.649	1.728	181.480	3.631	3.222	0.403



4.2.3.4 Optimization 3

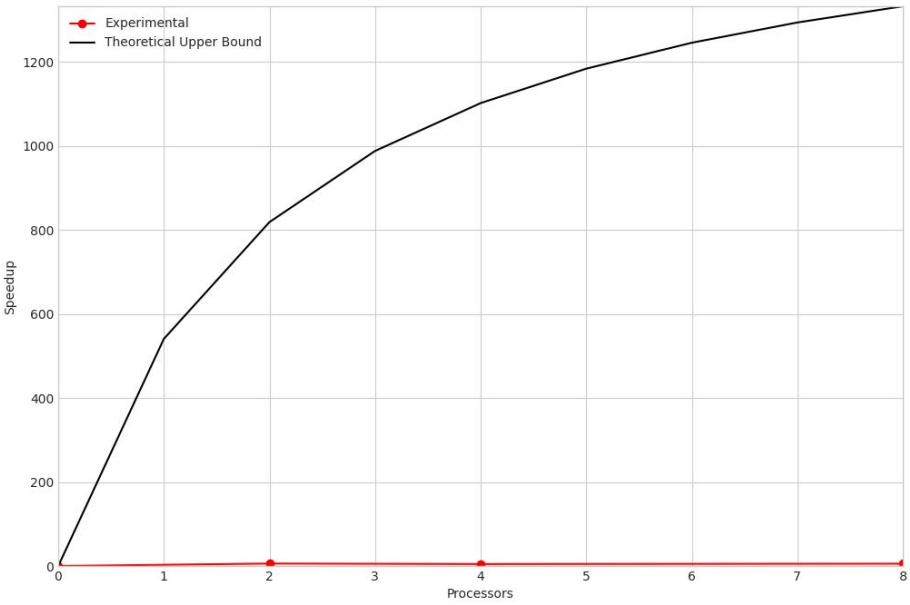
Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	150000.000	71094.000	0.006	0.036	no data	no data	no data	11.672	no data	no data	11.700	0.004	99.200	11.711	1.000	1.000
Parallel	2	150000.000	71094.000	0.054	0.525	0.026	0.000	0.525	1.102	0.030	0.000	3.387	0.121	150.400	2.493	4.697	2.349
Parallel	4	150000.000	71094.000	0.066	0.503	0.024	0.000	0.503	1.106	0.032	0.000	2.771	0.773	128.560	3.014	3.885	0.971
Parallel	8	150000.000	71094.000	0.068	0.629	0.025	0.000	0.629	1.110	0.032	0.000	4.438	1.656	176.720	3.548	3.300	0.413



4.2.4 Random-250000 graph

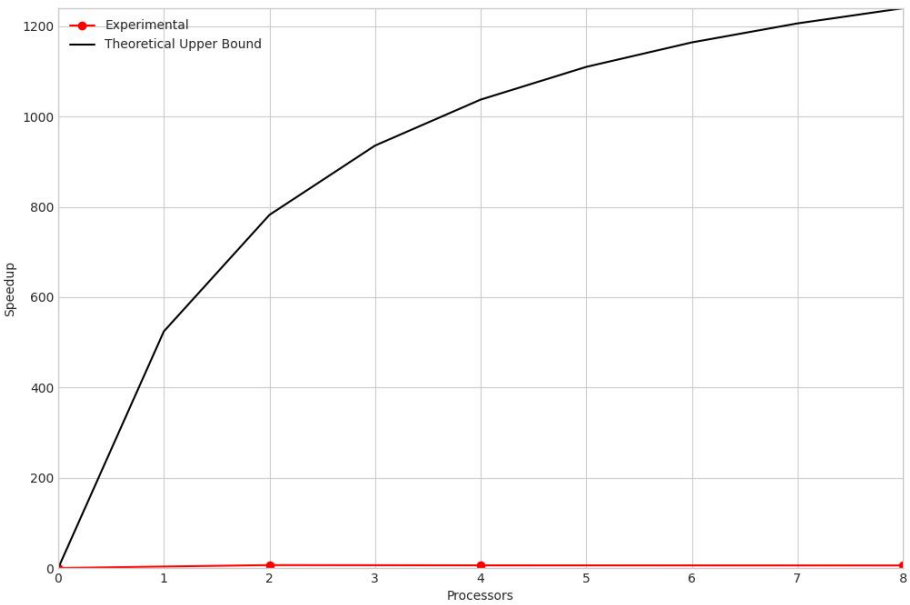
4.2.4.1 Optimization 0

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	250000.000	118498.000	0.010	0.041	no data	no data	no data	32.428	no data	no data	32.464	0.006	99.160	32.478	1.000	1.000
Parallel	2	250000.000	118498.000	0.062	0.794	0.057	0.000	0.794	3.538	0.052	0.000	8.828	0.156	167.880	5.550	5.852	2.926
Parallel	4	250000.000	118498.000	0.071	0.812	0.055	0.000	0.812	3.157	0.053	0.000	6.618	1.700	146.560	6.887	4.716	1.179
Parallel	8	250000.000	118498.000	0.069	0.941	0.055	0.000	0.941	3.146	0.052	0.000	9.888	3.436	228.200	5.908	5.498	0.687



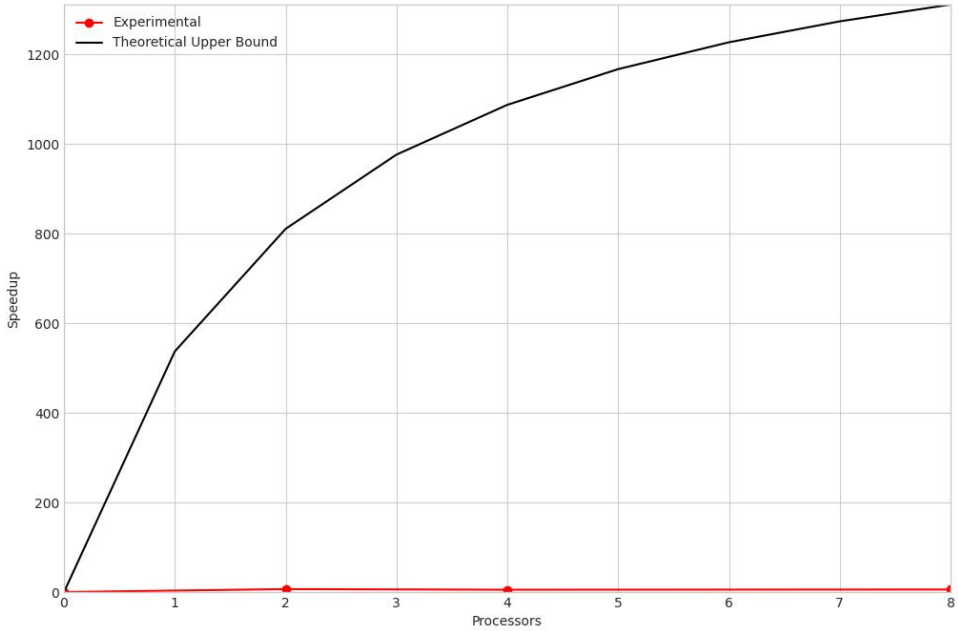
4.2.4.2 Optimization 1

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	250000.000	118498.000	0.010	0.041	no data	no data	no data	32.409	no data	no data	32.446	0.007	99.040	32.461	1.000	1.000
Parallel	2	250000.000	118498.000	0.068	0.798	0.058	0.000	0.798	3.138	0.057	0.000	8.052	0.164	170.720	4.979	6.519	3.260
Parallel	4	250000.000	118498.000	0.068	0.791	0.053	0.000	0.791	3.152	0.051	0.000	6.648	1.710	156.440	5.440	5.968	1.492
Parallel	8	250000.000	118498.000	0.065	0.796	0.053	0.000	0.796	3.143	0.050	0.000	9.512	3.227	227.320	5.688	5.707	0.713



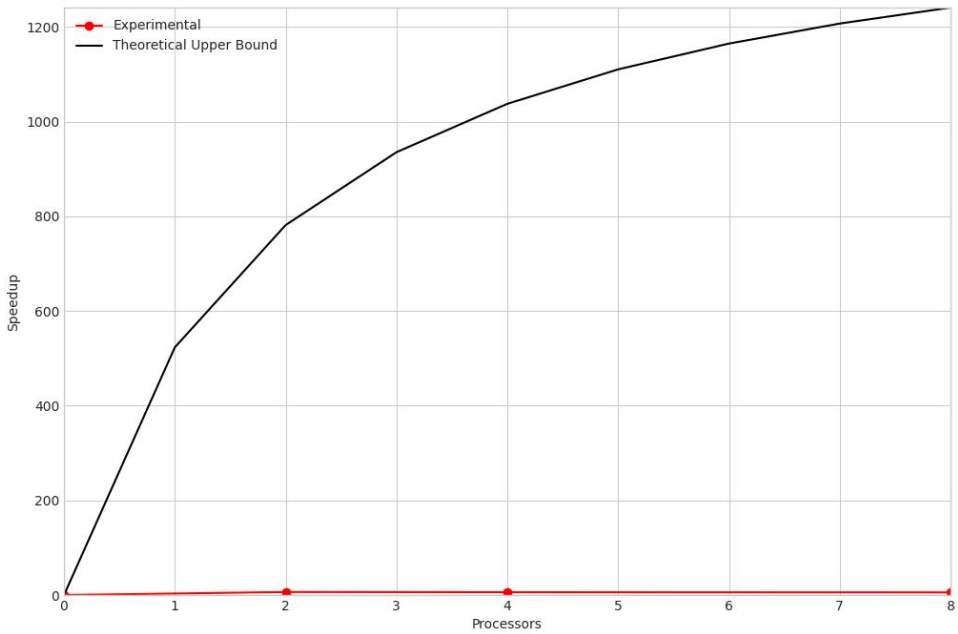
4.2.4.3 Optimization 2

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	250000.000	118498.000	0.010	0.041	no data	no data	no data	32.485	no data	no data	32.522	0.006	99.120	32.536	1.000	1.000
Parallel	2	250000.000	118498.000	0.057	0.807	0.058	0.000	0.807	3.052	0.058	0.000	7.890	0.145	170.040	4.922	6.610	3.305
Parallel	4	250000.000	118498.000	0.069	0.817	0.054	0.000	0.817	3.151	0.052	0.000	7.430	2.134	152.520	6.268	5.191	1.298
Parallel	8	250000.000	118498.000	0.069	0.807	0.054	0.000	0.807	3.144	0.052	0.000	9.595	3.255	228.000	5.717	5.691	0.711



4.2.4.4 Optimization 3

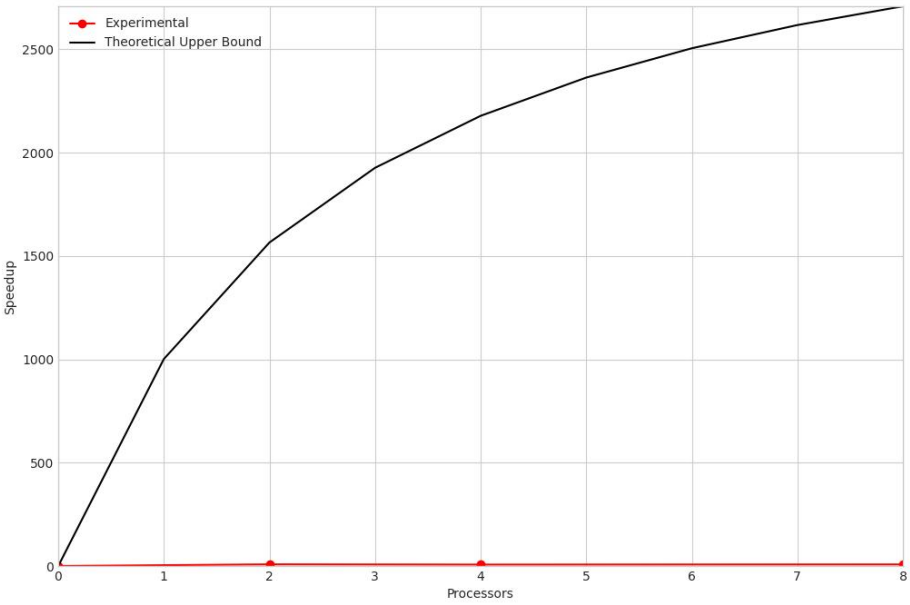
Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	250000.000	118498.000	0.010	0.041	no data	no data	no data	32.394	no data	no data	32.429	0.006	99.040	32.446	1.000	1.000
Parallel	2	250000.000	118498.000	0.062	0.789	0.058	0.000	0.789	3.056	0.058	0.000	7.868	0.153	167.160	5.004	6.483	3.242
Parallel	4	250000.000	118498.000	0.077	0.760	0.055	0.000	0.760	3.151	0.054	0.000	6.551	1.664	155.160	5.402	6.006	1.501
Parallel	8	250000.000	118498.000	0.067	0.752	0.053	0.000	0.753	3.145	0.052	0.000	9.530	3.220	225.000	5.766	5.627	0.703



4.2.5 Random-500000 graph

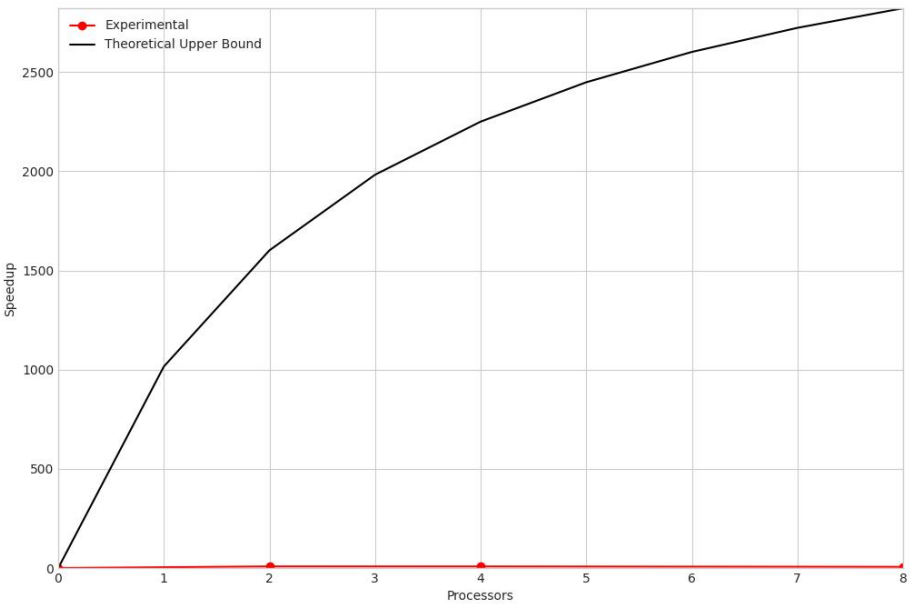
4.2.5.1 Optimization 0

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	500000.000	236518.000	0.020	0.092	no data	no data	no data	128.240	no data	no data	128.320	0.026	99.000	128.358	1.000	1.000
Parallel	2	500000.000	236518.000	0.079	1.963	0.145	0.000	1.963	12.299	0.162	0.000	28.839	0.261	184.520	15.776	8.136	4.068
Parallel	4	500000.000	236518.000	0.080	3.154	0.142	0.000	3.154	12.294	0.161	0.000	26.189	7.161	180.320	18.505	6.936	1.734
Parallel	8	500000.000	236518.000	0.083	1.541	0.141	0.000	1.541	12.405	0.117	0.000	33.152	10.570	264.400	16.555	7.753	0.969



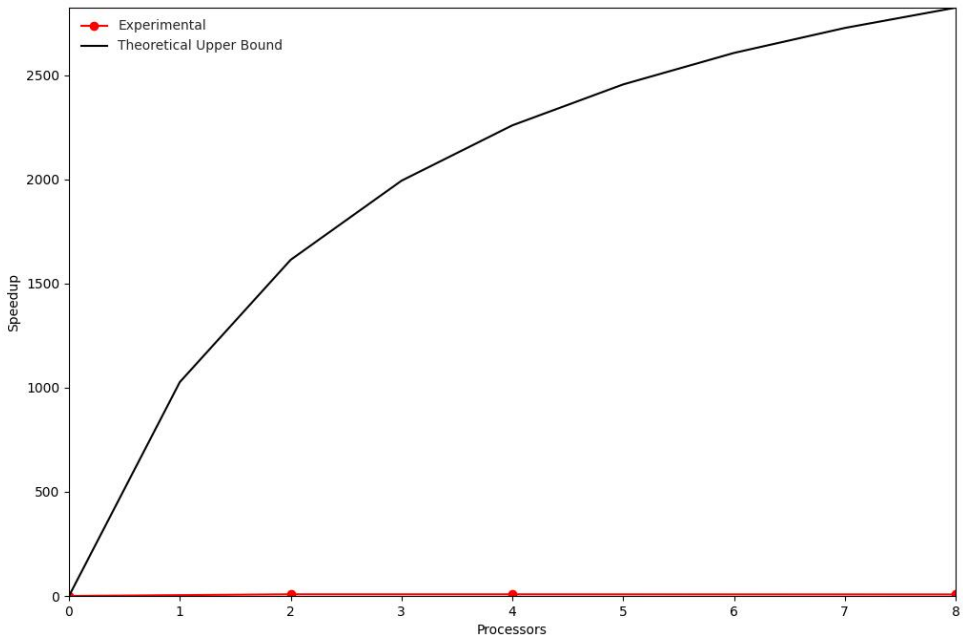
4.2.5.2 Optimization 1

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	500000.000	236518.000	0.020	0.093	no data	no data	no data	128.384	no data	no data	128.453	0.036	99.000	128.500	1.000	1.000
Parallel	2	500000.000	236518.000	0.070	2.070	0.138	0.000	2.071	12.299	0.158	0.000	29.027	0.264	185.080	15.839	8.113	4.056
Parallel	4	500000.000	236518.000	0.089	1.596	0.143	0.000	1.596	12.468	0.122	0.000	23.333	5.355	178.320	16.079	7.992	1.998
Parallel	8	500000.000	236518.000	0.086	1.612	0.139	0.000	1.612	12.451	0.121	0.000	40.378	13.993	258.360	21.480	5.982	0.748



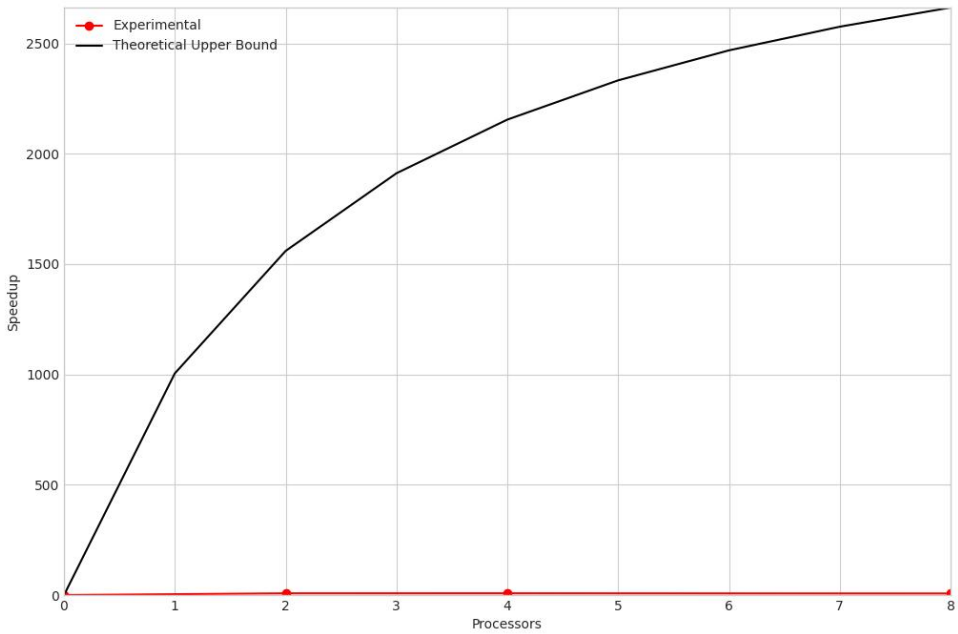
4.2.5.3 Optimization 2

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	500000.000	236518.000	0.020	0.091	no data	no data	no data	127.986	no data	no data	128.070	0.020	99.000	128.100	1.000	1.000
Parallel	2	500000.000	236518.000	0.080	2.086	0.148	0.000	2.086	12.287	0.162	0.001	29.058	0.274	183.440	15.987	8.013	4.006
Parallel	4	500000.000	236518.000	0.084	1.546	0.137	0.000	1.547	12.356	0.114	0.000	23.070	5.266	176.760	16.042	7.985	1.996
Parallel	8	500000.000	236518.000	0.070	2.170	0.130	0.000	2.170	12.766	0.105	0.000	34.882	11.195	260.640	17.620	7.270	0.909



4.2.5.4 Optimization 3

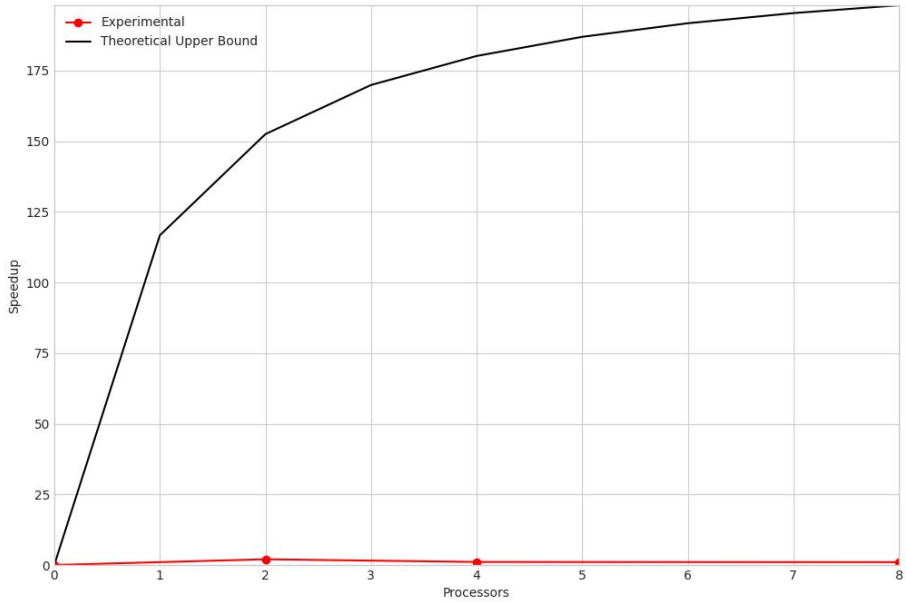
Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	500000.000	236518.000	0.020	0.091	no data	no data	no data	128.266	no data	no data	128.339	0.030	99.000	128.383	1.000	1.000
Parallel	2	500000.000	236518.000	0.072	2.113	0.139	0.000	2.113	12.297	0.158	0.000	29.112	0.265	183.280	16.040	8.004	4.002
Parallel	4	500000.000	236518.000	0.089	1.554	0.141	0.000	1.554	12.365	0.119	0.000	23.111	5.276	177.880	15.947	8.051	2.013
Parallel	8	500000.000	236518.000	0.074	1.755	0.132	0.000	1.755	12.451	0.112	0.000	35.264	11.575	262.000	17.890	7.176	0.897



4.2.6 Tile-52000 graph

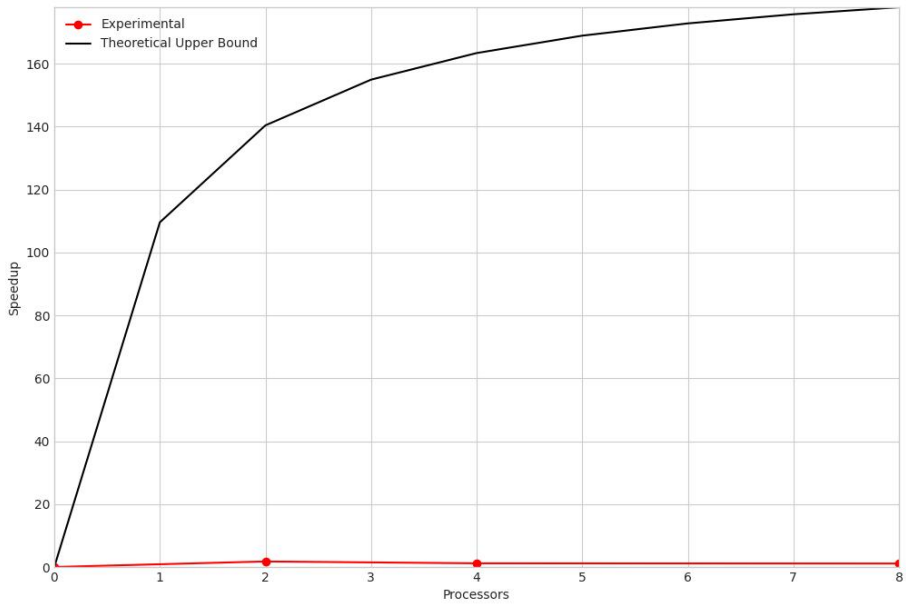
4.2.6.1 Optimization 0

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	51200.000	25088.000	0.002	0.007	no data	no data	no data	1.602	no data	no data	1.604	0.000	99.520	1.606	1.000	1.000
Parallel	2	51200.000	25088.000	0.052	0.057	0.003	0.006	0.058	0.114	0.008	0.000	0.458	0.089	89.120	0.788	2.037	1.019
Parallel	4	51200.000	25088.000	0.060	0.163	0.003	0.007	0.163	0.133	0.009	0.000	0.577	0.282	68.320	1.482	1.083	0.271
Parallel	8	51200.000	25088.000	0.060	0.146	0.003	0.006	0.147	0.133	0.009	0.000	0.968	0.469	102.120	1.582	1.015	0.127



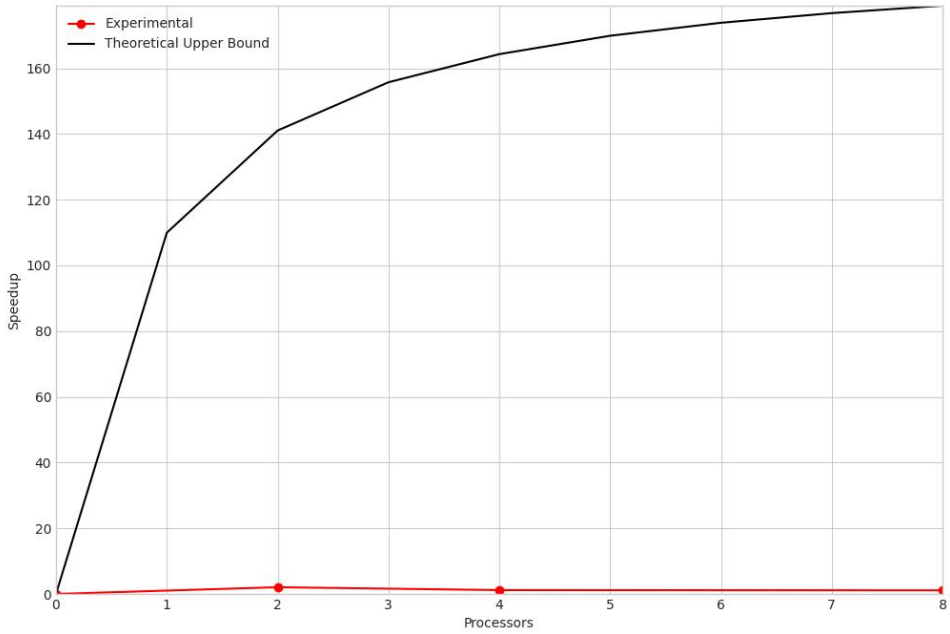
4.2.6.2 Optimization 1

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	51200.000	25088.000	0.002	0.006	no data	no data	no data	1.605	no data	no data	1.608	0.000	99.640	1.610	1.000	1.000
Parallel	2	51200.000	25088.000	0.050	0.060	0.003	0.006	0.060	0.110	0.009	0.000	0.453	0.091	82.160	0.910	1.769	0.885
Parallel	4	51200.000	25088.000	0.067	0.186	0.003	0.007	0.186	0.133	0.009	0.000	0.615	0.302	72.360	1.365	1.180	0.295
Parallel	8	51200.000	25088.000	0.055	0.199	0.003	0.006	0.200	0.133	0.008	0.000	1.062	0.526	113.120	1.451	1.109	0.139



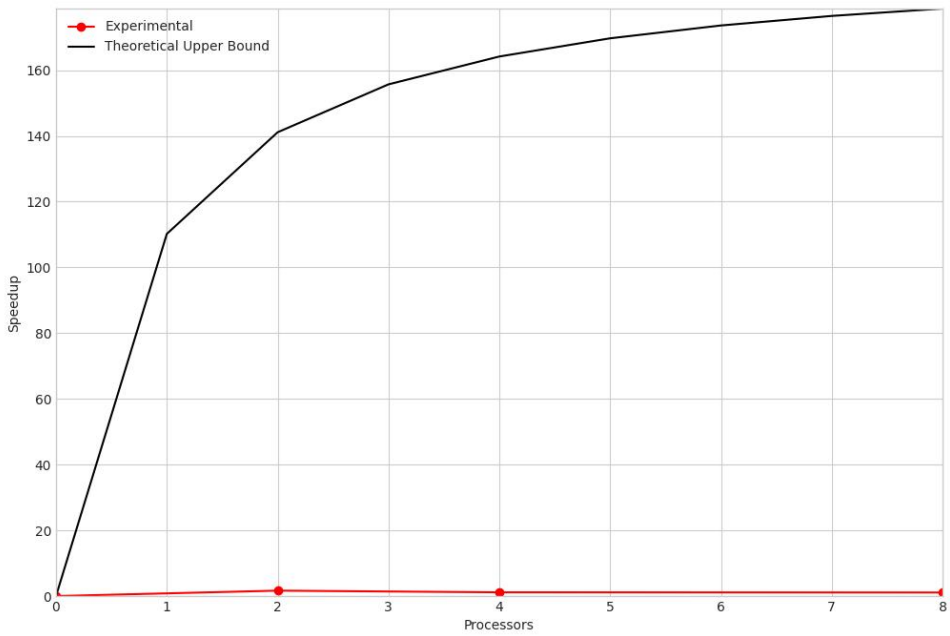
4.2.6.3 Optimization 2

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	51200.000	25088.000	0.002	0.006	no data	no data	no data	1.605	no data	no data	1.607	0.000	99.440	1.609	1.000	1.000
Parallel	2	51200.000	25088.000	0.045	0.058	0.003	0.006	0.058	0.109	0.008	0.000	0.446	0.082	86.720	0.784	2.053	1.026
Parallel	4	51200.000	25088.000	0.058	0.175	0.003	0.006	0.175	0.134	0.008	0.000	0.591	0.288	69.280	1.374	1.172	0.293
Parallel	8	51200.000	25088.000	0.048	0.136	0.003	0.006	0.136	0.133	0.008	0.000	0.909	0.448	103.240	1.452	1.109	0.139



4.2.6.4 Optimization 3

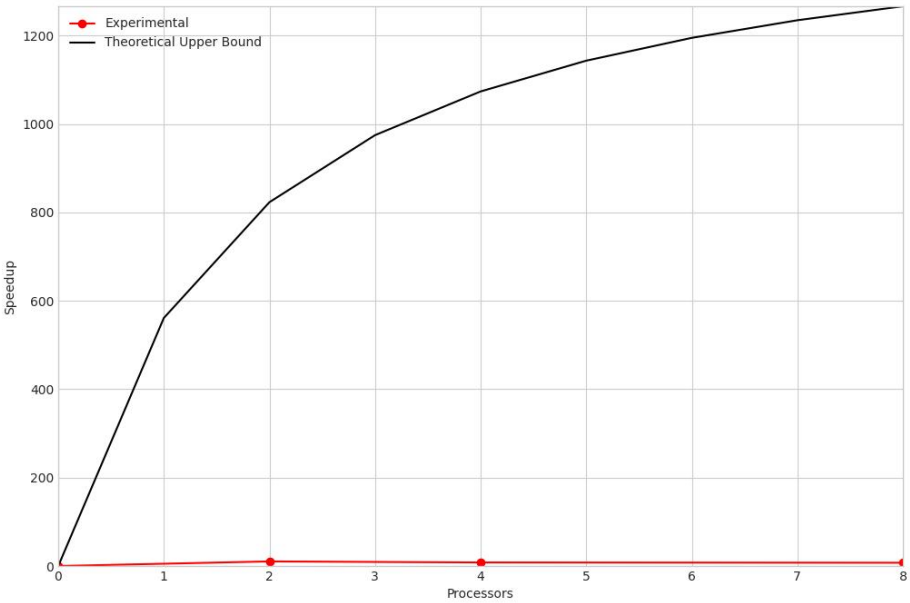
Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	51200.000	25088.000	0.002	0.006	no data	no data	no data	1.592	no data	no data	1.595	0.000	99.720	1.596	1.000	1.000
Parallel	2	51200.000	25088.000	0.054	0.058	0.003	0.006	0.059	0.111	0.009	0.000	0.460	0.091	85.200	0.967	1.651	0.826
Parallel	4	51200.000	25088.000	0.058	0.175	0.003	0.006	0.176	0.133	0.009	0.000	0.600	0.284	68.720	1.399	1.141	0.285
Parallel	8	51200.000	25088.000	0.049	0.222	0.003	0.006	0.223	0.133	0.008	0.000	1.116	0.544	115.640	1.473	1.084	0.135



4.2.7 Tile-205000 graph

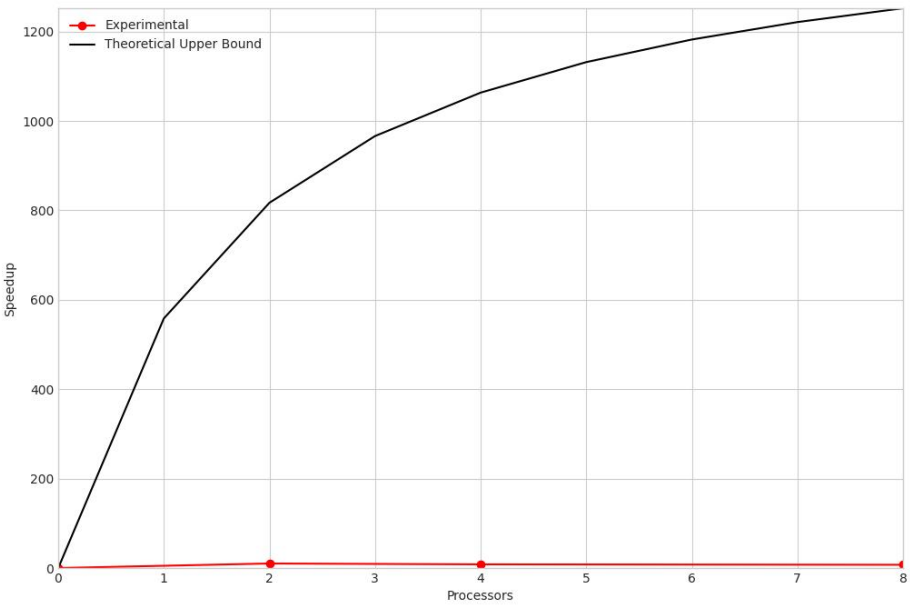
4.2.7.1 Optimization 0

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	204800.000	100352.000	0.008	0.029	no data	no data	no data	25.436	no data	no data	25.458	0.008	99.040	25.471	1.000	1.000
Parallel	2	204800.000	100352.000	0.057	0.392	0.024	0.023	0.394	1.365	0.034	0.001	3.669	0.125	162.080	2.440	10.439	5.219
Parallel	4	204800.000	100352.000	0.064	0.539	0.023	0.025	0.540	1.378	0.034	0.001	3.254	0.914	135.920	3.160	8.059	2.015
Parallel	8	204800.000	100352.000	0.067	0.641	0.023	0.026	0.643	1.377	0.034	0.001	5.059	1.875	206.920	3.430	7.425	0.928



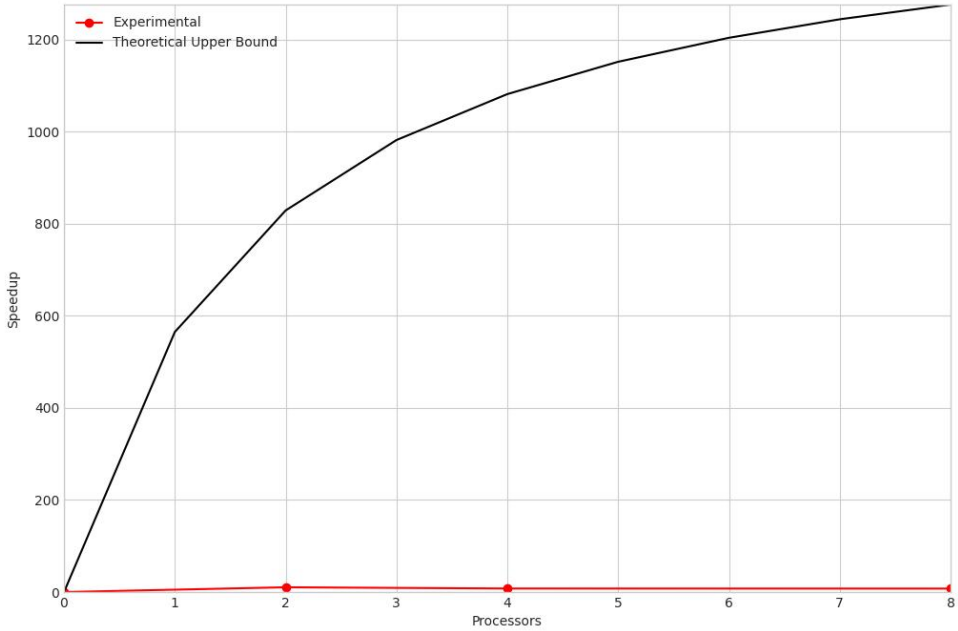
4.2.7.2 Optimization 1

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	204800.000	100352.000	0.008	0.029	no data	no data	no data	25.458	no data	no data	25.479	0.008	99.000	25.493	1.000	1.000
Parallel	2	204800.000	100352.000	0.054	0.393	0.023	0.022	0.395	1.361	0.033	0.001	3.654	0.123	160.200	2.520	10.118	5.059
Parallel	4	204800.000	100352.000	0.065	0.512	0.023	0.024	0.514	1.375	0.034	0.001	3.218	0.892	136.400	3.099	8.226	2.056
Parallel	8	204800.000	100352.000	0.070	0.631	0.023	0.027	0.633	1.388	0.036	0.001	5.092	1.803	202.440	3.509	7.265	0.908



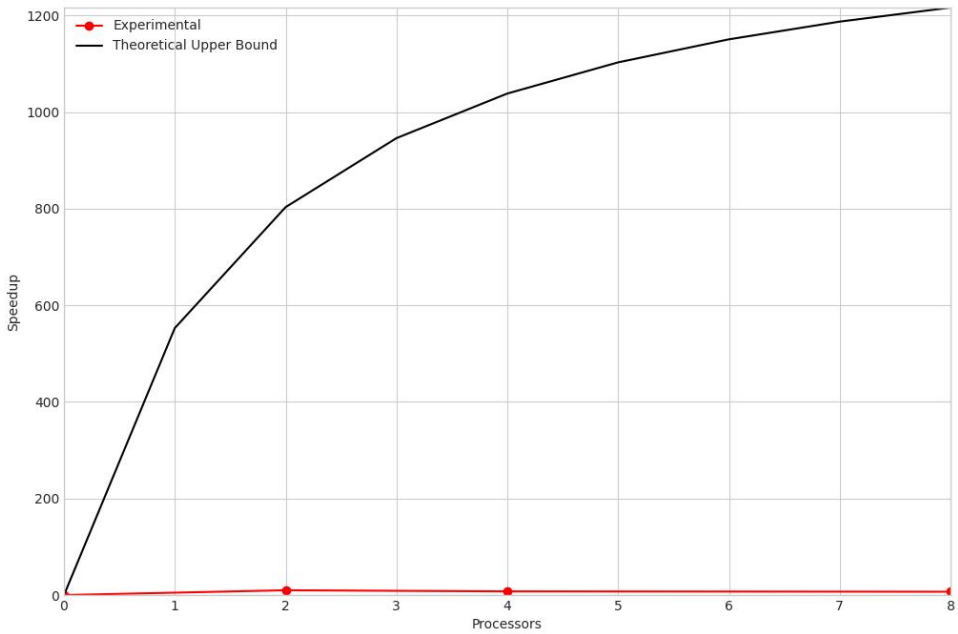
4.2.7.3 Optimization 2

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	204800.000	100352.000	0.008	0.029	no data	no data	no data	25.469	no data	no data	25.488	0.009	99.040	25.504	1.000	1.000
Parallel	2	204800.000	100352.000	0.051	0.397	0.023	0.022	0.399	1.362	0.032	0.001	3.667	0.118	161.400	2.450	10.411	5.206
Parallel	4	204800.000	100352.000	0.073	0.538	0.023	0.026	0.539	1.391	0.037	0.001	3.281	0.933	133.000	3.304	7.720	1.930
Parallel	8	204800.000	100352.000	0.076	0.625	0.024	0.028	0.626	1.378	0.038	0.001	5.042	1.799	206.760	3.374	7.558	0.945



4.2.7.4 Optimization 3

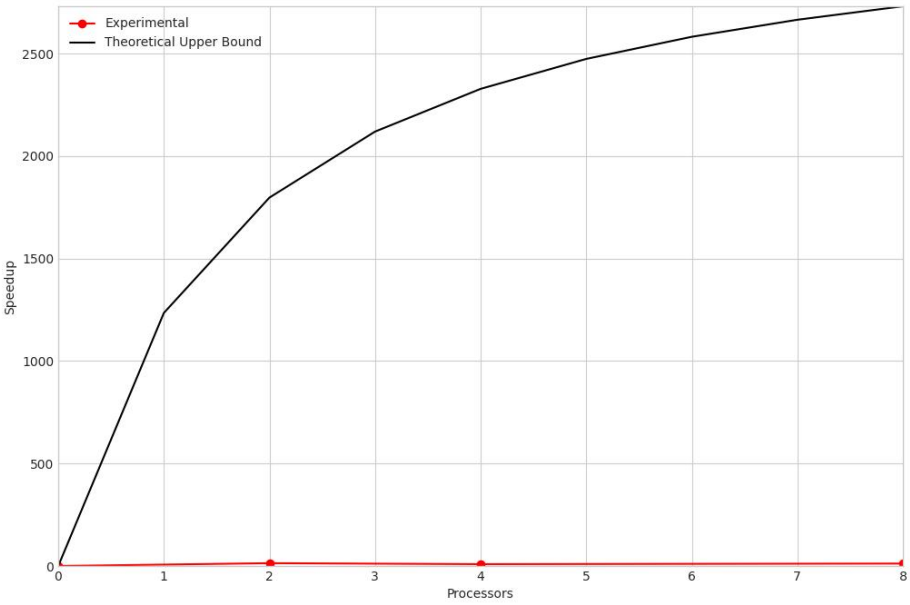
Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	204800.000	100352.000	0.008	0.029	no data	no data	no data	25.455	no data	no data	25.472	0.011	99.040	25.491	1.000	1.000
Parallel	2	204800.000	100352.000	0.066	0.380	0.024	0.024	0.382	1.362	0.036	0.001	3.659	0.128	157.960	2.558	9.967	4.983
Parallel	4	204800.000	100352.000	0.067	0.588	0.023	0.024	0.590	1.389	0.034	0.001	3.340	0.956	133.640	3.341	7.630	1.908
Parallel	8	204800.000	100352.000	0.060	0.629	0.022	0.027	0.631	1.378	0.035	0.001	5.064	1.807	197.440	3.682	6.924	0.865



4.2.8 Tile-410000 graph

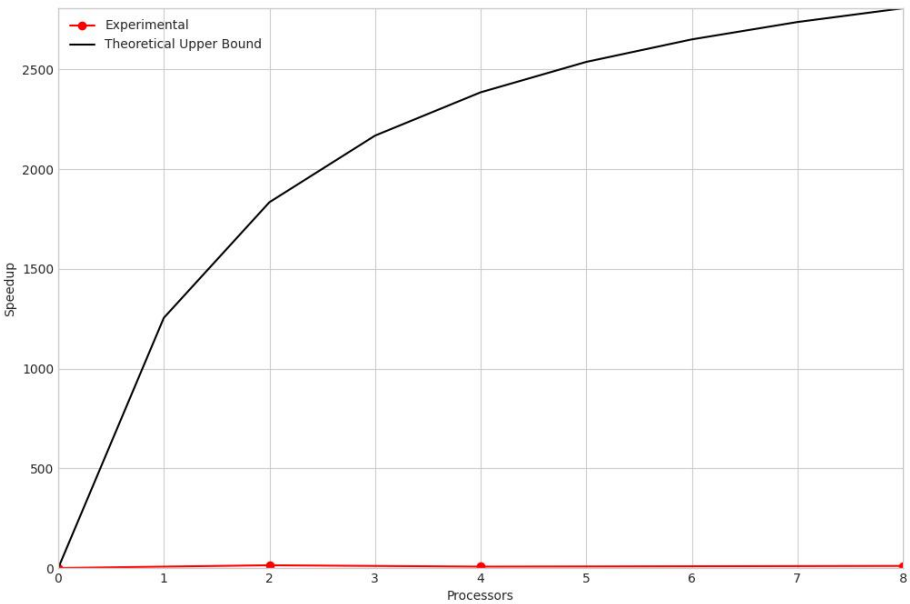
4.2.8.1 Optimization 0

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	409600.000	200704.000	0.017	0.052	no data	no data	no data	101.611	no data	no data	101.645	0.027	99.000	101.683	1.000	1.000
Parallel	2	409600.000	200704.000	0.061	0.928	0.053	0.048	0.930	5.288	0.080	0.002	12.655	0.155	177.040	7.302	13.925	6.963
Parallel	4	409600.000	200704.000	0.074	1.080	0.051	0.046	1.083	5.371	0.066	0.002	10.644	2.556	151.320	11.153	9.117	2.279
Parallel	8	409600.000	200704.000	0.079	1.161	0.054	0.052	1.164	5.345	0.068	0.002	15.561	5.064	244.840	8.482	11.988	1.499



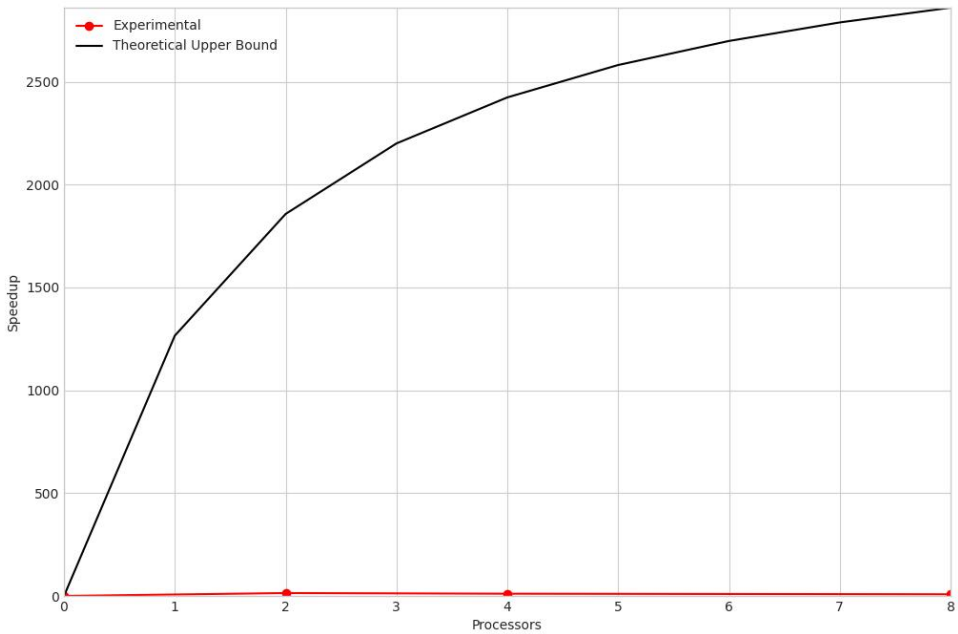
4.2.8.2 Optimization 1

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	409600.000	200704.000	0.017	0.051	no data	no data	no data	101.562	no data	no data	101.594	0.030	99.000	101.633	1.000	1.000
Parallel	2	409600.000	200704.000	0.077	0.899	0.056	0.051	0.902	5.329	0.086	0.002	12.704	0.176	178.480	7.289	13.944	6.972
Parallel	4	409600.000	200704.000	0.069	1.012	0.050	0.044	1.015	5.306	0.064	0.002	12.396	3.496	161.080	14.624	6.950	1.737
Parallel	8	409600.000	200704.000	0.075	1.112	0.050	0.048	1.116	5.295	0.067	0.002	16.413	5.206	237.160	9.633	10.550	1.319



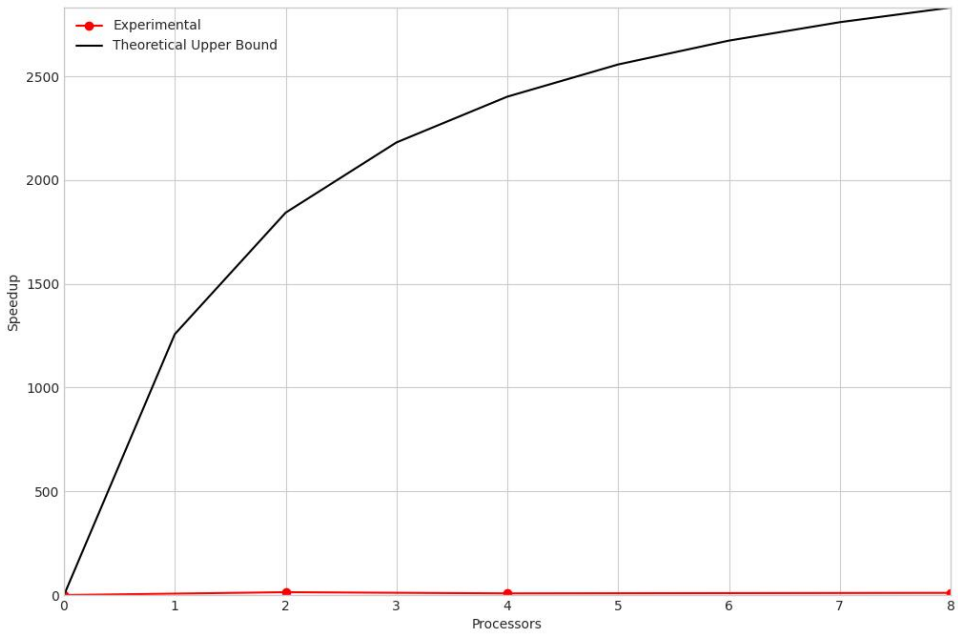
4.2.8.3 Optimization 2

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	409600.000	200704.000	0.017	0.051	no data	no data	no data	101.449	no data	no data	101.480	0.030	99.000	101.519	1.000	1.000
Parallel	2	409600.000	200704.000	0.076	0.910	0.057	0.054	0.913	5.208	0.098	0.002	12.500	0.181	178.240	7.193	14.114	7.057
Parallel	4	409600.000	200704.000	0.077	1.471	0.051	0.046	1.474	5.337	0.066	0.002	11.964	3.299	162.200	9.279	10.941	2.735
Parallel	8	409600.000	200704.000	0.077	1.814	0.053	0.052	1.817	5.307	0.070	0.002	17.381	6.161	234.320	12.364	8.211	1.026



4.2.8.4 Optimization 3

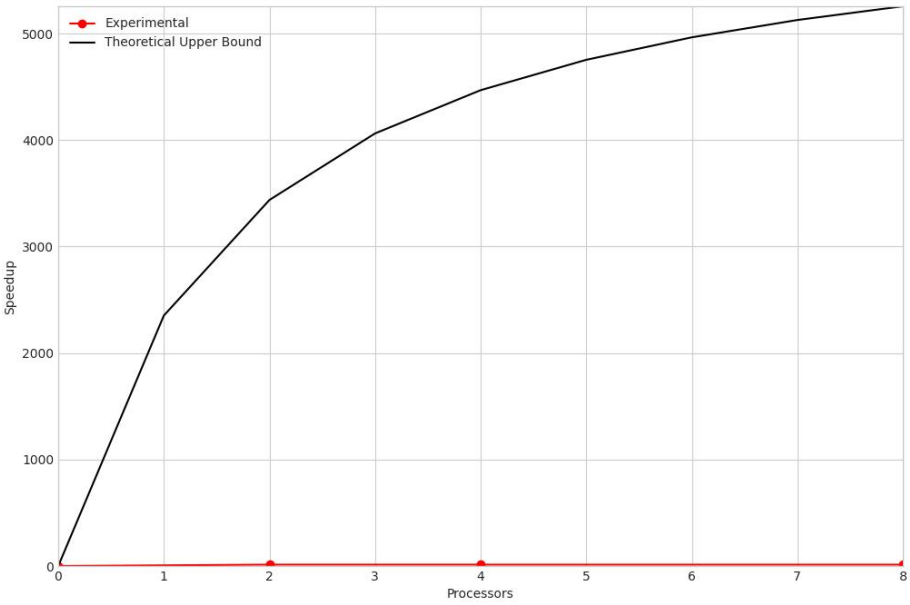
Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	409600.000	200704.000	0.017	0.051	no data	no data	no data	101.432	no data	no data	101.459	0.034	99.000	101.502	1.000	1.000
Parallel	2	409600.000	200704.000	0.078	0.919	0.058	0.052	0.922	5.289	0.089	0.002	12.670	0.178	177.280	7.329	13.849	6.924
Parallel	4	409600.000	200704.000	0.075	1.076	0.051	0.045	1.079	5.447	0.067	0.002	11.082	2.700	143.320	12.493	8.125	2.031
Parallel	8	409600.000	200704.000	0.071	1.546	0.052	0.050	1.549	5.317	0.065	0.002	16.418	5.745	237.360	9.591	10.583	1.323



4.2.9 Tile-820000 graph

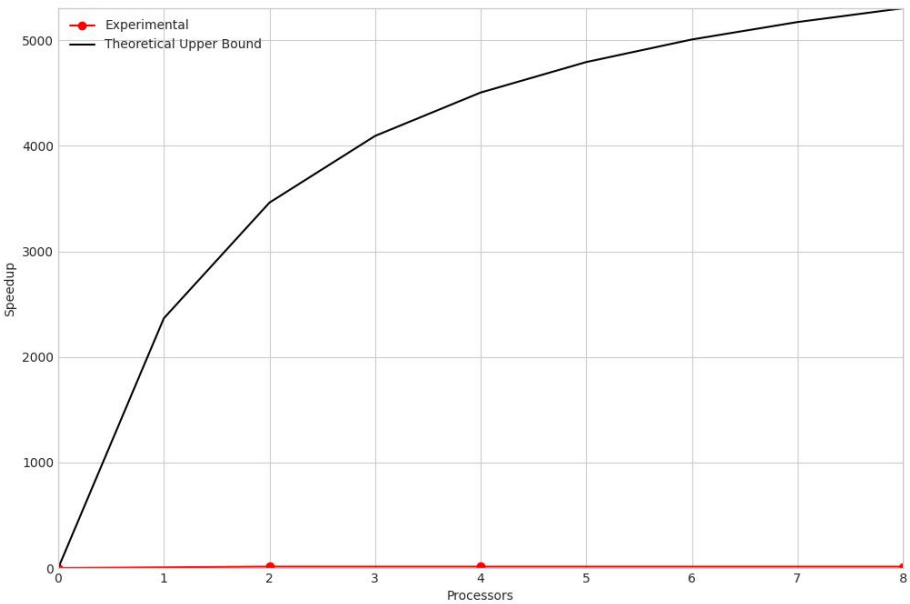
4.2.9.1 Optimization 0

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	819200.000	401408.000	0.034	0.108	no data	no data	no data	403.937	no data	no data	404.022	0.053	99.000	404.098	1.000	1.000
Parallel	2	819200.000	401408.000	0.114	2.191	0.124	0.095	2.196	26.270	0.164	0.005	57.303	0.302	186.000	30.901	13.077	6.539
Parallel	4	819200.000	401408.000	0.104	2.059	0.122	0.100	2.065	26.208	0.199	0.005	46.980	10.306	183.560	31.099	12.994	3.249
Parallel	8	819200.000	401408.000	0.093	2.213	0.121	0.103	2.219	26.173	0.183	0.005	66.149	20.410	277.800	31.109	12.990	1.624



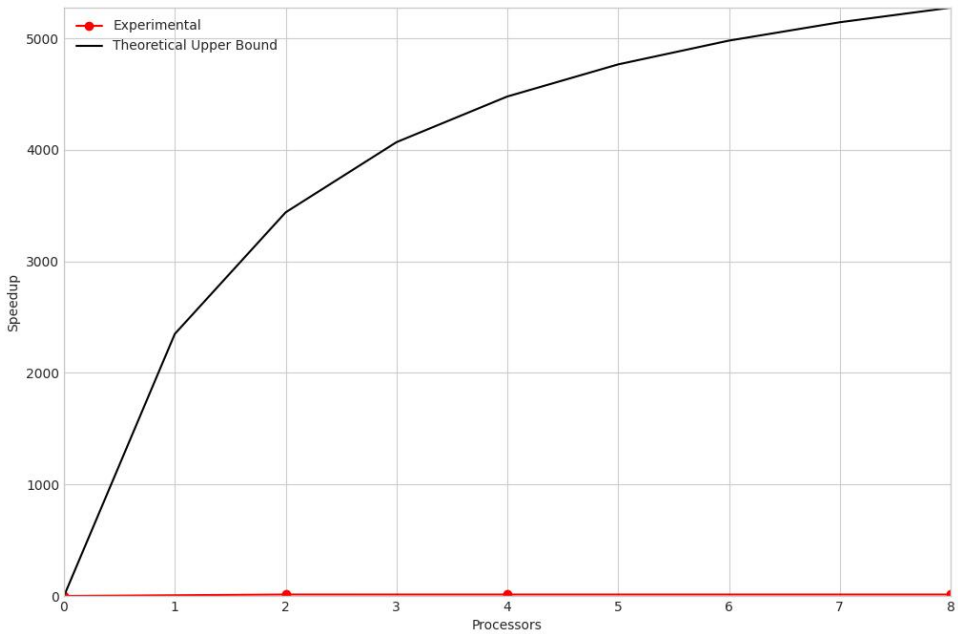
4.2.9.2 Optimization 1

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	819200.000	401408.000	0.034	0.108	no data	no data	no data	404.900	no data	no data	405.004	0.034	99.000	405.061	1.000	1.000
Parallel	2	819200.000	401408.000	0.094	2.287	0.125	0.099	2.293	26.171	0.191	0.005	57.308	0.290	186.280	30.839	13.135	6.567
Parallel	4	819200.000	401408.000	0.098	2.199	0.121	0.099	2.205	26.210	0.199	0.005	47.246	10.320	184.080	31.195	12.985	3.246
Parallel	8	819200.000	401408.000	0.099	2.271	0.123	0.106	2.277	26.194	0.197	0.005	66.465	20.436	275.040	31.532	12.846	1.606



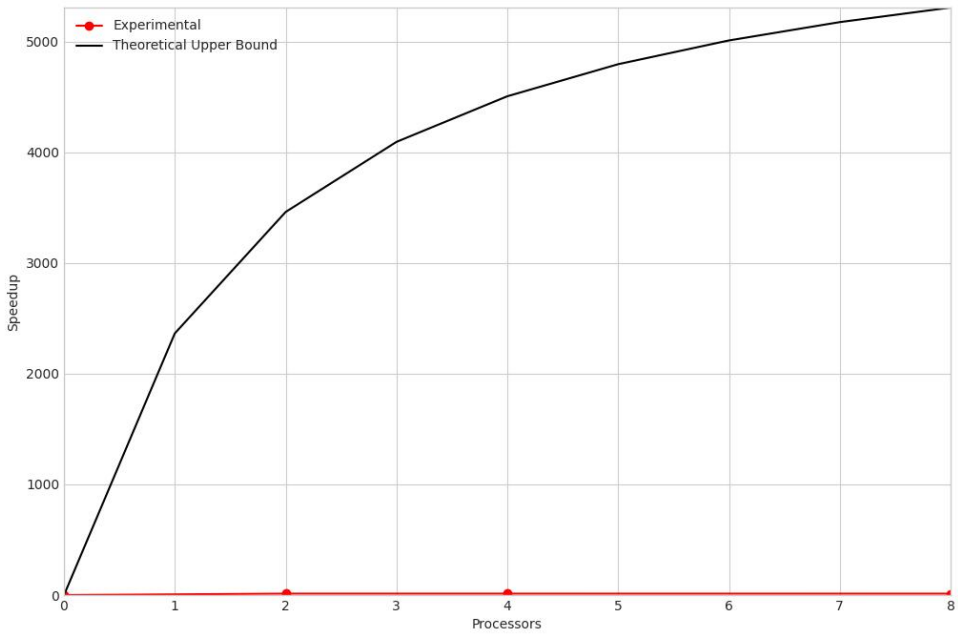
4.2.9.3 Optimization 2

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	819200.000	401408.000	0.034	0.109	no data	no data	no data	404.260	no data	no data	404.362	0.036	99.000	404.422	1.000	1.000
Parallel	2	819200.000	401408.000	0.097	2.348	0.126	0.102	2.354	26.197	0.198	0.005	57.481	0.312	185.520	31.067	13.018	6.509
Parallel	4	819200.000	401408.000	0.093	2.184	0.116	0.096	2.190	26.226	0.195	0.005	47.154	10.388	183.600	31.280	12.929	3.232
Parallel	8	819200.000	401408.000	0.094	2.281	0.123	0.105	2.287	26.163	0.192	0.005	66.398	20.417	276.920	31.298	12.921	1.615



4.2.9.4 Optimization 3

Version	Threads	vertices	vertices_after	init	tarjan	split	merge	total_only_mpi	preprocess	conversion	finalize	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	819200.000	401408.000	0.034	0.109	no data	no data	no data	404.698	no data	no data	404.793	0.042	99.000	404.859	1.000	1.000
Parallel	2	819200.000	401408.000	0.100	2.182	0.127	0.102	2.188	26.200	0.200	0.005	57.180	0.298	186.360	30.768	13.158	6.579
Parallel	4	819200.000	401408.000	0.101	2.119	0.118	0.096	2.125	26.223	0.196	0.005	47.070	10.354	185.280	30.930	13.090	3.272
Parallel	8	819200.000	401408.000	0.089	2.239	0.116	0.098	2.245	26.205	0.192	0.005	66.456	20.371	276.160	31.408	12.890	1.611



4.3 Considerations

Most of the time we managed to get a speedup greater than 1 thanks to the CUDA preprocessing. Notice that the theoretical upper bound for the speedup for a given number of processors P is often bigger than P. This is not an error because, in the graphs, only the P MPI processes are considered but multiple CUDA threads are implicitly used. More specifically, for this algorithm, the number of CUDA threads is set to be equal to the number of nodes in the input graph.

CHAPTER 5

CUDA IMPLEMENTATION WITH TEXTURE MEMORY

In this chapter we will discuss about our CUDA implementation of Tarjan's algorithm using texture memory insted of global memory (`cuda_texture`) and present the experimental results that this algorithm.

5.1 Algorithm idea

The algorithm is identical to the one described in 3.1.4, except that the `cuda_graph` is accessed through the texture memory of the device instead of the global memory. The term "texture memory" doesn't refer to a separate physical memory but rather it is an alternative way of accessing the global memory on the device [16]: textures are stored on the global memory, and they are accessed through a dedicated read-only cache. The cache includes hardware filtering which can perform linear floating point interpolation as part of the read process. The cache, however, is different to a conventional cache, in that it is optimised for spatial locality (in the coordinate system of the texture) and not locality in memory. For some applications, this is ideal and will give a performance advantage, but for others, it won't: textures can be slower because access involves a cache miss penalty in addition to the global memory read.

The algorithm described in this chapter exploits the texture memory of CUDA by defining the arrays `adj_lists` and `adj_list_indexes` as textures mapped on the global memory. The idea is that the algorithm could benefit from the memory caching provided by the texture memory mechanism. However, textures have historically come in 2 usage models: **texture references** and **texture objects**. Texture references were the original mechanism supplied with CUDA and texture objects were introduced with the Kepler generation of GPUs. During the High Performance Computing course, only texture references were covered, but their usage was deprecated in CUDA 11.3. In order to exploit texture memory, the refactoring process described in [17] was applied to the code seen in class.

Note that we didn't implement versions of the preprocessing algorithm exploiting shared or constant memories because they are too small for our purposes. In particular, observe that for each node of the graph, each CUDA thread must traverse the entirety of `adj_lists` to look for edges incoming into that node. This makes it difficult to split the work among thread blocks, and thus it would be difficult to exploit the advantages of shared memory.

5.2 Experimental results

In this subsection are presented the results of the parallel version of Tarjan's algorithm with CUDA with the use of the texture memory. Each graph is tested with all the compiler optimization (O0, O1, O2, O3) and with $\frac{n}{4}$, $\frac{n}{2}$, $\frac{3*n}{4}$, n CUDA threads, where n is the number of vertices of the graph tested. In the next pages for each graph, set the optimization, there are a table and a graph of the speedup. In the table there are:

- **Version**: is the version of the program. It can be "Serial" if it's the sequential with preprocessing version of the program, or it can be "Parallel" if it's the parallel version with CUDA.
- **Threads**: is the number of thread CUDA. For the "Serial" version, the value of this field is 1, because it's the sequential with preprocessing version of the program executed on only one processor of the CPU.
- **vertices**: is the number of vertices of the graph after the preprocessing.
- **init**: is the time spent to initialize the program, in particular it's the time spent to create and initialize the data structures, and to read the graph from a file.
- **preprocess**: is the time spent to do the preprocessing of the graph.
- **conversion**: is the time spent to convert the graph into a graph adapted for CUDA.
- **finalize**: is the time spent to deallocate the necessary data structures.
- **tarjan**: the time spent to execute Tarjan's algorithm.
- **user**: is the total number of CPU-seconds that the process used directly (in user mode), in seconds.
- **system**: is the total number of CPU-seconds used by the system on behalf of the process (in kernel mode), in seconds.
- **pCPU**: is the percentage of the CPU that this process got. This is just user + system times divided by the total running time.
- **elapsed**: is the elapsed real (wall clock) time used by the process, in seconds.
- **Speedup**: the speedup of the program.

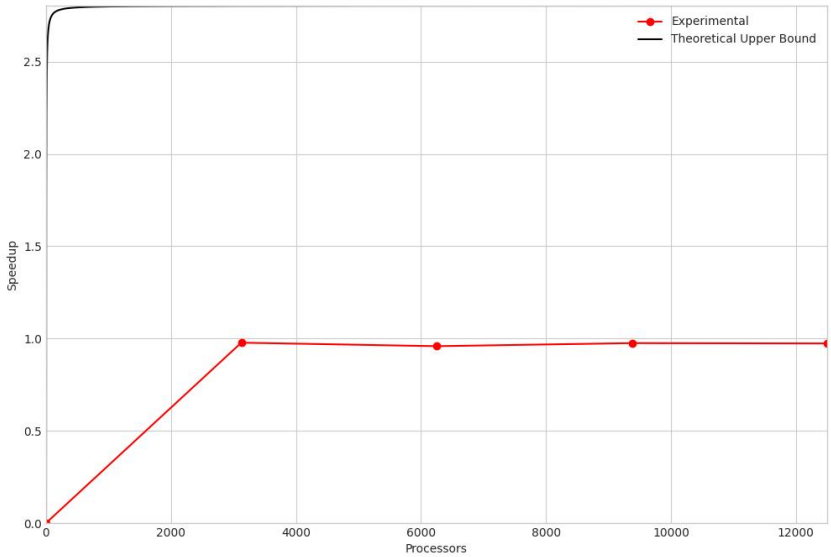
- **Efficiency**: the efficiency of the program.

All values except those in the `Version`, `Threads`, `Speedup`, and `Efficiency` columns are means. These means were calculated from the files of the measurements taken. `Speedup`, and `Efficiency` are calculated with the formulas in Cap.1 using the results of these means. For the sequential algorithm, the `finalize` and `conversion` fields were not measured.

5.2.1 Fully connected graph

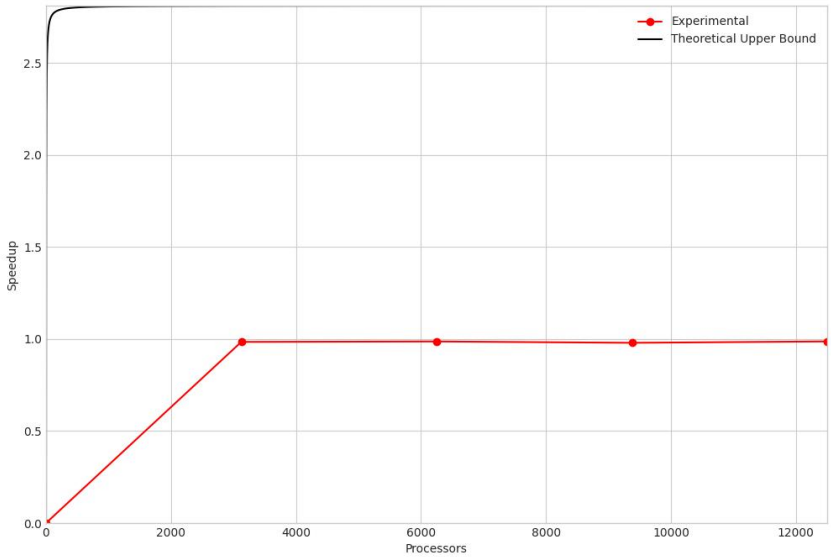
5.2.1.1 Optimization 0

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	12500.000	2.231	no data	9.478	no data	2.940	13.709	1.013	99.040	14.730	1.000	1.000
Parallel	3125	12500.000	2.384	0.015	0.006	9.615	2.964	14.005	1.039	99.000	15.067	0.978	0.000
Parallel	6250	12500.000	2.359	0.016	0.003	9.910	2.987	14.285	1.052	99.000	15.364	0.959	0.000
Parallel	9375	12500.000	2.377	0.016	0.003	9.670	2.953	14.020	1.060	99.000	15.104	0.975	0.000
Parallel	12500	12500.000	2.343	0.016	0.002	9.701	2.987	14.078	1.032	99.000	15.132	0.973	0.000



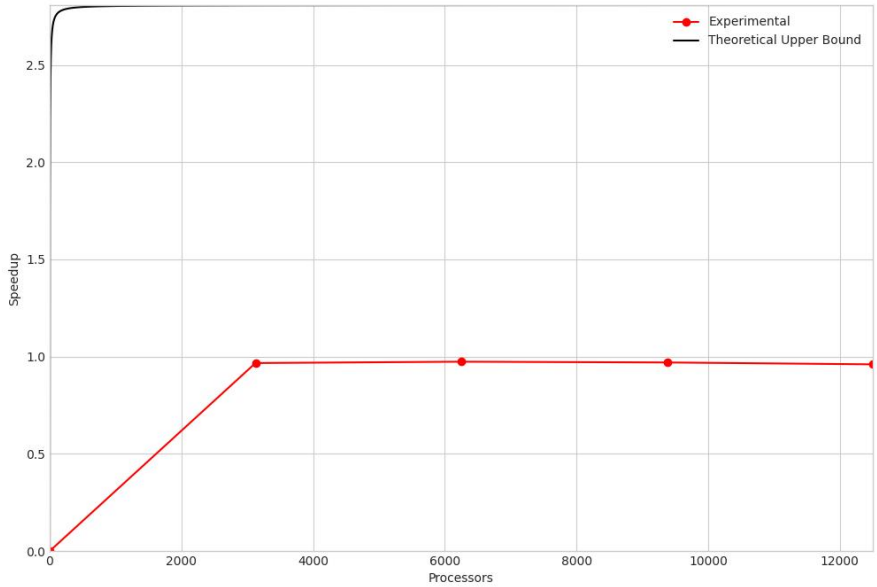
5.2.1.2 Optimization 1

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	12500.000	2.237	no data	9.523	no data	2.942	13.772	1.002	99.000	14.782	1.000	1.000
Parallel	3125	12500.000	2.344	0.015	0.006	9.582	2.990	13.971	1.029	99.000	15.022	0.984	0.000
Parallel	6250	12500.000	2.344	0.016	0.003	9.568	2.972	13.944	1.018	99.000	14.986	0.986	0.000
Parallel	9375	12500.000	2.343	0.016	0.003	9.640	3.012	14.028	1.047	99.000	15.098	0.979	0.000
Parallel	12500	12500.000	2.332	0.016	0.002	9.547	3.001	13.926	1.034	99.000	14.982	0.987	0.000



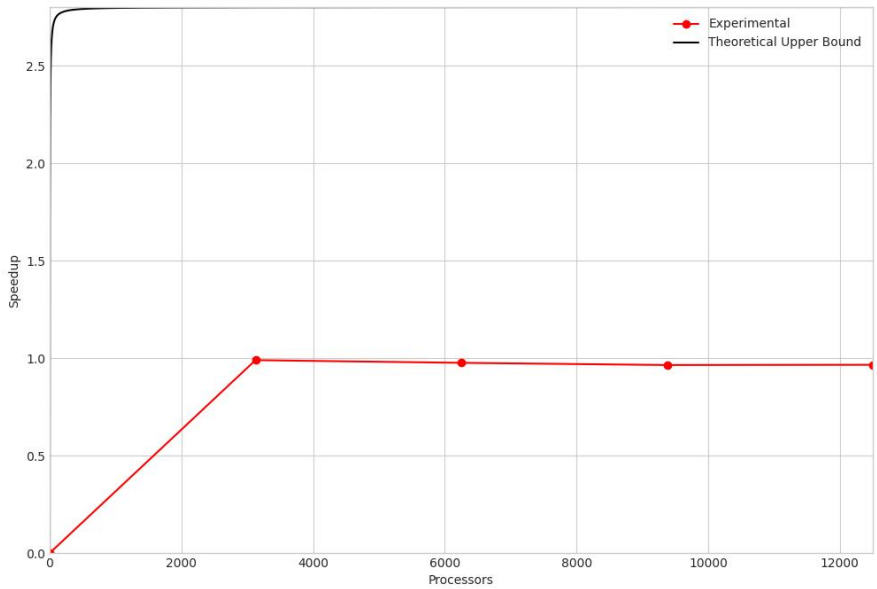
5.2.1.3 Optimization 2

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	12500.000	2.236	no data	9.454	no data	2.911	13.648	1.026	99.040	14.681	1.000	1.000
Parallel	3125	12500.000	2.351	0.016	0.006	9.693	3.036	14.116	1.045	99.000	15.185	0.967	0.000
Parallel	6250	12500.000	2.354	0.016	0.003	9.627	2.989	14.024	1.030	99.000	15.077	0.974	0.000
Parallel	9375	12500.000	2.352	0.016	0.003	9.664	3.018	14.062	1.054	99.000	15.138	0.970	0.000
Parallel	12500	12500.000	2.349	0.016	0.002	9.790	3.045	14.214	1.052	99.000	15.290	0.960	0.000



5.2.1.4 Optimization 3

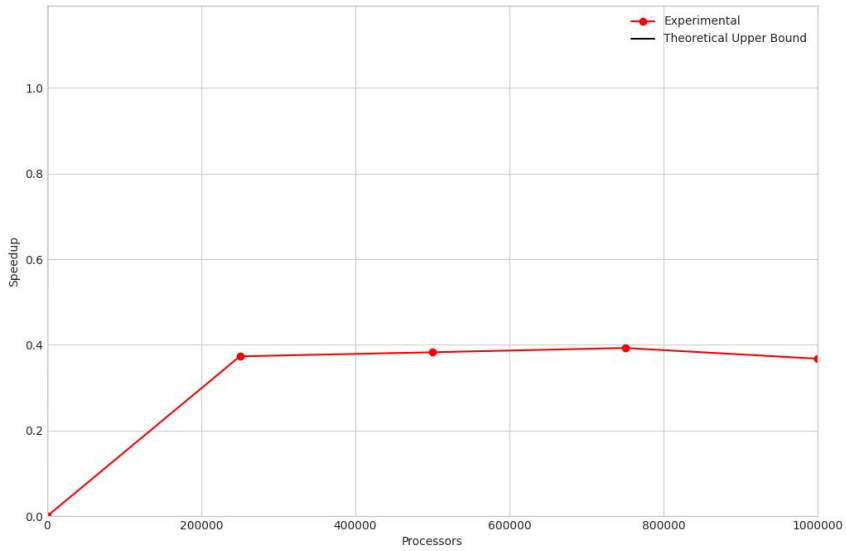
Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	12500.000	2.231	no data	9.460	no data	2.934	13.706	0.994	99.000	14.708	1.000	1.000
Parallel	3125	12500.000	2.337	0.015	0.006	9.467	2.946	13.807	1.027	99.000	14.855	0.990	0.000
Parallel	6250	12500.000	2.336	0.016	0.003	9.608	3.013	13.988	1.054	99.000	15.065	0.976	0.000
Parallel	9375	12500.000	2.352	0.016	0.003	9.767	3.017	14.187	1.032	99.000	15.241	0.965	0.000
Parallel	12500	12500.000	2.360	0.016	0.002	9.713	3.042	14.144	1.057	99.000	15.222	0.966	0.000



5.2.2 Fully disconnected graph

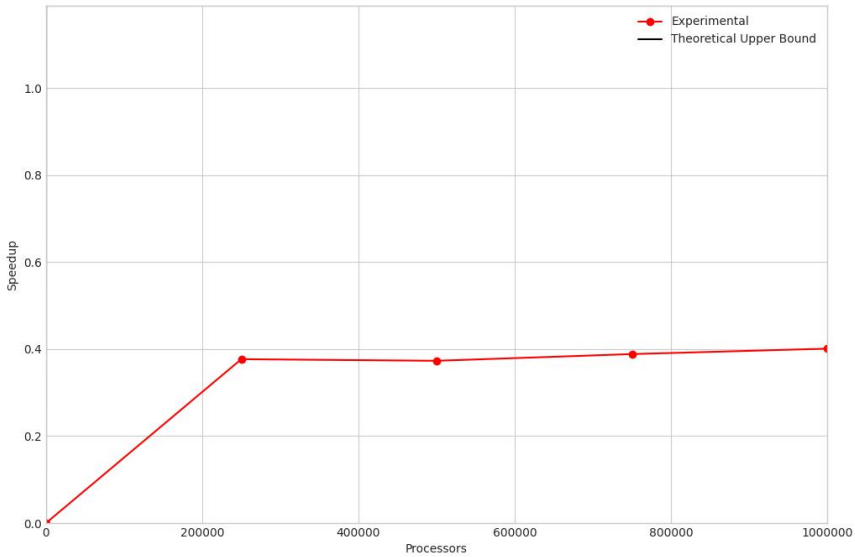
5.2.2.1 Optimization 0

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	0.000	0.026	no data	0.005	no data	0.000	0.024	0.000	97.480	0.030	1.000	1.000
Parallel	250000	0.000	0.062	0.001	0.000	0.001	0.000	0.027	0.040	88.600	0.080	0.373	0.000
Parallel	500000	0.000	0.061	0.000	0.001	0.001	0.000	0.026	0.039	89.440	0.078	0.383	0.000
Parallel	750000	0.000	0.061	0.000	0.001	0.001	0.000	0.026	0.038	90.160	0.076	0.393	0.000
Parallel	1000000	0.000	0.062	0.000	0.002	0.001	0.000	0.028	0.038	87.080	0.082	0.368	0.000



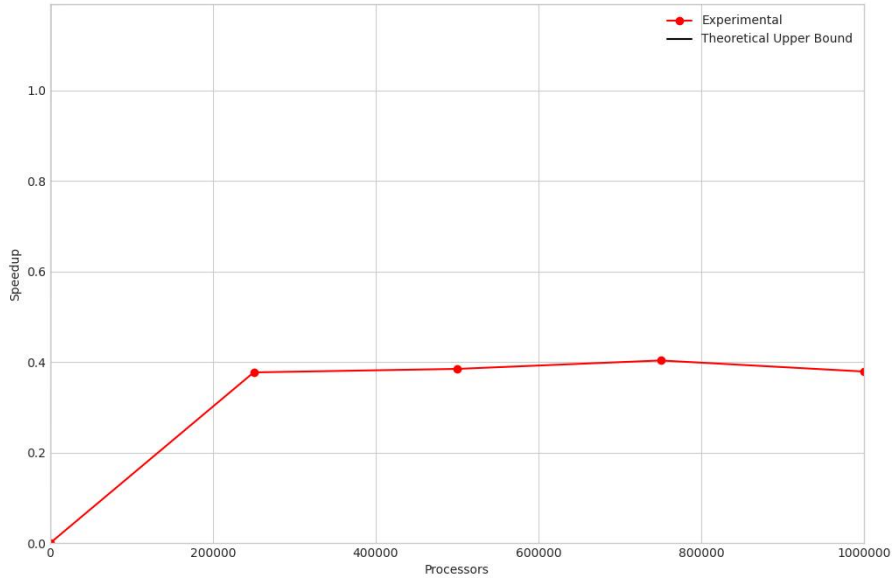
5.2.2.2 Optimization 1

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	0.000	0.026	no data	0.005	no data	0.000	0.023	0.000	97.320	0.030	1.000	1.000
Parallel	250000	0.000	0.063	0.001	0.000	0.001	0.000	0.028	0.040	90.320	0.080	0.377	0.000
Parallel	500000	0.000	0.060	0.000	0.001	0.001	0.000	0.028	0.037	88.280	0.080	0.373	0.000
Parallel	750000	0.000	0.061	0.000	0.001	0.001	0.000	0.026	0.038	90.400	0.077	0.389	0.000
Parallel	1000000	0.000	0.060	0.000	0.002	0.001	0.000	0.030	0.037	91.800	0.075	0.401	0.000



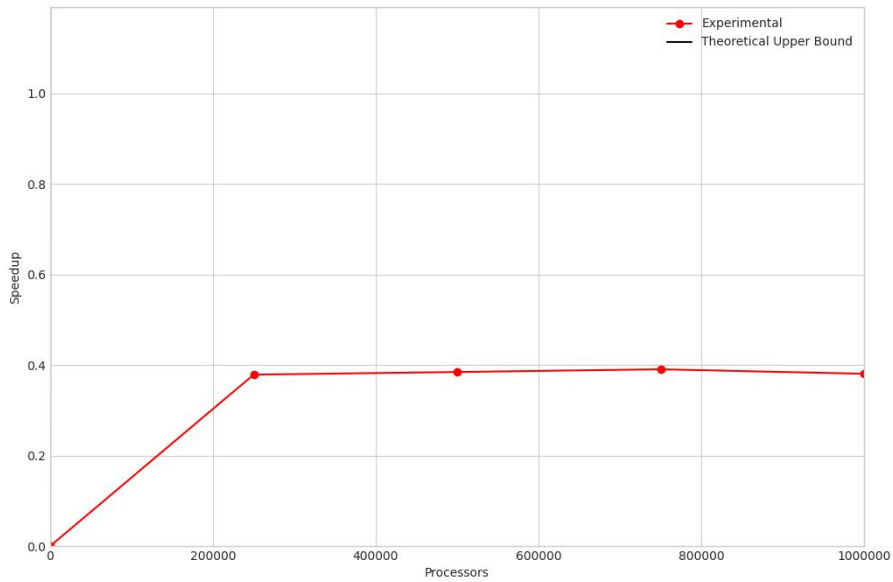
5.2.2.3 Optimization 2

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	0.000	0.026	no data	0.005	no data	0.000	0.026	0.000	98.240	0.030	1.000	1.000
Parallel	250000	0.000	0.060	0.000	0.000	0.001	0.000	0.026	0.038	87.160	0.080	0.377	0.000
Parallel	500000	0.000	0.060	0.000	0.001	0.001	0.000	0.028	0.038	88.960	0.078	0.385	0.000
Parallel	750000	0.000	0.060	0.000	0.001	0.001	0.000	0.028	0.038	92.720	0.074	0.403	0.000
Parallel	1000000	0.000	0.060	0.000	0.002	0.001	0.000	0.029	0.038	88.280	0.079	0.379	0.000



5.2.2.4 Optimization 3

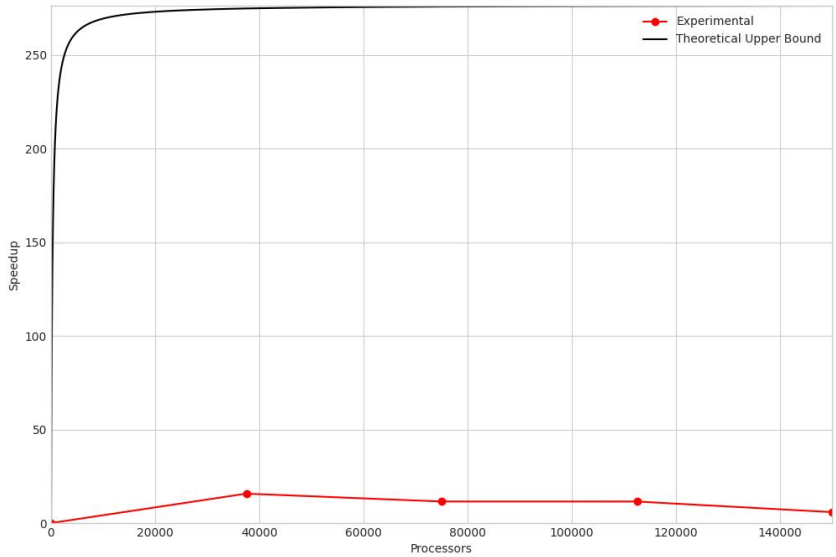
Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	0.000	0.026	no data	0.005	no data	0.000	0.023	0.000	97.440	0.030	1.000	1.000
Parallel	250000	0.000	0.060	0.001	0.000	0.001	0.000	0.028	0.038	89.560	0.079	0.379	0.000
Parallel	500000	0.000	0.060	0.000	0.001	0.001	0.000	0.026	0.037	88.640	0.078	0.385	0.000
Parallel	750000	0.000	0.061	0.000	0.001	0.001	0.000	0.027	0.036	90.800	0.077	0.391	0.000
Parallel	1000000	0.000	0.060	0.000	0.002	0.001	0.000	0.028	0.038	88.880	0.079	0.381	0.000



5.2.3 Random-150000 graph

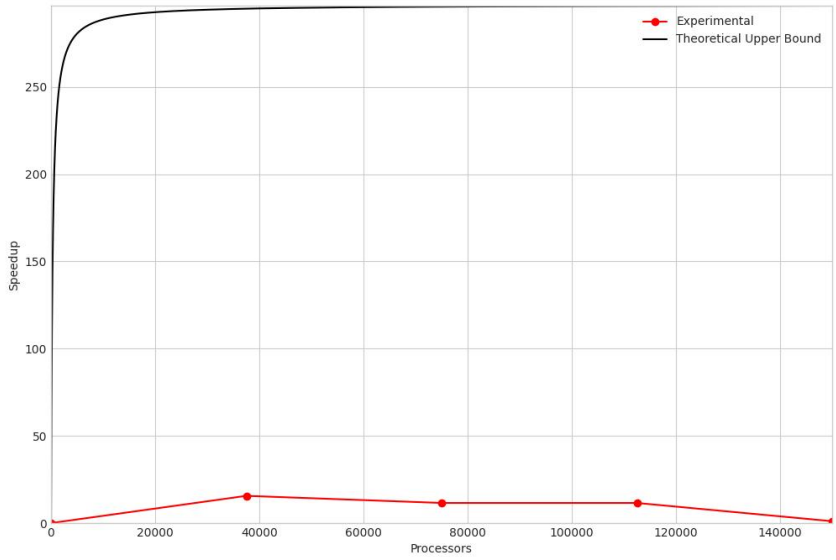
5.2.3.1 Optimization 0

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	71094.000	0.006	no data	11.645	no data	0.036	11.676	0.003	99.120	11.687	1.000	1.000
Parallel	37500	71094.000	0.040	0.000	0.612	0.026	0.038	0.678	0.042	96.960	0.743	15.726	0.000
Parallel	75000	71094.000	0.041	0.000	0.880	0.026	0.037	0.948	0.040	97.320	1.015	11.517	0.000
Parallel	112500	71094.000	0.041	0.000	0.879	0.027	0.038	0.947	0.040	97.440	1.016	11.503	0.000
Parallel	150000	71094.000	0.041	0.001	1.860	0.027	0.037	1.928	0.042	98.080	1.996	5.854	0.000



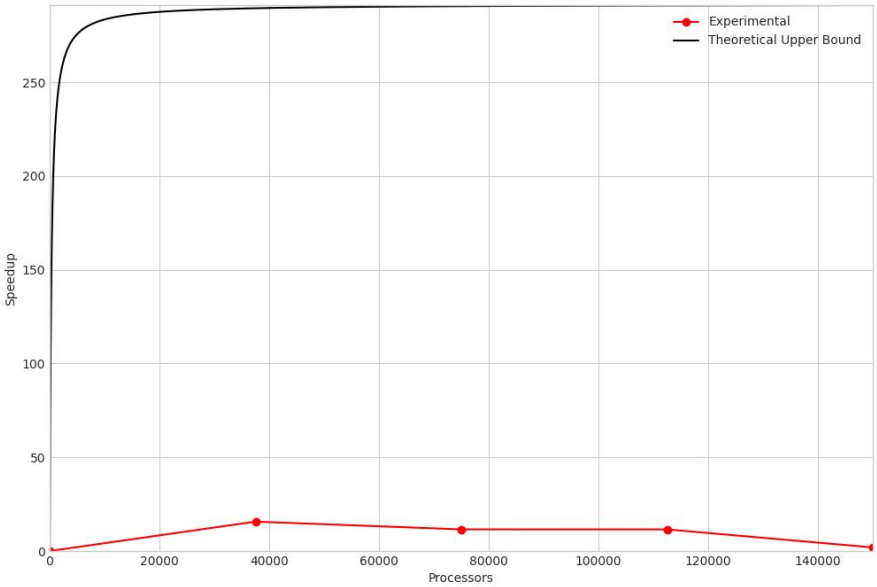
5.2.3.2 Optimization 1

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	71094.000	0.006	no data	11.675	no data	0.036	11.704	0.005	99.200	11.714	1.000	1.000
Parallel	37500	71094.000	0.041	0.000	0.614	0.027	0.038	0.683	0.041	96.720	0.750	15.611	0.000
Parallel	75000	71094.000	0.041	0.000	0.878	0.027	0.038	0.946	0.042	97.160	1.015	11.539	0.000
Parallel	112500	71094.000	0.041	0.000	0.879	0.027	0.038	0.946	0.042	97.280	1.016	11.534	0.000
Parallel	150000	71094.000	0.053	0.000	10.809	0.029	0.040	10.882	0.057	99.000	10.965	1.068	0.000



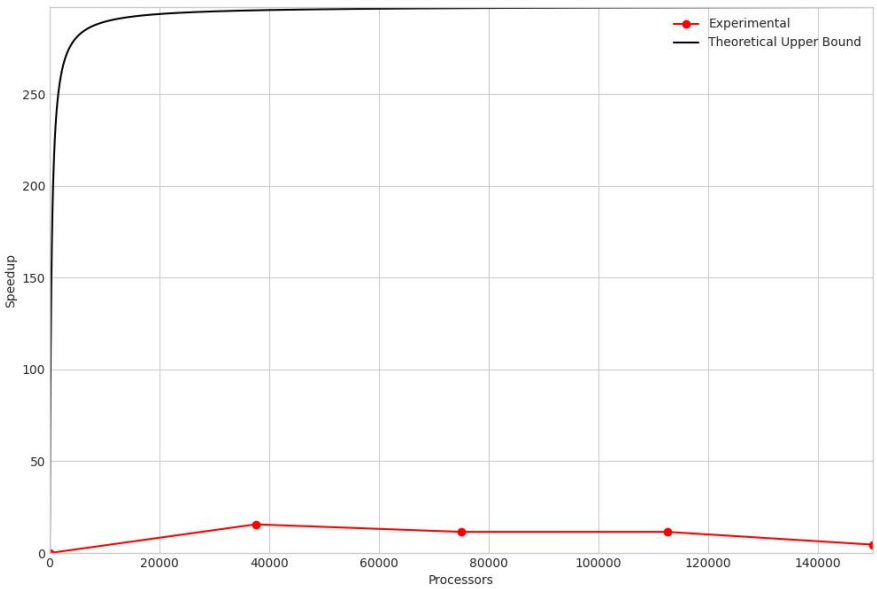
5.2.3.3 Optimization 2

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	71094.000	0.006	no data	11.658	no data	0.036	11.688	0.003	99.080	11.698	1.000	1.000
Parallel	37500	71094.000	0.042	0.000	0.615	0.027	0.038	0.680	0.045	97.280	0.747	15.664	0.000
Parallel	75000	71094.000	0.041	0.000	0.879	0.027	0.037	0.945	0.043	97.400	1.014	11.536	0.000
Parallel	112500	71094.000	0.041	0.000	0.879	0.027	0.038	0.946	0.043	97.400	1.015	11.523	0.000
Parallel	150000	71094.000	0.049	0.000	5.944	0.027	0.038	6.014	0.051	98.480	6.088	1.921	0.000



5.2.3.4 Optimization 3

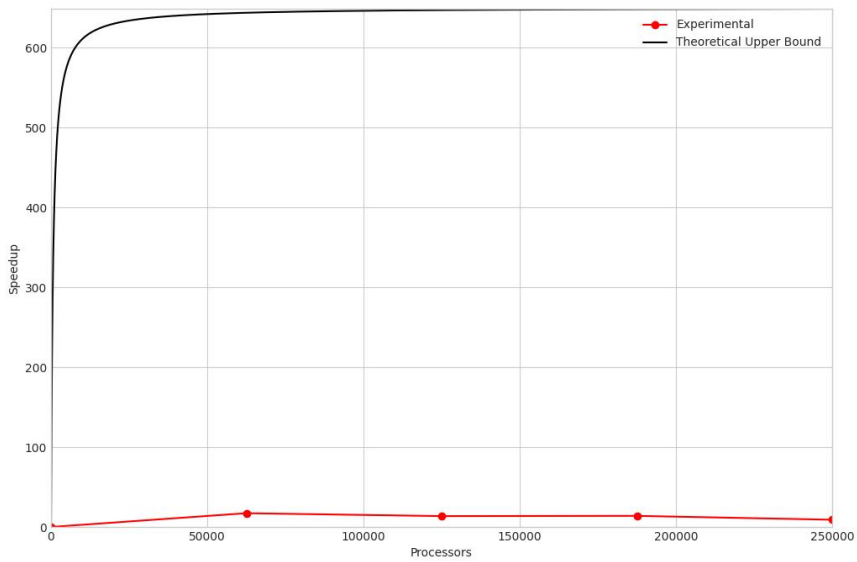
Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	71094.000	0.006	no data	11.672	no data	0.036	11.700	0.004	99.200	11.711	1.000	1.000
Parallel	37500	71094.000	0.043	0.000	0.614	0.027	0.037	0.682	0.043	96.880	0.750	15.614	0.000
Parallel	75000	71094.000	0.042	0.000	0.879	0.027	0.038	0.945	0.044	97.200	1.016	11.522	0.000
Parallel	112500	71094.000	0.042	0.001	0.880	0.026	0.037	0.947	0.042	97.240	1.017	11.517	0.000
Parallel	150000	71094.000	0.047	0.000	2.425	0.027	0.039	2.495	0.047	98.120	2.570	4.557	0.000



5.2.4 Random-250000 graph

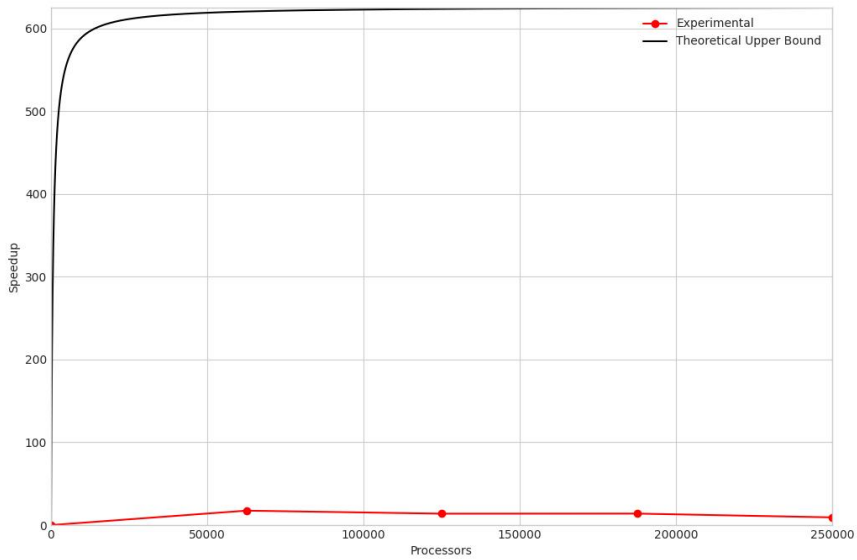
5.2.4.1 Optimization 0

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	118498.000	0.010	no data	32.428	no data	0.041	32.464	0.006	99.160	32.478	1.000	1.000
Parallel	62500	118498.000	0.047	0.000	1.721	0.044	0.043	1.809	0.050	98.160	1.885	17.228	0.000
Parallel	125000	118498.000	0.048	0.000	2.219	0.047	0.045	2.315	0.049	98.720	2.389	13.596	0.000
Parallel	187500	118498.000	0.051	0.000	2.153	0.052	0.049	2.260	0.051	98.640	2.338	13.894	0.000
Parallel	250000	118498.000	0.049	0.000	3.414	0.050	0.047	3.517	0.048	99.000	3.592	9.042	0.000



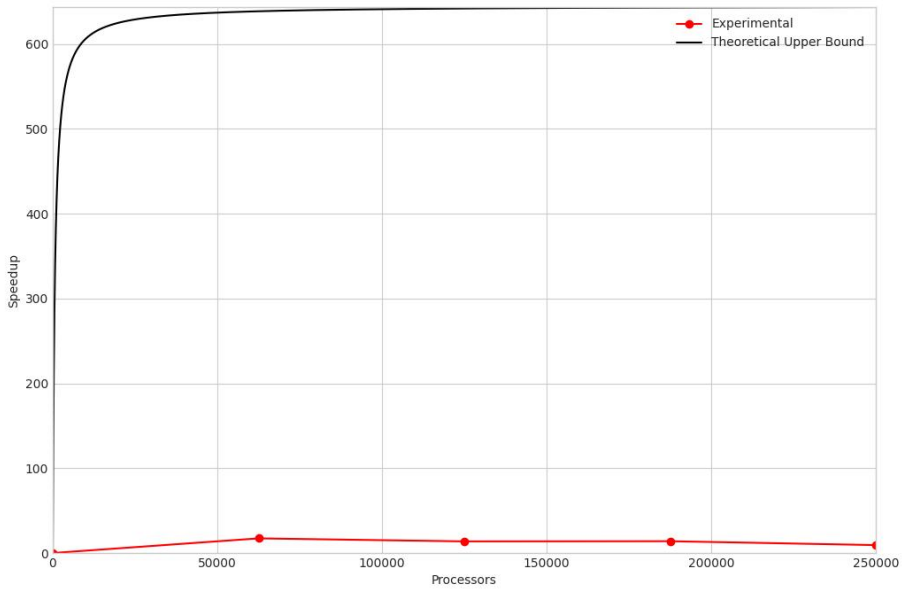
5.2.4.2 Optimization 1

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	118498.000	0.010	no data	32.409	no data	0.041	32.446	0.007	99.040	32.461	1.000	1.000
Parallel	62500	118498.000	0.048	0.000	1.695	0.045	0.043	1.788	0.048	98.040	1.862	17.437	0.000
Parallel	125000	118498.000	0.048	0.000	2.185	0.047	0.044	2.278	0.051	98.640	2.356	13.780	0.000
Parallel	187500	118498.000	0.050	0.000	2.152	0.052	0.048	2.258	0.051	98.200	2.338	13.882	0.000
Parallel	250000	118498.000	0.053	0.000	3.287	0.061	0.055	3.408	0.054	99.000	3.488	9.305	0.000



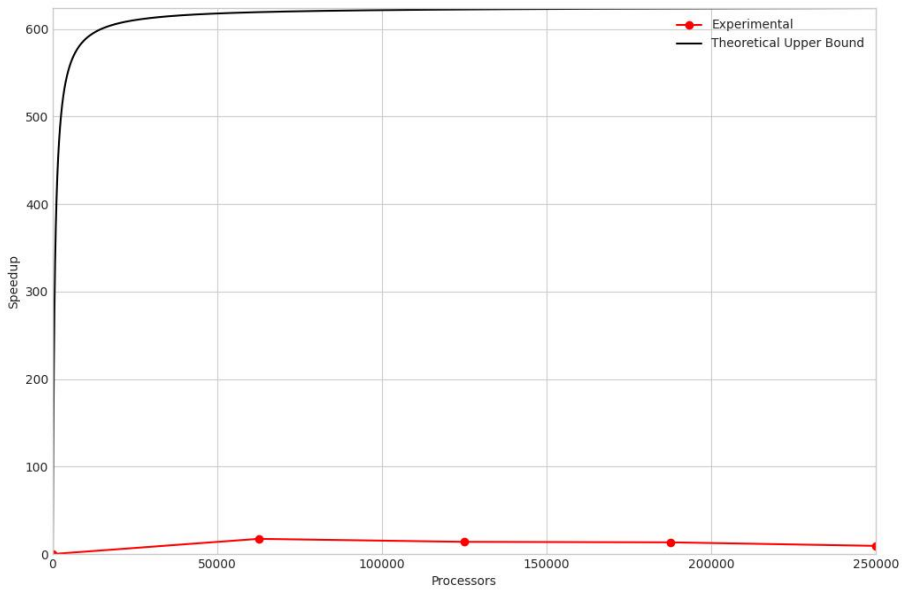
5.2.4.3 Optimization 2

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	118498.000	0.010	no data	32.485	no data	0.041	32.522	0.006	99.120	32.536	1.000	1.000
Parallel	62500	118498.000	0.046	0.000	1.716	0.046	0.044	1.809	0.048	98.080	1.886	17.251	0.000
Parallel	125000	118498.000	0.049	0.000	2.193	0.050	0.047	2.295	0.052	98.560	2.373	13.710	0.000
Parallel	187500	118498.000	0.049	0.000	2.155	0.051	0.048	2.261	0.050	98.600	2.338	13.914	0.000
Parallel	250000	118498.000	0.052	0.000	3.290	0.061	0.054	3.408	0.057	99.000	3.491	9.320	0.000



5.2.4.4 Optimization 3

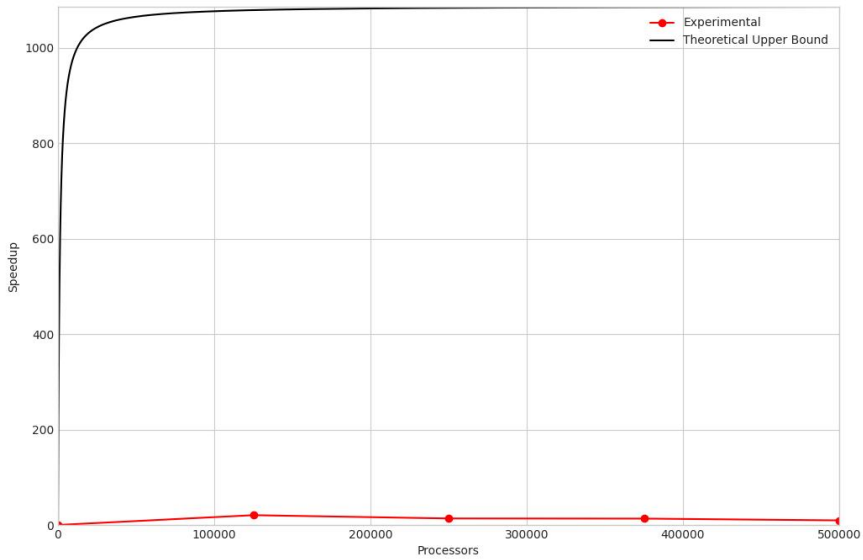
Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	118498.000	0.010	no data	32.394	no data	0.041	32.429	0.006	99.040	32.446	1.000	1.000
Parallel	62500	118498.000	0.047	0.000	1.705	0.046	0.044	1.800	0.047	98.080	1.872	17.336	0.000
Parallel	125000	118498.000	0.049	0.000	2.148	0.052	0.048	2.253	0.051	98.560	2.332	13.913	0.000
Parallel	187500	118498.000	0.047	0.000	2.262	0.045	0.043	2.354	0.049	98.800	2.430	13.352	0.000
Parallel	250000	118498.000	0.052	0.000	3.291	0.061	0.055	3.411	0.053	98.960	3.493	9.289	0.000



5.2.5 Random-500000 graph

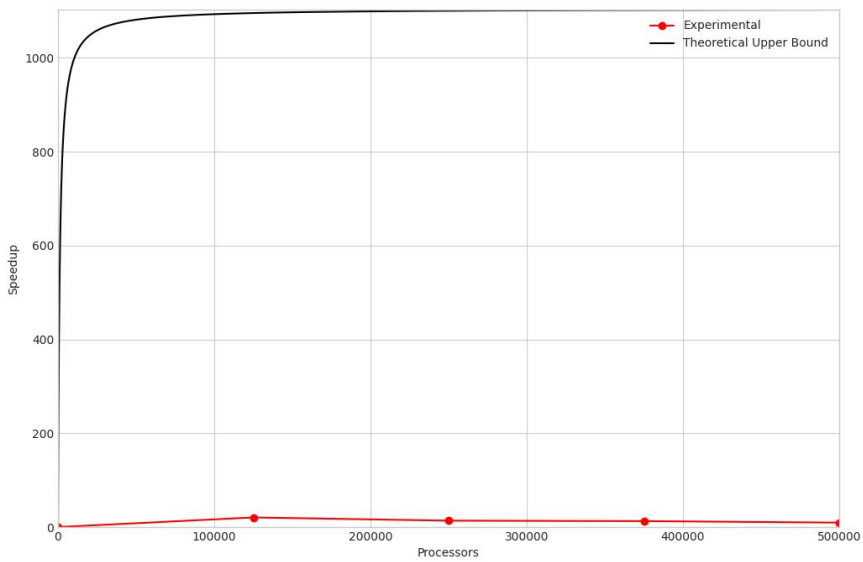
5.2.5.1 Optimization 0

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	236518.000	0.020	no data	128.240	no data	0.092	128.320	0.026	99.000	128.358	1.000	1.000
Parallel	125000	236518.000	0.066	0.000	5.872	0.151	0.124	6.148	0.073	99.000	6.247	20.548	0.000
Parallel	250000	236518.000	0.068	0.000	9.035	0.152	0.127	9.312	0.080	99.000	9.419	13.627	0.000
Parallel	375000	236518.000	0.065	0.000	9.271	0.136	0.119	9.525	0.075	99.000	9.628	13.332	0.000
Parallel	500000	236518.000	0.065	0.000	12.962	0.147	0.122	13.232	0.073	99.000	13.332	9.628	0.000



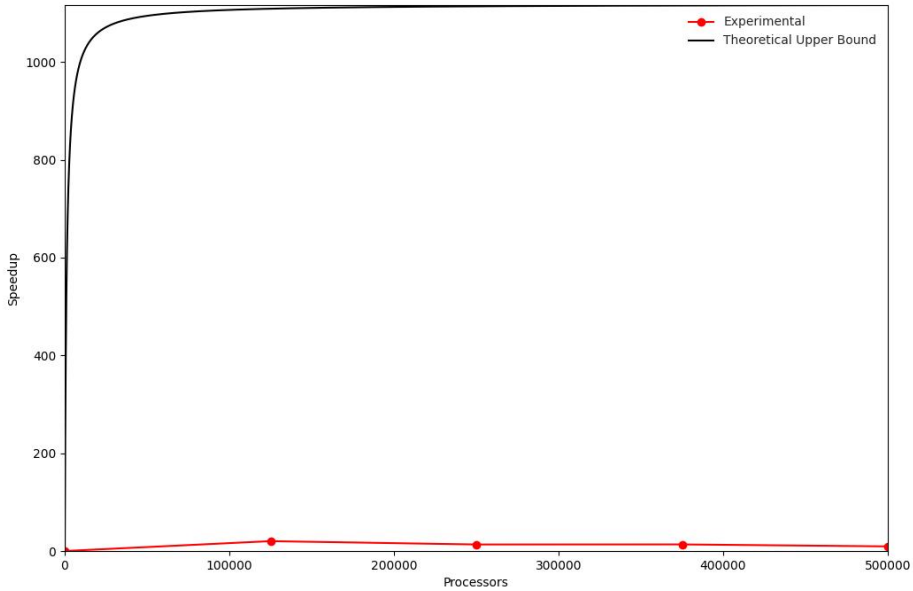
5.2.5.2 Optimization 1

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	236518.000	0.020	no data	128.384	no data	0.093	128.453	0.036	99.000	128.500	1.000	1.000
Parallel	125000	236518.000	0.068	0.000	5.873	0.153	0.127	6.153	0.076	99.000	6.253	20.551	0.000
Parallel	250000	236518.000	0.067	0.000	9.035	0.152	0.125	9.314	0.074	99.000	9.414	13.650	0.000
Parallel	375000	236518.000	0.059	0.000	9.806	0.103	0.100	10.010	0.063	99.000	10.101	12.722	0.000
Parallel	500000	236518.000	0.064	0.000	13.182	0.135	0.117	13.438	0.068	99.000	13.534	9.495	0.000



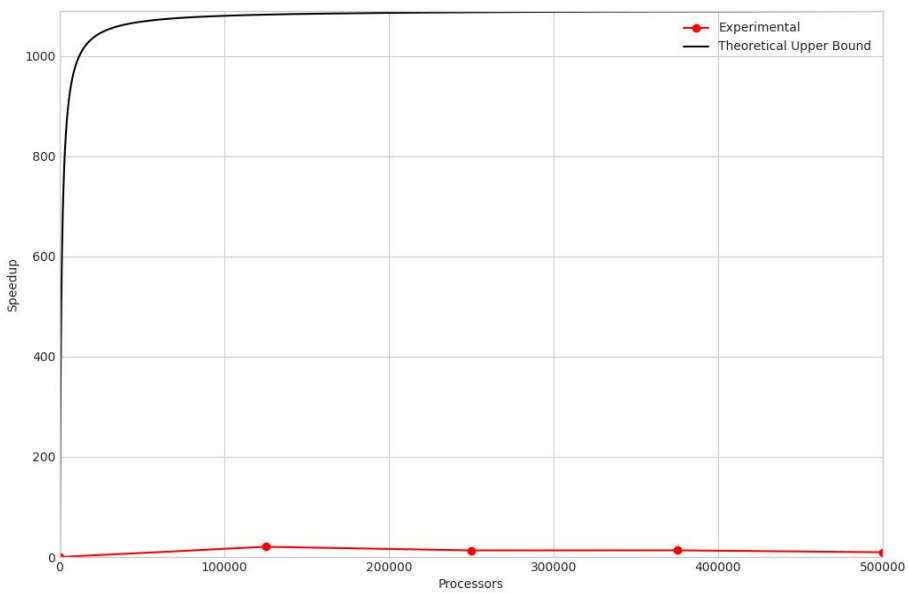
5.2.5.3 Optimization 2

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	236518.000	0.020	no data	127.986	no data	0.091	128.070	0.020	99.000	128.100	1.000	1.000
Parallel	125000	236518.000	0.063	0.000	6.015	0.126	0.112	6.261	0.063	99.000	6.350	20.173	0.000
Parallel	250000	236518.000	0.064	0.000	9.240	0.135	0.118	9.498	0.068	99.000	9.590	13.358	0.000
Parallel	375000	236518.000	0.065	0.000	9.118	0.149	0.124	9.391	0.073	99.000	9.496	13.489	0.000
Parallel	500000	236518.000	0.064	0.000	13.207	0.130	0.115	13.454	0.070	99.000	13.553	9.452	0.000



5.2.5.4 Optimization 3

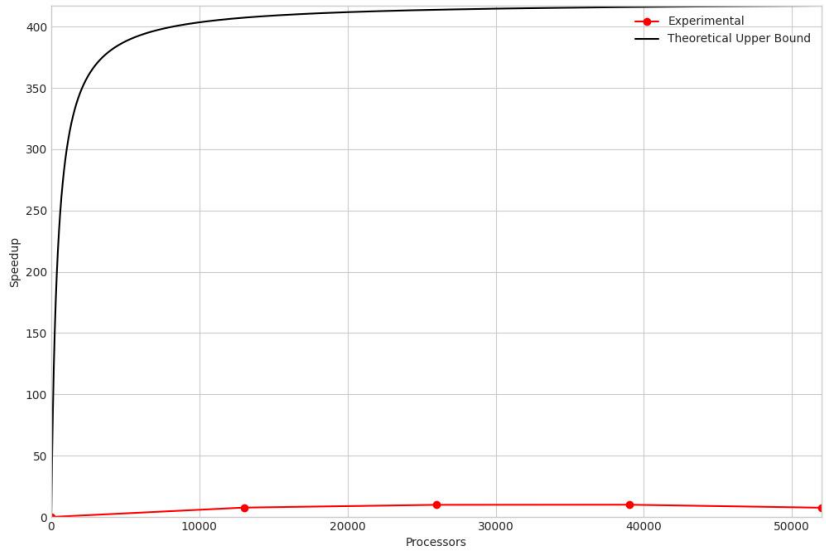
Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	236518.000	0.020	no data	128.266	no data	0.091	128.339	0.030	99.000	128.383	1.000	1.000
Parallel	125000	236518.000	0.064	0.000	5.924	0.147	0.121	6.196	0.070	99.000	6.291	20.407	0.000
Parallel	250000	236518.000	0.062	0.000	9.382	0.123	0.110	9.618	0.066	99.000	9.713	13.217	0.000
Parallel	375000	236518.000	0.065	0.000	9.246	0.137	0.118	9.506	0.068	99.000	9.604	13.368	0.000
Parallel	500000	236518.000	0.063	0.001	13.099	0.140	0.118	13.351	0.078	99.000	13.455	9.542	0.000



5.2.6 Tile-52000 graph

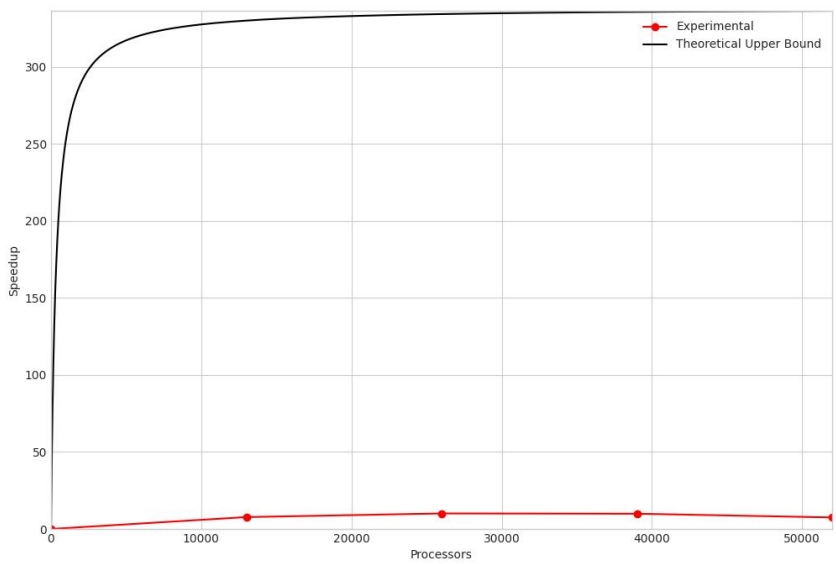
5.2.6.1 Optimization 0

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	25088.000	0.002	no data	1.602	no data	0.007	1.604	0.000	99.520	1.606	1.000	1.000
Parallel	13000	25088.000	0.037	0.001	0.139	0.007	0.007	0.154	0.038	93.960	0.210	7.633	0.001
Parallel	26000	25088.000	0.039	0.000	0.090	0.007	0.007	0.106	0.040	92.920	0.162	9.889	0.000
Parallel	39000	25088.000	0.038	0.000	0.090	0.007	0.007	0.105	0.040	93.360	0.161	9.988	0.000
Parallel	52000	25088.000	0.037	0.000	0.139	0.008	0.007	0.158	0.036	93.080	0.213	7.547	0.000



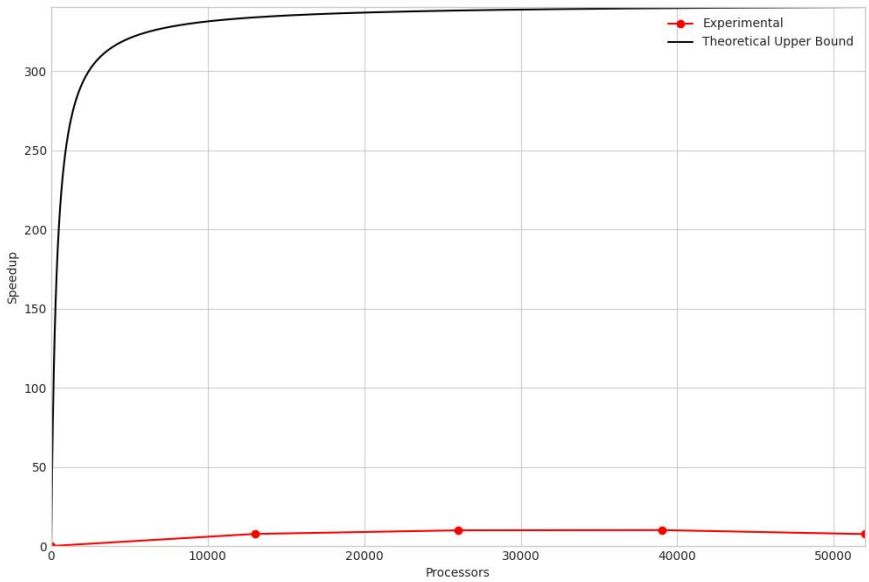
5.2.6.2 Optimization 1

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	25088.000	0.002	no data	1.605	no data	0.006	1.608	0.000	99.640	1.610	1.000	1.000
Parallel	13000	25088.000	0.037	0.000	0.139	0.007	0.007	0.155	0.039	94.360	0.209	7.711	0.001
Parallel	26000	25088.000	0.037	0.000	0.090	0.007	0.007	0.106	0.039	93.000	0.160	10.088	0.000
Parallel	39000	25088.000	0.038	0.000	0.090	0.007	0.007	0.106	0.040	92.320	0.162	9.914	0.000
Parallel	52000	25088.000	0.037	0.000	0.141	0.007	0.007	0.158	0.039	92.920	0.215	7.481	0.000



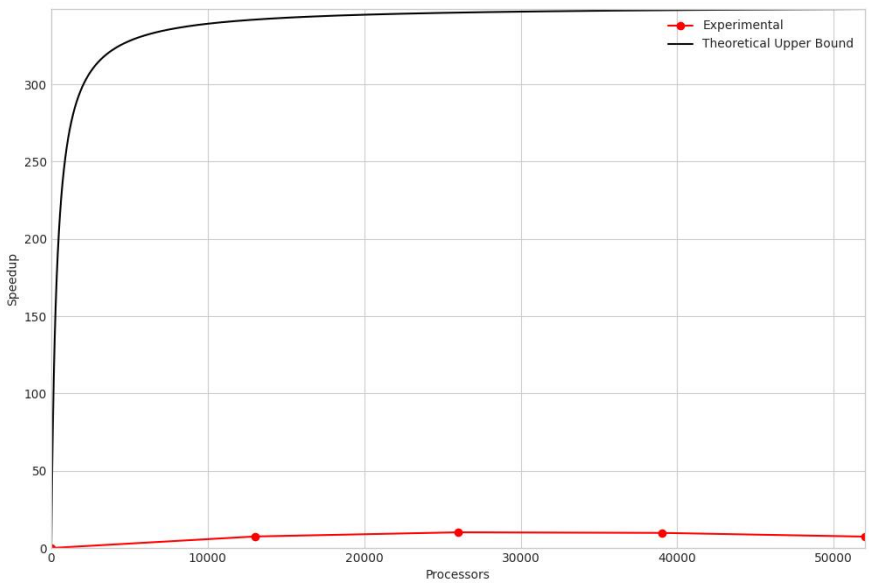
5.2.6.3 Optimization 2

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	25088.000	0.002	no data	1.605	no data	0.006	1.607	0.000	99.440	1.609	1.000	1.000
Parallel	13000	25088.000	0.037	0.000	0.139	0.007	0.007	0.154	0.039	94.000	0.211	7.619	0.001
Parallel	26000	25088.000	0.038	0.000	0.090	0.007	0.007	0.106	0.038	91.840	0.162	9.909	0.000
Parallel	39000	25088.000	0.037	0.000	0.090	0.007	0.007	0.104	0.038	92.760	0.160	10.058	0.000
Parallel	52000	25088.000	0.038	0.000	0.140	0.007	0.007	0.157	0.038	93.480	0.213	7.548	0.000



5.2.6.4 Optimization 3

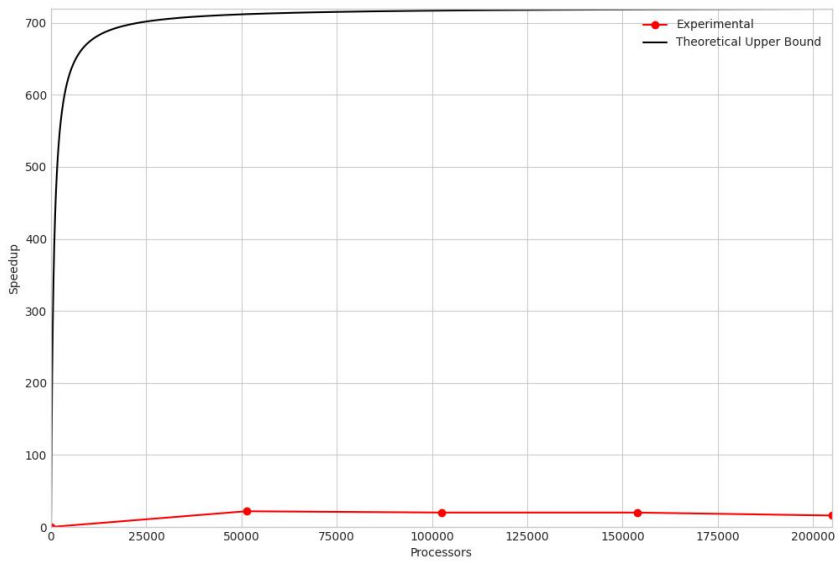
Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	25088.000	0.002	no data	1.592	no data	0.006	1.595	0.000	99.720	1.596	1.000	1.000
Parallel	13000	25088.000	0.038	0.000	0.139	0.008	0.007	0.155	0.040	93.000	0.214	7.446	0.001
Parallel	26000	25088.000	0.036	0.000	0.090	0.008	0.007	0.106	0.038	94.200	0.157	10.155	0.000
Parallel	39000	25088.000	0.037	0.000	0.090	0.007	0.007	0.107	0.038	91.400	0.163	9.782	0.000
Parallel	52000	25088.000	0.039	0.000	0.140	0.007	0.007	0.158	0.038	92.680	0.217	7.363	0.000



5.2.7 Tile-205000 graph

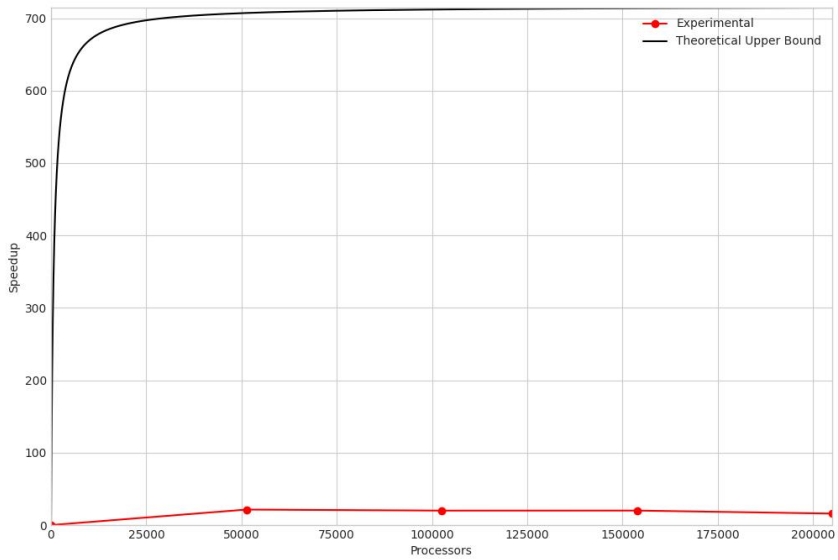
5.2.7.1 Optimization 0

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	100352.000	0.008	no data	25.436	no data	0.029	25.458	0.008	99.040	25.471	1.000	1.000
Parallel	51250	100352.000	0.045	0.001	1.028	0.030	0.030	1.092	0.047	97.760	1.165	21.867	0.000
Parallel	102500	100352.000	0.043	0.001	1.135	0.030	0.030	1.198	0.045	97.560	1.272	20.031	0.000
Parallel	153750	100352.000	0.044	0.001	1.136	0.030	0.030	1.200	0.046	97.800	1.272	20.031	0.000
Parallel	205000	100352.000	0.045	0.001	1.464	0.031	0.031	1.532	0.045	97.960	1.604	15.876	0.000



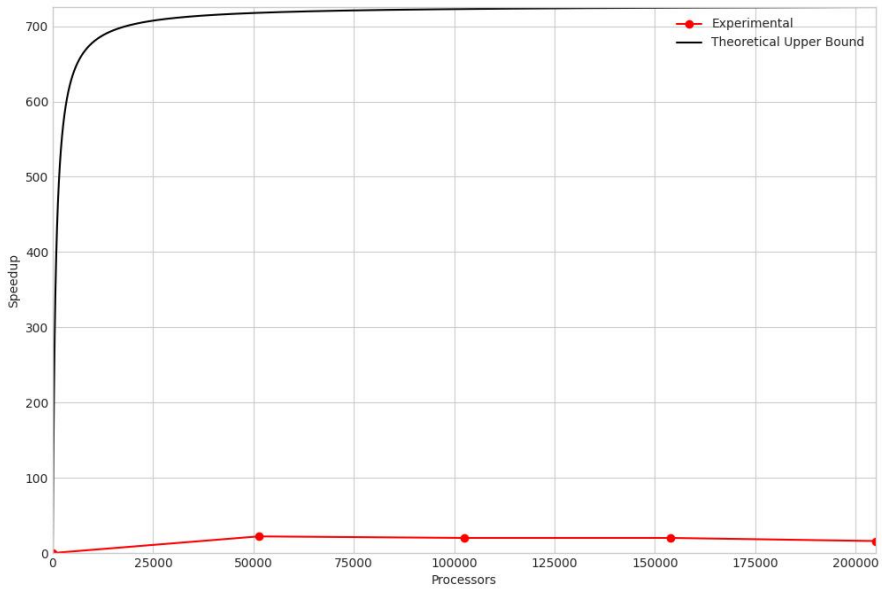
5.2.7.2 Optimization 1

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	100352.000	0.008	no data	25.458	no data	0.029	25.479	0.008	99.000	25.493	1.000	1.000
Parallel	51250	100352.000	0.045	0.001	1.056	0.029	0.029	1.120	0.047	97.720	1.190	21.416	0.000
Parallel	102500	100352.000	0.044	0.001	1.136	0.030	0.030	1.199	0.047	97.640	1.274	20.017	0.000
Parallel	153750	100352.000	0.043	0.001	1.136	0.030	0.030	1.201	0.043	97.760	1.270	20.067	0.000
Parallel	205000	100352.000	0.045	0.001	1.463	0.032	0.031	1.531	0.045	97.880	1.606	15.878	0.000



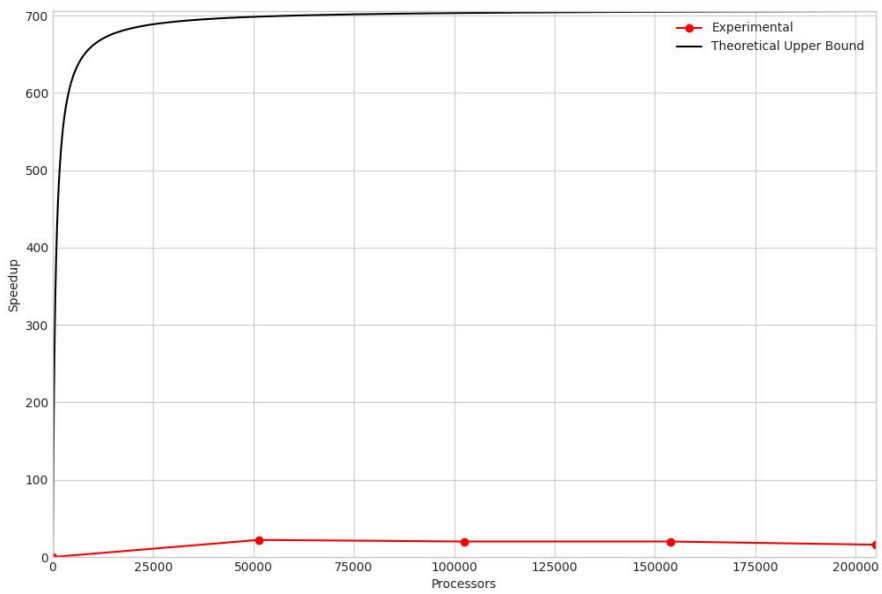
5.2.7.3 Optimization 2

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	100352.000	0.008	no data	25.469	no data	0.029	25.488	0.009	99.040	25.504	1.000	1.000
Parallel	51250	100352.000	0.043	0.001	1.021	0.029	0.029	1.085	0.043	97.640	1.155	22.085	0.000
Parallel	102500	100352.000	0.043	0.001	1.135	0.030	0.031	1.198	0.044	97.680	1.272	20.044	0.000
Parallel	153750	100352.000	0.044	0.001	1.134	0.030	0.030	1.196	0.046	97.600	1.272	20.057	0.000
Parallel	205000	100352.000	0.045	0.001	1.462	0.032	0.031	1.530	0.046	97.920	1.604	15.896	0.000



5.2.7.4 Optimization 3

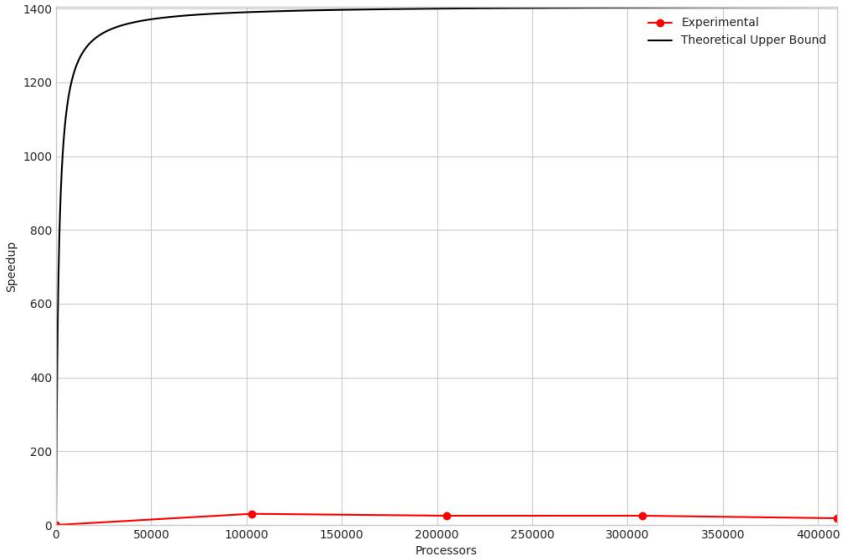
Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	100352.000	0.008	no data	25.455	no data	0.029	25.472	0.011	99.040	25.491	1.000	1.000
Parallel	51250	100352.000	0.043	0.001	1.022	0.029	0.029	1.082	0.046	97.800	1.155	22.074	0.000
Parallel	102500	100352.000	0.046	0.001	1.133	0.030	0.030	1.200	0.045	97.880	1.272	20.047	0.000
Parallel	153750	100352.000	0.043	0.001	1.134	0.030	0.030	1.196	0.046	97.600	1.272	20.047	0.000
Parallel	205000	100352.000	0.046	0.001	1.462	0.032	0.031	1.531	0.045	98.000	1.602	15.912	0.000



5.2.8 Tile-410000 graph

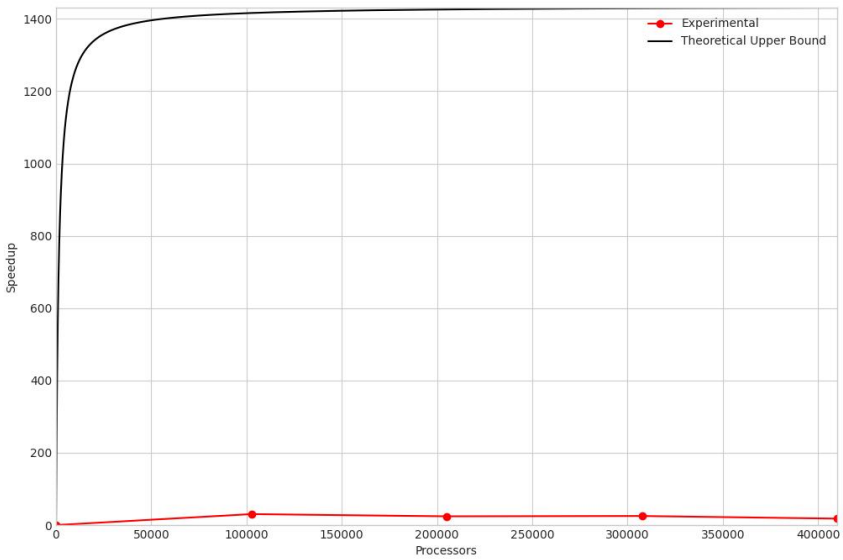
5.2.8.1 Optimization 0

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	200704.000	0.017	no data	101.611	no data	0.052	101.645	0.027	99.000	101.683	1.000	1.000
Parallel	102500	200704.000	0.061	0.001	3.114	0.077	0.067	3.268	0.062	99.000	3.354	30.317	0.000
Parallel	205000	200704.000	0.063	0.002	3.794	0.087	0.072	3.958	0.068	99.000	4.053	25.090	0.000
Parallel	307500	200704.000	0.064	0.002	3.789	0.087	0.073	3.959	0.066	99.000	4.050	25.107	0.000
Parallel	410000	200704.000	0.066	0.002	5.290	0.097	0.079	5.471	0.071	99.000	5.570	18.257	0.000



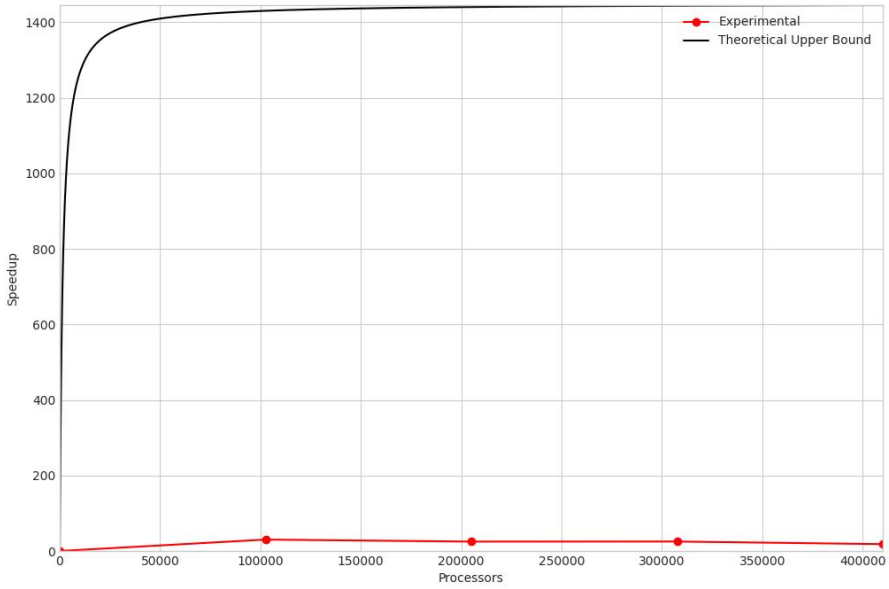
5.2.8.2 Optimization 1

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	200704.000	0.017	no data	101.562	no data	0.051	101.594	0.030	99.000	101.633	1.000	1.000
Parallel	102500	200704.000	0.060	0.001	3.112	0.077	0.067	3.259	0.066	99.000	3.351	30.331	0.000
Parallel	205000	200704.000	0.056	0.001	3.973	0.067	0.060	4.105	0.058	99.000	4.188	24.265	0.000
Parallel	307500	200704.000	0.062	0.002	3.790	0.086	0.073	3.952	0.068	99.000	4.047	25.112	0.000
Parallel	410000	200704.000	0.061	0.002	5.457	0.077	0.067	5.606	0.064	99.000	5.701	17.828	0.000



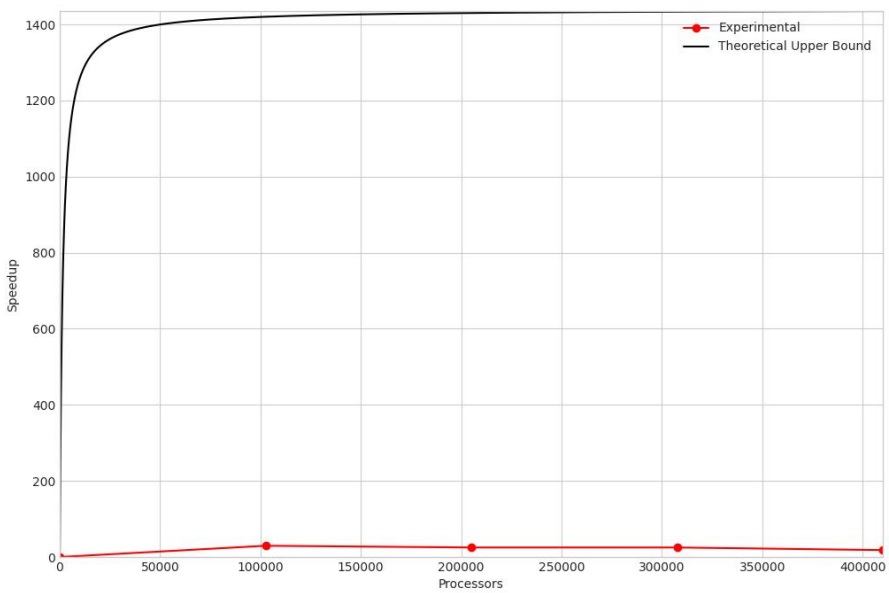
5.2.8.3 Optimization 2

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	200704.000	0.017	no data	101.449	no data	0.051	101.480	0.030	99.000	101.519	1.000	1.000
Parallel	102500	200704.000	0.059	0.001	3.120	0.076	0.066	3.263	0.065	98.960	3.356	30.246	0.000
Parallel	205000	200704.000	0.063	0.002	3.823	0.086	0.072	3.982	0.074	99.000	4.080	24.882	0.000
Parallel	307500	200704.000	0.062	0.002	3.790	0.087	0.072	3.951	0.069	99.000	4.047	25.086	0.000
Parallel	410000	200704.000	0.066	0.002	5.286	0.099	0.079	5.465	0.074	99.000	5.572	18.221	0.000



5.2.8.4 Optimization 3

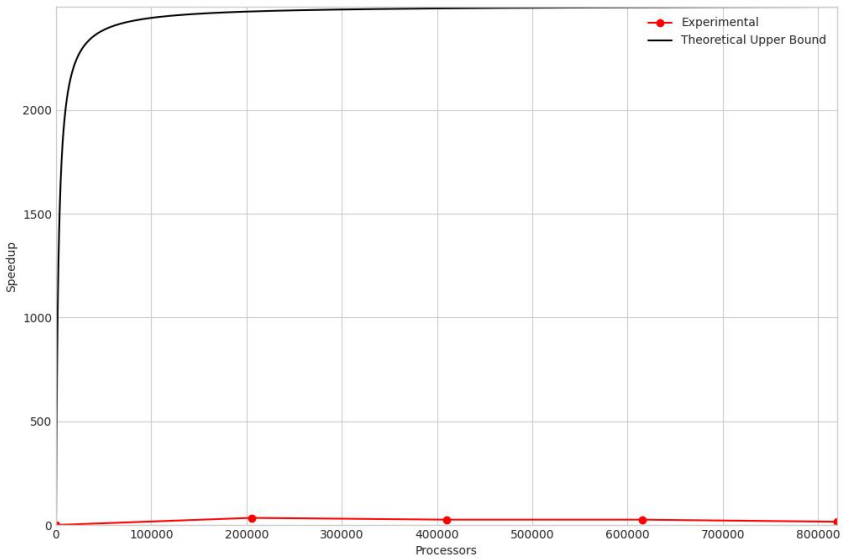
Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	200704.000	0.017	no data	101.432	no data	0.051	101.459	0.034	99.000	101.502	1.000	1.000
Parallel	102500	200704.000	0.053	0.001	3.229	0.061	0.056	3.349	0.057	98.960	3.433	29.565	0.000
Parallel	205000	200704.000	0.062	0.002	3.793	0.087	0.072	3.956	0.067	99.000	4.050	25.065	0.000
Parallel	307500	200704.000	0.063	0.002	3.793	0.088	0.073	3.962	0.067	99.000	4.053	25.042	0.000
Parallel	410000	200704.000	0.067	0.002	5.290	0.100	0.080	5.477	0.072	99.000	5.576	18.203	0.000



5.2.9 Tile-820000 graph

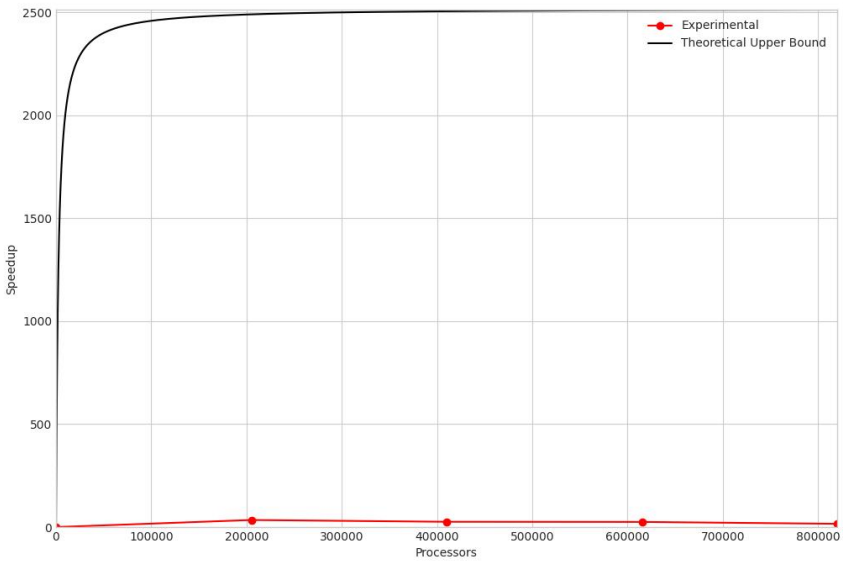
5.2.9.1 Optimization 0

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	401408.000	0.034	no data	403.937	no data	0.108	404.022	0.053	99.000	404.098	1.000	1.000
Parallel	205000	401408.000	0.082	0.003	11.333	0.187	0.150	11.677	0.089	99.000	11.791	34.272	0.000
Parallel	410000	401408.000	0.082	0.003	15.347	0.188	0.148	15.685	0.092	99.000	15.799	25.578	0.000
Parallel	615000	401408.000	0.081	0.003	15.337	0.188	0.148	15.673	0.094	99.000	15.788	25.595	0.000
Parallel	820000	401408.000	0.075	0.003	25.590	0.118	0.118	25.830	0.080	99.000	25.944	15.576	0.000



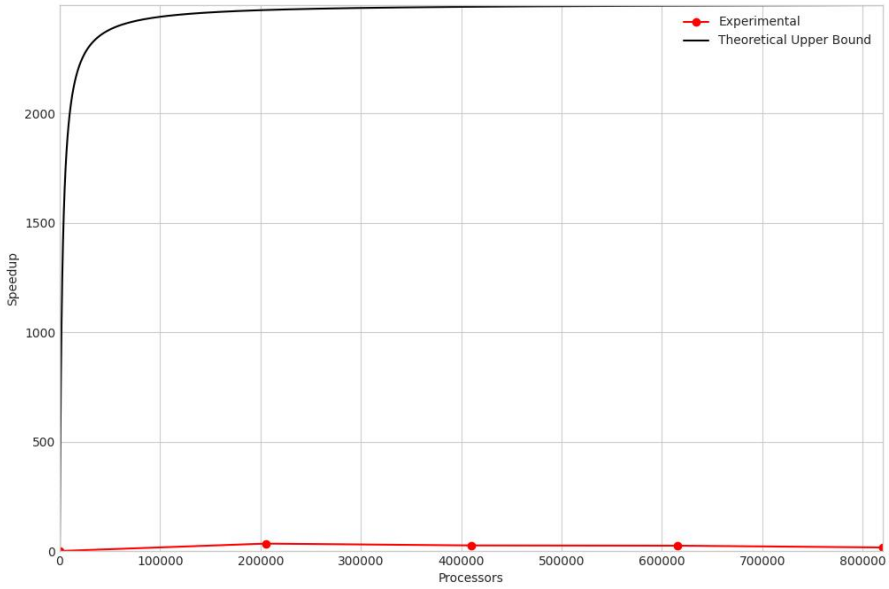
5.2.9.2 Optimization 1

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	401408.000	0.034	no data	404.900	no data	0.108	405.004	0.034	99.000	405.061	1.000	1.000
Parallel	205000	401408.000	0.081	0.003	11.333	0.188	0.149	11.666	0.096	99.000	11.787	34.364	0.000
Parallel	410000	401408.000	0.081	0.003	15.345	0.189	0.149	15.686	0.093	99.000	15.801	25.635	0.000
Parallel	615000	401408.000	0.078	0.003	15.808	0.153	0.134	16.100	0.087	99.000	16.210	24.988	0.000
Parallel	820000	401408.000	0.076	0.003	24.990	0.155	0.133	25.280	0.085	99.000	25.391	15.953	0.000



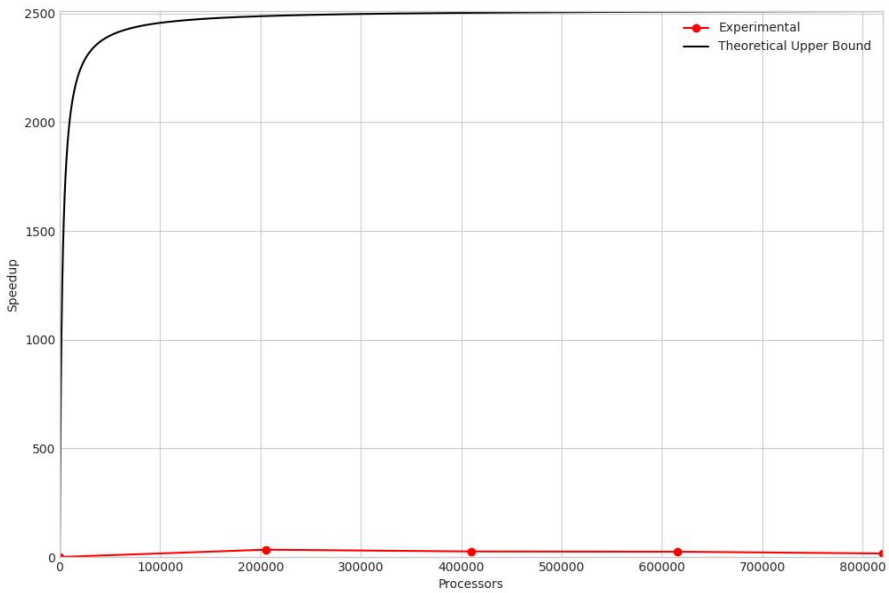
5.2.9.3 Optimization 2

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	401408.000	0.034	no data	404.260	no data	0.109	404.362	0.036	99.000	404.422	1.000	1.000
Parallel	205000	401408.000	0.079	0.003	11.508	0.173	0.142	11.832	0.082	99.000	11.938	33.876	0.000
Parallel	410000	401408.000	0.080	0.003	15.424	0.178	0.145	15.749	0.090	99.000	15.868	25.486	0.000
Parallel	615000	401408.000	0.072	0.003	16.217	0.117	0.117	16.457	0.077	99.000	16.559	24.423	0.000
Parallel	820000	401408.000	0.081	0.003	24.477	0.188	0.149	24.810	0.098	99.000	24.930	16.222	0.000



5.2.9.4 Optimization 3

Version	Threads	vertices	init	finalize	preprocess	conversion	tarjan	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	401408.000	0.034	no data	404.698	no data	0.109	404.793	0.042	99.000	404.859	1.000	1.000
Parallel	205000	401408.000	0.076	0.003	11.566	0.161	0.138	11.869	0.086	99.000	11.983	33.787	0.000
Parallel	410000	401408.000	0.081	0.003	15.348	0.189	0.149	15.686	0.093	99.000	15.800	25.624	0.000
Parallel	615000	401408.000	0.071	0.003	16.202	0.118	0.118	16.444	0.075	99.000	16.552	24.459	0.000
Parallel	820000	401408.000	0.084	0.003	24.489	0.190	0.152	24.829	0.101	99.000	24.946	16.229	0.000



5.3 Considerations

By comparing the obtained speedups with those of the CUDA implementation based on global memory, we conclude that, under most of the test conditions, the speedup was worsened by using the texture memory mechanism. Sometimes the speedup seems to be improved, but the improvement margin is too low to be considered statistically significant.

APPENDIX A

MPI MEASUREMENTS ON A RASPBERRY CLUSTER

A.1 Experimental results

In this appendix are presented the results of the parallel version of Tarjan's algorithm with MPI executed on the cluster of Raspberry. Each graph is tested with all the compiler optimization (O0, O1, O2, O3) and with 2, 4 and 8 processors MPI. In the next pages for each graph, set the optimization, there are a table and a graph of the speedup. In the table there are:

- **Version**: is the version of the program. It can be "Serial" if it's the sequential version of the program, or it can be "Parallel" if it's the parallel version with MPI.
- **Threads**: is the number of processors. For the "Serial" version, the value of this field is 1, because it's the sequential version of the program executed on only one processor of the CPU.
- **vertices**: is the number of vertices of the graph.
- **init**: is the time spent to initialize the program, in particular it's the time spent to create and initialize the data structures, and to read the graph from a file.
- **tarjan**: is the time spent to execute Tarjan's algorithm. In the "Parallel" version is the time it takes to run Tarjan's algorithm on each MPI slave node, along with the time it takes to partition the graph into chunks, exchange messages between processes, store found SCCs, and create supernodes.
- **split**: is the time spent to split the graph in the "Parallel" version. This phase is not provided by the sequential algorithm, in fact this field for this version is reported with the value "no data".

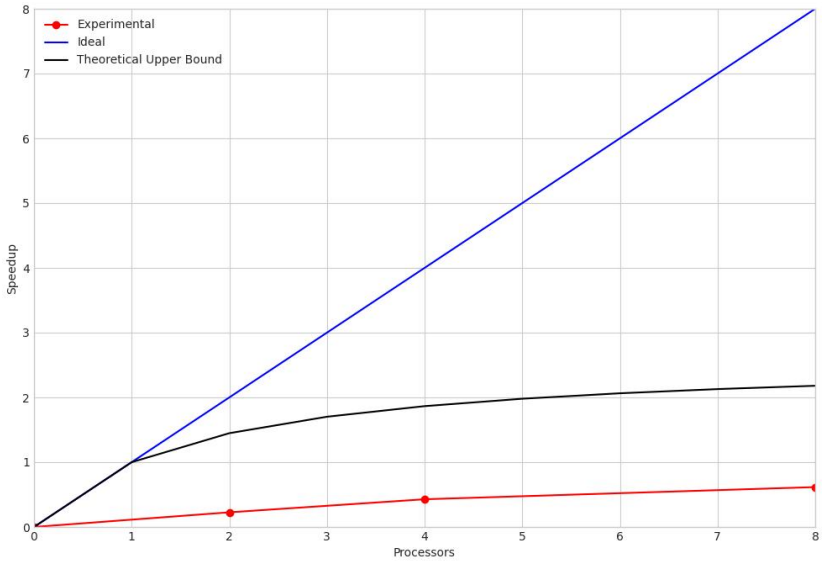
- **merge**: is the time spent to merge the SCCs found. This phase involves merging the nodes of an SCC found by the Tarjan's algorithm into a supernode, and storing the SCC found in a dedicated data structure. This phase is not provided by the sequential algorithm, in fact this field for this version is reported with the value "no data".
- **user**: is the total number of CPU-seconds that the process used directly (in user mode), in seconds.
- **system**: is the total number of CPU-seconds used by the system on behalf of the process (in kernel mode), in seconds.
- **pCPU**: is the percentage of the CPU that this process got. This is just user + system times divided by the total running time.
- **elapsed**: is the elapsed real (wall clock) time used by the process, in seconds.
- **Speedup**: is the speedup of the program.
- **Efficiency**: is the efficiency of the program.

All values except those in the `Version`, `Threads`, `Speedup`, and `Efficiency` columns are means. These means were calculated from the files of the measurements taken. `Speedup`, and `Efficiency` are calculated with the formulas in Cap.1 using the results of these means.

A.1.1 Fully disconnected graph

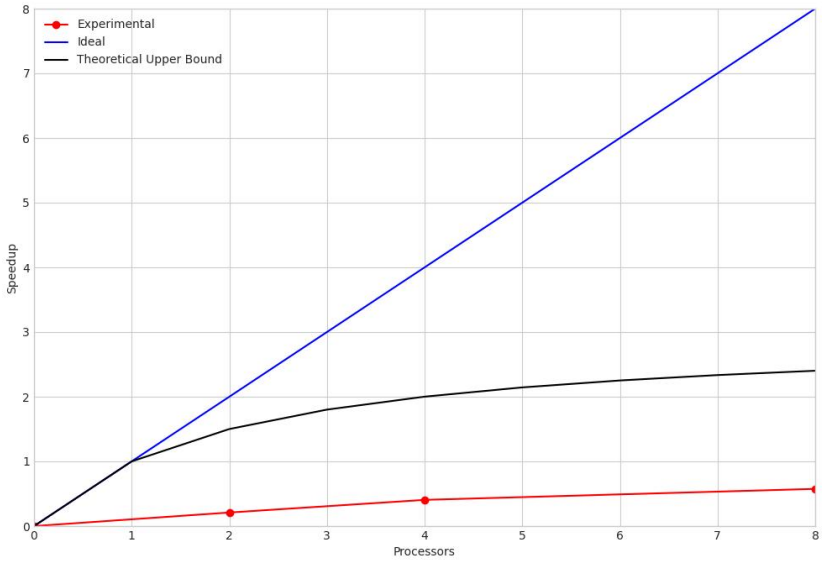
A.1.1.1 Optimization 0

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	1000000.000	1.612	3.494	no data	no data	4.927	0.205	92.960	5.649	1.000	1.000
Parallel	2	1000000.000	0.886	22.478	1.168	0.000	0.108	0.053	0.000	24.974	0.226	0.113
Parallel	4	1000000.000	0.896	10.383	1.200	0.000	0.100	0.057	0.920	13.210	0.428	0.107
Parallel	8	1000000.000	0.899	7.085	1.238	0.000	0.105	0.067	1.240	9.176	0.616	0.077



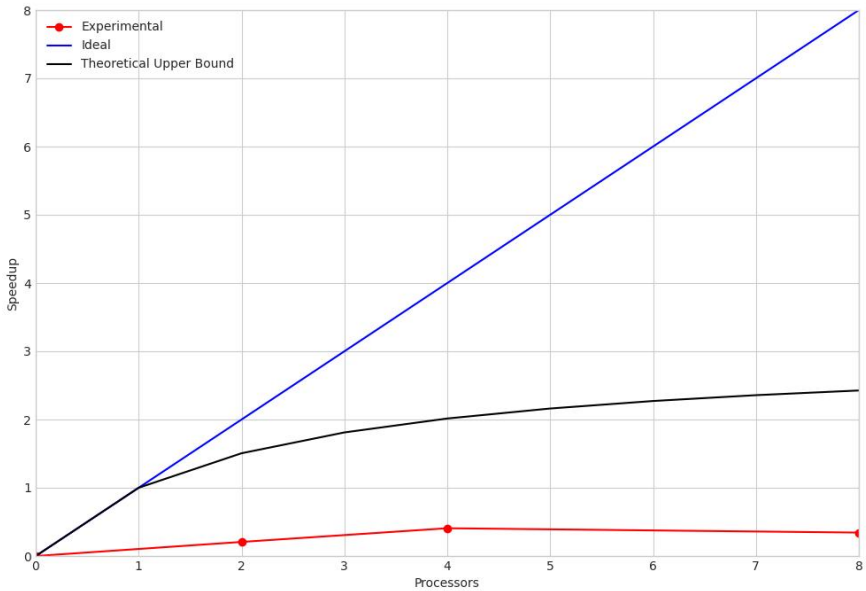
A.1.1.2 Optimization 1

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	1000000.000	1.611	3.496	no data	no data	4.919	0.208	97.720	5.241	1.000	1.000
Parallel	2	1000000.000	0.902	22.479	1.169	0.000	0.096	0.065	0.000	25.001	0.210	0.105
Parallel	4	1000000.000	0.900	10.388	1.202	0.000	0.096	0.063	0.960	12.939	0.405	0.101
Parallel	8	1000000.000	0.889	7.108	1.234	0.000	0.104	0.061	1.160	9.111	0.575	0.072



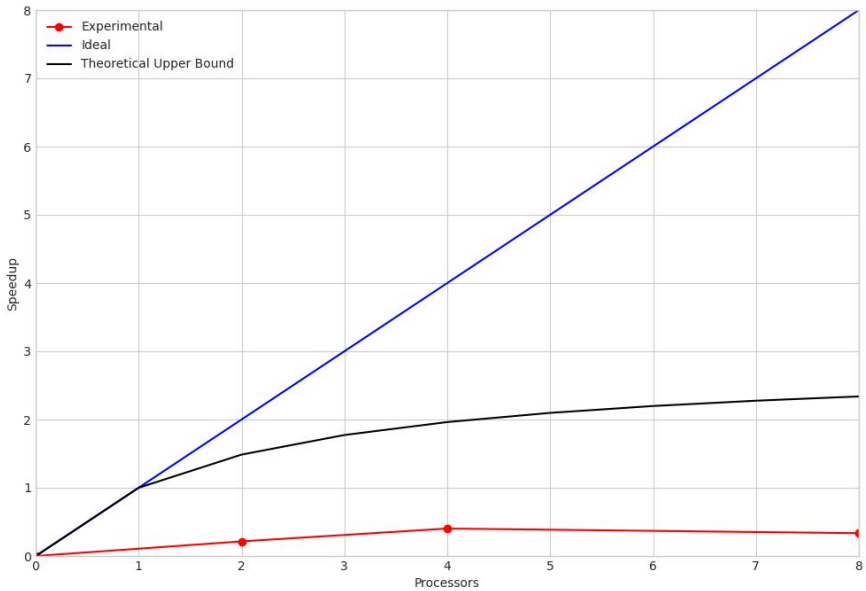
A.1.1.3 Optimization 2

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	1000000.000	1.609	3.498	no data	no data	4.927	0.201	98.000	5.208	1.000	1.000
Parallel	2	1000000.000	0.895	22.535	1.170	0.000	0.100	0.058	0.000	25.249	0.206	0.103
Parallel	4	1000000.000	0.893	10.364	1.202	0.000	0.100	0.057	1.000	12.797	0.407	0.102
Parallel	8	1000000.000	0.890	9.172	1.309	0.000	0.130	0.062	0.880	15.175	0.343	0.043



A.1.1.4 Optimization 3

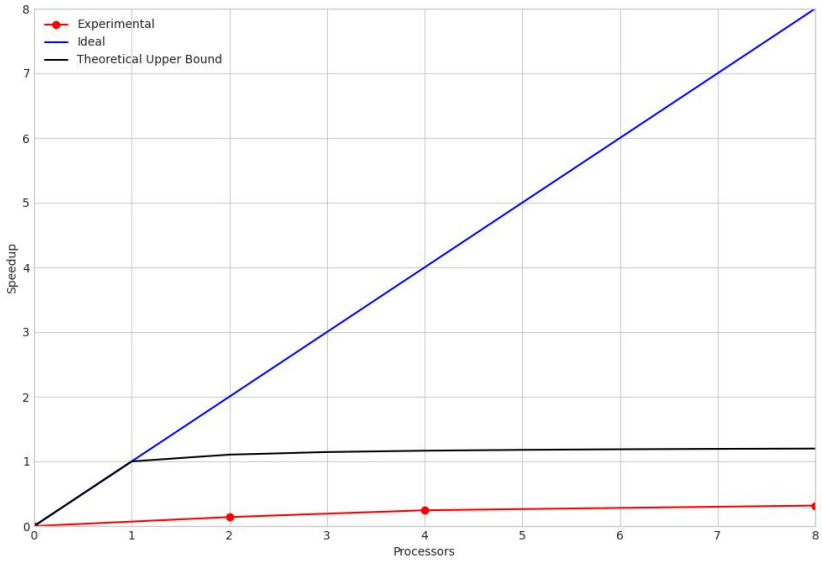
Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	1000000.000	1.608	3.493	no data	no data	4.920	0.204	96.400	5.340	1.000	1.000
Parallel	2	1000000.000	0.889	22.523	1.169	0.000	0.102	0.060	0.000	24.963	0.214	0.107
Parallel	4	1000000.000	0.894	10.384	1.201	0.000	0.104	0.054	0.920	13.274	0.402	0.101
Parallel	8	1000000.000	0.890	8.873	1.300	0.000	0.140	0.056	0.800	15.974	0.334	0.042



A.1.2 Random-150000 graph

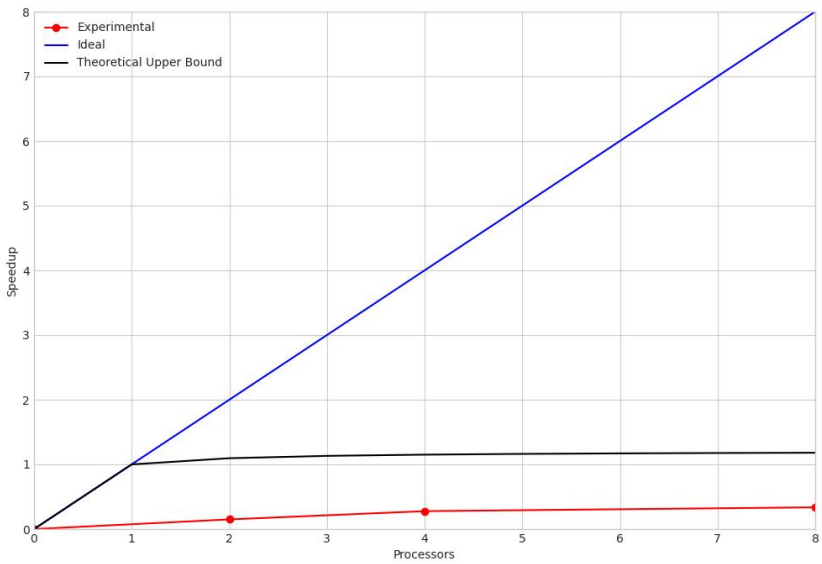
A.1.2.1 Optimization 0

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	150000.000	3.012	0.738	no data	no data	3.702	0.052	96.600	3.896	1.000	1.000
Parallel	2	150000.000	1.166	25.462	0.351	0.001	0.107	0.054	0.000	27.948	0.139	0.070
Parallel	4	150000.000	1.164	12.237	0.352	0.001	0.108	0.051	0.760	15.932	0.245	0.061
Parallel	8	150000.000	1.163	9.668	0.358	0.001	0.122	0.056	1.080	12.290	0.317	0.040



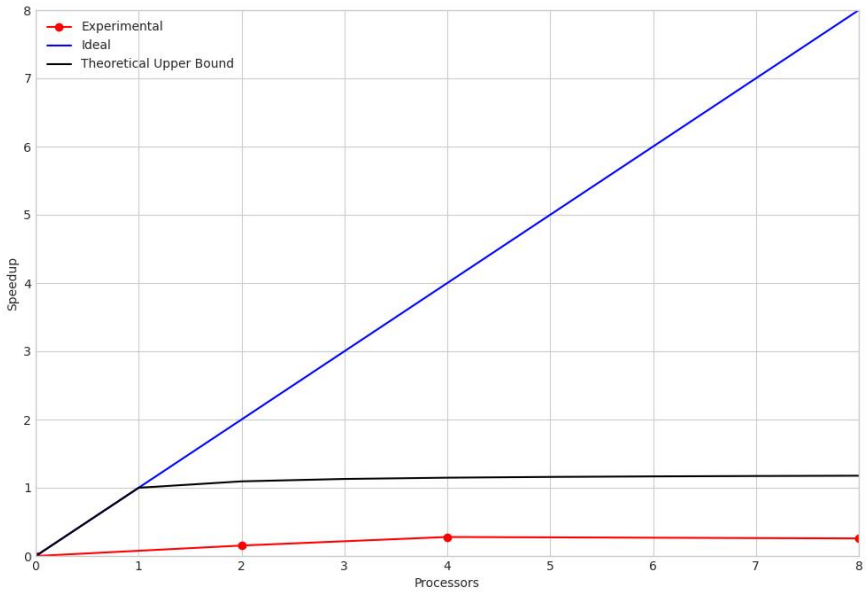
A.1.2.2 Optimization 1

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	150000.000	3.018	0.736	no data	no data	3.699	0.059	91.360	4.213	1.000	1.000
Parallel	2	150000.000	1.171	25.476	0.351	0.001	0.101	0.059	0.000	28.165	0.150	0.075
Parallel	4	150000.000	1.161	12.225	0.350	0.001	0.100	0.058	0.880	15.199	0.277	0.069
Parallel	8	150000.000	1.168	9.792	0.360	0.001	0.106	0.060	1.040	12.533	0.336	0.042



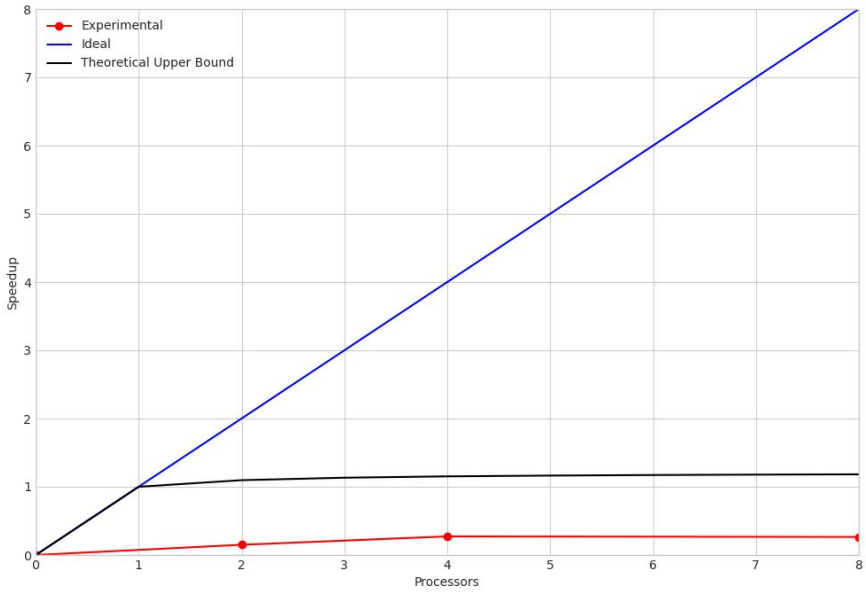
A.1.2.3 Optimization 2

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	150000.000	3.011	0.735	no data	no data	3.684	0.060	92.840	4.281	1.000	1.000
Parallel	2	150000.000	1.162	25.436	0.351	0.001	0.100	0.056	0.000	27.906	0.153	0.077
Parallel	4	150000.000	1.155	12.230	0.350	0.001	0.097	0.062	0.800	15.348	0.279	0.070
Parallel	8	150000.000	1.164	9.829	0.357	0.001	0.125	0.064	0.680	16.610	0.258	0.032



A.1.2.4 Optimization 3

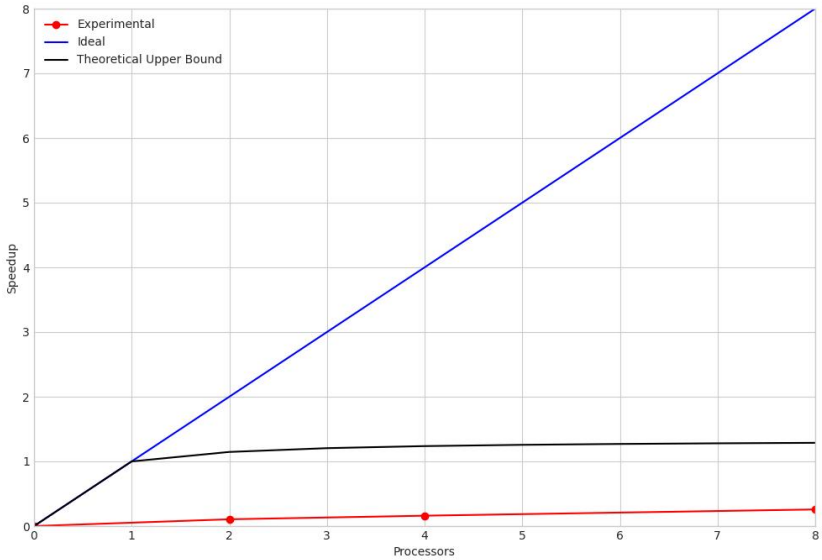
Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	150000.000	3.012	0.735	no data	no data	3.698	0.052	92.160	4.172	1.000	1.000
Parallel	2	150000.000	1.163	25.451	0.350	0.001	0.097	0.057	0.000	27.831	0.150	0.075
Parallel	4	150000.000	1.155	12.247	0.352	0.001	0.097	0.062	0.840	15.305	0.273	0.068
Parallel	8	150000.000	1.161	9.490	0.355	0.001	0.122	0.055	0.720	15.784	0.264	0.033



A.1.3 Random-250000 graph

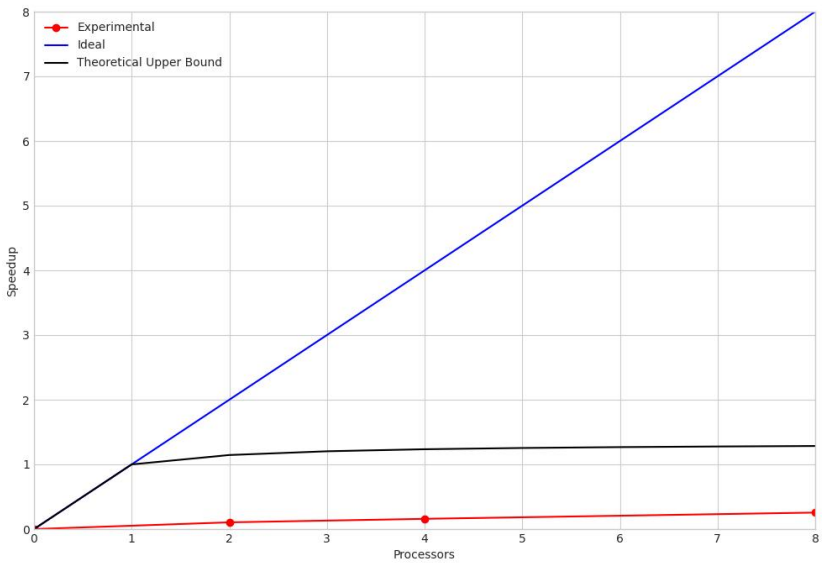
A.1.3.1 Optimization 0

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	250000.000	3.482	1.304	no data	no data	4.707	0.085	95.480	5.114	1.000	1.000
Parallel	2	250000.000	1.439	45.572	0.602	0.001	0.106	0.056	0.000	48.688	0.105	0.053
Parallel	4	250000.000	1.440	20.974	0.615	0.001	0.110	0.060	0.040	31.873	0.160	0.040
Parallel	8	250000.000	1.454	12.058	0.623	0.001	0.102	0.062	0.040	19.898	0.257	0.032



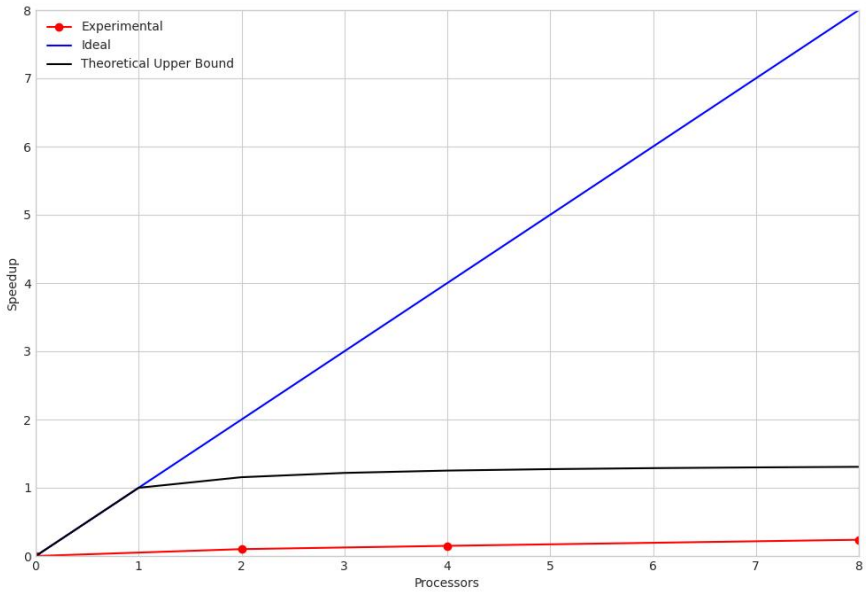
A.1.3.2 Optimization 1

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	250000.000	3.483	1.303	no data	no data	4.707	0.086	93.880	5.148	1.000	1.000
Parallel	2	250000.000	1.439	45.653	0.603	0.001	0.107	0.054	0.000	49.428	0.104	0.052
Parallel	4	250000.000	1.443	21.046	0.616	0.001	0.121	0.064	0.080	32.524	0.158	0.040
Parallel	8	250000.000	1.486	12.269	0.647	0.001	0.116	0.058	0.120	20.182	0.255	0.032



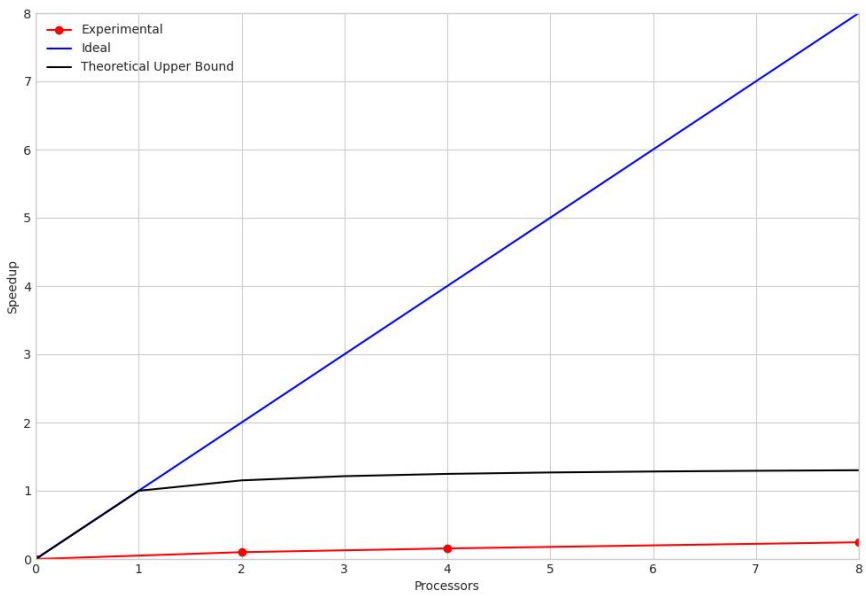
A.1.3.3 Optimization 2

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	250000.000	3.471	1.305	no data	no data	4.685	0.094	97.760	4.866	1.000	1.000
Parallel	2	250000.000	1.439	45.636	0.602	0.001	0.096	0.066	0.000	48.321	0.101	0.050
Parallel	4	250000.000	1.441	21.477	0.615	0.001	0.103	0.056	0.000	32.632	0.149	0.037
Parallel	8	250000.000	1.516	12.484	0.656	0.001	0.132	0.064	0.240	20.435	0.238	0.030



A.1.3.4 Optimization 3

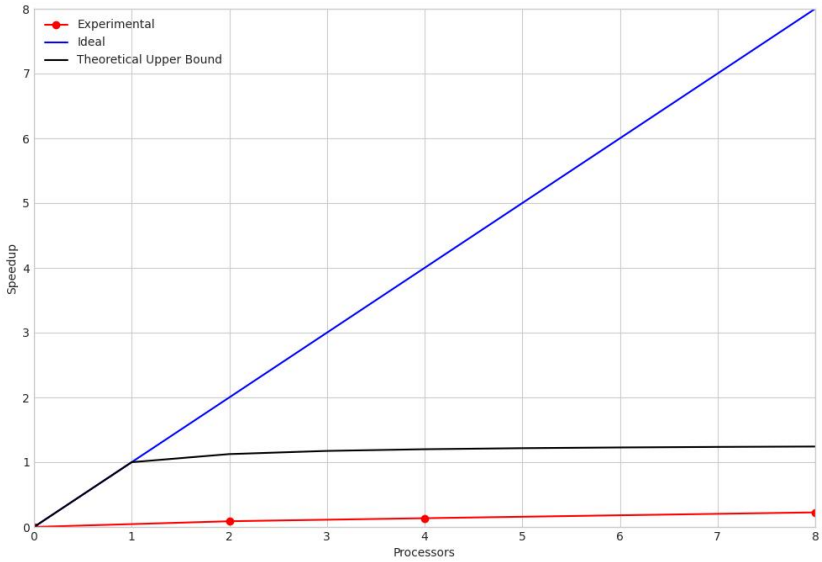
Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	250000.000	3.465	1.304	no data	no data	4.681	0.092	96.920	4.930	1.000	1.000
Parallel	2	250000.000	1.441	45.604	0.603	0.001	0.099	0.062	0.000	49.286	0.100	0.050
Parallel	4	250000.000	1.446	21.070	0.616	0.001	0.104	0.072	0.040	31.820	0.155	0.039
Parallel	8	250000.000	1.522	12.056	0.661	0.001	0.140	0.060	0.360	20.134	0.245	0.031



A.1.4 Random-500000 graph

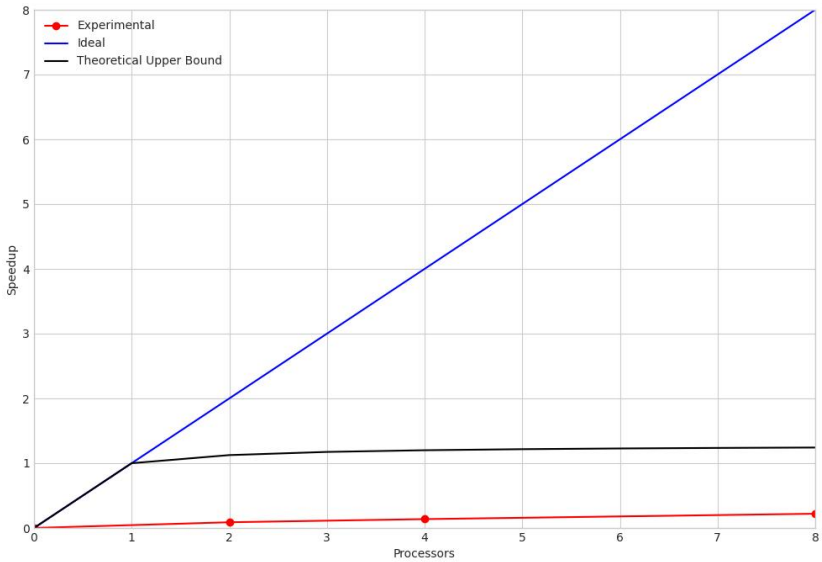
A.1.4.1 Optimization 0

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	500000.000	9.397	2.698	no data	no data	11.916	0.192	99.000	12.125	1.000	1.000
Parallel	2	500000.000	3.758	132.383	1.399	0.001	0.103	0.061	0.000	137.197	0.088	0.044
Parallel	4	500000.000	3.854	62.666	1.427	0.001	0.101	0.060	0.000	89.520	0.135	0.034
Parallel	8	500000.000	4.036	34.956	1.517	0.002	0.110	0.055	0.000	53.810	0.225	0.028



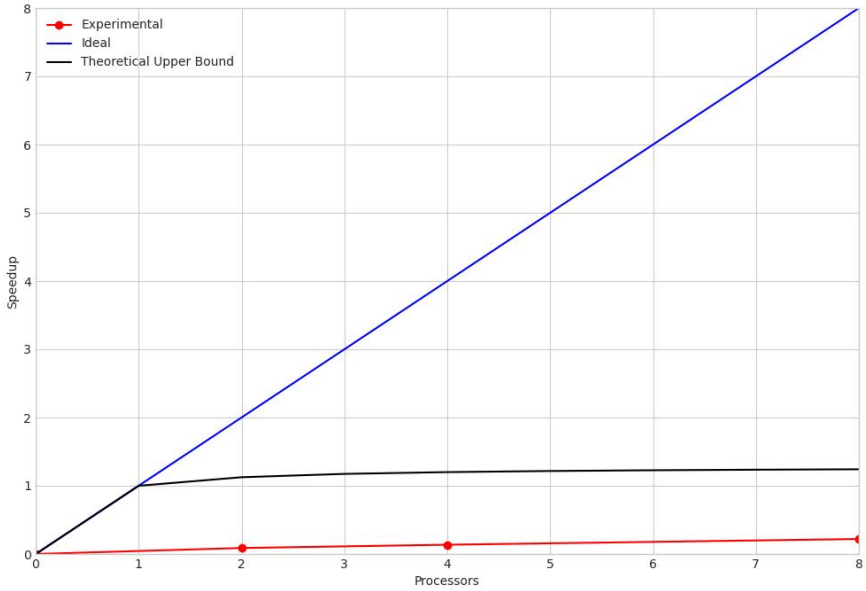
A.1.4.2 Optimization 1

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	500000.000	9.394	2.695	no data	no data	11.906	0.198	99.000	12.118	1.000	1.000
Parallel	2	500000.000	3.752	132.571	1.399	0.001	0.094	0.061	0.000	137.349	0.088	0.044
Parallel	4	500000.000	3.853	61.600	1.427	0.001	0.101	0.058	0.000	88.662	0.137	0.034
Parallel	8	500000.000	4.181	35.492	1.566	0.002	0.113	0.059	0.000	55.138	0.220	0.027



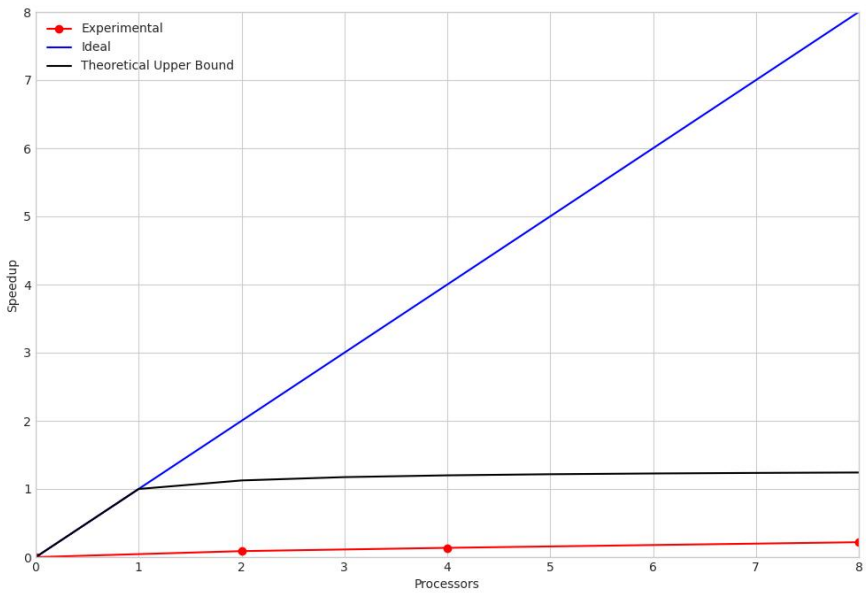
A.1.4.3 Optimization 2

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	500000.000	9.382	2.692	no data	no data	11.896	0.193	99.000	12.102	1.000	1.000
Parallel	2	500000.000	3.750	132.339	1.399	0.001	0.104	0.058	0.000	137.128	0.088	0.044
Parallel	4	500000.000	3.870	61.983	1.429	0.001	0.097	0.063	0.000	89.241	0.136	0.034
Parallel	8	500000.000	4.240	35.759	1.569	0.002	0.108	0.060	0.000	55.023	0.220	0.027



A.1.4.4 Optimization 3

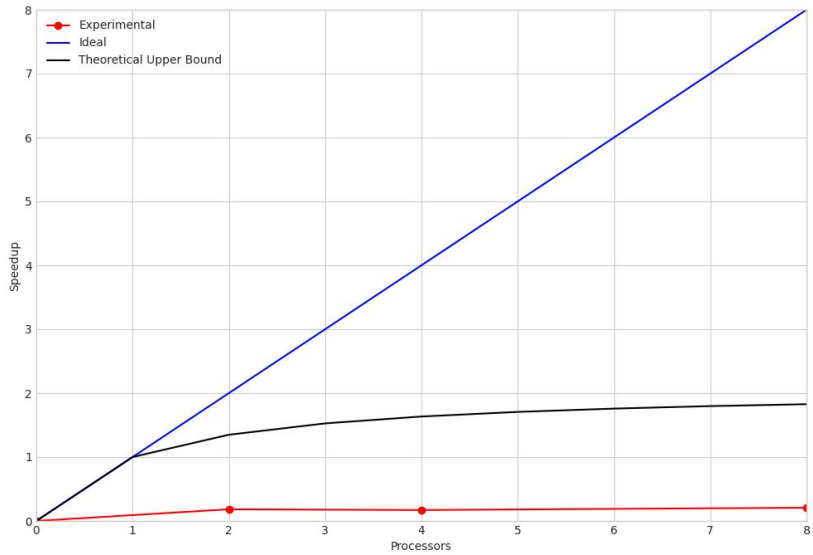
Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	500000.000	9.382	2.692	no data	no data	11.913	0.176	99.000	12.102	1.000	1.000
Parallel	2	500000.000	3.749	132.644	1.400	0.001	0.104	0.058	0.000	137.420	0.088	0.044
Parallel	4	500000.000	3.864	61.257	1.429	0.001	0.101	0.060	0.000	88.838	0.136	0.034
Parallel	8	500000.000	4.248	36.189	1.580	0.002	0.111	0.056	0.000	55.312	0.219	0.027



A.1.5 Tile-52000 graph

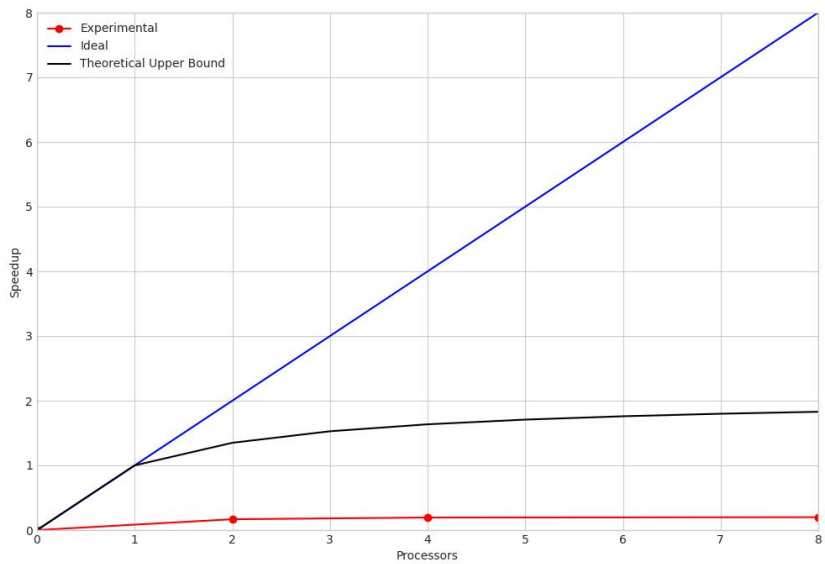
A.1.5.1 Optimization 0

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	51200.000	0.220	0.243	no data	no data	0.447	0.017	99.000	0.470	1.000	1.000
Parallel	2	51200.000	0.121	1.137	0.071	0.048	0.103	0.053	7.000	2.573	0.183	0.091
Parallel	4	51200.000	0.122	0.816	0.070	0.049	0.106	0.060	7.120	2.747	0.171	0.043
Parallel	8	51200.000	0.122	0.712	0.072	0.052	0.118	0.062	8.200	2.270	0.207	0.026



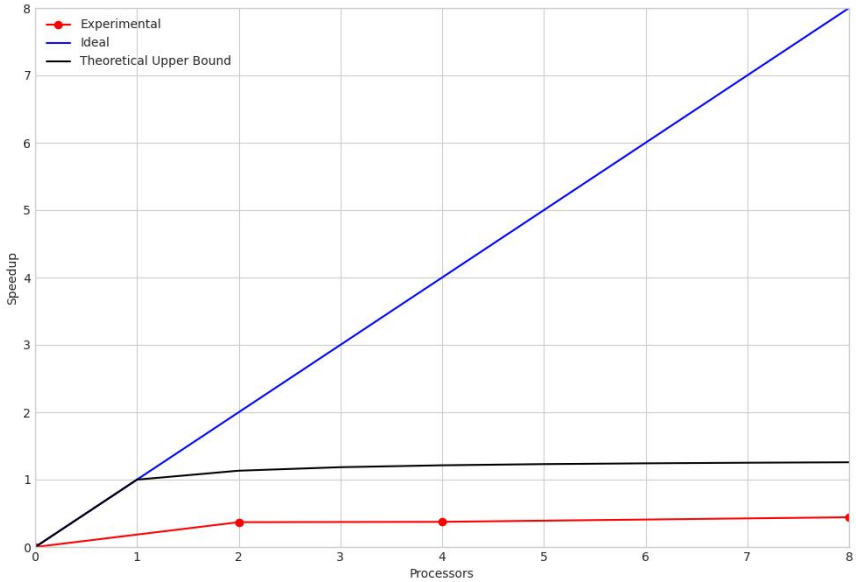
A.1.5.2 Optimization 1

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	51200.000	0.220	0.243	no data	no data	0.444	0.020	99.000	0.470	1.000	1.000
Parallel	2	51200.000	0.122	1.135	0.070	0.048	0.095	0.061	6.600	2.807	0.167	0.084
Parallel	4	51200.000	0.121	0.766	0.070	0.049	0.112	0.055	7.200	2.420	0.194	0.049
Parallel	8	51200.000	0.121	0.721	0.071	0.051	0.120	0.070	8.720	2.361	0.199	0.025



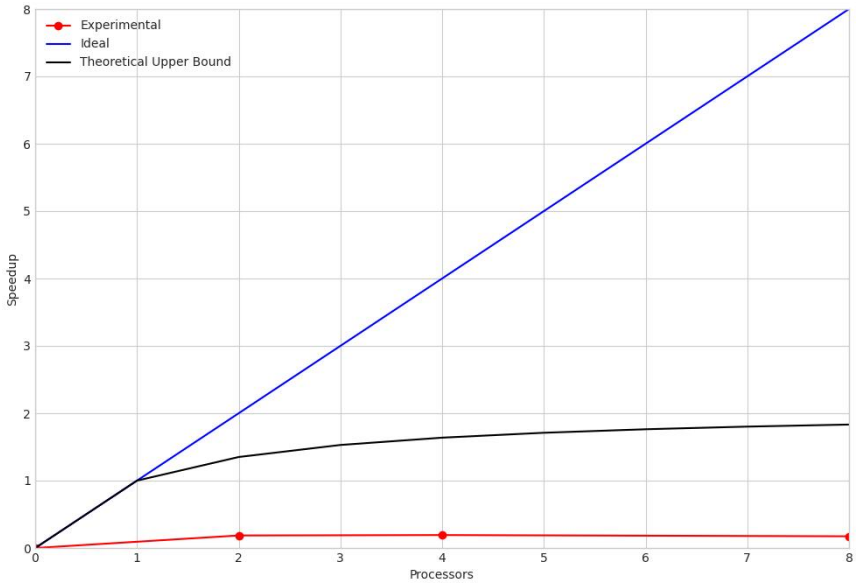
A.1.5.3 Optimization 2

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	51200.000	0.224	0.244	no data	no data	0.452	0.021	85.600	1.042	1.000	1.000
Parallel	2	51200.000	0.122	1.133	0.070	0.048	0.097	0.060	6.440	2.828	0.368	0.184
Parallel	4	51200.000	0.121	0.845	0.070	0.049	0.109	0.064	7.280	2.790	0.373	0.093
Parallel	8	51200.000	0.122	0.657	0.071	0.050	0.116	0.057	8.160	2.358	0.442	0.055



A.1.5.4 Optimization 3

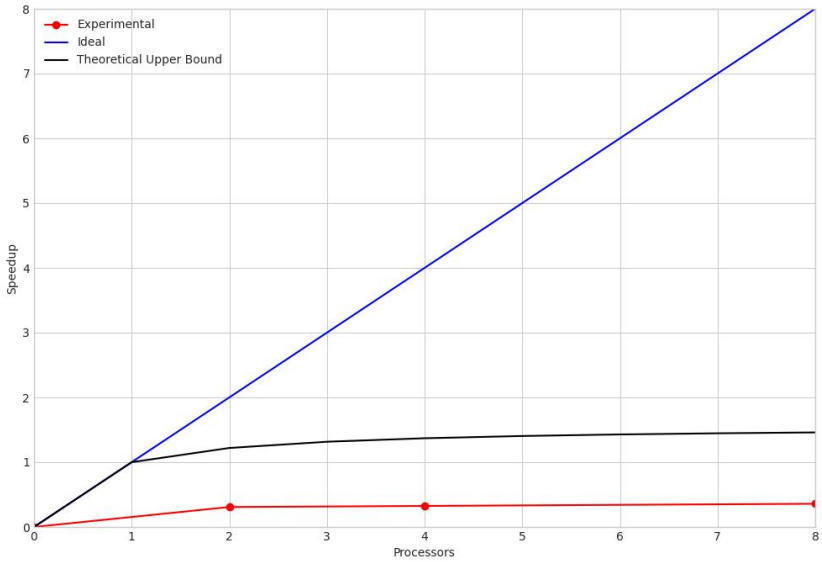
Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	51200.000	0.220	0.245	no data	no data	0.445	0.019	98.920	0.471	1.000	1.000
Parallel	2	51200.000	0.121	1.135	0.070	0.048	0.102	0.054	6.840	2.536	0.186	0.093
Parallel	4	51200.000	0.122	0.786	0.070	0.049	0.099	0.062	6.880	2.454	0.192	0.048
Parallel	8	51200.000	0.121	0.711	0.071	0.050	0.108	0.064	7.480	2.710	0.174	0.022



A.1.6 Tile-205000 graph

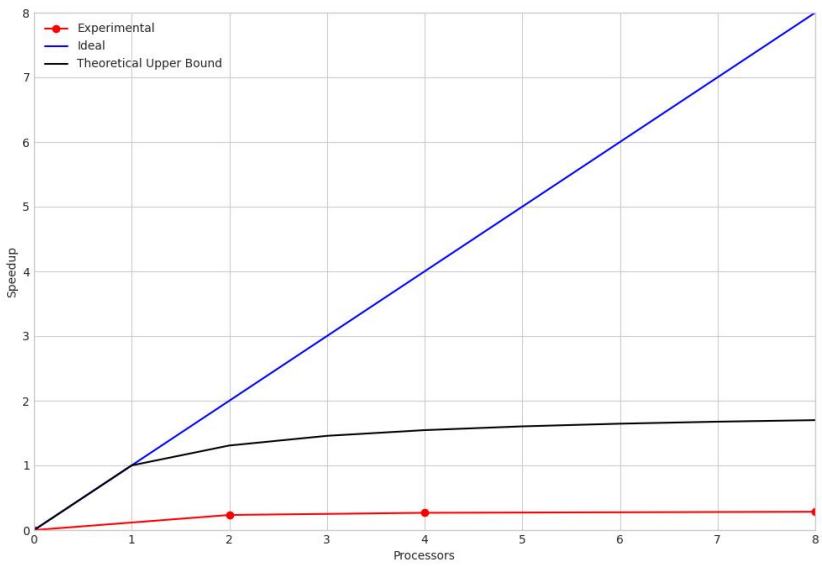
A.1.6.1 Optimization 0

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	204800.000	0.880	0.965	no data	no data	1.777	0.084	86.040	2.682	1.000	1.000
Parallel	2	204800.000	0.486	6.414	0.367	0.188	0.104	0.055	1.720	8.702	0.308	0.154
Parallel	4	204800.000	0.482	3.923	0.365	0.191	0.103	0.055	1.680	8.263	0.325	0.081
Parallel	8	204800.000	0.485	3.128	0.373	0.201	0.147	0.060	2.560	7.484	0.358	0.045



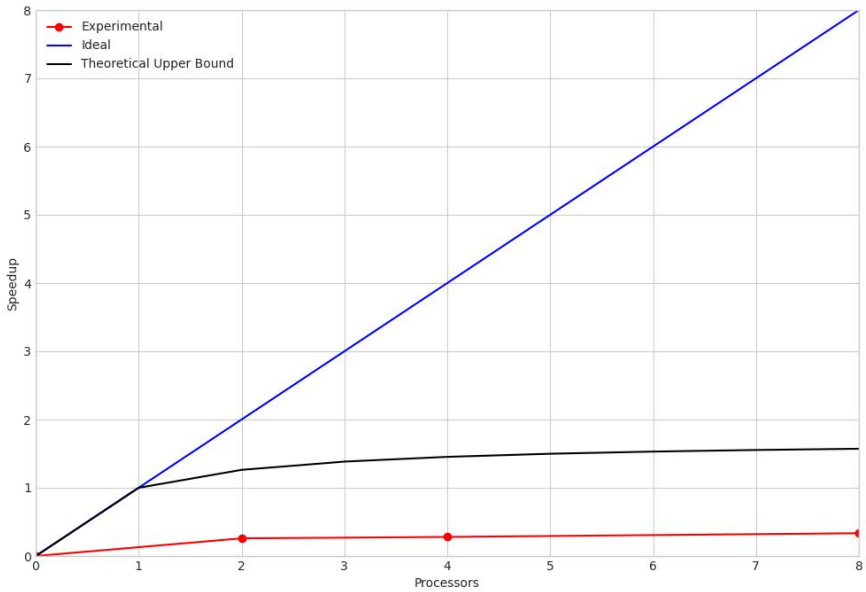
A.1.6.2 Optimization 1

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	204800.000	0.879	0.965	no data	no data	1.782	0.078	95.280	2.052	1.000	1.000
Parallel	2	204800.000	0.482	6.409	0.366	0.187	0.102	0.056	1.720	8.786	0.234	0.117
Parallel	4	204800.000	0.482	3.993	0.366	0.190	0.094	0.063	1.880	7.683	0.267	0.067
Parallel	8	204800.000	0.484	2.971	0.372	0.200	0.142	0.062	2.760	7.235	0.284	0.035



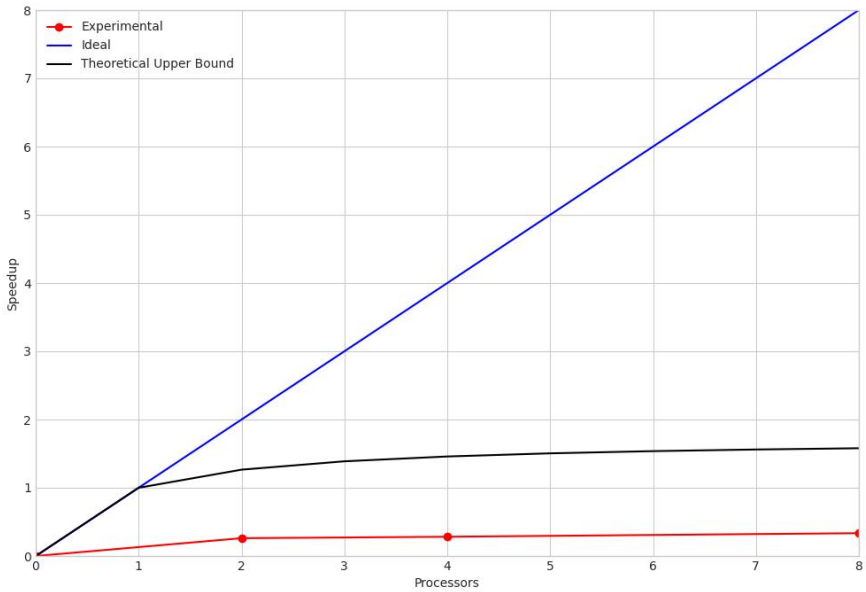
A.1.6.3 Optimization 2

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	204800.000	0.879	0.967	no data	no data	1.784	0.080	93.520	2.325	1.000	1.000
Parallel	2	204800.000	0.482	6.393	0.366	0.188	0.092	0.066	1.640	8.995	0.258	0.129
Parallel	4	204800.000	0.484	4.007	0.366	0.191	0.102	0.058	1.800	8.321	0.279	0.070
Parallel	8	204800.000	0.485	3.019	0.377	0.201	0.157	0.070	3.120	6.960	0.334	0.042



A.1.6.4 Optimization 3

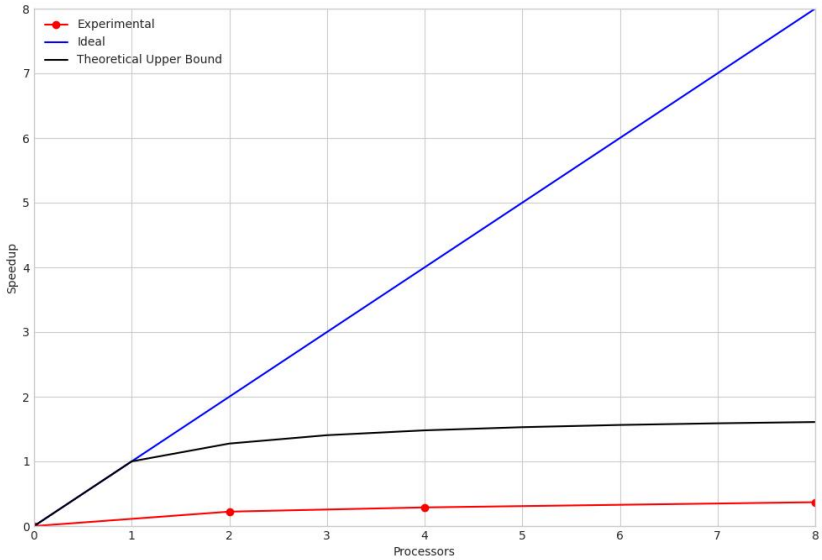
Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	204800.000	0.880	0.966	no data	no data	1.789	0.076	93.160	2.305	1.000	1.000
Parallel	2	204800.000	0.484	6.421	0.367	0.188	0.105	0.051	1.640	8.829	0.261	0.131
Parallel	4	204800.000	0.484	3.853	0.366	0.191	0.098	0.060	1.640	8.174	0.282	0.071
Parallel	8	204800.000	0.484	3.175	0.386	0.205	0.144	0.069	2.960	6.890	0.335	0.042



A.1.7 Tile-410000 graph

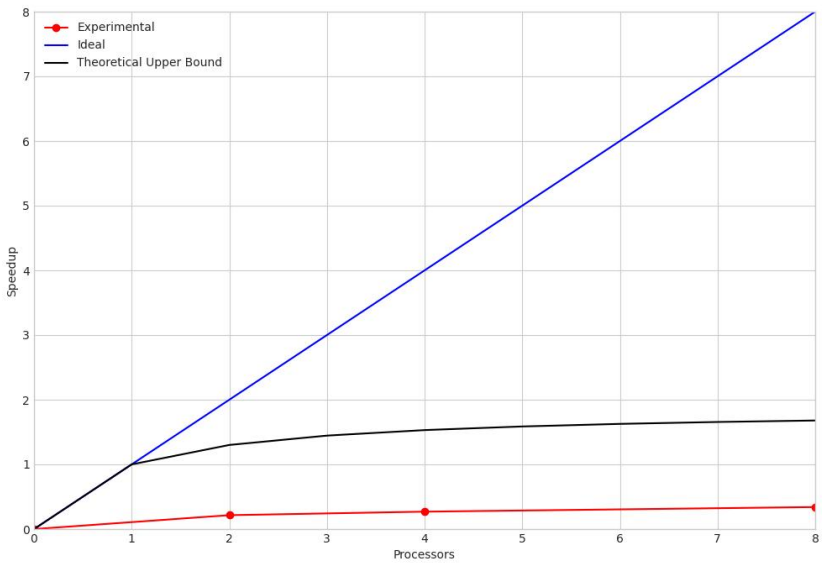
A.1.7.1 Optimization 0

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	409600.000	1.756	1.932	no data	no data	3.568	0.155	92.160	4.466	1.000	1.000
Parallel	2	409600.000	0.961	15.067	0.818	0.371	0.095	0.060	0.120	20.007	0.223	0.112
Parallel	4	409600.000	0.961	8.834	0.818	0.380	0.101	0.060	0.760	15.468	0.289	0.072
Parallel	8	409600.000	0.968	6.727	0.841	0.413	0.156	0.060	1.400	12.095	0.369	0.046



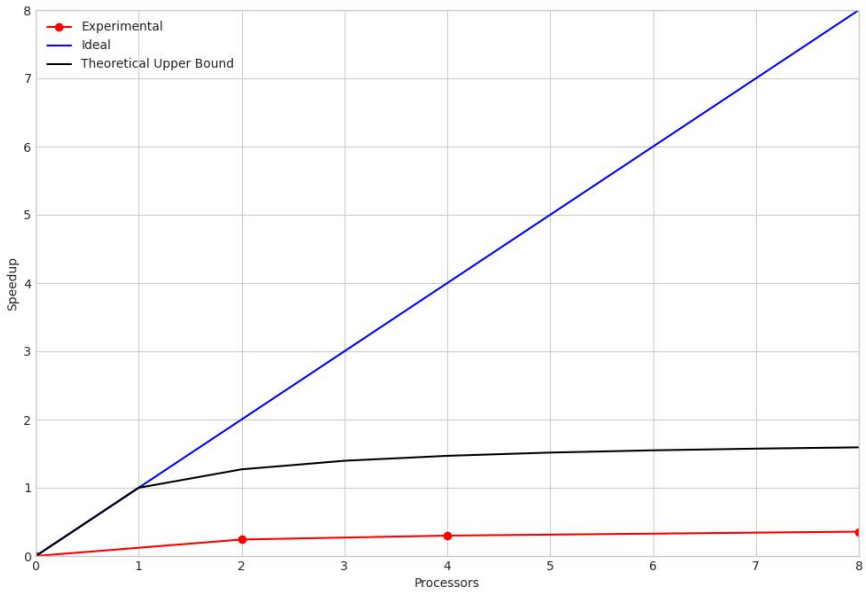
A.1.7.2 Optimization 1

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	409600.000	1.756	1.927	no data	no data	3.556	0.159	94.680	4.170	1.000	1.000
Parallel	2	409600.000	0.969	15.013	0.818	0.373	0.101	0.058	0.240	19.396	0.215	0.107
Parallel	4	409600.000	0.962	8.797	0.817	0.381	0.097	0.060	0.800	15.481	0.269	0.067
Parallel	8	409600.000	0.984	6.774	0.877	0.431	0.162	0.075	1.520	12.251	0.340	0.043



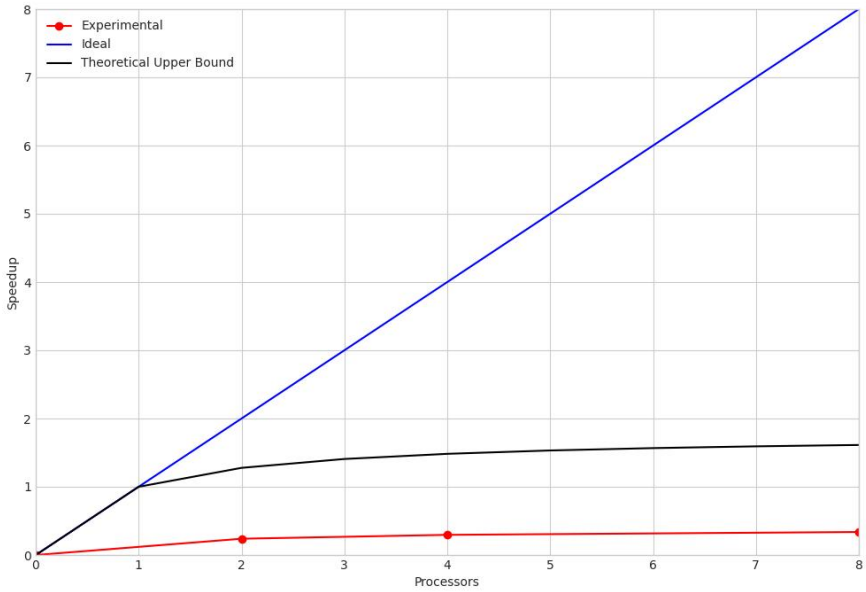
A.1.7.3 Optimization 2

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	409600.000	1.756	1.930	no data	no data	3.572	0.150	90.680	4.538	1.000	1.000
Parallel	2	409600.000	0.962	15.039	0.818	0.371	0.098	0.060	0.120	18.793	0.241	0.121
Parallel	4	409600.000	0.962	8.669	0.819	0.381	0.096	0.062	0.800	15.179	0.299	0.075
Parallel	8	409600.000	1.002	7.119	0.897	0.430	0.179	0.068	1.600	12.720	0.357	0.045



A.1.7.4 Optimization 3

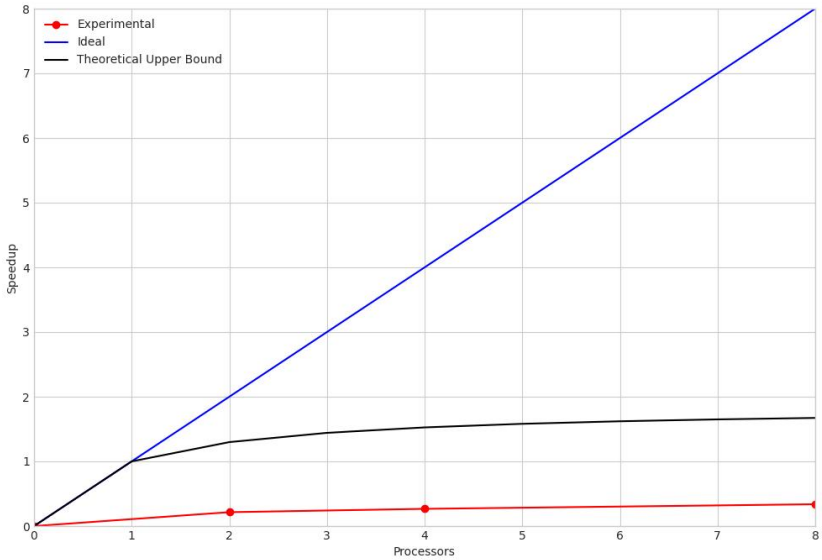
Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	409600.000	1.759	1.927	no data	no data	3.564	0.156	92.360	4.440	1.000	1.000
Parallel	2	409600.000	0.963	15.083	0.819	0.371	0.092	0.062	0.120	18.621	0.238	0.119
Parallel	4	409600.000	0.965	8.544	0.819	0.381	0.093	0.061	0.800	15.032	0.295	0.074
Parallel	8	409600.000	1.007	7.066	0.904	0.445	0.161	0.069	1.480	13.169	0.337	0.042



A.1.8 Tile-820000 graph

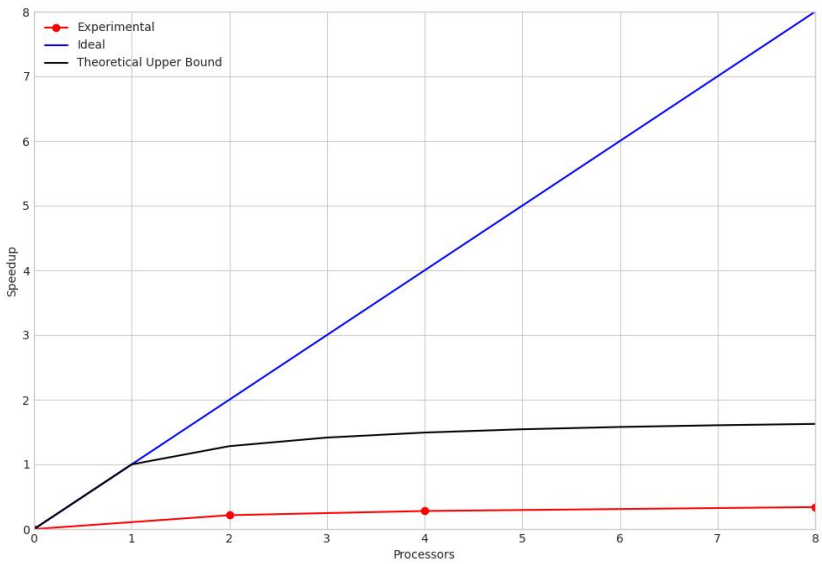
A.1.8.1 Optimization 0

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	819200.000	3.508	3.862	no data	no data	7.127	0.335	91.400	8.406	1.000	1.000
Parallel	2	819200.000	1.942	35.288	1.802	0.742	0.082	0.054	0.000	39.101	0.215	0.107
Parallel	4	819200.000	1.941	19.961	1.817	0.769	0.122	0.053	0.120	31.562	0.266	0.067
Parallel	8	819200.000	2.011	15.540	1.946	0.889	0.135	0.059	0.280	24.911	0.337	0.042



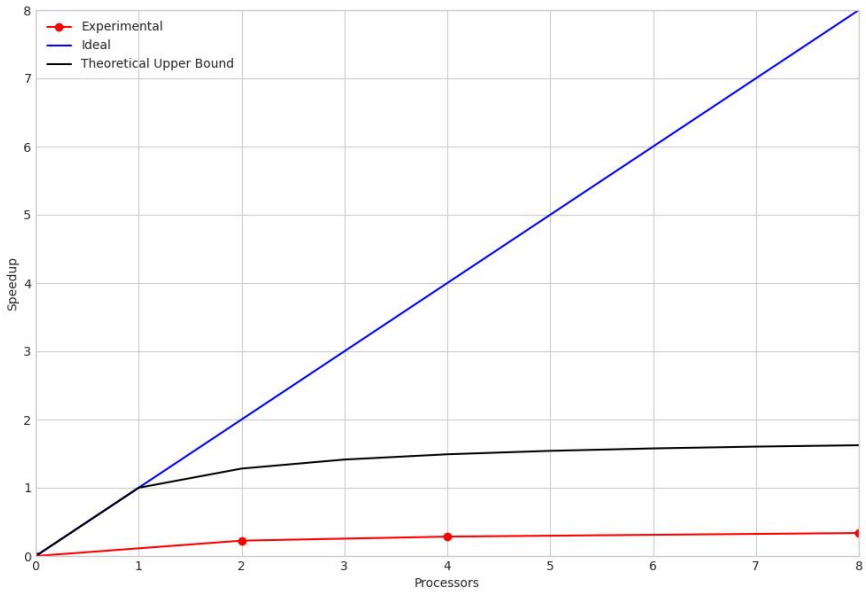
A.1.8.2 Optimization 1

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	819200.000	3.505	3.867	no data	no data	7.138	0.334	87.760	8.790	1.000	1.000
Parallel	2	819200.000	1.927	35.336	1.802	0.742	0.087	0.055	0.000	40.932	0.215	0.107
Parallel	4	819200.000	1.946	19.638	1.834	0.782	0.088	0.052	0.000	31.429	0.280	0.070
Parallel	8	819200.000	2.094	15.667	2.022	0.894	0.138	0.058	0.160	25.870	0.340	0.042



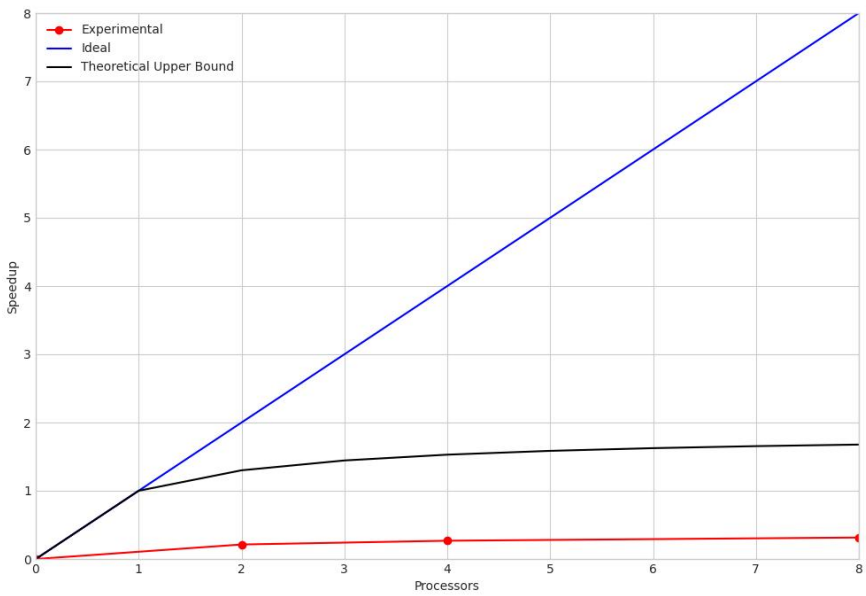
A.1.8.3 Optimization 2

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	819200.000	3.515	3.866	no data	no data	7.140	0.336	87.200	8.807	1.000	1.000
Parallel	2	819200.000	1.924	35.341	1.806	0.738	0.086	0.046	0.000	39.133	0.225	0.113
Parallel	4	819200.000	1.942	20.001	1.836	0.779	0.092	0.048	0.000	30.967	0.284	0.071
Parallel	8	819200.000	2.123	15.935	2.052	0.920	0.187	0.068	0.280	26.136	0.337	0.042



A.1.8.4 Optimization 3

Version	Threads	vertices	init	tarjan	split	merge	user	system	pCPU	elapsed	Speedup	Efficiency
Serial	1	819200.000	3.509	3.868	no data	no data	7.170	0.299	91.360	8.384	1.000	1.000
Parallel	2	819200.000	1.922	35.387	1.802	0.740	0.085	0.054	0.000	39.418	0.213	0.106
Parallel	4	819200.000	1.941	19.971	1.839	0.781	0.088	0.048	0.000	31.298	0.268	0.067
Parallel	8	819200.000	2.125	16.214	2.060	0.904	0.171	0.067	0.280	26.633	0.315	0.039



APPENDIX B

API DOCUMENTATION

The following API documentation were produced:

- HTML documentation accessible in ./docs/html/index.html
- PDF documentation accessible in ./docs/latex/refman.pdf

A copy of the PDF API documentation is also present in the root directory of the project.

REFERENCES

[1] Robert Tarjan. "Depth-First Search and Linear Graph Algorithms". In: *SIAM Journal on Computing* 1.2 (1972), pp. 146–160. DOI: 10.1137/0201010. eprint: https://doi.org/10.1137/0201010. URL: https://doi.org/10.1137/0201010.

[2] Wikipedia contributors. *Strongly connected component*. [Online; accessed 23-December-2022]. 2004. URL: https://en.wikipedia.org/wiki/Strongly_connected_component.

[3] Wikipedia contributors. *Tarjan's strongly connected components algorithm*. [Online; accessed 23-December-2022]. 2006. URL: https://en.wikipedia.org/wiki/Tarjan%27s_strongly_connected_components_algorithm.

[4] Wikipedia contributors. *Graphics processing unit*. [Online; accessed 17-January-2023]. 2005. URL: https://en.wikipedia.org/wiki/Graphics_processing_unit.

[5] Wikipedia contributors. *CUDA*. [Online; accessed 17-January-2023]. 2007. URL: https://en.wikipedia.org/wiki/CUDA.

[6] Wikipedia contributors. *Message Passing Interface*. [Online; accessed 18-January-2023]. 2005. URL: https://en.wikipedia.org/wiki/Message_Passing_Interface.

[7] Wikipedia contributors. *Speedup*. [Online; accessed 19-January-2023]. 2009. URL: https://en.wikipedia.org/wiki/Speedup.

[8] Wikipedia. *Adjacency list — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Adjacency%20list&oldid=1102846197. [Online; accessed 20-January-2023]. 2023.

[9] Attractivechaos. *Klib/khash.h at master · attractivechaos/klib*. 2018. URL: https://github.com/attractivechaos/klib/blob/master/khash.h.

[10] frankfeng579 et al. *Tarjan's Algorithm to find Strongly Connected Components*. 2022. URL: https://www.geeksforgeeks.org/tarjan-algorithm-find-strongly-connected-components/.

[11] Freeman. "Parallel Algorithms for Depth-First Search". In: *Stanford* 1 (1991), pp. 1–34. URL: http://snap.stanford.edu/class/cs224w-2012/projects/cs224w-001-final.pdf.

[12] Reif. "OPTIMAL COST PARALLEL ALGORITHMS FOR LEXICOGRAPmCAL ORDERING". In: *Purdue* (1986). URL: https://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=1520&context=cstech.

[13] Alok Aggarwal, Richard J. Anderson, and Ming-Yang Kao. "Parallel Depth-First Search in General Directed Graphs". In: *SIAM Journal on Computing* 19.2 (1990), pp. 397–409. DOI: 10.1137/0219025. eprint: https://doi.org/10.1137/0219025. URL: https://doi.org/10.1137/0219025.

[14] Jiri Barnat et al. "Computing strongly connected components in parallel on Cuda". In: *2011 IEEE International Parallel; Distributed Processing Symposium* (2011). DOI: 10.1109/ipdps.2011.59.

[15] Jiri Barnat et al. *Computing strongly connected components in parallel on Cuda - Synthesis*. 2011. URL: https://usermanual.wiki/m/35c81553978e1bd3cc4c6641ad9d28783637b1ebaafcdb115b14524eb2993ba6.pdf.

[16] Nvidia. *CUDA C++ Programming Guide*. 2022. URL: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#texture-memory.

[17] *CUDA Pro Tip: Kepler Texture Objects Improve Performance and Flexibility*. en-US. Feb. 2013. URL: https://developer.nvidia.com/blog/cuda-pro-tip-kepler-texture-objects-improve-performance-and-flexibility/ (visited on 01/21/2023).

LIST OF FIGURES

3.1 Key steps of the Forward-Backward Algorithm 52
3.2 CUDA-friendly graph representation 53
3.3 Variation of warp occupancy with respect to block size 55

This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/4.0/ or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.